ISWC 2013
Sydney, Australia

The 12th International Semantic Web Conference
and the 1st Australasian Semantic Web Conference
21-25 October 2013, Sydney, Australia

Proceedings of the ISWC 2013 Posters & Demos Track

ISWC 2013 Posters & Demos

October 21-25, 2013

Editors:

Eva Blomqvist
Tudor Groza

Preface

The ISWC 2013 Posters and Demonstrations Track complements the Research and In-Use tracks of the conference and offers an opportunity for presenting late-breaking research results, ongoing research projects, and speculative or innovative work in progress. The informal setting of the Posters and Demonstrations Track encourages presenters and participants to engage in discussions about the presented work. Such discussions can be invaluable inputs for the future work of the presenters, while offering participants an effective way to broaden their knowledge of the emerging research trends and to network with other researchers.

These proceedings represent the collection of abstracts corresponding to the posters and demos presented at ISWC 2013. Technical posters report on work in progress or completed Semantic Web software systems, while demonstrations showcase innovative Semantic Web related implementations and technologies. In total, there were 104 submissions, of which we accepted 45 demonstrations and 27 posters. We thank the authors of all submissions for their contribution to the program of ISWC 2013.

Furthermore, we would like to thank all the members of the program committee, as well as the additional reviewers who have spent their valuable time within a very tight schedule in prime holiday time to make this program reality. We are grateful to all of these dedicated people for their valuable discussions and feedback, and we wholeheartedly appreciate their voluntary and enthusiastic cooperation. They supported us in creating a balanced mix of posters and demos that address diverse Semantic Web aspects and which will result in an exciting session at the conference. Lastly, we want to thank our fellow organizers of ISWC, foremost the general chair, Chris Welty, and Kerry Taylor for the local organization.

September 2013 Eva Blomqvist & Tudor Groza

Posters & Demos Track – Organization

Track chairs

Eva Blomqvist, Linköping University, Sweden
Tudor Groza, The University of Queensland, Australia

Program Committee

Alessandro Adamou
Sofia Angeletou
Marie-Aude Aufaure
Marut Buranarach
Vinay Chaudhri
Gong Cheng
Oscar Corcho
Mathieu D'Aquin
Yuzhang Feng
Miriam Fernandez
Anna Lisa Gentile
Gunnar Aastrand Grimnes
Gerd Gröner
Peter Haase
Armin Haller
Karl Hammar
Pascal Hitzler
Aidan Hogan
Jason J. Jung
Hanmin Jung
Haklae Kim
Hong-Gee Kim
Yuan-Fang Li
Yuan Ni
Andrea Giovanni Nuzzolese
Viktoria Pammer
Alexandre Passant
Valentina Presutti
Guilin Qi
Marta Sabou
Bernhard Schandl
Francois Scharffe
Giorgos Stoilos
Nenad Stojanovic
Mari Carmen Suárez-Figueroa
Jing Sun
Vojtěch Svátek
Hideaki Takeda
Raphaël Troncy
Tania Tudorache
Victoria Uren
Maria Esther Vidal
Boris Villazón-Terrazas
Haofen Wang
Kewen Wang
Yi Zhou

Additional Reviewers

Panos Alexopoulos
Jean-Paul Calbimonte
David Carral
Jae-Hong Eom
Aldo Gangemi
Jangwon Gim
Adila A. Krisnadhi
Harshit Kumar
Sungin Lee
Raghava Mutharaju
Hyun Namgoong
Jung Ho Um
Zhe Wang
Zhangquan Zhou

Table of Contents

Part I: Demonstrations

Part II: Posters

Part I: Demonstrations

RelClus: Clustering-based Relationship Search

Yanan Zhang, Gong Cheng, and Yuzhong Qu

State Key Laboratory for Novel Software Technology, Nanjing University,
Nanjing 210023, P.R. China
`ynzhang@smail.nju.edu.cn`, `{gcheng, yzqu}@nju.edu.cn`

Abstract. Searching and browsing relationships between entities is an important task in many domains. To support users in interactively exploring a large set of relationships, we present a novel relationship search engine called RelClus, which automatically groups search results into a dynamically generated hierarchy with meaningful labels. This hierarchical clustering of relationships exploits their schematic patterns and a similarity measure based on information theory.

Keywords: Association discovery, exploratory browsing, hierarchical clustering, path finding, relationship search.

1 Introduction

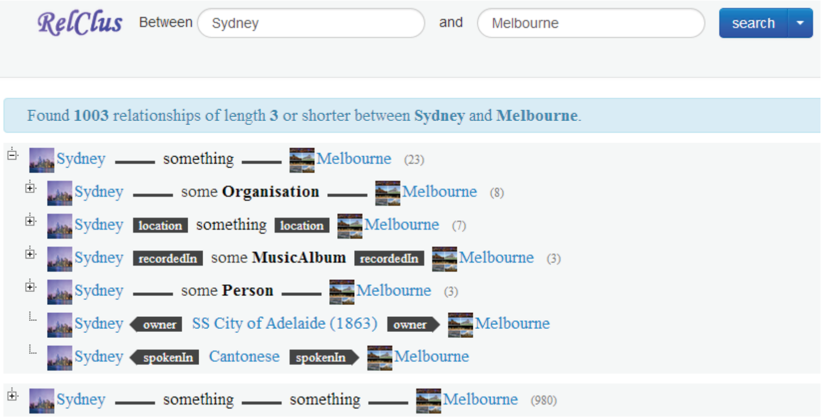
Many information needs in various domains can be met by using an information system that supports searching and browsing relationships (a.k.a. associations) between entities, which are represented as paths in RDF graph. Whereas path finding has been efficiently implemented (e.g. [4]), a major challenge that remains is how to organize the results, which could be a large set of relationships and cause information overload. To address this issue, efforts (e.g. [1]) have been made to rank the results according to various criteria. Another line of work such as [3], inspired by recent advances in exploratory search [2], provides faceted categories to organize search results into groups and serve as filters, each of which characterizes a common feature of the underlying relationships such as their length, or a type of a node (i.e. a class) or edge (i.e. a property) involved. Differently, in this demo we will present a relationship search engine called RelClus[1] that practices another implementation of exploratory search, namely clustering. RelClus measures the similarity between relationships based on their schematic patterns by using information theory, and returns a dynamically generated hierarchical clustering with meaningful labels, to effectively guide the exploration of search results. Figure 1 shows a screenshot of the system.

2 Design and Implementation

RelClus is based on the DBpedia data set (`dbpedia.org`), and consists of four components: keyword mapping, path finding, relationship clustering, and result presentation, which will be detailed in the following.

---

[1] `http://ws.nju.edu.cn/relclus/`.



Found **1003** relationships of length **3** or shorter between **Sydney** and **Melbourne**.

Sydney — something — Melbourne (23)
  Sydney — some **Organisation** — Melbourne (8)
  Sydney location something location Melbourne (7)
  Sydney recordedIn some **MusicAlbum** recordedIn Melbourne (3)
  Sydney — some **Person** — Melbourne (3)
  Sydney owner SS City of Adelaide (1863) owner Melbourne
  Sydney spokenIn Cantonese spokenIn Melbourne
Sydney — something — something — Melbourne (980)

Fig. 1. A screenshot of RelClus.

2.1 Keyword Mapping

User interaction starts with two keyword phrases, e.g. *Sydney* and *Melbourne* in Fig. 1, describing two entities between which the relationships are requested. Featured by the autocomplete functionality implemented based on Apache Lucene (`lucene.apache.org`), RelClus can help the user conveniently determine the mapping from keyword phrases to entities. In addition, when necessary, the user can change the default length limit on the relationships to be returned, by choosing an appropriate value from the drop-down list next to the search button.

2.2 Path Finding

Once the search button is clicked, RelClus will start to find all the paths (subject to a length limit) between the two entities specified by the user. In particular, edges in a relationship are not required to go the same direction because the inverse of a property also has meaning for human readers. Figure 2 illustrates an RDF graph containing five relationships, R_1–R_5, from Sydney to Melbourne, as our running example.

Paths are found by using bidirectional breadth-first search (bi-BFS), which runs two simultaneous searches from the two entities given and finds paths when the two meet in the middle. According to our experimental results, bi-BFS is generally faster than a single BFS or DFS, though requiring more memory than DFS. To further reduce the time needed, our bi-BFS runs concurrently in multiple threads. Besides, a cache is used to avoid repeated path finding for repeated queries in the future.

2.3 Relationship Clustering

As a key step of RelClus, all the relationships found by bi-BFS will be clustered into a hierarchy. For simplicity, the relationships are firstly grouped by length,

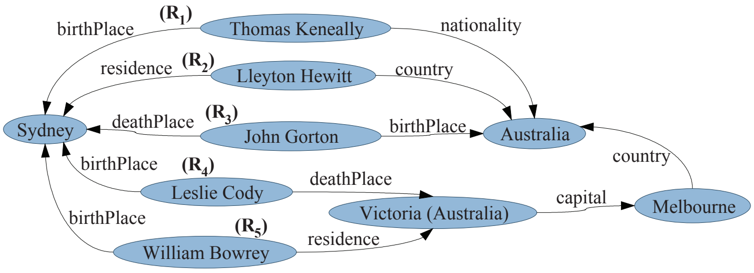


Fig. 2. An RDF graph containing five relationships from `Sydney` to `Melbourne`.

and then each group is processed individually. We will illustrate our clustering algorithm by using R_1–R_5 in Fig. 2, all of which are of length three.

Our clustering algorithm follows an agglomerative manner; that is, each relationship starts in its own cluster, and a hierarchy of clusters is built by progressively merging the most similar pair of clusters.

Before describing the similarity measure, we need to introduce how we assign a meaningful and representative label to each cluster. The label of a cluster is a *relationship pattern*, which is a high-level abstraction of relationships where the nodes can be either entities or classes, and the directions of edges are omitted; the label of a singleton cluster is just the unique relationship it contains. For instance, in Fig. 3, R_4 labels the singleton cluster $\{R_4\}$; P_1 is a relationship pattern that labels the cluster $\{R_4, R_5\}$, where $\top_{\mathbb{P}}$ denotes the top property that is a superproperty of all the properties.

We call P_1 a *superpattern* of R_4 and R_5 in the sense that for each entity, class, and property in R_4 and R_5, the element in the corresponding position in P_1 is either the same or its type, superclass, and superproperty, respectively; in particular, it is the *least common element* among the possibilities. For instance, Leslie Cody in R_4 and William Bowrey in R_5 are instances of both Person and Athlete, and we choose Athlete in P_1 because it is the least common type given Athlete being a subclass of Person, and thus contains the most *information content* as discussed in [5].

We define the similarity between two clusters as *the information content associated with the relationship pattern that labels their union*, which indicates how many commonalities the two clusters share. The information content associated with a relationship pattern is the sum of the information contents of its elements. So, at the beginning of the clustering process in our running example, $\{R_4\}$ and $\{R_5\}$ are merged before $\{R_1\}$ and $\{R_2\}$ because the label of $\{R_4, R_5\}$, which would be P_1, contains more information content than P_2 which would label $\{R_1, R_2\}$, mainly because P_2 contains one more top property, the information content of which is trivially zero.

In addition, after successively forming $\{R_4, R_5\}$ labeled with P_1 and $\{R_1, R_2\}$ labeled with P_2, we will immediately merge $\{R_1, R_2\}$ and $\{R_3\}$ into $\{R_1, R_2, R_3\}$, still labeled with P_2, because R_3 also matches this pattern. Finally, the two

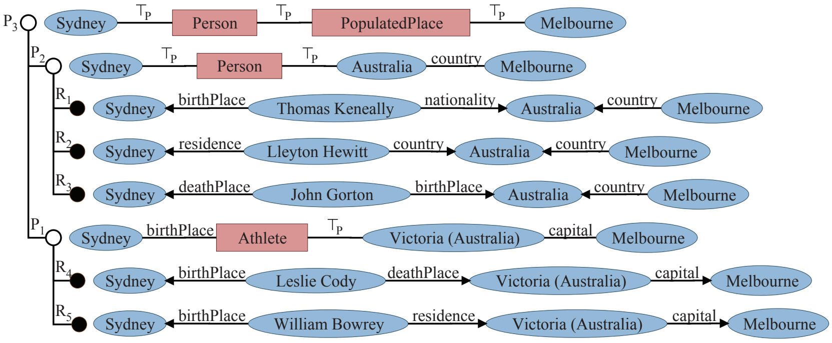


Fig. 3. A hierarchical clustering of five relationships from `Sydney` to `Melbourne`.

clusters labeled with P_1 and P_2 are merged into the root cluster labeled with P_3, and a hierarchy is formed.

2.4 Result Presentation

A hierarchical clustering is visualized as a collapsible/expandable tree, as illustrated in Fig. 1. Each entity is prefixed by a thumbnail if available; each class is prefixed by *some*; the top property is omitted; and the top class is shown as *something*. Each non-singleton cluster is suffixed by its size in parentheses, and sibling clusters are sorted by their sizes decreasingly.

Acknowledgments. This work was supported in part by the NSFC under Grant 61100040 and 61223003, and in part by the JSNSF under Grant BK2012723.

References

1. Aleman-Meza, B., Halaschek-Wiener, C., Arpinar, I.B., Ramakrishnan, C., Sheth, A.P.: Ranking Complex Relationships on the Semantic Web. IEEE Internet Comput. 9(3), 37–44 (2005)
2. Hearst, M.A.: Clustering versus Faceted Categories for Information Exploration. Comm. ACM 49(4), 59–61 (2006)
3. Heim, P., Lohmann. S., Stegemann, T.: Interactive Relationship Discovery via the Semantic Web. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part I. LNCS, vol. 6088, pp. 303–317. Springer, Heidelberg (2010)
4. Janik, M., Kochut, K.: BRAHMS: A WorkBench RDF Store and High Performance Memory System for Semantic Association Discovery. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 431–445. Springer, Heidelberg (2005)
5. Resnik, P.: Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In: 14th International Joint Conference on Artificial Intelligence, Volume 1, pp. 448–453. Morgan Kaufmann, San Francisco (1995)

DiTTO: Diagrams Transformation inTo OWL

Aldo Gangemi[1,2] and Silvio Peroni[1,3]

[1] STLab-ISTC, Consiglio Nazionale delle Ricerche (Italy)
[2] LIPN, Universit Paris 13 (France)
[3] Department of Computer Science and Engineering, University of Bologna (Italy)
aldo.gangemi@cnr.it, essepuntato@cs.unibo.it

Abstract. In this paper we introduce DiTTO, an online service that allows one to convert an E/R diagram created through the yEd diagram editor into a proper OWL ontology according to three different conversion strategies.

1 Introduction

Ontology design is an activity that involves the use of many different languages, resources, and technologies. When dealing with formal or semi-formal languages used by different people such as domain experts, knowledge engineers or final users, a correct transformation strategy can be crucial to guarantee an effective design. For instance, it seems preferable to adopt intuitive languages when the ontology should be introduced to and/or discussed with an heterogeneous audience, which may not be expert of formal languages and knowledge representation. In these cases, graphical languages seem to support well both ontology understanding and the discussion between knowledge engineers and final users.

The recent project ran by the Italian National Research Council (CNR)[4] and SOGEI[5], the Information and Communication Technology company owned by the Italian Ministry of Economy and Finance, presents the aforementioned scenario. The aim of the CNR-SOGEI project is twofold. One goal is the re-engineering of the E/R (Entity-Relationship) models SOGEI use to describe fiscal entities (such as taxpayers, laws, deposits, etc.) into standard Semantic Web languages, such as OWL 2. Another goal is to propose tools that facilitate future interactions (changes, re-use in different application contexts, etc.) with these semantic models.

In this paper we present one of the aforementioned tools called *DiTTO*, which stands for *Diagrams Transformations inTo OWL*. As its name suggest, DiTTO is able to translate E/R diagrams expressed in crow's foot notation and created with *yEd*[6], an open source application to quickly and effectively generate high-quality diagrams, into OWL ontologies.

---

[4] CNR homepage: http://www.cnr.it.

[5] SOGEI homepage: http://www.sogei.it.

[6] yEd homepage: http://www.yworks.com/en/products_yed_about.html.

In Section 2 we present some relevant diagram models originally developed for particular purposes (e.g. software engineering, databases, AI), and progressively adapted to also model OWL ontologies. In Section 3 we introduce DiTTO, describing its implementation and showing how to use it for diagram transformation. We also discuss a set of transformation rules to convert E/R diagrams into OWL. Finally, in Section 4, we present our plans for future developments of the tool, in terms of both new transforming features and diagrams support.

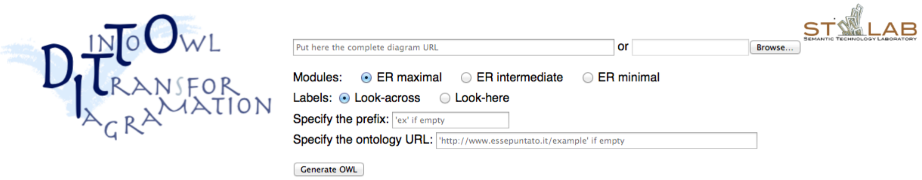
2 Related work

One of the most common graphical notations for logical languages is a *semantic network*, which can be defined as a "graphic notation for representing knowledge in patterns of interconnected nodes and arcs" [5]. Ontology classes and individuals are defined as nodes of a graph; at the same time, direct and labelled arcs can interlink nodes in order to represent predicates between them.

Gasevic *et al.* [3] and Brockmans *et al.* [2] propose another UML profile that enables one to define OWL entities using an extended set of UML-based graphic notations. The industry consortium responsible of UML, the Object Management Group, released an official UML profile [4] for defining OWL ontologies, called *Ontology Definition Metamodel* (*ODM*). TopBraid Composer[7] is a commercial tool featuring a diagramming component that adopts a proprietary UML profile to represent a substantial part (focusing on subclasses and associations, including restrictions and class constructions) of OWL semantics.

3 From E/R to OWL

DiTTO has been implemented as a Web service[8], as shown in Fig. 1.



Put here the complete diagram URL or

Modules: ● ER maximal ○ ER intermediate ○ ER minimal

Labels: ● Look-across ○ Look-here

Specify the prefix: 'ex' if empty

Specify the ontology URL: 'http://www.essepuntato.it/example' if empty

Fig. 1. The home page of DiTTO.

The core of DiTTO is a set of XSLT 2.0 documents included in a Java Web Application Archive (i.e. a WAR file) served as a Tomcat application. These XSLT documents apply several rewriting templates to the source file of the E/R diagram created by means of yEd, which is stored in GraphML format[9].

---

[7] http://www.topbraidcomposer.com

[8] http://www.essepuntato.it/ditto

[9] GraphML specification: http://graphml.graphdrawing.org/specification.html.

Using the service is quite simple. One needs to specify a URL referring to a yEd diagram, or, alternatively, to choose such a diagram from the file system. Then, after choosing among the available options and after specifying the prefix for naming ontological entities and the full URI of the ontology itself, the "Generate OWL" button can be used to produce an RDF/XML file containing the OWL ontology result of the conversion.

The transformation rules DiTTO implements are as follows. E/R *entities*, *attributes* and *relations* are converted into OWL *classes*, *data properties* and *object properties* respectively. All the subtype relations between E/R entities -▷ are converted into rdfs:subClassOf axioms. In addition, DiTTO adds appropriate restrictions to classes according to the edges that link E/R entities. In particular:

- a link between an E/R entity and an attribute results in restricting the related OWL class with a qualified cardinality;
- a symbol -○< (i.e. zero-to-many) of a relation results in restricting the domain class with a universal quantifier. This restriction is also added if any of the symbols in the following point is specified;
- the symbols |< (i.e. one-to-many), || (i.e. one-to-one) and |○- (i.e. zero-to-one) result in restricting the domain class with an existential quantifier, a qualified cardinality and a qualified maximum cardinality respectively.

In addition to these rules, DiTTO allows one to chochooose what E/R semantics to apply for the transformation. We have identified three alternative conversion strategies, which depends on the application of two assumptions:

- *global semantics* (*GS*) is a characteristic of OWL ontologies (but not typically of E/R), and has the effect of unifying the formal interpretation of domain and range axioms, property characteristics, and all the restrictions that act at a global level. When GS does not hold, one is not allowed to assume such unification, even when the axioms regard two constants with the same name;
- *unique name assumption* (*UNA*), which is a characteristic in E/R semantics (but not of OWL), and whose consequence is that two objects named differently always refer to different entities in the world.

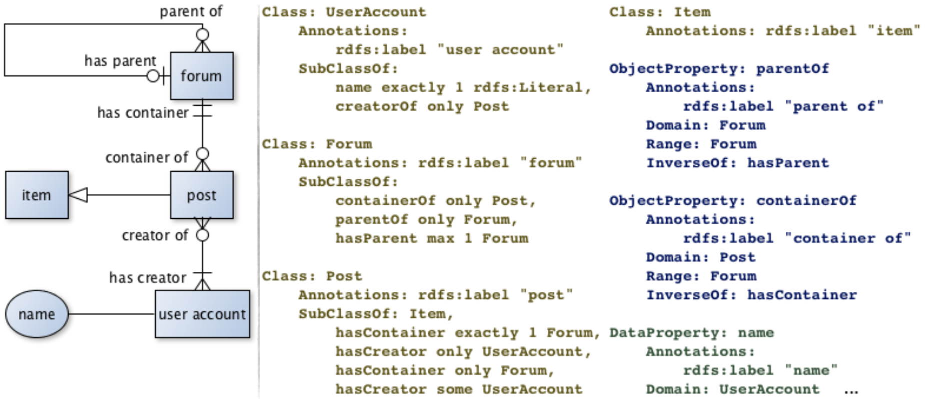
In particular, the *minimal strategy* interprets the semantics of E/R in its purest form (cf. [1]) by not using GS, while using UNA, the *intermediate strategy* does not use either GS or UNA, and finally the *maximal strategy*, which is the closest to OWL semantics, use GS, but UNA does not hold.

Different strategies proved useful in the aforementioned project because they address different requirements. The minimal strategy is of course conservative with the possible conceptualisations admitted by the original specification, while the maximal one allows us to simulate the consequences of E/R into OWL semantics, with its pros and cons, e.g. suggesting domains and ranges, or unification of properties, as well as spotting potential issues when applying the simulation.

In Fig. 2 we illustrate a small E/R diagram based on SIOC[10], and an excerpt of the OWL ontology returned by DiTTO by using the maximal strategy for the conversion.

---

[10] SIOC Ontology: http://sioc-project.org/ontology.



```
Class: UserAccount                    Class: Item
    Annotations:                          Annotations: rdfs:label "item"
        rdfs:label "user account"
    SubClassOf:                       ObjectProperty: parentOf
        name exactly 1 rdfs:Literal,      Annotations:
        creatorOf only Post                   rdfs:label "parent of"
                                          Domain: Forum
Class: Forum                              Range: Forum
    Annotations: rdfs:label "forum"       InverseOf: hasParent
    SubClassOf:
        containerOf only Post,        ObjectProperty: containerOf
        parentOf only Forum,              Annotations:
        hasParent max 1 Forum                 rdfs:label "container of"
                                          Domain: Post
Class: Post                               Range: Forum
    Annotations: rdfs:label "post"        InverseOf: hasContainer
    SubClassOf: Item,
        hasContainer exactly 1 Forum, DataProperty: name
        hasCreator only UserAccount,      Annotations:
        hasContainer only Forum,              rdfs:label "name"
        hasCreator some UserAccount       Domain: UserAccount   ...
```

Fig. 2. The exemplar E/R diagram developed through yEd and the conversion (shown in Manchester Syntax) produced by DiTTO using the maximal strategy.

4 Conclusions

In this paper we presented DiTTO, an online service that converts E/R diagrams created with the yEd editor into proper OWL ontologies according to three different conversion strategies: minimal, intermediate and maximal. Future extensions of the tool will address the management of additional E/R features (such as primary and foreign keys, and multiple and optional attributes), as well as different kinds of diagrams (e.g. UML and Graffoo[11]) as input.

References

1. Berardi, D., Calvanese, D., De Giacomo, G. (2005). Reasoning on UML class diagrams. In Artificial Intelligence, 168 (1-2): 70-118. DOI: 10.1016/j.artint.2005.05.003
2. Brockmans, S., Haase, P., Hitzler, P., Studer, R. (2006): A Metamodel and UML Profile for Rule-Extended OWL DL Ontologies. In Proceedings of the 3rd European Semantic Web Conference. DOI: 10.1007/11762256_24
3. Gasevic, D., Djuric, D., Devedzic, V., Damjanovic, V. (2004). Converting UML to OWL Ontologies. In Proceedings of the 13th international World Wide Web Conference on Alternate track papers & posters. DOI: 10.1145/1013367.1013539
4. Object Management Group (2009). Ontology Definition Metamodel (ODM) Version 1.0. http://www.omg.org/spec/ODM/1.0/PDF
5. Sowa, J. F. (1987). Semantic Networks. In Encyclopedia of Artificial Intelligence. John Wiley & Sons. ISBN: 0471503053.

---

[11] Graffoo: http://www.essepuntato.it/graffoo.

CEDAR: a Fast Taxonomic Reasoner Based on Lattice Operations

Samir Amir and Hassan Aït-Kaci

Université Claude Bernard Lyon 1, France

`{samir.amir,hassan.ait-kaci}@univ-lyon1.fr`

Abstract. Taxonomy classification and query answering are the core reasoning services provided by most of the Semantic Web (SW) reasoners. However, the algorithms used by those reasoners are based on Tableau method or Rules. These well-known methods in the literature have already shown their limitations for large-scale reasoning. In this demonstration, we shall present the CEDAR system for classifying and reasoning on very large taxonomies using a technique based on lattice operations. This technique makes the CEDAR reasoner perform on par with the best systems for concept classification and several orders-of-magnitude more efficiently in terms of response time for query-answering. The experiments were carried out using very large taxonomies (Wikipedia: 111599 sorts, MESH: 286381 sorts, NCBI: 903617 sorts and Biomodels: 182651 sorts).[1] The results achieved by CEDAR were compared to those obtained by well-known Semantic Web reasoners, namely FaCT++, Pellet, HermiT, TrOWL, SnoRocket and RacerPro.

Keywords: Semantic Reasoning, Lattice Operations, Partial-Order Encoding

1 Introduction

The demo will demonstrate how an implementation of a system based on lattice operations can be used for taxonomic reasoning in a robust and scalable way. Indeed, this challenge was defined on the frame of CEDAR project. [2] CEDAR system is a Boolean reasoner that can support a huge amount of sorts without any noticeable degradation of query evaluation performance. The essential technique we used for implementing the CEDAR reasoner is based on bit-vector encoding. This method was proposed over 20 years ago for implementing efficient lattice operations [1]. Since the common aspect of all Semantic Web reasoning systems is the representation and processing of taxonomic data, we implemented a taxonomic concept classification and Boolean query-answering system based

---

[1] We use "sort" as a synonym of atomic "class" or "concept." In other words, sorts are partially ordered symbols.

[2] Constraint Event-Driven Automated Reasoning—`http://cedar.liris.cnrs.fr`

on the method described above. We made measurements over several very large taxonomies under the exact same conditions for so-called TBox reasoning. A comparative evaluation was conducted to assess the performance of CEDAR over the mentioned systems which have been implemented by using OWL-API. [3] In terms of query-answering response time, CEDAR is several orders-of-magnitude more efficient than that of the best existing SW reasoning systems.

2 Lattice Operations for Taxonomic Reasoning

The CEDAR reasoner is an implementation in Java of the technique described as bottom-up transitive-closure encoding in [1]. This technique consists in representing the elements of a taxonomy (an arbitrary poset) as bit vectors. Thus, each element has a code (a bit vector) carrying a "`1`" in the position corresponding to the index of any other elements that it subsumes. In this manner, the three Boolean operations on sorts are readily and efficiently performed as their corresponding operations on bit-vectors. This is possible if the bit-vectors encoding the sorts comprising a taxonomy are obtained by computing the reflexive-transitive closure of the "is-a" relation derived from the subsort declarations.

3 Demonstration

The demo will show how CEDAR differs from existing and known reasoners in terms of classification (Figures 1 and 2) and query answering (Figures 3 and 4) where it is several orders-of-magnitude more efficient than other reasoners. Developed software integrates six other reasoners to provide a comparison with CEDAR (HermiT [4], FaCT++ [7], RacerPro [2], TrOWL [6], Pellet [5] and SnoRocket [3] all of which use the OWL-API interface). The proposed structure of the demonstration is the following:

- Classification performance using very large taxonomies as Wikipedia[4] (111599 sorts), NCBI[5](903617 sorts), MESH[6] (286381 sorts) and Biomodels[7] (182651 sorts). The demo will show the results illustrated in Figures 1 and 2 where CEDAR is always among the best three out of six reasoners.
- Query Answering using boolean queries (`and`, `or` and `not`) involving a large number of concepts (up to 100 concepts). The obtained results will be compared with those of traditional reasoners as shown in Figures 3 and 4.
- Saving and reusing an encoded taxonomy. With CEDAR, there is no need to perform a classification each time. A classified taxonomy can be saved and reused.
- Detecting Cycles: We will show how to detect cycles in taxonomies, which are a particular case of inconsistency resulting from modeling errors.

---

[3] http://owlapi.sourceforge.net

[4] http://www.h-its.org/english/research/nlp/download/wikitaxonomy.php

[5] http://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html/

[6] http://www.nlm.nih.gov/mesh/meshhome.html

[7] https://code.google.com/p/sbmlharvester/

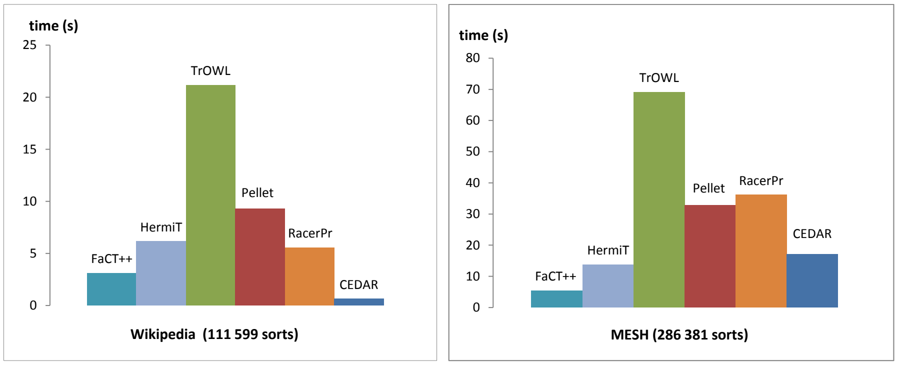


Fig. 1. Classification time per reasoner for the Wikipedia and MeSH taxonomies

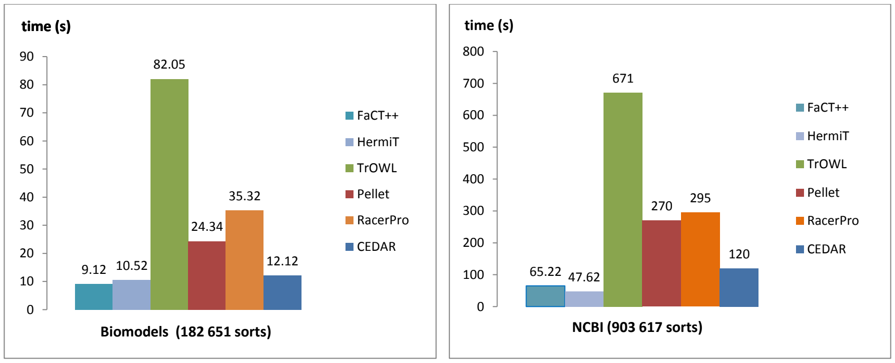


Fig. 2. Classification time per reasoner for the Biomodels and NCBI taxonomies

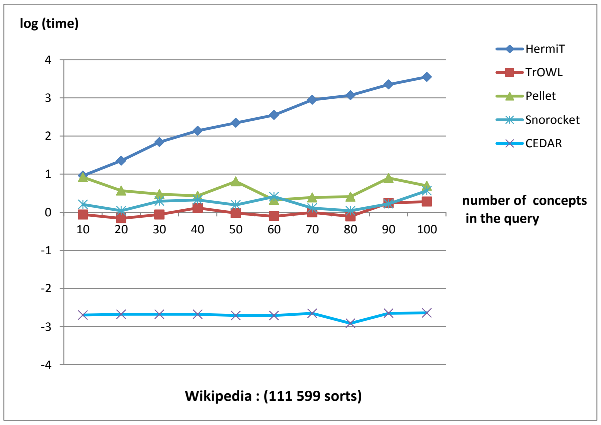


Fig. 3. Query response time per reasoner for the Wikipedia taxonomy

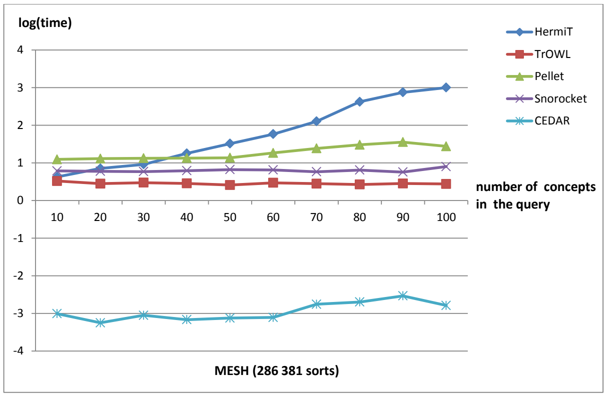


Fig. 4. Query response time per reasoner for the MESH taxonomy

4 Acknowledgements

The authors wish to thank Mohand-Saïd Hacid for his comments. This work was carried out as part of the CEDAR Project (Constraint Event-Driven Automated Reasoning) under the *Agence Nationale de la Recherche* (ANR) Chair of Excellence grant N^o **ANR-12-CHEX-0003-01** at the *Université Claude Bernard Lyon 1*.

References

1. Aït-Kaci, H., Boyer, R., Lincoln, P., and Nasr, R. Efficient implementation of lattice operations. *ACM Transactions on Programming Languages and Systems 11*, 1 (January 1989), 115–146.
2. Haarslev, V., Hidde, K., Möller, R., and Wessel, M. The RacerPro knowledge representation and reasoning system. *Semantic Web Journal 1* (March 2011), 1–11.
3. Lawley, M. J., and Bousquet, C. Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner. In *Proceedings of the 2nd Australasian Ontology Workshop: Advances in Ontologies* (Adelaide, Australia, December 2010), T. Meyer, M. A. Orgun, and K. Taylor, Eds., AOW'10, ACS, pp. 45–50.
4. Shearer, R., Motik, B., and Horrocks, I. HermiT: A highly-efficient OWL reasoner. In *Proceedings of the 5th International Workshop on OWL Experiences and Directions* (Karlsruhe, Germany, October 2008), U. Sattler and C. Dolbear, Eds., OWLED'08, CEUR Workshop Proceedings.
5. Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics 5*, 2 (June 2007), 51–53.
6. Thomas, E., Pan, J. Z., and Ren, Y. TrOWL: Tractable OWL 2 reasoning infrastructure. In *Proceedings of the 7th Extended Semantic Web Conference* (Heraklion, Greece, May-June 2010), ESWC'10, Springer-Verlag, pp. 431–435.
7. Tsarkov, D., and Horrocks, I. FaCT++ description logic reasoner: System description. In *Proceedings of the 3rd International Joint conference on Automated Reasoning* (Seattle, WA, USA, August 2006), U. Furbach and N. Shankar, Eds., IJCAR'06, Springer-Verlag, pp. 292–297.

Demonstrating The Entity Registry System: Implementing 5-Star Linked Data Without the Web

Marat Charlaganov[1], Philippe Cudré-Mauroux[2], Cristian Dinu[1],
Christophe Guéret[1], Martin Grund[2], and Teodor Macicas[2]*

[1] DANS, Royal Dutch Academy of Sciences—The Netherlands
`{firstname.lastname}@dans.knaw.nl`

[2] eXascale Infolab, University of Fribourg—Switzerland
`{firstname.lastname}@unifr.ch`

Abstract. Linked Data applications often assume that connectivity to data repositories and entity resolution services are always available. This may not be a valid assumption in many cases. Indeed, there are about 4.5 billion people in the world who have no or limited Web access. Many data-driven applications may have a critical impact on the life of those people, but are inaccessible to such populations due to the architecture of today's data registries. In this demonstration, we show how our new open-source ERS system can be used as a general-purpose entity registry suitable for deployment in poorly-connected or ad-hoc environments.

1 Introduction

There is an estimated number of 2 billion individuals who have access to the Internet and can thus use centralized cloud hosted solutions for sharing data. Many of these centralized solutions are well-known (Facebook, Wikipedia, WikiData, etc.) and make it possible to share semi-structured data about entities. Linked Data comes into this picture as a solution to interlink the isolated data silos by linking those entities through semantically rich connections. The expected outcome being a globally connected data space everyone can contribute to.

Unfortunately those who do not have access to seamless data connectivity and web hosting services can not benefit from Linked Data. Even when computers are interconnected through local mesh networks, the dependency on web platforms makes it impossible to de-reference the description of an entity.

For example, let us consider the case of the XO laptops deployed by the OLPC (One-Laptop-Per-Child) foundation[3]. OLPC brings Information and Communication Technology (ICT) to young learners in the poorest areas of the world so that they can develop new skills and work collaboratively using multimedia applications. So far, two million children world-wide have received an XO and use it to work with their peers. Data-sharing is however limited to synchronous messages using XMPP-based channels between two running instances of an application. In

---

* Authors are listed in alphabetical order.

[3] `http://one.laptop.org/`

this context, the asynchronous editing of a database shared by different applications is a challenging architectural problem. External data-hosting, pre-defined schemas and data-caching can be a solution: “Sugar Network”[4], a data-sharing service built for Sugar—the learning environment of the XO—implements such a platform for community support.

This kind of approach is however limited in scope and requires to have some connectivity to the central server. The goal of the Entity Registry System (ERS)[5] is to provide a lightweight, versatile, linked data publication tool that does not rely on third party data hosting or services. ERS replaces the Web as a platform for publishing linked data. It lets a swarm of small devices interconnected in an intermittent way create/update/delete entities within a globally shared data-space. By having the triples hosted directly on the machines creating them, the system supports different connectivity contexts.

ERS tackles one of the three challenges for accelerating the adoption of Linked Data and data-intensive applications in developing parts of the world [1, 3].

2 The Entity Registry System (ERS)

ERS is designed around lightweight components: Contributors, Bridges, and Global Servers, which collaboratively support data-sharing and data-intensive applications in intermittently connected settings. It is compatible with the RDF data model and makes use of the available connectivity to share data, but does not base its content publication strategy on the Web. No single component is required to hold a complete copy of the registry. The global content consists of the union of what every component decides to share. We hereafter briefly describe the components and the implementation. The interested reader is invited to consult [2] for more details on the system and on performance considerations.

2.1 Components

Contributor: Contributors read and edit the content of the registry. They may create and delete entities, look for entities, and contribute to the description of entities. Every contribution is identified by the contributor name so that the collectively-created description of an entity can be traced back to individual contributors. Contributors are free to make any statement about any entity in the system. They use a local data-store in which they persist their contributions to the description of the entities. They may also cache the contributions of others when appropriate.

Bridge: Bridges do not directly contribute to the content of the registry. They are used to connect isolated closed networks and improve the availability of the individual descriptions shared by the contributors. Bridges can theoretically store content coming from any contributor, but will typically store the data only for a limited amount of time.

---

[4] http://wiki.sugarlabs.org/go/Sugar_Network

[5] `http://worldwidesemanticweb.org/projects/entity-registries/`

Global Server: ERS deployments can feature any number of bridges and contributors. In addition, some use-cases may require the presence of global servers that contain a copy of all the data going through the bridges. A global server provides a single entry point to the registry content. It exposes the contents of an ERS to other systems, for instance to the Web of Data.

2.2 Implementation

URNs of the form `urn:ers:<path>:<identifier>` are used to uniquely identify entities and contributors within an ERS. Individual contributions, in the form of triples, are stored in CouchDB instances run by the contributors. CouchDB's synchronisation system is used to propagate these contributions in the network by replicating them with other contributors or bridges. In addition, a search feature enables running federated queries over a set of CouchDB instances. The system source code is available at https://github.com/ers-devs/ers under an open licence.

3 Demonstration scenario

Figure 1 shows the sample deployment we created for this demonstration featuring three different physical locations, eight contributors, two bridges, and a global server. In our setup, we create one physical class-room scenario with multiple semi-connected devices consisting of multiple OLPC XO laptops, a class-room bridge server on a RaspberryPi and a dedicated global server that is connected via Internet from Fribourg, Switzerland.

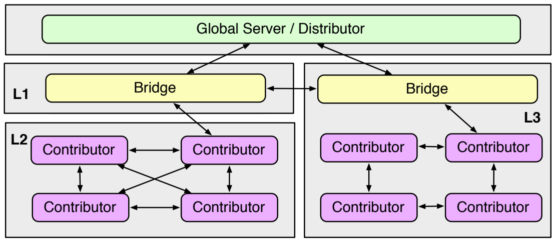
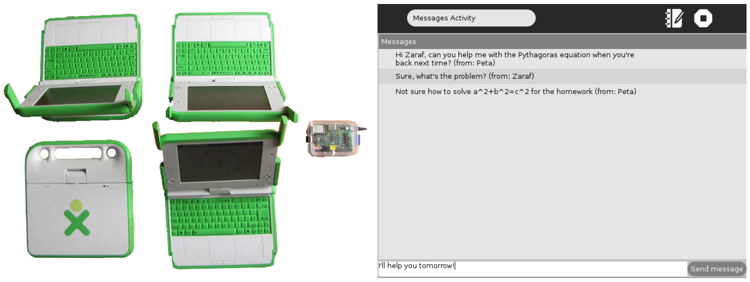


Fig. 1. An example ERS deployment across three different locations

The contributors (XO laptops) are creating, consuming and storing structured data about entities. One bridge is used to ensure information flow and data distribution between the nodes, even if there is no reliable direct connection between two contributors. The global server is used to expose the entities within ERS as de-referencable HTTP URIs.

In our sample application, we support asynchronously discussion among school pupils. ERS is used to edit the content of a global Q&A database. In contrast to common approaches, the messages are stored and served from the laptop



Hi Zaraf, can you help me with the Pythagoras equation when you're back next time? (from: Peta)

Sure, what's the problem? (from: Zaraf)

Not sure how to solve a^2+b^2=c^2 for the homework (from: Peta)

I'll help you tomorrow!

(a) Four contributors and a bridge (b) The messaging application

Fig. 2. Demo setup 2(a) and messaging application 2(b)

of their publishers directly. These questions and answers are stored as entities described and interlinked using common vocabularies (SIOC, RDF, etc.). To post a message, the software creates a new entity and puts the text, the name of the creator and a visibility status (public/private) as part of the description of the said entity. When appropriate, these triples gets then automatically replicated to other devices, eventually transiting through a bridge. Links between messages are established by referring to the identifiers of the entities when adding new messages, thereby creating conversation threads.

A video has been recorded to show the asynchronous dispatch of messages between XO devices. This scenario involves two XOs from the 2007 generation and a RaspberryPi model B used as a bridge. The video can be seen on Vimeo at `https://vimeo.com/70883238`.

Acknowledgment

This work was supported by the Verisign 2012 *Internet Infrastructure Grant* program.

References

1. The World Wide Semantic Web community. `http://worldwidesemanticweb.org/`, visited Aug 20, 2013.
2. Marat Charlaganov, Philippe Cudré-Mauroux, Cristian Dinu, Christophe Guéret, Martin Grund, and Teodor Macicas. The Entity Registry System: Implementing 5-Star Linked Data Without the Web. *arXiv preprint*, August 2013.
3. Christophe Guéret, Stefan Schlobach, Victor De Boer, Anna Bon, and Hans Akkermans. Is data sharing the privilege of a few ? Bringing Linked Data to those without the Web. In *Proceedings of ISWC2011 - "Outrageous ideas" track, Best paper award*, pages 1–4. Best paper award, 2011.

NoHR: Querying $\mathcal{EL}$ with Non-monotonic rules

Vadim Ivanov[1,2], Matthias Knorr[1], and João Leite[1]

[1] CENTRIA & Departamento de Informática, Universidade Nova de Lisboa, Portugal
[2] Department of Computing Mathematics and Cybernetics, Ufa State Aviation Technical University, Russia

Abstract. We present NoHR, a Protégé plug-in that allows the user to take an $\mathcal{EL}^{+}_{\bot}$ ontology, add a set of non-monotonic (logic programming) rules – suitable e.g. to express defaults and exceptions – and query the combined knowledge base. Provided the given ontology alone is consistent, the system is capable of dealing with potential inconsistencies between the ontology and the rules, and, after an initial brief pre-processing period utilizing OWL 2 EL reasoner ELK, returns answers to queries at an interactive response time by means of XSB Prolog.

1 Introduction

Ontology languages have become widely used to represent and reason over taxonomic knowledge, and often such knowledge bases are expressed within the language of the OWL 2 profile OWL 2 EL.[1] For example, the clinical health care terminology SNOMED CT,[2], arguably the most prominent example in the area of medicine and currently used for electronic health record systems, clinical decision support systems, or remote intensive care monitoring, to name only a few, builds on a fragment of OWL 2 EL and its underlying description logic (DL) $\mathcal{EL}^{++}$ [2].

Since OWL and its profiles are based on DLs [3], hence monotonic by nature, which means that once drawn conclusions persist when adopting new additional information, the ability to model defaults and exceptions with a closed-world view is frequently requested as a missing feature. For example, in clinical health care terminology, it would be advantageous to be able to express directly that normally the heart is on the left side of the body unless the person is a dextrocardiac, which matters when applying ECG or defibrillation to a patient.

In recent years, there has been a considerable amount of effort devoted to extending DLs with non-monotonic features – see, e.g., related work in [8] – many of the existing approaches focusing on combining DLs and non-monotonic rules. The latter are one of the most well studied formalisms (in the area of Logic Programming) that admit expressing defaults, exceptions, and also integrity constraints in a declarative way. As such, they are part of the RIF,[3] the other language for the Semantic Web whose standardization is driven by the W3C.[4]

---

[1] http://www.w3.org/TR/owl2-profiles/
[2] http://www.ihtsdo.org/snomed-ct/
[3] http://www.w3.org/TR/rif-overview/
[4] http://www.w3.org

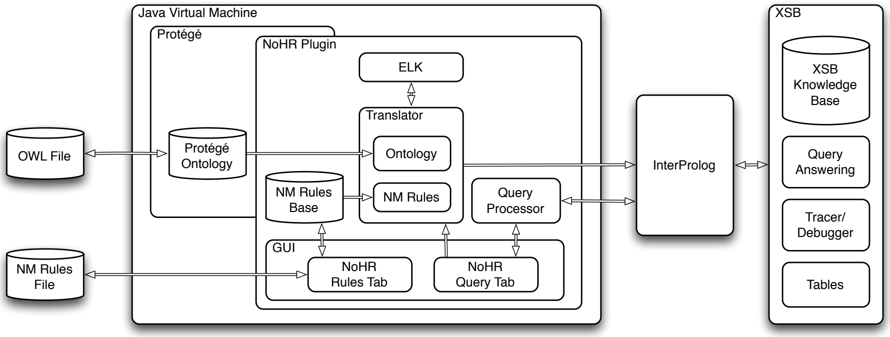


Fig. 1. System Architecture of NoHR

Here, we focus on Hybrid MKNF under the well-founded semantics [7] combining ontologies and such rules, because, as argued for the preceding semantics in [8], the overall framework is very general and flexible, and unlike [8], [7] has a polynomial data complexity and admits top-down query-answering based only on the information relevant for the query, and without computing the entire model – no doubt a crucial feature when dealing with large ontologies such as SNOMED with over 300,000 classes.

In our ISWC 2013 Research Track paper [5], we describe a system, realized as a plug-in for the ontology editor Protégé 4.X,[5] that allows the user to query combinations of $\mathcal{EL}_{\bot}^{+}$ ontologies and non-monotonic rules in a top-down manner. To the best of our knowledge, it is the first Protégé plug-in to integrate non-monotonic rules and top-down queries. Our approach is theoretically founded on the abstract procedure **SLG($\mathcal{O}$)** [1] and developed upon the usage of the consequence-driven, concurrent $\mathcal{EL}$ reasoner ELK [6] to classify the ontology part, whose result is translated into rules which, together with the non-monotonic rules, subsequently serve as input for the top-down query engine XSB Prolog.[6] Additional features of the plug-in include: the possibility to load and edit rule bases, and define predicates with arbitrary arity; guaranteed termination of query answering, with a choice between one/many answers; robustness w.r.t. potential inconsistencies between the ontology and the rules in case the $\mathcal{EL}_{\bot}^{+}$ ontology contains $DisjointWith$ axioms; leveraging of XSB tabling mechanisms to improve performance, and trace/debug features, e.g., to provide explanations.

2 System Description

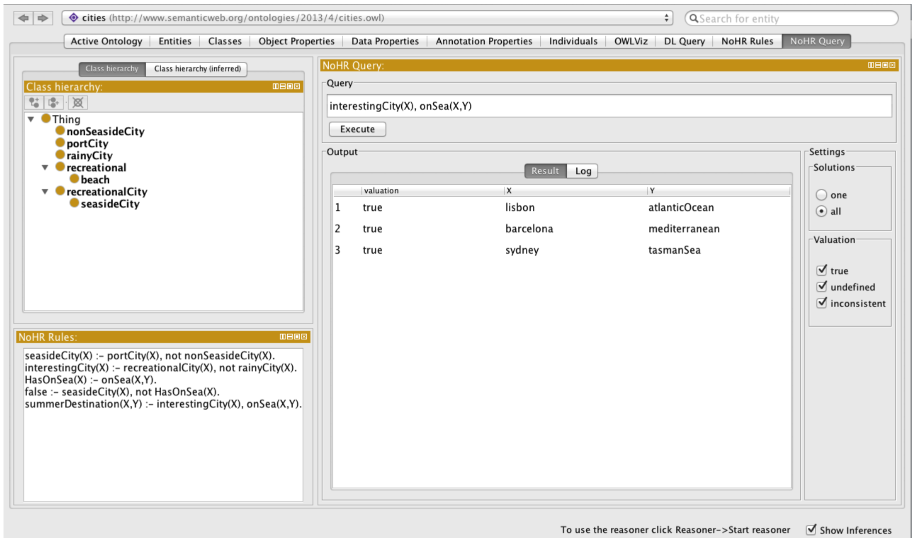
In this Section, we briefly describe the architecture of NoHR, our plug-in for Protégé, as shown in Fig. 1 and discuss some features of our implementation and querying in XSB. For the technical details and the evaluation of our approach, we refer to [5].

The input for our plug-in consists of an OWL file, which can be manipulated as usual in Protégé, and a rule file. For the latter, we provide a tab called NoHR Rules that

---

[5] `http://protege.stanford.edu`

[6] `http://xsb.sourceforge.net`



Query: interestingCity(X), onSea(X,Y)

	valuation	X	Y
1	true	lisbon	atlanticOcean
2	true	barcelona	mediterranean
3	true	sydney	tasmanSea

Solutions: ○ one ● all

Valuation: ☑ true ☑ undefined ☑ inconsistent

```
seasideCity(X) :- portCity(X), not nonSeasideCity(X).
interestingCity(X) :- recreationalCity(X), not rainyCity(X).
HasOnSea(X) :- onSea(X,Y).
false :- seasideCity(X), not HasOnSea(X).
summerDestination(X,Y) :- interestingCity(X), onSea(X,Y).
```

Fig. 2. NoHR Query Tab with a query $interestingCity(X), onSea(X, Y)$

allows the user to load, save and edit rule files in a text panel. The syntax follows Prolog conventions, so that one rule from Ex. 2 in [5] can be represented, e.g., by

```
SeaSideCity(X) :- PortCity(X), not NonSeaSideCity(X).
```

The NoHR Query tab as shown in Fig. 2 also allows for the visualization of the rules (in the lower left corner), but its main purpose is to provide an interface for querying the combined KB. Whenever the first query is posed by pushing "Execute", the translator is started, initiating the ELK reasoner to classify the ontology and return the result to the translator. It is verified whether $DisjointWith$ axioms appear in $\mathcal{O}$ which determines whether the transformation into rules has to contain means to check for potential inconsistencies or not. Then, accordingly, a joint (non-monotonic) rule set is created in which predicates and constants, i.e., all terms, are encoded using MD5. This requires the user to write case-sensitive rules (w.r.t. to the ontology), but ensures full compatibility with XSB Prolog's more restrictive admitted input syntax. The resulting program is transfered to XSB via InterProlog [4], which is an open-source Java front-end that provides the ability to communicate between Java and a Prolog engine.

Next, the query can be sent via InterProlog to XSB, and answers are returned to the query processor, which collects them and sets up a table showing for which variable substitutions we obtain *true*, *undefined*, or *inconsistent* valuations (or just shows the truth value for a ground query). The table itself is shown in the Result tab of the Output panel (see Fig. 2), while the Log tab shows measured times and system messages, including those from XSB via InterProlog. XSB not only answers queries very efficiently in a top-down manner, with tabling, it also avoids infinite loops.

Once the query has been answered, the user may pose other queries, and the system will simply send them to XSB directly without any repeated preprocessing. If the user changes data in the ontology or in the rules, then the system offers the option to recompile, but always restricted to the part that actually changed.

During the demo exhibition, we take a given/chosen ontology loaded into Protégé, and we show, first how to edit, load, and save rules, and subsequently, run queries on the combined knowledge base. In particular, we interactively demonstrate how changing the ontology and the rules affects the query results, also in the presence of inconsistencies between the ontology and the rule set. For that purpose, we use data sets of two kinds, namely toy examples for which the query result can be verified right away and some of the real world ontologies, utilized already during testing in [5] (cf. Fig.3), to which we add non-monotonic rules.

Our plug-in is under active development and the most recent version is available at `https://code.google.com/p/nohr-reasoner/`. The example file sets for testing can also be found on this web page.

Acknowledgments. We would like to thank Miguel Calejo for his help with InterProlog, Pavel Klinov for his help with ELK, Terry Swift for his help with XSB, and Gonca Güllü for her collaboration. Vadim Ivanov was partially supported by a MULTIC – Erasmus Mundus Action 2 grant. Matthias Knorr and João Leite were partially supported by FCT funded project ERRO – Efficient Reasoning with Rules and Ontologies (PTDC/EIA-CCO/121823/2010) and Matthias Knorr also by FCT grant SFRH/BPD/86970/2012.

References

1. Alferes, J.J., Knorr, M., Swift, T.: Query-driven procedures for hybrid MKNF knowledge bases. ACM TOCL 14(2) (2013)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: IJCAI'05: 19th Int. Joint Conf. on Artificial Intelligence. pp. 364–369. Morgan Kaufmann (2005)
3. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 3rd edn. (2010)
4. Calejo, M.: Interprolog: Towards a declarative embedding of logic programming in java. In: Alferes, J.J., Leite, J.A. (eds.) JELIA. Lecture Notes in Computer Science, vol. 3229, pp. 714–717. Springer (2004)
5. Ivanov, V., Knorr, M., Leite, J.: A query tool for $\mathcal{EL}$ with non-monotonic rules. In: ISWC 2013. Springer (2013), to appear
6. Kazakov, Y., Krötzsch, M., Simančík, F.: Concurrent classification of $\mathcal{EL}$ ontologies. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) Proceedings of the 10th International Semantic Web Conference (ISWC'11). LNCS, vol. 7032. Springer (2011)
7. Knorr, M., Alferes, J.J., Hitzler, P.: Local closed world reasoning with description logics under the well-founded semantics. Artif. Intell. 175(9–10), 1528–1554 (2011)
8. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5) (2010)

A Search Interface for Researchers to Explore Affinities in a Linked Data Knowledge Base

Laurens De Vocht[1], Erik Mannens[1], Rik Van de Walle[1],
Selver Softic[2], and Martin Ebner[3]

[1] Ghent University - iMinds, Multimedialab
Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium
`{laurens.devocht,erik.mannens,rik.vandewalle}@ugent.be`
[2] Virtual Vehicle Research Center - Area Information and Process Management
Inffeldgasse 21a, 8010 Graz, Austria
`selver.softic@tugraz.at`
[3] Graz University of Technology, IICM - Institute for Information Systems and Computer Media
Inffeldgasse 16c, 8010 Graz, Austria
`martin.ebner@tugraz.at`

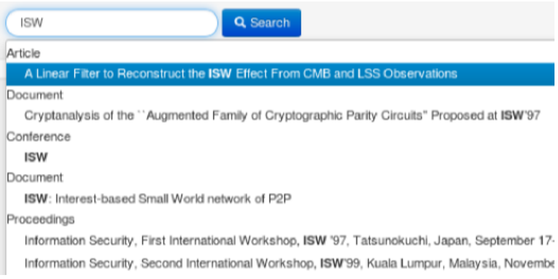
Abstract. Research information is widely available on the Web. Both as peer-reviewed research publications or as resources shared via (micro)blogging platforms or other Social Media. Usually the platforms supporting this information exchange have an API that allows access to the structured content. This opens a new way to search and explore research information. In this paper, we present an approach that visualizes interactively an aligned knowledge base of these resources. We show that visualizing resources, such as conferences, publications and proceedings, expose affinities between researchers and those resources. We characterize each affinity, between researchers and resources, by the amount of shared interests and other commonalities.

1 Introduction

Research 2.0 as adaptation of the Web 2.0 for researchers defines researchers as main consumers of information. Typically researchers define queries with a set of keywords when searching for information related to their work, for example using Google or digital archives such as PubMed. Linked Data technologies offer an entity based infrastructure to resolve the meanings of the keywords and the relations between them. Combining keyword resolution and resource expansion with Linked Data entities and filtering the results with personal preferences enhances the search precision. Currently many researchers have a Social Media account, such as on Twitter or Mendeley. We use these accounts to personalize the search. Our interface supports searching for scientific events, authors or groups of authors, as well as finding publications, and proceedings. The interface uses a search engine which relies on Linked Data knowledge base containing research related and personal information.

2 Real-time Keyword Disambiguation

We chose a real-time keyword disambiguation to guide the researchers in expressing their research needs. We do this by allowing users to select the correct meaning from a drop down menu that appears below the search box. Presenting candidate query expansion terms in real-time, as users typed their queries, can be useful during the early stages of the search [1]. In this is case it is very important that the users understand meaning of the suggested terms. Therefore we use an as straightforward as possible representation of the keyword mappings as shown in Figure 1.



Article
A Linear Filter to Reconstruct the **ISW** Effect From CMB and LSS Observations
Document
Cryptanalysis of the ``Augmented Family of Cryptographic Parity Circuits" Proposed at **ISW**'97
Conference
ISW
Document
ISW: Interest-based Small World network of P2P
Proceedings
Information Security, First International Workshop, **ISW** '97, Tatsunokuchi, Japan, September 17-
Information Security, Second International Workshop, **ISW**'99, Kuala Lumpur, Malaysia, Novembe

Fig. 1. Mapping of keywords to Linked Data entities.

3 Exploring Resources

Researchers can improve the definition of their "intended" search goal over several iterations. Each time a combination of various resources is visualized. The visualization suggests new queries: they are generally most useful for refining the system's representation of the researcher's need. In case they have no idea which entity to focus on or what topic to investigate next they get an overview of possible entities of interest, like points of interest on a street map. By profiling their activities and contributions on Social Media and other platforms such as their own research publications, the affinity with the proposed resources is enhanced. An iteration can consist of either one of two actions:

1. **Query Expansion:** The user expands the query space by clicking the results retrieved by initial keyword based search.The resolution of results happens based upon the properties of Linked Data like *rdf:label*, *owl:sameAs*, *rdf:seeAlso*, *dc:title* or *dc:description*.
2. **Additional Query Formulation:** Additional query expansion happens either through adding further keywords as well as through keyword combinations already entered where the back-end tries to deliver additional results based upon connection paths between the resources.

4 Visualizing Relations between Resources

We find relations between resources after matching the input given by the researcher in the knowledge base. With the delivery of first results, our engine expands the query and enhances the context. For this purpose we used a model and an implementation that builds upon on our earlier work on the “Everything is Connected” engine (EiCE) [2] and semantic profiling [3].
In the visualization we emphasize the affinities by showing, on a radial map [4], how the current focused entity is related to the other found entities. It is based on the concept of affinity that can be appropriately expressed in visual terms as a spatial relationship: proximity [5]. We additionally express the amount of unexpectedness as *novelty* of a resource in each particular search context. To further enhance the guidance of users during search we have used two other visual aids:

1. **Color:** Every entity has a type and associated unique color. For a certain result set the user gets an immediate impression of the nature of the found resources.
2. **Size:** We rank each entity according to novelty and relation to the context and enlarge those that should attract attention from the researcher first. A goal of the search is to explore information not seen before which makes it difficult to define an accurate search goal. Besides allowing to search specific entities, our visualization facilitates exploratory browsing. This is particularly useful when information seeking with unclear defined search targets [6].

Figure 2 shows how researchers can track the history of their search: the explored relations are marked red and clearly highlight the context of a search. Researchers can click on a list of resources they have searched to focus the visualization. A screencast of the search interface is available online[4]. In this screencast, we show how researchers interact with the search interface and the above described visualization.

5 Conclusions

We have developed an interface for personalized search in a social driven knowledge base for researchers. Combining the latest Linked Data technologies with an advanced indexing and path finding system, EiCE and Web 2.0 technologies (such as JQuery and Django). The result is a semantic search application providing both a technical demonstration and a visualization that could be applied in many other disciplines beyond Research 2.0. The main contribution of our work is, besides retrieving resources from Linked Data repositories, allowing users to interactively explore relations between the resources and find out the affinity with each resource.

---

[4] `http://semweb.mmlab.be/search_interface_for_researchers`

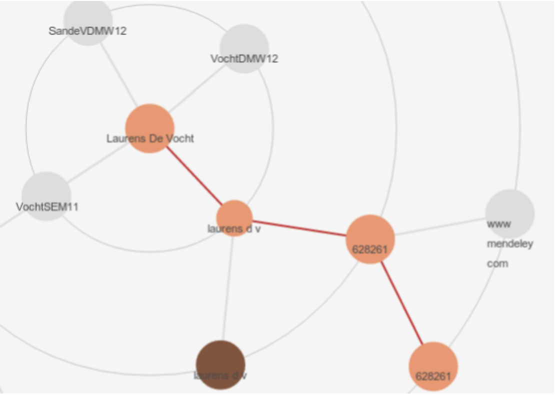


Fig. 2. A red line marks the explored relations in the visualized search context.

6 Acknowledgement

The research activities that have been described in this paper were funded by Ghent University, iMinds (Interdisciplinary institute for Technology) a research institute founded by the Flemish Government, Graz University of Technology, the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research-Flanders (FWO-Flanders), and the European Union. The authors would like to acknowledge the financial support of the "COMET K2 - Competence Centres for Excellent Technologies Programme" of the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Austrian Federal Ministry of Economy, Family and Youth (BMWFJ), the Austrian Research Promotion Agency (FFG), the Province of Styria and the Styrian Business Promotion Agency (SFG).

References

1. White, R.W., Marchionini, G.: Examining the effectiveness of real-time query expansion. Information Processing & Management **43**(3) (2007) 685–704
2. De Vocht, L., Coppens, S., Verborgh, R., Vander Sande, M., Mannens, E., Van de Walle, R.: Discovering meaningful connections between resources in the web of data. In: Proceedings of the 6th Workshop on Linked Data on the Web (LDOW). (2013)
3. De Vocht, L., Van Deursen, D., Mannens, E., Van de Walle, R.: A semantic approach to cross-disciplinary research collaboration. International Journal of Emerging Technologies in Learning (iJET) **7**(S2) (2012) 22–30
4. Yee, K.P., Fisher, D., Dhamija, R., Hearst, M.: Animated exploration of dynamic graphs with radial layout. In: Proceedings of the IEEE Symposium on Information Visualization (INFOVIS). (2001)
5. Pintado, X.: The affinity browser. In: Object-oriented software composition. Prentice Hall (1995) 245–272
6. Pace, S.: A grounded theory of the flow experiences of web users. International journal of human-computer studies **60**(3) (2004) 327–363

Cite4Me: A Semantic Search and Retrieval Web Application for Scientific Publications

Bernardo Pereira Nunes[1,2], Besnik Fetahu[1], Stefan Dietze[1], and Marco A. Casanova[2]

[1]L3S Research Center, Leibniz University Hannover, Appelstr. 9a, 30167 Hannover, Germany
{nunes, fetahu, dietze}@L3S.de
[2]Department of Informatics, Pontifical Catholic University of Rio de Janeiro,
Rio de Janeiro/RJ – Brazil, CEP 22451-900
{bnunes, casanova}@inf.puc-rio.br

Abstract. Cite4Me is a Web application that leverages Semantic Web technologies to provide a new perspective on search and retrieval of bibliographical data. The Web application presented in this work focuses on: (i) semantic recommendation of papers; (ii) novel semantic search & retrieval of papers; (iii) data interlinking of bibliographical data with related data sources from LOD; (iv) innovative user interface design; and (v) sentiment analysis of extracted paper citations. Finally, as this work also targets some educational aspects, our application provides an in-depth analysis of the data that guides a user on his research field.

1 Introduction

The huge amount of Web data and resources, particularly in the academic area, calls for strategies to analyse and explore resources and data.

While scientific disciplines are very data- and knowledge-intensive, the lack of semantic tools hampers information management and decision making. This includes scientific data as well as unstructured academic publications as one of the key outcome of scientific work. This is due to information access offered by digital library providers such as ACM Digital Library[1] and Elsevier[2] being mostly based on free text search and hierarchical classification[3].

Thus, we present a novel Web application for exploratory search, retrieval and visualization of scientific publications. *Cite4Me* aims at providing a single access point for accessing papers and, therefore, assisting searchers on finding relevant topics, papers, and unveiling new nomenclature more efficiently. For this, we use reference datasets such as DBpedia[4] to explore semantic relationships between scientific papers and user queries. We also perform a topic coverage analysis to provide an overview of different bibliographic datasets. *Cite4Me* is a Web application which exploits results of previous research works [2–5].

---

[1] `http://dl.acm.org`

[2] `http://www.elsevier.com`

[3] `http://www.acm.org/about/class/`

[4] `http://dbpedia.org`

2 Cite4Me - The Application

Cite4Me implements semantic and co-occurrence-based methods to search and retrieve academic papers and suggest related work in a user-friendly interface that assists users in exploring relationships between authors, institutions, papers and query terms. Due to space restrictions, we present in this paper the most relevant features of *Cite4Me* to the Semantic Web field.

2.1 Search and Retrieval

Cite4Me implements standard techniques, such as free text search, to search and retrieve scientific publications. In this section, we emphasize the semantic and exploratory search mechanisms.

Exploratory Search. The *exploratory search* or *graph search* component assists users to discover related work, people and institutions that are working on a specific topic. A crucial step to provide this type of search is the annotation of the publications' content. For this, we used DBpedia Spotlight API[5] for extracting entities, entity types and their categories. For instance, the categories of the extracted concepts are used to interlink publications through the topics they cover. In cases where two publications share the same category (*dcterms:subject* property), then a link between both publications is created. Figure 1 shows an example of topically related publications.

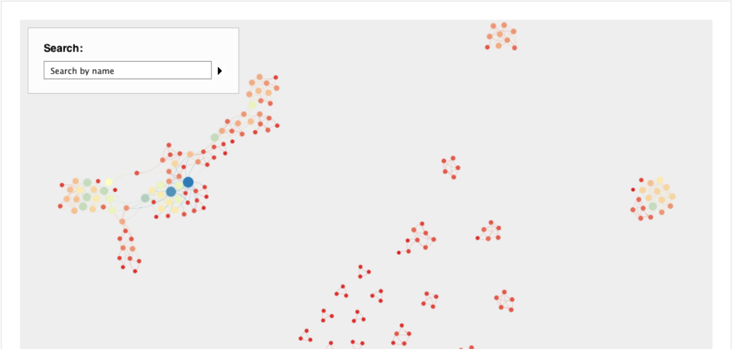


Fig. 1. Preview of the exploratory search funcionality.

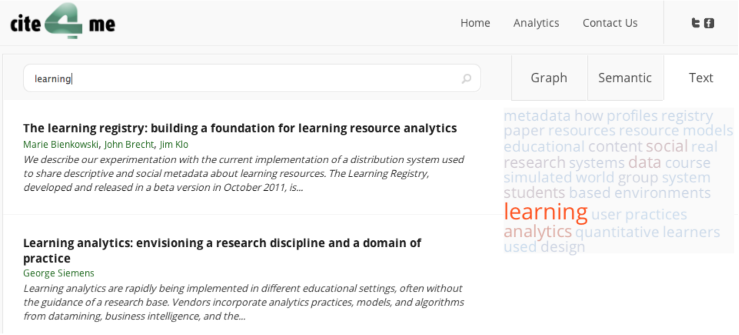
Semantic Search. The *semantic search* component of *Cite4Me* is similar to the *explicit semantic analysis* (ESA) technique [1]. After running the annotation process aforementioned, the relatedness score between the enriched concepts (DBpedia entities) found

---

[5] `http://dbpedia.org/spotlight`

in the user query terms and the publications' content are computed and ranked. The relatedness score is computed based on the *tf-idf* score for the entities found in the publications' content. The ranking of the retrieved documents is based on the sum of the *tf-idf* scores of the matching concepts.

Figure 2 illustrates the semantic search functionality. Alongside the results of the semantic search a tag cloud shows the most prominent terms for a given user query. The tag cloud is updated while browsing through the list of results. The tags are selected based on the *tf-idf* score for the entities found in the abstract of the retrieved papers.


cite 4 me
Home
Analytics
Contact Us
Graph
Semantic
Text
metadata how profiles registry
paper resources resource models
educational content social real
research systems data course
simulated world group system
students based environments
learning user practices
analytics quantitative learners
used design

learning

The learning registry: building a foundation for learning resource analytics

Marie Bienkowski, John Brecht, Jim Klo

We describe our experimentation with the current implementation of a distribution system used to share descriptive and social metadata about learning resources. The Learning Registry, developed and released in a beta version in October 2011, is...

Learning analytics: envisioning a research discipline and a domain of practice

George Siemens

Learning analytics are rapidly being implemented in different educational settings, often without the guidance of a research base. Vendors incorporate analytics practices, models, and algorithms from datamining, business intelligence, and the...

Fig. 2. Preview of the semantic search funcionality.

Paper recommendation. Another important feature of *Cite4Me* and which differentiates it from similar tools is the *semantic paper recommendation*. Given a scientific publication, the tool recommends a related paper based on a score calculated according to direct and lateral relationships between the publication of interest and the remaining papers in our corpus.

To compute the relatedness score, we rely on previous work by Nunes et al. [2, 4], where the paths connecting two enriched concepts in the scientific publications are analysed using a variation of the Katz index, a measure based on Social Network Theory, and quantifying the weight of the connectivity between two concepts given a knowledge graph (in our case DBpedia graph).

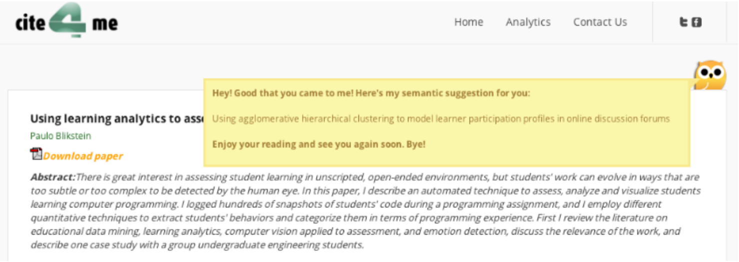
After computing the relatedness scored between enriched concepts, the paper recommendation relies on an aggregated measure that takes into account the relatedness inter-documents. Finally, we generate a ranked list of pairwise publications according to the overall score (see [5] for more details). Thus, the top-ranked publication is recommended to the user, as shown in Figure 3.

3 Datasets

Currently, *Cite4Me* is linked to a dataset (*LAK Dataset*[6]) which contains semi-structured research publications from the ACM Digital Library (under a special license)

---

[6] `http://www.solaresearch.org/resources/lak-dataset/`



Hey! Good that you came to me! Here's my semantic suggestion for you:

Using agglomerative hierarchical clustering to model learner participation profiles in online discussion forums

Enjoy your reading and see you again soon. Bye!

Using learning analytics to ass

Paulo Blikstein

Abstract: *There is great interest in assessing student learning in unscripted, open-ended environments, but students' work can evolve in ways that are too subtle or too complex to be detected by the human eye. In this paper, I describe an automated technique to assess, analyze and visualize students learning computer programming. I logged hundreds of snapshots of students' code during a programming assignment, and I employ different quantitative techniques to extract students' behaviors and categorize them in terms of programming experience. First I review the literature on educational data mining, learning analytics, computer vision applied to assessment, and emotion detection, discuss the relevance of the work, and describe one case study with a group undergraduate engineering students.*

Fig. 3. An example of paper recommendation based on SCS_w.

and other public datasets (see also [6] for details). The dataset contains 315 full papers along with their descriptive metadata while new publications are added continuously. Metadata as well as the full text body are freely available in a variety of formats, including RDF accessible via a public SPARQL endpoint. We are currently working on expanding the number of papers available in *Cite4Me*. However, due to copyright reasons, the process to expose scientific publications from publishers is still under discussion.

4 Conclusion

This paper presented the application of previous works in the Semantic Web field within *Cite4Me*, a Web application that assists users in finding relevant scientific papers by exploring semantic relationships between them. For more information about the *Cite4Me* Web application please refer to `http://www.cite4me.com`.

References

1. E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of IJCAI'07*, pages 1606–1611, San Francisco, CA, USA, 2007.
2. B. Pereira Nunes, S. Dietze, M. A. Casanova, R. Kawase, B. Fetahu, and W. Nejdl. Combining a co-occurrence-based and a semantic measure for entity linking. In *ESWC*, 2013 (to appear).
3. B. Pereira Nunes, B. Fetahu, and M. A. Casanova. Cite4me: Semantic retrieval and analysis of scientific publications. In M. d'Aquin, S. Dietze, H. Drachsler, E. Herder, and D. Taibi, editors, *LAK (Data Challenge)*, volume 974 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.
4. B. Pereira Nunes, R. Kawase, S. Dietze, D. Taibi, M. A. Casanova, and W. Nejdl. Can entities be friends? In *Proceedings of WOLE, in conjuction with the ISWC'12*, volume 906 of *CEUR-WS.org*, pages 45–57, Nov. 2012.
5. B. Pereira Nunes, R. Kawase, B. Fetahu, S. Dietze, M. A. Casanova, and D. Maynard. Interlinking documents based on semantic graphs. In *Proceedings of KES'13*, 2013 (to appear).
6. D. Taibi and S. Dietze. Fostering analytics on learning analytics research: the lak dataset. In *Proceedings of the LAK Data Challenge, held at LAK2013*, April 2013.

Best-effort Linked Data Query Processing with time constraints using ADERIS-Hybrid

Steven Lynden, Isao Kojima, Akiyoshi Matono, and Akihito Nakamura

Information Technology Research Institute
National Institute of Advanced Industrial Science and Technology (AIST), Japan
{steven.lynden|isao.kojima|a.matono|nakamura-akihito}@aist.go.jp

Abstract. Answering SPARQL queries over the Web of Linked Data is a challenging problem. Approaches based on distributed query processing provide up-to-date results but can suffer from delayed response times, indexing-based approaches provide fast response times but results can be out-of-date and the costs of indexing the growing Web of Linked Data are potentially huge. Hybrid approaches try to offer the best of both. In this demo paper we describe a system for answering SPARQL queries within fixed time constraints by accessing SPARQL endpoints and the Web of Linked Data directly.

1 Introduction

Answering Linked Data queries in a timely manner is a challenging problem. An example of one approach towards this is Sindice [1], which provides a SPARQL query interface over RDF data that has been indexed via crawling the Web of Linked Data. Other examples include approaches based on distributed query processing over SPARQL endpoints [5], in addition to link traversal, live exploration and hybrid approaches as surveyed in [3], which access the Web of Linked Data directly during query execution. However, such approaches may result in unpredictable query execution times in the order of minutes for even basic queries, and while there are obviously applications that would utilise such results, it is sometimes more important that an approximate or incomplete answer is provided within a shorter time frame. In this paper we introduce a system based on a hybrid approach where SPARQL endpoints such as Sindice and the Web of Linked Data are accessed in parallel to answer queries within fixed time constraints. The system can be found at `http://aderis.linkedopendata.net`.

2 Hybrid Linked Data Query Processing with Time Constraints

The approach, illustrated in Figure 1, proceeds as follows:

1. A federated SPARQL query is parsed and compiled into a set of triple patterns. The query is entirely declarative written without knowledge of the

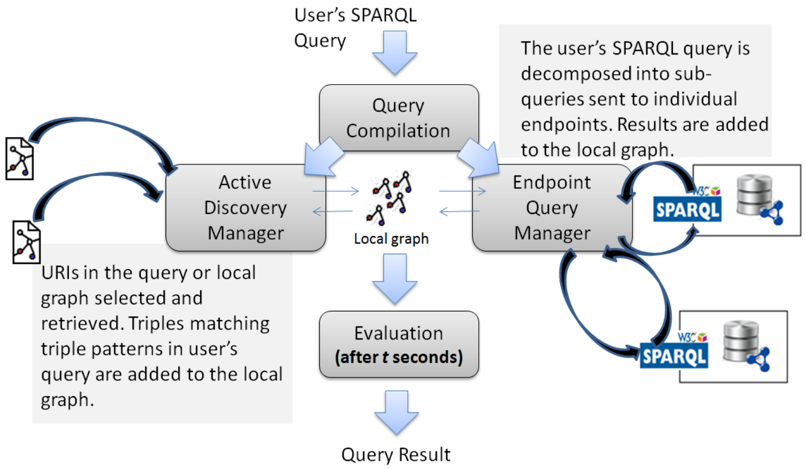

User's SPARQL Query
Query Compilation
The user's SPARQL query is decomposed into sub-queries sent to individual endpoints. Results are added to the local graph.
Active Discovery Manager
Local graph
Endpoint Query Manager
SPARQL
SPARQL
URIs in the query or local graph selected and retrieved. Triples matching triple patterns in user's query are added to the local graph.
Evaluation
(after *t* seconds)
Query Result

Fig. 1. System details

The active discovery manager and endpoint query manager run in parallel for a fixed time, are then terminated and the local graph is converted into a query result.

location of data, in contrast to, for example the SPARQL 1.1 Federation [2] extensions.

2. A *local graph* component is initialised to store intermediate results, i.e. triples which have been found to match the set of triple patterns in the query.
3. Two components are executed, the *endpoint query manager* and the *active discovery manager*. The endpoint query manager sends queries to SPARQL endpoints and the active discovery manager dereferences URIs and matches the RDF triples retrieved with triple patterns in the query.
4. After a time t, for which the query is scheduled to run, the endpoint query manager and active discovery manager are terminated and the local graph component is used to obtain the result of the federated query.

For a detailed description of the optimisation strategies implemented by the active discovery manager and endpoint query manager, please refer to [4]. For the purposes of an effective demonstration we have chosen t to be 10 seconds for the optimisation and configuration of the system, however the value can be changed. The rationale being that this is a response time within which at least some useful answers can usually be obtained and for which users are generally willing to wait. Compared with the work as presented in [4], we have extended the system with a cache and more extensive statistics from the Web of Linked Data aimed at prioritising the retrieval of fresh, up-to-date data by the active discovery manager. As queries are answered on a best-effort basis, it is important to give the user an idea of how complete the results are estimated to be. An estimate of the completeness of the results (low, medium, or high) is given to the user based on the number of relevant URIs that could not be dereferenced by the active discovery manager in the time allowed, combined with other indicators such as overlap between the triples retrieved from SPARQL endpoints and URIs (i.e. a high degree of overlap indicates that URIs not yet dereferenced would be likely to provide triples already retrieved from endpoints and be therefore unlikely to provide additional query results).

The proposed approach provides increased coverage and fault tolerance due to the fact that multiple data sources are used and the effects of individual data source unavailability can be mitigated. The system provides parallel query execution by pushing down query fragments to individual SPARQL endpoints and automatically optimises the queries sent to individual endpoints to comply with fair-use restrictions such as bounds on query execution time.

3 ADERIS-Hybrid Web Application

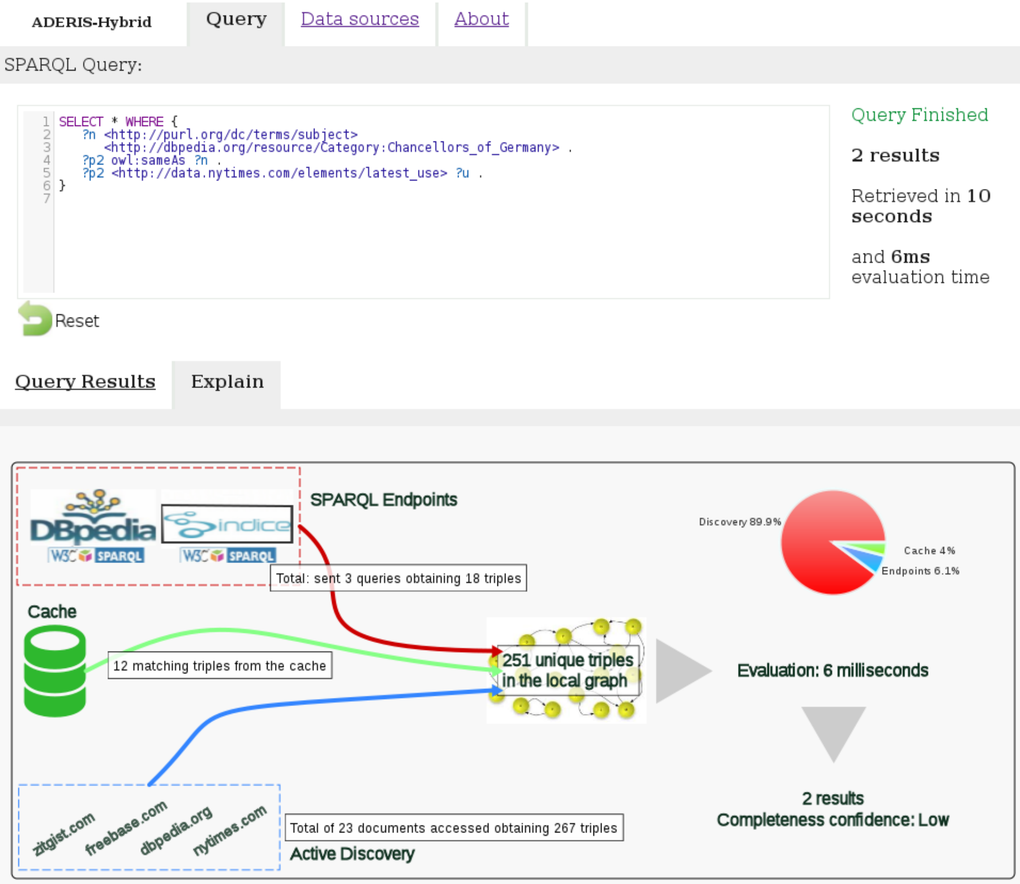
The proposed demo presents the ADERIS-Hybrid Web application implementing the previously described approach, built on our previous work on the Adaptive Distributed Endpoint Integration System (ADERIS) [5], to provide a Web application implementing the hybrid approach described in this paper. The proposed demonstration will highlight the salient aspects of the system including the construction of SPARQL queries, where a set of example queries are provided which can be easily edited by the user; results of queries are presented to the user using the Google Visualization API complemented by a visual representation of the query execution process, as shown in Figure 2. A summary of the statistics used by the query processor is also presented.

4 Conclusion

Answering SPARQL queries over the Web of Linked data with reasonable response times is an important, challenging problem. The proposed demo is a Web application based on our approach in [4], extended with a cache, additional data source statistics, and an estimate of the completeness of the result.

References

1. Sindice: The Semantic Web Index. http://sindice.com.
2. SPARQL 1.1 Federation Extensions. http://www.w3.org/2009/sparql/docs/fed/gen.html.
3. O. Hartig. An Overview on Execution Strategies for Linked Data Queries. *Datenbank-Spektrum*, 13(2):89–99, 2013.
4. S. Lynden, I. Kojima, A. Matono, A. Nakamura, and M. Yui. A Hybrid Approach to Linked Data Query Processing with Time Constraints. In C. Bizer, T. Heath, T. Berners-Lee, and M. Hausenblas, editors, *WWW2013 Workshop on Linked Data on the Web - LDOW 2013*.
5. S. Lynden, I. Kojima, A. Matono, and Y. Tanimura. ADERIS: Adaptively integrating RDF data from SPARQL endpoints (Demo Paper). In *Proceedings of the Database Systems for Advanced Applications (DASFAA) Conference 2010*, 2010.



```
SELECT * WHERE {
   ?n <http://purl.org/dc/terms/subject>
      <http://dbpedia.org/resource/Category:Chancellors_of_Germany> .
   ?p2 owl:sameAs ?n .
   ?p2 <http://data.nytimes.com/elements/latest_use> ?u .
}
```

Query Finished

2 results

Retrieved in **10 seconds**

and **6ms** evaluation time

Reset
Query Results
Explain
DBpedia
W3C SPARQL
indice
W3C SPARQL
SPARQL Endpoints
Discovery 89.9%
Cache 4%
Endpoints 6.1%
Total: sent 3 queries obtaining 18 triples
Cache
12 matching triples from the cache
251 unique triples in the local graph
Evaluation: 6 milliseconds
2 results
Completeness confidence: Low
zitgist.com
freebase.com
dbpedia.org
nytimes.com
Total of 23 documents accessed obtaining 267 triples
Active Discovery

Fig. 2. Web application

The figure is a screenshot of the Web application. Here, the user has executed a SPARQL query, which can be seen in the upper left-hand portion of the screen, and obtained 2 results (not shown in this screenshot due to space restriction). The user is utilising the system's "explain" feature to view how the results were obtained. The percentage of RDF triples that make up the local graph from SPARQL endpoints, the cache and the Web of Linked Data are shown, in addition to a confidence measure that the results are complete.

Assisted Policy Management for SPARQL Endpoints Access Control

Luca Costabello, Serena Villata*, Iacopo Vagliano, and Fabien Gandon

INRIA Sophia Antipolis, France
{`firstname.lastname`}`@inria.fr`

Abstract. Shi3ld is a context-aware authorization framework for protecting SPARQL endpoints. It assumes the definition of access policies using RDF and SPARQL, and the specification of named graphs to identify the protected resources. These assumptions lead to the incapability for users who are not familiar with such languages and technologies to use the authorization framework. In this paper, we present a graphical user interface to support dataset administrators to define access policies and the target elements protected by such policies.

1 Introduction

Shi3ld[1] [2] is an access control framework for querying Web of Data servers. It protects RDF stores from incoming SPARQL queries, whose scope is restricted to triples included in accessible named graphs only [1]. In particular, Shi3ld determines the list of accessible graphs by evaluating pre-defined access policies against client attributes sent with the query. It adopts exclusively Semantic Web languages, reuses existing proposals, and protects data up to triple level. The drawback of such framework is that it relies on the assumption that dataset administrators have a proficient knowledge of RDF and SPARQL, and that they are able to manage vocabularies and define new named graphs. In this paper, we address this open issue by presenting a web application that allows non-expert dataset administrators to manage Shi3ld context-aware access control policies, by hiding the complexity of RDF and SPARQL. The Shi3ld policy manager allows the definition of context-aware access conditions featuring user, environment (time and location above all), and device attributes. Moreover, such application allows a simpler definition of new named graphs over a set of existing triples. The work presented in this paper can be classified among the works trying to hide the complexity of SPARQL and the Semantic Web to end users [4–6]. Such proposals mainly consist in GUIs to query, search, visualize, browse and edit triples published on the Web of Data. In our work, we deal with querying issues and we tackle the problem of providing a user-friendly interface for the creation of context-aware access control policies for triple stores.

---

* The author acknowledges support of the DataLift Project ANR-10-CORD-09 founded by the French National Research Agency.

[1] `http://wimmics.inria.fr/projects/shi3ld/`

2 Our Proposal

The Shi3ld policy management GUI[2] is designed to support the interaction with two kinds of dataset administrators: *non-experts*, which are assumed not to know the SPARQL query language and RDF, and *experts*, which are able to edit access policies source code. In particular, the following functionalities are proposed:

- **Policies visualization and modification**: the application shows the list of policies stored in the triple store through a grid view. Each policy is an expandable row that, if selected, shows the main features of the policy like the policy target (i.e. the named graphs protected by the policy), the privilege granted by the policy (Create, Update, Read, Delete), and the access conditions (SPARQL 1.1 `ASK` queries) which specify the requirements that need to be satisfied to access the target resource. Users can edit all these elements, e.g., they associate the policy to another named graph, add or remove privileges, or modify the defined access conditions. Two different views are proposed to the user: i) a graphical view where operations are performed without the need to write policies using SPARQL and RDF to support *non-expert* administrators, and ii) a textual editor which allows to directly write policies using SPARQL and RDF for *expert* administrators.
- **Policies creation**: the creation of a new context-aware policy is managed by a wizard. In particular, the wizard proposes the following views: *i)* the definition of the policy name (which is then "translated" into an `rdfs:label`), the target named graph (it is possible to select one of the already defined named graphs included in the triple store, or to define a new one as we will detail later), and the privilege(s) to associate to the policy; *ii)* the view concerning the User dimension, that consists in a text box where the administrator inserts the features that must be satisfied by the user accessing the target resource, e.g., `foaf:knows :ACME_boss`. The text box provides autocompletion and it suggests a list of properties showing the associated vocabulary (to date, we use the `foaf`[3] and `relationship`[4] vocabularies, but other vocabularies can be added); *iii)* the view concerning the Environment dimension, that consists in two parts: the first one defines temporal conditions, and the second one deals with geographical conditions. Temporal conditions are expressed with a time picker, to select the desired time interval in which the access is granted. The definition of the geographical condition is done with a map interface[5], enriched with a movable marker and a resizable radius; *iv)* the view concerning the Device dimension, similar to the User view, that suggests the access properties related to the device used to access the target

---

[2] Video available at `http://wimmics.inria.fr/projects/shi3ld/`

[3] `http://xmlns.com/foaf/spec/`

[4] `http://purl.org/vocab/relationship/`

[5] `http://developers.google.com/maps/`

resource (we use the Delivery Context vocabulary[6] but further vocabularies can be added). At the end of the wizard, the access policy is automatically generated and stored in the triple store.

- **Named graphs creation**: the administrator is assisted in the definition of a new named graph. Shi3ld access policies must be associated to named graphs, and this leads to a number of difficult tasks for non-expert users, since it involves the use of non-trivial SPARQL features. We thus provide a GUI to mask such complexity, by letting administrators define a new named graph starting from the set of triples they want to associate to such newly defined named graph. The application asks for the label of the named graph to be created and it presents the template of a `SELECT` query, to be completed with the desired triple pattern. A preview of the selected triples is shown, thus letting the administrator check which triples will be added to the named graph. If results are satisfying, the new named graph is created and it can be used as the access policy target.

Figure 1 shows how user actions are translated into SPARQL and RDF by the Shi3ld Policy Manager. The application supports the administrator in creating, editing and deleting both policies and target named graphs. SPARQL queries are completely masked to end users, unless the embedded SPARQL textual editor is opened.

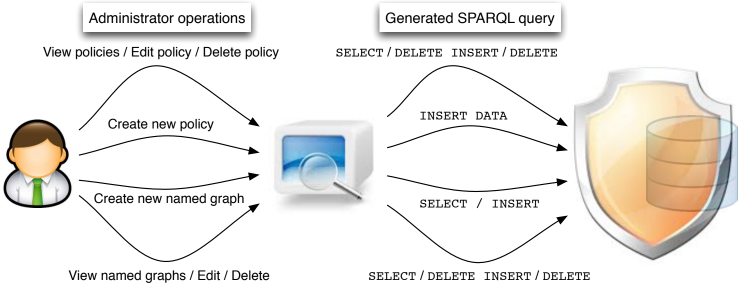

Administrator operations
Generated SPARQL query
View policies / Edit policy / Delete policy
SELECT / DELETE INSERT / DELETE
Create new policy
INSERT DATA
Create new named graph
SELECT / INSERT
View named graphs / Edit / Delete
SELECT / DELETE INSERT / DELETE

Fig. 1: The administrator operations and the resulting `SPARQL` query.

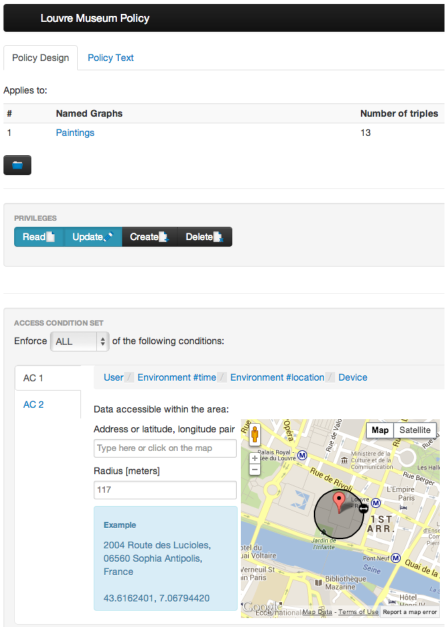
The Shi3ld Policy Manager is a web application developed in JavaScript and backed by a Fuseki SPARQL 1.1 triple store[7]. The server-side relies on the Node.js platform[8], and the front-end is built over jQuery, the Twitter Bootstrap framework[9], and Backbone.js[10] as structure. The SPARQL editor is provided by Flint[11].

---

[6] `http://www.w3.org/TR/dcontology/`
[7] `http://jena.apache.org/documentation/serving_data/`
[8] `http://nodejs.org/`
[9] `http://twitter.github.io/bootstrap/`
[10] `http://backbonejs.org/`
[11] `http://openuplabs.tso.co.uk/demos/sparqleditor`

3 Future Perspectives

We have presented a user interface to declare context-aware policies for the Shi3ld authorization framework. There are several issues to be considered as future research. First, since Shi3ld has been recently extended to manage also HTTP access to resources [3], we will extend this application such that also policies for Shi3ld-HTTP would be defined and manageable, i.e., access conditions are defined as RDF triples instead of `ASK` SPARQL queries. Second, we will integrate our interface with the Linked Open Vocabulary catalogue[12] such that administrators are supported in including new vocabularies used to define the access conditions. Third, we plan to favour policy reuse across datasets by adding a "policy template" sharing functionality. Moreover, we envision a "deep" properties validation, (i.e. checking that a certain URI actually corresponds to a `foaf` profile). Finally, we will add a sandbox to test the access policies effectiveness on the protected triples.



Applies to:

#	Named Graphs	Number of triples
1	Paintings	13

Enforce ALL of the following conditions:

Data accessible within the area:

Address or latitude, longitude pair: Type here or click on the map

Radius [meters]: 117

Example

2004 Route des Lucioles,
06560 Sophia Antipolis,
France

43.6162401, 7.06794420

Fig. 2: The Shi3ld user interface.

References

1. Carroll, J.J., Bizer, C., Hayes, P.J., Stickler, P.: Named graphs. J. Web Sem. 3(4), 247–267 (2005)
2. Costabello, L., Villata, S., Gandon, F.: Context-Aware Access Control for RDF Graph Stores. In: Procs of ECAI. Frontiers in Artificial Intelligence and Applications, vol. 242, pp. 282–287. IOS Press (2012)
3. Costabello, L., Villata, S., Rocha, O.R., Gandon, F.: Access Control for HTTP Operations on Linked Data. In: Procs of ESWC. Lecture Notes in Computer Science, vol. 7882, pp. 185–199. Springer (2013)
4. Lopez, V., Uren, V.S., Sabou, M., Motta, E.: Is Question Answering fit for the Semantic Web?: A survey. Semantic Web 2(2), 125–155 (2011)
5. Ngomo, A.C.N., Bühmann, L., Unger, C., Lehmann, J., Gerber, D.: Sorry, i don't speak SPARQL: translating SPARQL queries into natural language. In: Procs of WWW. pp. 977–988. ACM (2013)
6. Sonntag, D., Heim, P.: A Constraint-Based Graph Visualization Architecture for Mobile Semantic Web Interfaces. In: Procs of SAMT. Lecture Notes in Computer Science, vol. 4816, pp. 158–171. Springer (2007)

---

[12] `http://lov.okfn.org/dataset/lov/`

OU Social: Reaching Students in Social Media

Miriam Fernandez, Harith Alani, Stuart Brown

Knowledge Media Institute, UK
`m.fernandez, h.alani, stuart.brown`@open.ac.uk

Abstract. This work describes OU Social, an application that collects and analyses data from public Facebook groups set up by students to discuss particular Open University courses. This application exploits semantic technologies to monitor the behaviour of users over time as well as the topics that emerge from Facebook group discussions. The paper describes the architecture of OU Social and provides a brief overview of the analysis results obtained from 44 different Facebook groups examined over a 6 year period (2007-2013)

Keywords: semantics, social media, education

1 Introduction

The Open University (OU) is the largest university in the United Kingdom and the leading distance teaching institution in the world. The OU courses are directly available throughout Europe and, by means of partnership agreements with other institutions, in many other parts of the world.

Given its on-line learning profile, one of the key goals of the OU is the constant research and development of online learning and teaching solutions based on the analysis of usage data and on the feedback provided by the users about their experiences. To acquire this feedback the OU has created several websites and applications were students can discuss the different courses and share their learning experiences. This information, as well as OU's website usage data, is currently being collected and processed as part of the OU's usage data analytics process. However, there are other rich and rapidly growing sources of user feedback that are external to the OU, which could also be collected and investigated. With the emergence of social media, online learning is no longer restricted to particular in-house sites, but there is a clear tendency for students to share and discuss their learning material, methodologies and experiences on popular social networking sites, such as Facebook and Twitter. [1]

This paper presents OU Social, a prototypical tool for collecting and analysing content from a large set of relevant Facebook public groups. These groups have been specifically set up by Facebook users to bring together other students who

---

[1] http://www.eric.ed.gov/PDFS/ED535130.pdf
http://www.topuniversities.com/sites/qs.topuni/files/Students-Online-Useage-Global-Trends-Report-2013-nc.pdf

enrolled in particular OU courses or modules. The designed tool integrates two semantic analysis modules: (i) the behaviour analysis module, which categorises users into different behavioural roles (leaders, followers, etc.) by using a semantic-rule based methodology and, (ii) the topic analysis module, which extracts and monitors the concepts that emerge from Facebook groups discussions (using a semantic annotation system). Data extracted from these modules is enriched via the OU's liked data portal (data.open.ac.uk) to provide a better overview of the courses under analysis.

The rest of the paper is structured as follows: Section 2 presents the architecture of the system and provides a brief overview of the semantic analysis modules. Section 3 presents some preliminary results obtained by this tool after analysing 44 different Facebook groups over a 6 year period (2007-2013). Section 4 concludes the paper and outlines future work.

2 System Architecture

This section describes the architecture of OU Social and provides an overview of its components. A video of this demo is available under http://people.kmi.open.ac.uk/miriam/OUSocial/OUSocialVideo.mov. The demo is not publicly accessible to avoid disclosing private student information.

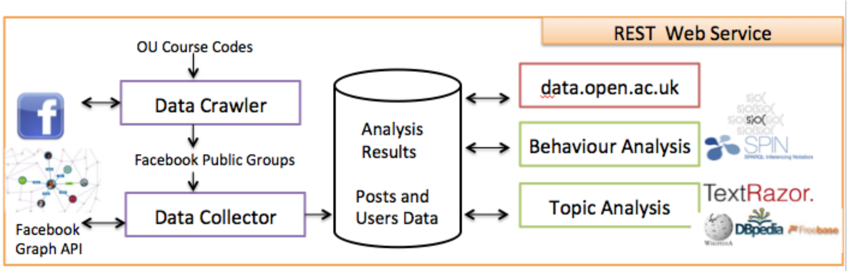


Fig. 1. OU Social Architecture

Data Collection: Departing from the list of OU course codes, a crawler has been developed that extracts the name of all those Facebook public groups and pages containing a course code (e.g., B120) and the words "Open University" or "OU". For each of these groups information about users and posts is extracted (by using the Facebook Graph API[2]) and stored in an internal database.

Behaviour Analysis: The Behaviour Analysis component shows the type of people discussing a particular topic or concept of interest with regard to their online behaviour. It allows the OU's course managers to focus on a smaller, more manageable, set of students (read their contributions, monitor their opinion, etc.). The analysis module not only identifies those students that are mostly active (e.g., leaders, contributors) but also those who are generally inactive and may need additional learning support (e.g., lurkers, followers). Specifically, this

---

[2] https://developers.facebook.com/docs/reference/api/

analysis distinguish among eight types of user roles: Lurker, Follower, Daily User Contributor, Broadcaster, Leader, Celebrity and Super User. This analysis module makes use of the OUBO (Open University Behaviour Analysis Ontology) and the SIOC (Semantically Interlinked Online Communities ontology) ontologies to model the behaviour of users in the different Facebook groups. To infer the different roles that a user adopts over time the module applies semantic rules encoded using SPIN (e.g, if popularity=high and contribution=high then role=leader). For more details of this model, the ontologies and the role extraction process, the reader is referred to the following publication [1].

Topic Analysis: Questions, answers, discussions, learning material, etc. are distributed and shared via social networking sites. Detecting what are the topics that emerge from these discussions can help to identify emerging issues with respect to certain elements of interest to the OU. To obtain the topics for each post the analysis component makes use of TextRazor [3], a natural language processing tool based on knowledge bases such as Wikipedia, DBPedia and Freebase. TextRazor identifies key entities and topics in a piece of text, returning a mapping between each posts and a list of URIs.

Data Enrichment To complement the results of the analyses with concrete information about the courses discussed in the Facebook groups, course information is extracted via SPARQL queries from the OU's linked data site (data.open.ac.uk). E.g., the OU maintains a taxonomy of course categories (science, chemistry, ...) that complements the results obtained by the topic analysis.

3 Analysis Results

An initial analysis have been conducted for 44 different Facebook groups over a period of six years, from 2007 till 2013, including a total of 136,704 posts and 19,094 users. The demo aims to show conference attendees how different analyses can be performed using this tool and how the use of semantic technologies can help course managers and university staff to productively exploit social media to obtain relevant feedback. Figures 1a and 1b display two examples of the OU Social analyses. Figure 1a displays a tag cloud visualising the relevant topics across all Facebook groups. Among these top topics we can find People, Works, Network Protocols, Behavioural Sciences and Educational Technology. This visualisation can also be obtained for each group individually. Additionally, the tool also allows the visualisation of the evolution of topics over time. Wikipedia links are provided for each of the displayed topics thanks to the information provided by TextRazor. Regarding the behaviour analysis, Figure 1b displays an example of the role composition for the community built around the M263 Facebook group. We can see the evolution of the different roles over time. At the beginning, during the creation of the group, it was mostly composed by inactive users (lurkers and followers) but between the end of 2011 and the beginning of 2012 a mixed of different roles was present in the community, including leaders

---

[3] http://www.textrazor.com/

and super users, which are the most active an engaged roles. The application allows monitoring the role composition over time for all Facebook groups as well as monitoring the behaviour of individual users. For a particular user the application displays her role path (the different roles that the user adopts over time) in all Facebook groups where the user has participated. Studying the user's behavioural paths can help us to detect particular patterns that students follow before dropping or loosing interest about course. Note that each of the studied Facebook groups is linked to a particular OU course. To obtain information about the course, therefore complementing the results of the analyses, we make use of the OU's linked data portal (data.open.ac.uk). Using different SPARQL queries we can obtain information about OU courses, their title, description, available locations, required level and categorisation, among others.

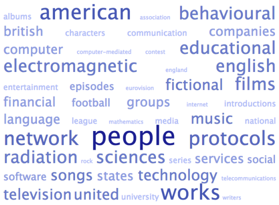


Figure 2.1 Top Topics across Facebook groups

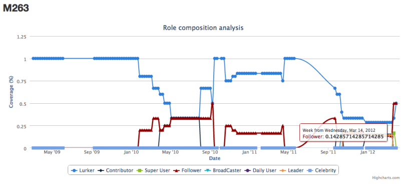


Figure 2.2 Behaviour analysis for group M263

4 Conclusions and Future work

This paper presents OU Social, a semantic social media analysis platform developed to collect and analyse students' feedback about OU courses expressed in Facebook public groups. The prototype is based on two semantic analysis modules that identify the behaviour of users and the emergent topics over time. Information about the courses is also integrated into the system by exploiting the OU's linked data portal. As seen by the video, extensions can be added to this prototype to facilitate the daily work of course managers. We are currently looking at tools like Google Trends (http://www.google.co.uk/trends/), Meltwater Buzz (buzz.meltwater.com), etc. to select visualisations that can better display the result of the analyses. Despite the existence of many other social media analysis tools in the market, OU Social is specifically designed to fulfil the needs of OU course managers. Apart from the interface extensions we are currently working on the integration of students historical data. This will enable correlating their behaviour and topic interests with their performance in different courses, providing the bases for more sophisticated analyses.

References

1. M. Rowe, M. Fernandez, S. Angeletou and H. Alani. (2012). Community analysis through semantic rules and role composition derivation. Web Semantics: Science, Services and Agents on the World Wide Web.

Demonstration: Semantic Web Enabled Smart Farm with GSN

Raj Gaire[1], Laurent Lefort[1], Michael Compton[1], Gregory Falzon[2], David Lamb[2] and Kerry Taylor[1]

[1] CSIRO Computational Informatics, Acton, Australia
{raj.gaire, laurent.lefort, michael.compton, kerry.taylor}@csiro.au
[2] Precision Agriculture Research Group, University of New England, Australia
{gfalzon2, dlamb}@une.edu.au

Abstract. GSN is an open source middleware designed for managing data produced by sensors deployed in a sensor network. We have extended the GSN to enable (i) semantically aware preparation, exchange and processing of the data (ii) user specified event processing for alerts, and (iii) associate sensor data to *things*. Here, we demonstrate our smart farm as a use case of a semantically aware sensor network for better integration of sensor data.

1 Introduction

Sensing devices are used in agriculture to measure and control farming activities. Smarter use of these measurements requires integration with other information. For example, soil condition measurements such as temperature and volumetric water content together with historical and weather forecast data can help make decisions about the time to sow a particular crop. Similarly, the cattle location data together with the current weather data can help monitor the cattle welfare. Since such data are often distributed across different organisations, the semantic web can be used as an integration mechanism for making better decisions.

Sensors produce data streams continuously or at short intervals generating large volumes of data. Therefore, data management is a prominent issue with sensor networks. Global Sensor Network (GSN) [1], an open source middleware, provides some foundation for the management of the streaming sensor data. In order to enable integration of this data across the web, it is necessary to employ ontologies like SSN [2] and techniques like Linked Data [3] that are widely accepted in the semantic web community. The disparity between existing tools like GSN and the need for providing open access for an effective integration of sensor network data exists as a barrier.

Our Kirby Smart Farm is a prototypical 269 hectare livestock property located in Armidale, NSW. In this paper, we use our smart farm to illustrate how GSN can be extended to enable integration of sensor network data with external data, define situation monitoring conditions to produce alerts, and extend the sensor measurements to implement *web of things* in a farm. Specifically, in Section 2, we describe the architecture of smart farm system. Section 3 highlights our extensions of GSN and semantic web aspects, followed by the conclusion in Section 4. Furthermore, the semantic network aspect of our work in a farming environment is described in [4] while the business aspect is explained in [5]. This system can be accessed from our website `http://smartfarm-ict.it.csiro.au`.

2 Architecture

Our farm contains a mixture of environmental and livestock tracking sensor nodes: 100 soil sensors, 2 weather stations and 65 cattle tags. A soil sensor node contains sensors measuring ground temperature, soil temperature, volumetric water content (VWC) and electric conductivity (EC). A weather station node contains sensors measuring air temperature, photo-synthetically active radiation (PAR), pressure, wind, rain and hail measurements. These nodes also contain sensors to measure temperature, battery status, solar voltage and current of the platform in which the sensors are embedded. Finally, the cattle have active tags attached to their ears, which send radio signals to base stations. Based on the time lapsed to receive the signal at three base stations, the locations of cattle are determined[3].

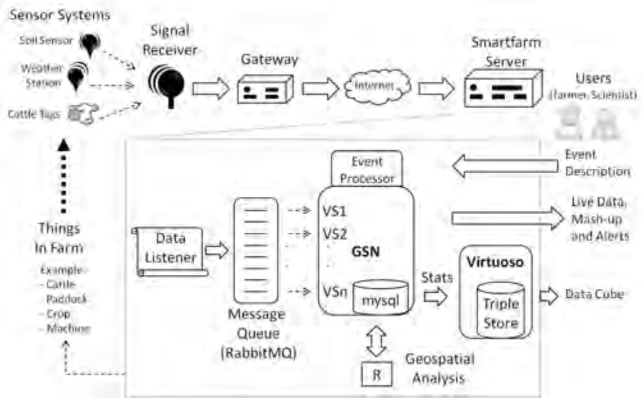


Fig. 1: The Smart Farm System Architecture

The architecture design of the smart farm is shown in Fig 1. Here, the signal received from all the sensors are collected by a gateway located on the farm and sent to smartfarm servers through a high speed broadband network. The smartfarm server contains four software components: data listener (a python script library), RabbitMQ[4] (a message queue system), GSN (a sensor network middleware) and Virtuoso[5] (a triple store enabled DBMS). The data listener

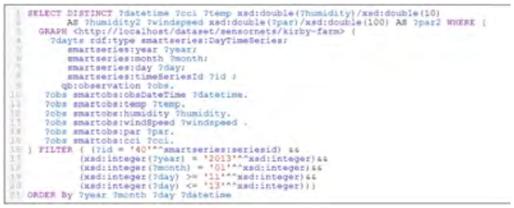
---

[3] http://www.taggle.com.au/livestock.php

[4] http://www.rabbitmq.com

[5] http://virtuoso.openlinksw.com/

directly receives data from the farm, transforms them to text messages and then publishes the messages to the message queue. The GSN is configured with virtual sensors which subscribe to these messages.



```
SELECT DISTINCT ?datetime ?cci ?temp xsd:double(?humidity)/xsd:double(10)
        AS ?humidity2 ?windspeed xsd:double(?par)/xsd:double(100) AS ?par2 WHERE {
  GRAPH <http://localhost/dataset/sensornets/kirby-farm> {
    ?dayts rdf:type smartseries:DayTimeSeries;
        smartseries:year ?year;
        smartseries:month ?month;
        smartseries:day ?day;
        smartseries:timeSeriesId ?id ;
      qb:observation ?obs.
   ?obs smartobs:obsDateTime ?datetime.
   ?obs smartobs:temp ?temp.
   ?obs smartobs:humidity ?humidity.
   ?obs smartobs:windSpeed ?windspeed .
   ?obs smartobs:par ?par.
   ?obs smartobs:cci ?cci.
} FILTER ( (?id = '40'^^smartseries:seriesid) &&
          (xsd:integer(?year) = '2013'^^xsd:integer)&&
          (xsd:integer(?month) = '01'^^xsd:integer)&&
          (xsd:integer(?day) >= '11'^^xsd:integer)&&
          (xsd:integer(?day) <= '13'^^xsd:integer))}
ORDER By ?year ?month ?day ?datetime
```

Fig 2. CCI SPARQL query

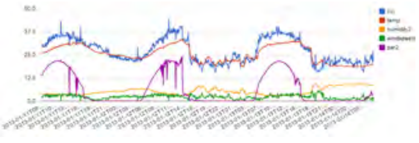


Fig 3. CCI as a composite variable of temp, humidity, wind speed and PAR

Our sensor network requires management of both static and dynamic data. We have used a hybrid approach to manage them using GSN and Virutoso [4]. Live linked data in RDF format are provided through GSN, while Virtuoso is used to provide archived data in data cube [6] format and visualised using VisualBox[6] (see Fig 2,3).

3 Semantically Enabled GSN

GSN is particularly useful in micro-management of live sensor data. However, GSN in its original form has a few limitations: it requires upgrades of dependent libraries; it lacks some important concepts (e.g. it is not possible to specify the units of measurements); it supports limited situation monitoring queries; and it does not provide data in a format expected by the semantic web community. We have modified GSN to overcome these limitations. In addition, we have extended GSN to provide additional features. Firstly, a combination of Java/R algorithms

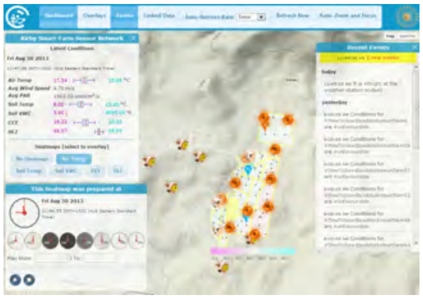


Fig 4. The smart farm map interface

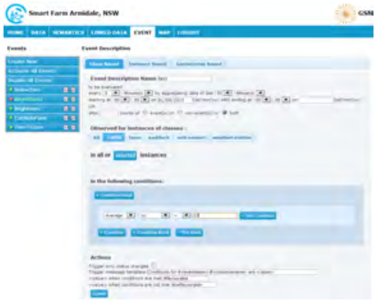


Fig 5. The interface for user defined events

---

[6] http://visualbox.org

is implemented for geo-spatial data processing to produce spatially aggregated cross-sectional data, generate heatmaps and infer relevant measurements corresponding to the cattle location (see Fig 4). Secondly, complex event processing system has been created based on semantic event descriptions [7], which is also embedded in GSN. We have identified and implemented a number of alert conditions, such as 'sowing time' for a crop, 'cattle not in farm', 'frost', and 'soil compaction' which are particularly useful to farmers. At the same time, users can specify their own alerts (see Fig 5), enabling them to embed their knowledge into the system. Thirdly, composite variables as cattle welfare indicators comprehensive climate index (CCI) [8] and heat load index (HLI) [9] have been implemented. Finally, GSN is extended to produce RDF data in linked data formats for both live and archived data.

4 Conclusion

In this paper, we presented Kirby 'Smart' Farm as a prototype farm installed with various sensors and connected with a broadband network. We demonstrated that by enabling a farm with the semantic web and providing query capability on both the static (i.e. archived) and the dynamic (i.e. live) linked data, we can fulfil the needs of the farmers and help them make better decisions.

References

1. Aberer, K., Hauswirth, M., Salehi, A.: A middleware for fast and flexible sensor network deployment. In: Proceedings of the 32nd international conference on Very large data bases, VLDB Endowment (2006) 1199–1202
2. Lefort, L., Henson, C., Taylor, K., Barnaghi, P., Compton, M., Corcho, O., Garcia-Castro, R., Graybeal, J., Herzog, A., Janowicz, K., et al.: Semantic sensor network xg final report. W3C Incubator Group Report **28** (2011)
3. Bizer, C., Heath, T., Berners-Lee, T.: Linked data-the story so far. International Journal on Semantic Web and Information Systems (IJSWIS) **5**(3) (2009) 1–22
4. Gaire, R., Lefort, L., Compton, M., Falzon, G., Lamb, D.W., Taylor, K.: Semantic web enabled smart farming. In: Proceedings of the 1st International Workshop on Semantic Machine Learning and Linked Open Data (SML2OD) for Agricultural and Environmental Informatics, ISWC (2013) accepted
5. Griffith, C., Heydon, G., Lamb, D., Lefort, L., Taylor, K., Trotter, M., Wark, T.: Smart farming: Leveraging the impact of broadband and the digital economy (2013)
6. Cyganiak, R., Reynolds, D., Tennison, J.: The rdf data cube vocabulary, w3c working draft 05 april 2012. World Wide Web Consortium (2012)
7. Taylor, K., Leidinger, L.: Ontology-driven complex event processing in heterogeneous sensor networks. In: The Semanic Web: Research and Applications. Springer (2011) 285–299
8. Mader, T., Johnson, L., Gaughan, J.: A comprehensive index for assessing environmental stress in animals. Journal of Animal Science **88**(6) (2010) 2153–2165
9. Gaughan, J., Mader, T.L., Holt, S., Lisle, A.: A new heat load index for feedlot cattle. Journal of Animal Science **86**(1) (2008) 226–234

Exploring Linked Open Data with Tag Clouds

Xingjian Zhang, Dezhao Song, Sambhawa Priya, and Jeff Heflin

Department of Computer Science and Engineering, Lehigh University
19 Memorial Drive West, Bethlehem, PA 18015, USA
`{xiz307,des308,sps210,heflin}@cse.lehigh.edu`

Abstract. In this paper we present the contextual tag cloud system: a novel application that helps users explore a large scale RDF dataset. Unlike folksonomy tags used in most traditional tag clouds, the tags in our system are ontological terms (classes and properties), and a user can construct a context with a set of tags that defines a subset of instances. Then in the contextual tag cloud, the font size of each tag depends on the number of instances that are associated with that tag and all tags in the context. Each contextual tag cloud serves as a summary of the distribution of relevant data, and by changing the context, the user can quickly gain an understanding of patterns in the data. Furthermore, the user can choose to include RDFS taxonomic and/or domain/range entailment in the calculations of tag sizes, thereby understanding the impact of semantics on the data. The system runs on the BTC2012 dataset with more than 1.4 billion triples from which we extract over 380,000 tags. Several scalability challenges must be overcome in order to achieve a responsive interface.

Keywords: Tag Cloud, Data Visualization, Data Exploration

1 Introduction

We present the contextual tag cloud system[1] as an attempt to address the following questions: How can we help casual users explore the Linked Open Data (LOD) cloud? Can we provide a more detailed summary of linkages beyond the LOD cloud diagram[2]? Can we help data providers find potential errors or missing links in a multi-source dataset of mixed quality?

In analogy to traditional Web 2.0 tag cloud systems, an instance is like a web document or photo, but is "tagged" with formal ontological classes, as opposed to folksonomies. Tags are then another name for the categories of instances. We extend the expressiveness and treat classes, properties and inverse properties as tags that are assigned to any instances that use these ontological terms in their triples. The font sizes in the tag cloud reflect the number of matching instances for each tag, indicating the distribution of tags used by instances in this dataset. We allow the user to change their focus on a specific subset of instances in the

---

[1] `http://gimli.cse.lehigh.edu:8080/btc/`

[2] `http://lod-cloud.net/`

dataset by specifying a combination of ontological terms as the context on the fly. The context is a set of tags or the negation of tags, where the negation of a tag means that the tag is not assigned to the instances. Then the resulting contextual tag cloud will resize tags to display the conditional distribution of tags for the instances that satisfy this context.

Although materialization can lead to many interesting facts, a single erroneous axiom in the uncurated dataset could generate thousands of errors. Rather than attempting to guess which axioms are worthwhile, our system supports multiple levels of inference; and at any time a user can view tag clouds with the same context under different entailment regimes, which helps users understand the dataset better and helps data providers investigate the errors in the dataset.

To demonstrate the scalability of our system, we load the entire BTC2012 dataset. This complex dataset contains 1.4 billion triples, from which we extract 198.6M unique instances, and assign more than 380K tags to these instances. This multi-source, large-scale dataset brings us challenges in achieving acceptable performance and user-interface design as well as affordable preprocessing. Details of the technical aspects can be found in our research paper[3].

2 System Features

The initial tag cloud has no context (or semantically `owl:Thing`), and the tags in the cloud reflect the absolute sizes of instances related to each tag. We put classes and properties into two separate views, so that users will not treat a property called "author" (which may have domain Publication) as a class name by mistake. To emphasize that difference, we also add an icon with "C" or "P" in front of each tag. If a tag is clicked, it will be added to the current context, and then a new tag cloud will be shown for the updated context. A user can add/remove any tags to/from the context, and explore any dynamically defined types of instances. A user can also switch to Instance View to investigate the detailed triples of instances specified by the context.

A user can also change the inference regime: (1) No Inference; (2) Taxonomy Inference; (3) Domain/Range Inference; and (4) Both Inference. If a set of tags are entailed to be equivalent, we choose a cannoncial tag to group them under. We display a $\equiv$ after the canonical tag to indicate this; clicking it will display the equivalent tags. Also for any tag cloud, we can turn on the negation mode, and then the tag sizes indicate how many instances do not have this tag under the current context and inference level. A negation tag can be also added to the context, which mathematically means the relative complement. Fig. 1 shows an example of the property tag cloud with negation tag in the context.

We show tag clouds in pages when there are too many tags. To help users locate specific tags in the tag cloud, we initially sort the tags by their local names alphabetically. When the system receives a request (context and inference level),

---

[3] Xingjian Zhang, Dezhao Song, Sambhawa Priya and Jeff Heflin. Infrastructure for Efficient Exploration of Large Scale Linked Data via Contextual Tag Clouds. 12th International Semantic Web Conference. Sydney, Australia. 2013.

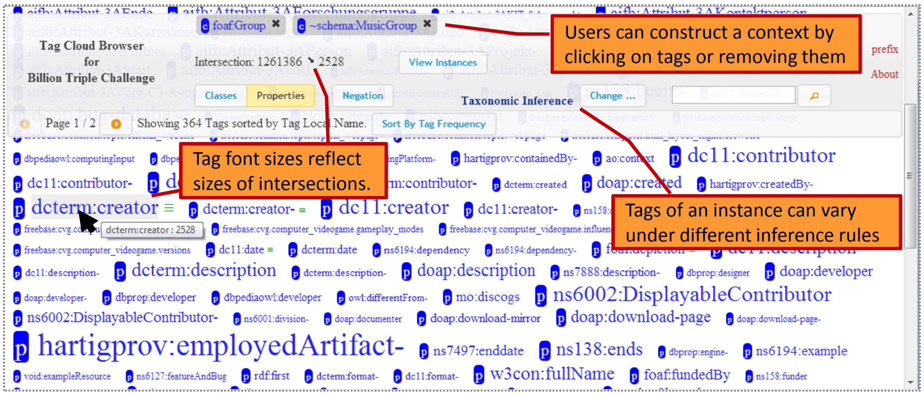

Users can construct a context by clicking on tags or removing them
Tag font sizes reflect sizes of intersections.
Tags of an instance can vary under different inference rules
Tag Cloud Browser for Billion Triple Challenge
foaf:Group
~schema:MusicGroup
Intersection: 1261386 ↘ 2528
View Instances
Classes
Properties
Negation
Taxonomic Inference
Change ...
prefix
About
Page 1 / 2
Showing 364 Tags sorted by Tag Local Name.
Sort By Tag Frequency
dcterm:creator : 2528

Fig. 1. Property tag cloud shows property usages of instances of `foaf:Group` that are not instances of `schema:MusicGroup`.

it will process tags in alphabetic order, and then stream out whatever is available for the requested page. If the user chooses to browse tags alphabetically, then the streaming of results is generally able to stay ahead of the user by pre-fetching results for tags on subsequent pages. Instead of browsing, a user can also search for tags by keywords. We index the local name, `rdfs:label` and `rdfs:comment` (if it exists) for each tag to support such keyword search. The retrieved tags will then be shown in the tag cloud sorted by their relevance to the keyword with their frequencies under the current context and inference regime. In addition, we provide sorting by tag frequency as another option, so that users can easily see the most popular tags under the current context and inference. However, we have to wait until all the frequencies are computed to enable this sort option. For some contexts, it can take a few minutes for the overall computation of thousands of pages of results. We show a progress bar of the computation and the estimated time left; but before sort is enabled, users can still browse by alphabetical order or search with keywords.

3 Use Cases

We believe our system can be used for multiple purposes. Here we shall briefly describe four scenarios where a user explores the BTC dataset.

Choose the right terms for SPARQL. A user wants to build a SPARQL query on lakes, but does not know what classes about lakes are available. Then by starting with a keyword search “lake”, the user is presented with a tag cloud containing all tags that match the keyword, and finds that `dbpediaowl:Lake` contains the most instances. After picking this class, the user wonders what property to use for quering the area of a lake. Then by searching again with keyword “area”, the user is presented with the contextual tag cloud with keyword-matched tags whose sizes reflect the intersection of the instances of the tags and `dbpediaowl:Lake`. It turns out `dbpediaowl:areaTotal` is the best choice of the property.

Learn interesting facts. A casual user tries a keyword search on "Manhattan". There are classes of parks, streets, etc. located in Manhattan. However, it also has the class `yago:ManhattanProjectPeople`; the user adds this to the context to explore in more detail. In the resulting tag cloud, the user finds various categories for such people, and then searches again for "scientist". Then surprisingly there is a tag `freebase:computer.computer_scientist`. The user is intrigued, because she did not know that any computer scientists were involved in the effort to build the first atomic bomb. By adding that tag and switching to the Instance View, she finally learns that this scientist is John von Neumann.

Detect Co-reference Mistakes. Sometimes when two tags have a small unexpected intersection, it is due to an error, rather than an interesting fact. For example, a user finds the tag `yago:BritishComputerScientist` has one common instance with `dbpediaowl:MusicalArtist` (as shown by a very small tag). By adding this tag and looking into the triple details in the Instance View, we can see the two `dbpediaowl:abstract` values clearly refer to two different people who have the same name but different birth years but have been connected by an erroneous owl:sameAs statement.

Examine ontological errors. Under Domain/Range Inference, a user finds that `foaf:Person` appears in the tag cloud of context `dbpediaowl:Software`, implying that some people are software, or vice versa! If the user turns off the Domain/Range Inference, this error will disappear. What is wrong with this inference? If a property is claimed as having domain `foaf:Person`, then any instance using this property will be classified as the instance of this class. With this assumption in mind, the user adds both `foaf:Person` and `dbpediaowl:Software` to the context, selects the property view and Domain/Range Inference, and sorts the properties by frequency. Then the top tag is `foaf:homepage`, which has all the instances in the current context (by hovering the mouse over the tag, we can see the frequency of this tag). This is very suspicious, and by clicking on the "P" icon before `foaf:homepage`, the user can see that `foaf:Person` is an inferred super tag of this tag, and that causes the error. By checking the raw ontology we find that although the domain of `foaf:homepage` is `owl:Thing` in the `foaf` schema, two other sources in the BTC dataset make the claim that the domain is `foaf:Person` and `foaf:Agent` respectively.

4 Conclusions

In this paper we introduce the features and use cases of the contextual tag cloud system. The contextual tag cloud system is a novel tool that helps both casual users and data providers explore the BTC dataset: by treating classes and properties as tags, we can visualize patterns of co-occurrence and get summaries of the instance data. From the common patterns users can better understand the distribution of data in the KB; and from the rare co-occurrences users can either find interesting special facts or errors in the data.

Comparing ontologies with *ecco*

Rafael S. Gonçalves, Bijan Parsia, and Uli Sattler

School of Computer Science, University of Manchester, Manchester, United Kingdom

Abstract. In this paper we present the diff tool *ecco*, which detects changes to both axioms and concepts between OWL ontologies. Furthermore, the tool aligns axiom changes between each other, according to a fine-grained change categorisation, and subsequently aligns axiom changes with the concepts that each of those directly affect. The diff is open source, and made available as a standalone command-line tool, as well as a Web-based application.

1 Introduction

The diff tool presented in this paper, *ecco*, incorporates structural and semantic techniques to detect differences between OWL ontologies at both axiom and concept level.

At the axiom level, *ecco* uses structural difference to detect additions and removals, and subsequently verifies whether these changes have any logical impact (i.e., whether they are *logically effectual* or *ineffectual*). Based on these two, coarse-grained categories, we derive finer-grained ones that reflect the apparent impact of the changes detected, as described in [1]. For instance, by further constraining an axiom $A \sqsubseteq B$ into $A \sqsubseteq B \sqcap C$ we "strengthen" it, and the relation between the stronger axiom and its preceding version is made explicit by our categorisation (that is, we align the source and target of the change), and suitably presented by our tool. Such a categorisation of changes is shown to facilitate the navigation through, and analysis of axioms in the diff.

In addition to detecting axiom changes, *ecco* computes entailment and term differences between ontologies. Differences at the entailment level are computed according to several entailment grammars explained in [2], and only shown to users upon request. These entailment differences are used to retrieve the sets of concepts that were specialised or generalised, that is, concepts which have a new superconcept or a new subconcept, respectively. Finally, the tool aligns concept changes with axiom changes, where each (effectual) axiom is aligned with the concepts it directly affects.

The diff *ecco* is freely available as a command-line tool with advanced features, as well as a Web-based application. Both of these output an XML change set file and a transformation of that into HTML, which allows users to browse through and focus on those changes of utmost interest using any Web browser and operating system.

2 Related Work

Structural difference, based on OWL's notion of structural equivalence, is used in several tools to present axiom changes between ontologies; specifically within ContentCVS [3], Bubastis [6] and OWLDiff [5]. However, none of these produce any form of alignment between axioms. The tool ContentCVS also computes entailment differences between ontologies, but does not extrapolate affected concepts from the entailments in the diff. The tool CEX [4] computes entailment and concept differences between acyclic $\mathcal{EL}$ terminologies with role hierarchies and range restrictions. In addition to the major restriction on its input, CEX does not compute changes between axioms, nor does it produce any form of alignment between axioms and the terms they affect.

3 *ecco*: A hybrid diff for OWL 2 ontologies

The diff *ecco* is an open source Java tool available at `https://github.com/rsgoncalves/ecco`. Most major (i.e., computationally expensive) operations are performed in parallel, taking advantage of new concurrency features in Java 7. The command line interface allows tuning the diff using advanced options, all of which can naturally be used programatically as well. Additionally, there is a Web-based front end that allows users to use the system on small to medium ontologies, without having to download it. A demo instance of the Web-based version of *ecco* is deployed at `http://owl.cs.manchester.ac.uk/diff`.[1] In order to demonstrate the functionality of the tool, as well as how its output can be interpreted, we carry out an example diff walkthrough using the toy ontologies in Table 1, and further on we show the output of *ecco* on those same ontologies.

Table 1: Example ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$.

	$\mathcal{O}_1$		$\mathcal{O}_2$
α_1 :	$A \sqsubseteq B$	β_1 :	$A \sqsubseteq B$
α_2 :	$B \sqsubseteq C$	β_2 :	$B \sqsubseteq C \sqcap F$
α_3 :	$C \sqsubseteq \exists r.X$	β_3 :	$C \sqsubseteq \exists r.X$
α_4 :	$\exists r.X \sqsubseteq \exists r.Y$	β_4 :	$X \sqsubseteq D$
α_5 :	$X \sqsubseteq D \sqcap E$	β_5 :	$F \sqsubseteq \exists r.Y \sqcap G$
α_6 :	$F \sqsubseteq \exists r.Y$		

To start with, *ecco* computes the sets of additions and removals between $\mathcal{O}_1$ and $\mathcal{O}_2$ according to structural equivalence. From these changes the tool distinguishes between those that have logical impact (effectual) and those that do not (ineffectual), that is, we check which removed axioms are entailed by $\mathcal{O}_2$ (ineffectual removals), and analogously for added axioms. This coarse-grained categorisation is shown in Table 2.

Table 2: Coarse-grained categorisation of changes in $\text{diff}(\mathcal{O}_1, \mathcal{O}_2)$.

Removals		**Additions**	
Effectual	**Ineffectual**	**Effectual**	**Ineffectual**
$\{\alpha_4, \alpha_5\}$	$\{\alpha_2, \alpha_6\}$	$\{\beta_2, \beta_5\}$	$\{\beta_4\}$

---

[1] The code is hosted at `https://github.com/rsgoncalves/ecco-webui`.

Subsequently, *ecco* performs a fine-grained categorisation of the changes according to entailment and justification relations, allowing us to identify and align changes between ontologies. The categorisation is shown in Table 3, where we denote effectual additions as $\mathrm{EffAdds}(\mathcal{O}_1, \mathcal{O}_2)$, ineffectual additions as $\mathrm{IneffAdds}(\mathcal{O}_1, \mathcal{O}_2)$, and analogously for removals.

Table 3: Fine-grained categorisation of changes in $\mathrm{diff}(\mathcal{O}_1, \mathcal{O}_2)$.

	Coarse-grained category	**Fine-grained category**	**Axiom change**	**Axiom alignment**
Additions	$\mathrm{EffAdds}(\mathcal{O}_1, \mathcal{O}_2)$	$\mathrm{Strgth}(\mathcal{O}_1, \mathcal{O}_2)$	β_2	$\{\alpha_2\}$
		$\mathrm{StrgthNT}(\mathcal{O}_1, \mathcal{O}_2)$	β_5	$\{\alpha_6\}$
	$\mathrm{IneffAdds}(\mathcal{O}_1, \mathcal{O}_2)$	$\mathrm{NewRed}(\mathcal{O}_1, \mathcal{O}_2)$	β_4	$\{\alpha_5\}$
Removals	$\mathrm{EffRems}(\mathcal{O}_1, \mathcal{O}_2)$	$\mathrm{Weakng}(\mathcal{O}_1, \mathcal{O}_2)$	α_5	$\{\beta_4\}$
		$\mathrm{PrRem}(\mathcal{O}_1, \mathcal{O}_2)$	α_4	–
	$\mathrm{IneffRems}(\mathcal{O}_1, \mathcal{O}_2)$	$\mathrm{NewRed}(\mathcal{O}_1, \mathcal{O}_2)$	α_2 α_6	$\{\beta_2\}$ $\{\beta_5\}$

The fine-grained categorisation shown in Table 3 reveals such changes as α_2 being strengthened into β_2 with shared terms, and similarly α_6 into β_5 though using new terms. In the set of ineffectual changes we have only new retrospective (resp. prospective) redundancies, that is, added axioms (resp. removed axioms) for which there are more constraining axioms in $\mathcal{O}_1$ (resp. $\mathcal{O}_2$). For instance, β_4 is a weaker version of α_5. Within the effectual removals we have α_5 which is weakened into β_4 using shared terms, and α_4 which has no identifiable relation with axioms in $\mathcal{O}_2$.

After axiom changes are categorised, *ecco* detects which atomic terms had their meaning affected between ontologies.[2] In order to do that, *ecco* first computes differences w.r.t. finite sets of entailments, defined according to some entailment grammar.[3] From the sets of lost and gained entailments, so called *witness axioms*, the tool extrapolates the affected terms depending on whether these occur on the left hand side of an entailment difference (specialised term), or right hand side (generalised term). Furthermore, *ecco* distinguishes between whether a concept A is directly affected, or indirectly affected via some other concept change which propagates to A. Finally, the tool aligns term and axiom changes based on witness axioms and their justifications: if the justification for a witness axiom that witnesses a direct (resp. indirect) change to A contains an effectual change α, then α is said to directly (resp. indirectly) affect A, denoted $\{\alpha\} \to^d A$ (resp. $\{\alpha\} \to^i A$). In Table 4 we show the result of computing and aligning term differences according to entailments between atomic concepts.

---

[2] Currently restricted to concept changes only, though roles are easily added.

[3] The tool supports all entailment grammars specified in [2], e.g., differences w.r.t. atomic subsumptions, or differences over subsumptions involving asserted subconcepts.

Table 4: Affected concepts with corresponding witness axioms and justifications.

	Affected concept	**Effect**	**Witness axioms**	**Justification(s)**	**Axiom alignment**
Specialised	A	Gained superconcept	$A \sqsubseteq F$	$\{\beta_1, \beta_2\}$	$\{\beta_2\} \rightarrow^i A$
			$A \sqsubseteq G$	$\{\beta_1, \beta_2, \beta_5\}$	$\{\beta_5\} \rightarrow^i A$
	B	Gained superconcept	$B \sqsubseteq F$	$\{\beta_2\}$	$\{\beta_2\} \rightarrow^d B$
			$B \sqsubseteq G$	$\{\beta_2, \beta_5\}$	$\{\beta_5\} \rightarrow^i B$
	X	Lost superconcept	$X \sqsubseteq E$	$\{\alpha_5\}$	$\{\alpha_5\} \rightarrow^d X$
Generalised	G	Gained subconcept	$F \sqsubseteq G$	$\{\beta_5\}$	$\{\beta_5\} \rightarrow^d G$
			$B \sqsubseteq G$	$\{\beta_2, \beta_5\}$	$\{\beta_2\} \rightarrow^i G$
			$A \sqsubseteq G$	$\{\beta_1, \beta_2, \beta_5\}$	— ” —
	F	Gained subconcept	$B \sqsubseteq F$	$\{\beta_2\}$	$\{\beta_2\} \rightarrow^d F$
			$A \sqsubseteq F$	$\{\beta_1, \beta_2\}$	— ” —
	E	Lost subconcept	$X \sqsubseteq E$	$\{\alpha_5\}$	$\{\alpha_5\} \rightarrow^d E$

4 Discussion

In this paper we presented the diff tool *ecco*, and exemplified how its categorisation mechanisms and presentation of differences can facilitate change analysis. By categorising axioms the tool presents the changed axioms and what they are a change of. Thus we can group and align changes according to their impact, allowing users to shift their attention to specific types of changes rather than going through an unstructured change set while inspecting both ontologies. Based on the combination and alignment of axiom and term changes, *ecco* provides a comprehensive and intelligible change report that would suit most ontology engineers, whether they are only interested in term or axiom changes. In particular, it facilitates the understanding of the impact of axiom changes on the ontology, and the meaning of its terms (via entailment differences). The diff tool is freely distributed as both a command line tool and a Web-based application.

References

1. Gonçalves, R.S., Parsia, B., Sattler, U.: Categorising logical differences between OWL ontologies. In: Proc. of CIKM-11 (2011)
2. Gonçalves, R.S., Parsia, B., Sattler, U.: Concept-based semantic difference in expressive description logics. In: Proc. of ISWC-12 (2012)
3. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga Llavori, R.: Supporting concurrent ontology development: Framework, algorithms and tool. DKE 70(1), 146–164 (2011)
4. Konev, B., Ludwig, M., Wolter, F.: Logical difference computation with CEX 2.5. In: Proc. of IJCAR-12 (2012)
5. Křemen, P., Šmíd, M., Kouba, Z.: OWLDiff: A practical tool for comparison and merge of OWL ontologies. In: Proc. of DEXA-11 (2011)
6. Malone, J., Holloway, E., Adamusiak, T., Kapushesky, M., Zheng, J., Kolesnikov, N., Zhukova, A., Brazma, A., Parkinson, H.: Modeling sample variables with an experimental factor ontology. Bioinformatics 26(8), 1112–1118 (2010)

Linked Scientometrics: Designing Interactive Scientometrics with Linked Data and Semantic Web Reasoning

Grant McKenzie[1], Krzysztof Janowicz[1], Yingjie Hu[1], Kunal Sengupta[2], and Pascal Hitzler[2]

[1] University of California, Santa Barbara, CA, USA
[2] Wright State University, Dayton, OH, USA

Abstract. In this demo paper we introduce a Linked Data-driven, **S**emantically-**e**nabled **J**ournal **P**ortal (SEJP) that offers a variety of interactive scientometrics modules. SEJP allows editors, reviewers, authors, and readers to explore and analyze (meta)data published by a journal. Besides Linked Data created from the journal's internal data, SEJP also links out to other sources and includes them to develop more powerful modules. These modules range from simple descriptive statistics, over the spatial analysis of visitors and authors, to topic trending modules. While SEJP will be available for multiple journals, this paper shows its deployment to the Semantic Web journal by IOS Press. Due to its open & transparent review process, SWJ offers a wide variety of additional information, e.g., about reviewers, editors, paper decisions, and so forth.

1 Introduction

Scientometrics are playing an increasingly important role in facilitating the understanding of different research fields as well as the research topics within them. In combination with Semantic Web reasoning, Linked Data provides the principles to structure and interlink data in a way that facilitates data integration and knowledge discovery and can therefore enhance scientometric analysis. In this paper, we present a Linked Data-driven, semantically-enabled journal portal (SEJP) that currently supports over 20 different interactive analysis modules. These modules range from simple descriptive statistics such as the acceptance rate of a journal to more complex modules that provide spatial analysis and topic modeling. For instance, SEJP allows editors to visualize authors together with keywords reflecting their expertise to better select reviewers for a new paper submission. While SEJP will be deployed to other IOS Press journals in the future, this paper showcases the SEJP functionality by showing its application to the structured data from the Semantic Web journal (SWJ).[3] SWJ adopts a unique open and transparent review process. This provides a rich amount of data related to the review and publication process, including not only papers' contents, but also reviewers and their reviews, assigned editors, paper decisions, resubmission time lines, and so forth.

---

[3] The current SEJP version can be used at *http://sejp.geog.ucsb.edu/SWJPortal*

2 The Linked Data Portal for SWJ

2.1 Structuring and Publishing Data

SWJ employs a highly customized version of the popular Drupal content management system (CMS)[1]. All submissions, reviews, notifications, and feedbacks are contributed through the CMS, storing content in a relational database management system. The first step in developing the portal was to export all data from the database, and convert it to the Resource Description Framework (RDF) format, making use of the bibliographic ontology BIBO [2]. While most of the data accessed from SWJ can be modeled by BIBO, the ontology was extended to include aspects such as the versioning of articles (*AcademicArticleVersion*). Once the data was organized and relationships were defined (e.g., Article *hasAuthor*), a custom Java converter was constructed using the OWL API and published online via Apache Jena's SPARQL server *Fuseki*; see [3] for details.

2.2 User Interface

Once the back-end data was organized, structured, and published to the Web, a modular user interface was developed to allow visual analysis of the SWJ data. A modular approach was taken with a *plug and play* mentality, allowing the analysis modules to be separated and configured based on the particular requirements of the applications. Built through HTML5, CSS, JavaScript, D3, and ExtJS, the front-end interface for the application is light-weight and compatible with any modern W3-compatible browser. Given the separation between the back-end data and the front-end analysis modules, SEJP is able to integrate data from other SPARQL endpoints and APIs; in our case the Semantic Web Dog Food portal and Microsoft Academic Search.

3 Modules

A variety of scientometric modules were developed for analyzing the SWJ data. Visual-analytic tools range from pie charts showing paper submission types to Cartograms of website visitors to edge-node graphs showing links between collaborating authors. Two of the more unique *trend* modules are discussed here.

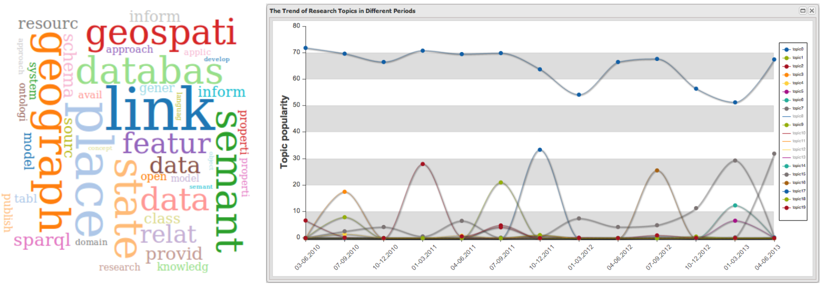
3.1 Research Topic Trends

This module shows how the research topics contained in the SWJ trend over time. In order to construct this module, a topic modeling approach was taken to extract latent topics in papers submitted to the SWJ. First, the text for all original submissions between March 2010 and April 2013 were accessed, cleaned of standard English stop-words and non-alpha numeric characters and stemmed[4]. The submissions were then grouped by time periods (3 month are considered as

---

[4] Using the Snowball stemmer - http://snowball.tartarus.org

one period), combining text from all articles within this period in to one single document. This produced a total of 13 documents. Latent Dirichlet allocation (LDA) was then applied to the documents with the purpose of extracting a set number of latent topics. LDA is an unsupervised, generative probabilistic model used to infer latent topics in a textual corpus [4]. In this case, LDA is applied across the set of 13 documents and topics are discovered, represented as a multinomial distribution over words. Based on the co-occurrence of words in the corpus and a numerical value for the resulting topics, LDA produces probability values for each word in each topic and for each topic in each document. The LDA model was tested with 50, 20 and 10 topics, and 20 topics produce the most human comprehensible results for this module. An example of one of these topics is shown in Figure 1a with font-size indicating relative probability of the word existing in the topic.



(a) Word cloud showing an example topic.

(b) Research topic trending module showing how topics change over time

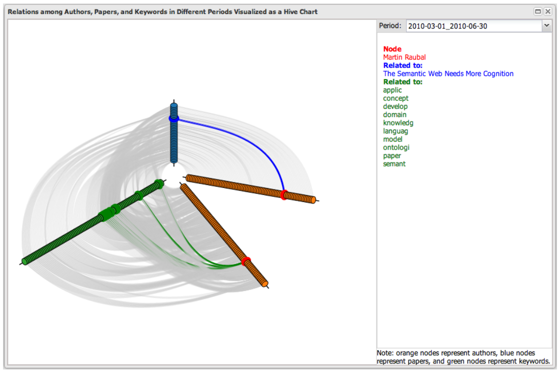
Fig. 1: Research topic trending module

The topics are then displayed through the user interface via an interactive line graph constructed with the JavaScript D3 charting library (Figure 1b). LDA defines each document (publications grouped by time period) as a distribution over all topics with the total probability across all topics summing to 1. Visually this is represented with time period shown on the X-axis and probabilities for each topic (multiplied by 100) shown on the Y-axis. Initially the 20 topics are color coded and shown on the chart with the option to show or hide each topic through a click-able legend on the right. Hovering one's mouse over a line produces a pop-up bubble that informs the user of the topic strength as well as the top ten words most probable to that topic.

3.2 Author-paper-keyword Hive Chart

The author-paper-keyword hive chart module (Figure 2) is a unique interactive visualization showing the relationship between authors, papers, and keywords. This module has the capability to help users to discover the possibly hidden relations among the three distinct types of data.

Hovering over a node on the authors axis (orange) produces a one-to-many relationship on the keyword axis (green) showing all the keywords mentioned by


Relations among Authors, Papers, and Keywords in Different Periods Visualized as a Hive Chart

Period: 2010-03-01_2010-06-30

Node
Martin Raubal
Related to:
The Semantic Web Needs More Cognition
Related to:
applic
concept
develop
domain
knowledg
languag
model
ontologi
paper
semant

Note: orange nodes represent authors, blue nodes represent papers, and green nodes represent keywords.

Fig. 2: Hive chart showing relationships between authors, papers and keywords.

a specific author. Additionally, there is a one-to-many relationship between the selected author and the papers (blue axis) that he or she has contributed to the SWJ. The same relationships are true of selecting any node in either the paper or the keywords axis, allowing for exploration of the data from any node. This module allows users to find authors who are concerned with similar research topics, and can also help visually discover all the coauthors of a researcher. Editors can use the module to find suitable reviewers.

4 Conclusions

This demo paper presents a Linked Data-driven, semantically-enabled journal portal for scientometrics and deploys it to the Semantic Web journal. SEJP uses a journal's internal data and also connects to other (Linked Data) sources to includes them in the analysis. Two scientometric analysis modules were discussed in the prior sections focusing on the changing of topics over time as well as the relations between authors, papers and keywords. These modules, however, are only a small subset of a suite of interactive modules developed for the portal. As development continues to progress, new modules and tools will be added, further advancing the portal's capability for scientometrics. In the near future, SEJP will be deployed to other journals as well.

References

1. Hitzler, P., Janowicz, K., Sengupta, K.: The new manuscript review system for the semantic web journal. Semantic Web **4**(2) (2013) 117–117
2. D'Arcus, B., Giasson, F.: Bibliographic ontology specification. Online: http://bibliontology.com/specification (November 2009) Last accessed 2013-5-12.
3. Hu, Y., Janowicz, K., McKenzie, G., Sengupta, K., Hitzler, P.: A linked data-driven semantically-enabled journal portal for scientometrics. International Semantic Web Conference (October 2013)
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. the Journal of machine Learning research **3** (2003) 993–1022

The Benefits of Incremental Reasoning in OWL EL

Yevgeny Kazakov and Pavel Klinov

The University of Ulm, Germany
{ yevgeny.kazakov, pavel.klinov } @uni-ulm.de

Abstract. This demo will present the advantages of the new, bookkeeping-free method for incremental reasoning in OWL EL on incremental classification of large ontologies.[1] In particular, we will show how the typical experience of a user editing a large ontology can be improved if the reasoner (or ontology IDE) provides the capability of instantaneously re-classifying the ontology in the background mode when a change is made. In addition, we intend to demonstrate how incremental reasoning helps in other tasks such as answering DL queries and computing explanations of entailments. We will use our OWL EL reasoner ELK and its Protege plug-in as the main tools to highlight these benefits.

1 Introduction

The $\mathcal{EL}$ family of Description Logics (DLs) are tractable extensions of the DL $\mathcal{EL}$ featuring conjunction and existential restriction. It is the formal basis of the OWL EL profile [2] of the Web ontology language OWL 2 specifically aimed at applications that require management of large terminologies, which is common in biology, health care and life sciences. Ontology classification is the core reasoning task used by such applications. It requires computing all entailed (implicit) subsumption relations between atomic classes. Specialized $\mathcal{EL}$ reasoners, such as CEL [3], ELK [4], jcel [5], and Snorocket [6] are able to compute the classification for ontologies as large as SNOMED CT [7] with about 300,000 axioms. Classification plays the key role during ontology development, e.g., for detecting modeling errors that result in mismatches between terms. But even with fast classification procedures, frequent re-classification of ontologies can introduce significant delays in the development workflow, especially as ontologies grow over time. This motivates development of *incremental reasoning* methods which do not recompute the entire class hierarchy after local changes but manage to incorporate the changes incrementally.

The demo will present a novel incremental reasoning procedure implemented in ELK 0.4.0 and its positive impact on re-classification and related reasoning problems.[2]

1.1 State of the Art

Several incremental reasoning procedures have been developed for ontology languages. Most procedures maintain extra information to trace conclusions back to the axioms in order to deal with axiom deletions.

---

[1] This submission complements the accepted research paper which explains the developed incremental reasoning method in full technical detail [1].

[2] A screencast will be available at `https://code.google.com/p/elk-reasoner/`.

The Pellet reasoner [8] implements a technique called *tableau tracing* to keep track of the axioms used in tableau inferences [9]. Tracing maps tableau elements (nodes, labels, and relations) to responsible axioms. Upon deletion of axioms, the corresponding elements get deleted. This method is memory-intensive for large tableaux and currently supports only ABox changes.

The *module-based* incremental reasoning method does not perform full tracing of inferences, but instead maintains a collection of modules for derived conclusions [10]. The modules are (not necessarily minimal) subsets of the ontology that entail the respective conclusion. If no axiom in the module was deleted then the entailment is still valid. Unlike tracing, the method does not require changes to the reasoning algorithm, but still incurs the cost of computing and storing the collection of modules.

Managing the extra information such as traces or modules, broadly referred to as *bookkeeping*, typically incurs only a linear overhead. However, even that can substantially hurt user experience on large ontologies such as SNOMED CT.

1.2 Common Use Cases for Incremental Reasoning

The most frequently occurring scenarios when incremental reasoning is beneficial can be summarized as follows:

Continuous Classification The typical ontology development workflow consists of adding, removing, or modifying axioms and occasionally invoking a reasoner to classify the ontology. The latter is done to verify that the changes do not trigger any unwanted entailments and that all desirable entailments are there. Since classification tends to get slower as the ontology grows large, the ontology engineers often do it "offline" after a considerable set of changes has been accumulated. This is sub-optimal because if an error did occur it can become a needle in the haystack to find. A better approach is to classify the ontology continuously in the background mode, i.e., similarly to how modern IDEs continuous compile software's source code and immediately point out errors. Of course, this approach requires a fast, *incremental* incorporation of changes.

DL Queries Certain applications make use of a form of queries, also called *DL Queries*, based on complex class expressions. Every DL query is a class expression for which inferred superclasses, subclasses, or individuals need to be computed. One example of such application is the Virtual Fly Brain project.[3] DL Queries can be straightforwardly implemented by introducing fresh class names. Suppose superclasses need to be computed for a complex class C. Then one can introduce a fresh class name C', add the axiom $C' \sqsubseteq C$ to the ontology, and then re-classify it so that C' finds its place in the class hierarchy. Obviously, the last step is better be implemented incrementally.

2 Incremental Reasoning in ELK

This section briefly describes the main aspects of the incremental reasoning procedure implemented in ELK. Full technical details, including the $\mathcal{EL}^+$ inference rules, algo-

---

[3] `http://www.virtualflybrain.org/`

rithms, proofs, and experiments can be found in the research track paper [1]. Also, the interested reader can run the code examples provided on the ELK Web page.[4]

There are two main ideas behind our method. The first has been borrowed from the known DRed (over-delete, re-derive) method for maintaining materialized views in databases [11]. When an axiom is deleted or modified, conclusions of all $\mathcal{EL}^+$ inferences in which the axiom was used (as a side-condition) are deleted. Then the same happens to conclusions of all inferences which use deleted conclusions as premises (until a fixpoint). It is well-known that it may lead to over-deletion since some conclusions may have alternative derivations. Our second idea is based on *partitioning* of all conclusions to identify those which may need to be restored. Crucially, partitions are *not* stored, as modules or traces, during the forward classification and do not incur any overhead. Due to space limitations, we only illustrate the method on a small example.

Example 1. Consider the following $\mathcal{EL}^+$ ontology $\mathcal{O}$:

(ax1): $A \sqsubseteq \exists R.B$ (ax2): $\exists R.B \sqsubseteq C$ (ax3): $B \sqsubseteq \exists S.A$
(ax4): $\exists S.C \sqsubseteq C$ (ax5): $C \sqsubseteq D$ (ax6): $\exists S.\top \sqsubseteq D$

One can see that $\mathcal{O}$ entails the following atomic subsumptions: $A \sqsubseteq C$, $B \sqsubseteq C$, and $B \sqsubseteq D$ (we omit the intermediate inferences). Now, let us see what will happen if (ax4) is deleted. The axiom was used to derive $B \sqsubseteq C$ (together with (ax3) and another conclusion $A \sqsubseteq C$) which is retracted first. Then $B \sqsubseteq D$ is also deleted since it was produced by an inference which had $B \sqsubseteq C$ as a premise (using (ax5)). After that the conclusions whose left hand-side is one of $\{\exists S.C, \exists S.A, B\}$ are repaired (intuitively, these are classes whose superclasses changed during the deletion). The repair stage re-applies the inference rules w.r.t. the remaining axioms and restores the conclusion $B \sqsubseteq D$ using (ax3) and (ax6). Other conclusions, e.g., for A or C on the left, are intact.

Our experiments demonstrate that in practice the partitioning tends to be pretty fine and the changes are rather local, i.e., not many partitions need to be repaired. This is the reason why for large ontologies, such as SNOMED CT, incremental classification is 10–40 times faster than full classification, making re-classification nearly instantaneous (see [1] for more details and a comparison with the modularity-based method).

3 Structure of the Demonstration

Finally we describe what we intend to demonstrate during the demo session. Our general goal is to demonstrate performance gains resulted from incremental reasoning to give ontology developers a sense of how their user experience can be improved.

3.1 Continuous Classification

We will use large ontologies, such SNOMED CT, an $\mathcal{EL}^+$ version of GALEN, or others suggested by participants, to demonstrate the sub-second re-classification for a typical ontology editing workflow. We will use the ELK Protege 4+ plug-in[5] which allows users

---

[4] `https://code.google.com/p/elk-reasoner/wiki/IncrementalReasoning`

[5] Available at `https://code.google.com/p/elk-reasoner/downloads/list`

to turn incremental reasoning on and off to highlight the performance differences. We plan to prepare some changesets, including those introducing errors, e.g., class unsatisfiability, but will also let the participants make their own changes to ontology axioms.

3.2 Fast DL Query Answering

We will demonstrate fast answering of DL queries based on incremental reasoning. For large ontologies, such as SNOMED CT, it will be visible that answering a single query takes considerably less time than re-classification. We plan to use the DL Query plugin for Protege for interactive query answering (so that participants can enter their own queries). To people more interested in answering DL queries via a programming or Web interface, we will show how a simple Web service can handle parallel DL queries posted over HTTP (by computing the corresponding $\mathcal{EL}$ saturations incrementally).

References

1. Kazakov, Y., Klinov, P.: Incremental reasoning in OWL EL without bookkeeping. In: International Semantic Web Conference. (2013) to appear.
2. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C., eds.: OWL 2 Web Ontology Language: Profiles. W3C Recommendation (27 October 2009) Available at `http://www.w3.org/TR/owl2-profiles/`.
3. Baader, F., Lutz, C., Suntisrivaraporn, B.: Efficient reasoning in $\mathcal{EL}^+$. In Parsia, B., Sattler, U., Toman, D., eds.: Proc. 19th Int. Workshop on Description Logics (DL'06). Volume 189 of CEUR Workshop Proceedings., CEUR-WS.org (2006)
4. Kazakov, Y., Krötzsch, M., Simančík, F.: Concurrent classification of $\mathcal{EL}$ ontologies. In Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E., eds.: Proc. 10th Int. Semantic Web Conf. (ISWC'11). Volume 7032 of LNCS., Springer (2011) 305–320
5. Mendez, J., Ecke, A., Turhan, A.Y.: Implementing completion-based inferences for the $\mathcal{EL}$-family. In Rosati, R., Rudolph, S., Zakharyaschev, M., eds.: Proc. 24th Int. Workshop on Description Logics (DL'11). Volume 745 of CEUR Workshop Proceedings., CEUR-WS.org (2011) 334–344
6. Lawley, M.J., Bousquet, C.: Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner. In Taylor, K., Meyer, T., Orgun, M., eds.: Proc. 6th Australasian Ontology Workshop (IAOA'10). Volume 122 of Conferences in Research and Practice in Information Technology., Australian Computer Society Inc. (2010) 45–49
7. Schulz, S., Cornet, R., Spackman, K.A.: Consolidating SNOMED CT's ontological commitment. Applied Ontology **6**(1) (2011) 1–11
8. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. of Web Semantics **5**(2) (2007) 51–53
9. Halaschek-Wiener, C., Parsia, B., Sirin, E.: Description logic reasoning with syntactic updates. In: OTM Conferences (1). (2006) 722–737
10. Cuenca Grau, B., Halaschek-Wiener, C., Kazakov, Y., Suntisrivaraporn, B.: Incremental classification of description logics ontologies. J. of Autom. Reason. **44**(4) (2010) 337–369
11. Gupta, A., Mumick, I.S., Subrahmanian, V.S.: Maintaining views incrementally. In Buneman, P., Jajodia, S., eds.: Proc. 1993 ACM SIGMOD Int. Conf. on Management of Data, Washington, D.C., ACM Press (May 26-28 1993) 157–166

Curating Semantic Linked Open Datasets for Software Engineering

Kavi Mahesh, Aparna Nagarajan,
Apoorva Rao Balevalachilu, and Karthik Prasad

Centre for Knowledge Analytics and Ontological Engineering - KAnOE,
PES Institute of Technology, Bangalore 560085 India
`{drkavimahesh,aparna.nagarajan26,`
`apoorva.rao.b,karthikprasad008}@gmail.com`
`http://kanoe.org/`

Abstract. A typical software engineer spends a significant amount of time and effort reading technical manuals to find answers to questions especially those related to features, versions, compatibilities and dependencies of software and hardware components, languages, standards, modules, libraries and products. It is currently not possible to provide a semantic solution to their problem primarily due to the non-availability of comprehensive semantic datasets in the domains of information technology. In this work, we have extracted, integrated and curated a linked open dataset (LOD) called LOaD-IT exclusively on this domain from a variety of sources including other LODs such as Freebase and DBPedia, technical documentation such as JavaDocs and others. Further, we have built a technical helpdesk system using a semantic query engine that derives answers from LOaD-IT. Our system demonstrates how productivity of the software engineer can be improved by eliminating the need to read through lengthy technical manuals. We expect LOaD-IT to become more comprehensive in the future and to find other related practical applications.

Keywords: Linked Open Data, software engineering, technical helpdesk, information retrieval, semantic search.

1 Introduction

Software developers, test engineers, researchers and students face technical queries in the course of their daily work. The solution to their problems, more often than not, is to go through some technical documentation, search for information, look for a similar question on on-line forums, or to ask somebody who is considered an expert in that domain. Unfortunately, information from these sources is rather unstructured, distributed and difficult to obtain. The process of going through the technical documentation is tedious and time consuming. What the person ideally wants is a quick and precise answer to the question perhaps augmented by a suitable visualization to aid comprehension. The objective of our work is to enhance workplace productivity of information technology workers by reducing the effort and time spent in searching for technical information.

There is currently no readily available LOD that covers the factual knowledge sought by a typical information technology (IT) worker. Thus, our solution begins by semanticizing the relevant documentation by creating a linked open data store, which we call LOaD-IT (available as an RDF dump in N-Quads format at http://datahub.io/dataset/load-it). The potential of this highly specialized linked data is tapped by building Kappa, a technical assistant application prototype (Kappa is hosted at http://kanoe.org/lod/loadit-kappa.html). Kappa takes keyword queries as input and provides specific factual answer(s) instead of large number of HTML pages that may or may not contain the correct and complete answer. It also saves the user from scrolling through long manuals and documents to manually extract the answer. In order to provide results in the form of facts, the application first performs query interpretation as described in [1]. The user of the application need not have any knowledge of the structure of LOaD-IT or knowledge of RDF formats. Simple keywords entered in a search bar suffice. Search results are presented as a list of facts along with hyperlinks to relevant pages. Further, a graphical representation of the relevant RDF sub-graph is shown to help the user visualize the facts. Use of anontology to implement semantic search together with Web search has been demonstrated before (see, e.g., [2]). However, such methods typically require semantic meta-data or annotations to be manually added to existing documents or web pages.

2 Creating the Semantic LOD on Information Technology: LOaD-IT

Existing semantic datasets [3] like DBPedia, Freebase and YAGO are quite large and cover multiple domains while focusing heavily on common areas of interest rather than technical domains such as IT. It is unlikely that a retrieval engine on such a general-interest LOD will provide the necessary recall or precision for IT users. Our aim is to curate an LOD specific to the IT and software engineering domains which will also be of smaller size than the above LODs.

Another concurrent objective is to generate RDF quads from technical manuals, user guides and other documentation that are widely available but demand considerable amount of time and effort to sift through. Semanticization of this kind of documentation will contribute to raising the quality of answers (i.e., recall and precision) that the dataset is capable of providing at a fine grain of detail. Extraction of RDF facts from free flowing paragraphs of explanatory texts amounts to solving core problems of natural language processing and semantic information retrieval. Therefore, we take the practical approach of selecting ‘low-hanging fruit? by focusing on relatively well-structured technical documentation. Java Platform Standard Edition 7 Documentation API was selected for this purpose because of its definite structure in HTML tables and consistent layout, thereby facilitating extraction of clean and accurate facts. The widespread use of Java in industry and academia is another factor why this documentation API is a good choice for our purposes.

The methodology adopted for constructing the LOD has two parts:

2.1 Filtering Existing Datasets

A set of carefully chosen seed words from the domain of IT and software engineering is used to select matching resources from Freebase and DBPedia. For each of these resources, all triples matching the resource in their subject fields are selected. All other triples are filtered out as irrelevant to our domain of interest. These operations were carried out using Unix Shell scripts. The numbers of triples obtained by this method from Freebase and DBPedia are shown in Table 1 below.

2.2 Extracting triples from technical documentation

1. A crawler is built to cover the entire documentation collection.
2. A screen scraping package is designed and built to parse the pages and extract facts after a thorough study of the layout of each type of page.
3. Rules are written for logical mapping into quads:
   (a) Check if any reuse of predicates is possible from existing vocabularies like DC, FOAF, SKOS, and SIOC. Reuse wherever possible.
   (b) If compatible predicates are not found, create new predicates that can appropriately describe the relationships. For instance, to link a Java method to its return type or modifier, a new predicate is required.

   (c) Create and publish the required explanatory pages for predicates in RDF and HTML formats. URLs are used to generate quads: (Subject, Predicate, Object, Context) according to the conventions of Linked Open Data.
4. A suitable software library is used to load the quads into a database or data store.

The documentation of Java 7 SE is obtained from the official web site and parsed using a Python Library called BeautifulSoup4 [4]. The LOaD-IT dataset thus obtained has 1.4 million quads. Statistics of the dataset are given in Table 1.

Table 1. Numbers and Data Sizes in LOaD-IT

Source	*OriginalSize*	*FilteredSize*	*RetainedQuads*
Freebase	49 GB	128 MB	740149
DBPedia	44 GB	184 MB	433834
Javadocs	< 1 GB	65 MB	268275
TOTAL LOaD-IT		377 MB	1442258

3 Kappa: The LOaD-IT Retrieval Engine

Kappa first constructs the top k query candidates for the given keywords using the methodology outlined in [1]. Once the SPARQL query candidates are generated and the best query is selected, the results are obtained by sending an HTTP request to a REST API provided by our wrapper for LOaD-IT. This API access to LOaD-IT can be used also by other applications in the future. Results obtained as JSON objects are rendered both as triples and graphically to help the user in visualizing the relevant RDF sub-graph of LOaD-IT. An easy-to-use browser-based interface is provided for this purpose. In addition, the query is also sent to external Web search engines (e.g., Bing) as well as on-line technical discussion forums (e.g., StackOverflow). Results from external engines are also displayed to the user to help them look for external content related to the query. Fig. 1(a) shows the overall architecture of LOaD-IT and Kappa.

4 Example Scenarios

Two example scenarios and results are presented below to illustrate the usage of Kappa and LOaD-IT.

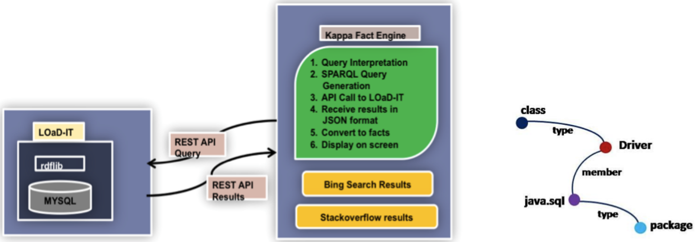
Scenario 1 The user wants to know what to write in the import statement in a Java program if he has a need to use a method from the Driver Class in Java. In fact, the user may not even know whether Driver is the name of a Class or a Package.
Keywords entered: Package Driver
Facts retrieved:

java.sql	type	package
java.sql	member	Driver
Driver	type	class

A graphical rendering of these three triples in shown in Fig. 1(b)

Scenario 2 The user wants to know the details of the Integer Class in Java.
Keywords entered: Integer
Facts retrieved:

Integer	comment	Integer class wraps a value of the primitive type int in an object
Integer	member	highestOneBit,hashCode,valueOf,getInteger..
Integer	type	class



(a) Schematic diagram of the architecture of Kappa

(b) Visualization of Facts Returned by Kappa

Fig. 1.

5 Conclusion and Future Work

Our primary aim was to create a linked open dataset for the IT domain. After filtering existing datasets, we created a crawler and used screen scraping to extract RDF quads from semi-structured technical documentation. We integrated and normalized the dataset to ensure its compatibility with other LODs and our application Kappa. A reasonably large technical dataset was obtained and its potential was illustrated by a simple yet meaningful and useful application, a fact-finding engine that serves as a technical helpdesk for practicing software engineers.

The dataset can be further enlarged and enriched to include RDF quads extracted from other technical documentation and on-line discussion forums. Further, it can be refined and grown over time by adopting a crowd-sourcing approach. Additionally, a vocabulary can be created specifically for predicates that are useful in a software engineering context and can be standardized to bring in global acceptance. The facts that are rendered can be enriched by natural language processing techniques to bring them closer to proper English.

We expect LOaD-IT to find several other useful applications in the near future. An immediate possibility is to use it as a controlled vocabulary plus ontology for corporate knowledge management. It can also be used to further automate the generation and management of program comments and documentation in software engineering by integrating it with IDE for various programming languages and platforms. None of these is readily possible without LOaD-IT wherein a federated retrieval engine would have to query various general-purpose LODs as well as document repositories and the Web. LOaD-IT together with Kappa promises to deliver the levels of precision and recall to make them useful in real-world software engineering situations.

A demonstration of our application Kappa is available at `http://kanoe.org/lod/loadit-kappa.html`.

References

1. Tran, T., Wang, H., Rudolph, S. et al.: Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (Rdf) Data. (2009) 405-416
2. Karanth, P., Mahesh, K.: Integrating Knowledge Base Retrieval with Web Search using Semantic Roles In: Lecture Notes in Engineering and Computer Science: Proceedings of the International MultiConference of Engineers and Computer Scientists 2012 (IMECS 2012, ICCS 2012), Vol. 1, ISBN 978-988-19251-1-4 pp. 344-349 IAENG Hong Kong (2012)
3. Bizer, C., Jentzsch, A., Cyganiak, R.: State of the LOD Cloud. Version 0.3 (September 2011), (2011)
4. Richardson, L.: Beautiful Soup-HTML. XML parser for Python, (2008)

Optique 1.0: Semantic Access to Big Data[⋆]

The Case of Norwegian Petroleum Directorate's FactPages

E. Kharlamov[1,⋆⋆], M. Giese[2], E. Jiménez-Ruiz[1], M. G. Skjæveland[2], A. Soylu[2],
D. Zheleznyakov[1], T. Bagosi[3], M. Console[5], P. Haase[4], I. Horrocks[1],
S. Marciuska[3], C. Pinkel[4], M. Rodriguez-Muro[3], M. Ruzzi[5], V. Santarelli[5],
D. F. Savo[5], K. Sengupta[4], M. Schmidt[4], E. Thorstensen[2], J. Trame[4], and
A. Waaler[2]

[1] University of Oxford, UK; [2] University of Oslo, Norway;
[3] Free University of Bozen-Bolzano, Italy; [4] fluid Operations AG, Germany;
[5] Sapienza Università di Roma, Italy

Abstract. The Optique project aims at developing an end-to-end system for semantic data access to Big Data in industries such as Statoil ASA and Siemens AG. In our demonstration we present the first version of the Optique system customised for the Norwegian Petroleum Directorate's FactPages, a publicly available dataset relevant for engineers at Statoil ASA. The system provides different options, including visual, to formulate queries over ontologies and to display query answers. Optique 1.0 offers installation wizards that allow to extract ontologies from relational schemata, extract and define mappings connecting ontologies and schemata, and align and approximate ontologies. Moreover, the system offers highly optimised techniques for query answering.

1 Introduction

Accessing the *relevant* data in Big Data scenarios is increasingly difficult both for end-user and IT-experts, due to the *volume*, *variety*, *velocity*, and *complexity* dimensions of Big Data. This brings a high cost overhead in data access for large enterprises. For instance, in the oil and gas industry, engineers spend 30–70% of their time gathering and assessing the quality of data. The Optique project[1] [1, 2] advocates for a next generation of the well known *Ontology-Based Data Access* (OBDA) approach to address the data access problem. The project aims at solutions that reduce the cost of data access dramatically. In our demonstration we present the first version of the Optique system which we customised for the Norwegian Petroleum Directorate's (NPD) FactPages.[2]

OBDA systems address the data access problem by presenting a general ontology-based and end-user oriented query interface over heterogeneous data sources. The core elements in a classical OBDA systems are an *ontology*, describing

---

[⋆] The research was supported by the FP7 grant Optique (n. 318338).

[⋆⋆] Corresponding author: `evgeny.kharlamov@cs.ox.ac.uk`

[1] `http://www.optique-project.eu/`

[2] `http://factpages.npd.no`

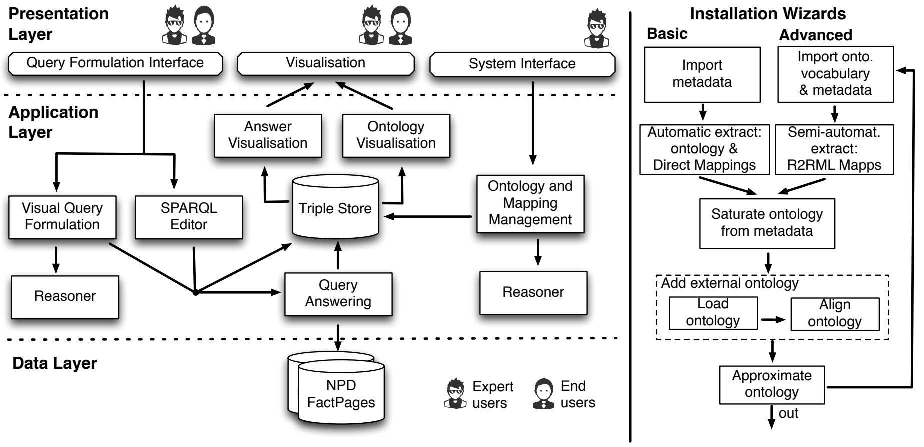


Fig. 1. *Left*: General architecture of the Optique 1.0 system; *Right*: installation process

the application domain, and a set of *mappings*, relating the ontological terms with the schemata of the underlying data sources. End-users formulate queries using the ontological terms and thus they are not required to understand the structure of the data sources. These queries are then automatically translated using the ontology and mappings into an executable code over the data sources.

State of the art OBDA systems, however, have shown among others the following limitations:

- The *usability* of OBDA systems is hampered by the need to use a formal query language. Even if the users know the ontological vocabulary, they may find difficult to formulate queries with several concepts and relationships.
- The *prerequisites* of OBDA, i.e., ontology and mappings, are in practice expensive to obtain. Additionally, they are not static artefacts and should evolve according to the new end-users' information requirements.
- The *efficiency* of the translation process and the execution of the queries is usually not sufficiently addressed in OBDA systems.

The first version of the Optique system, i.e., Optique 1.0, aims at partially overcoming the above limitations. Demonstration videos are available at following address: `http://www.cs.ox.ac.uk/isg/projects/Optique/demos/iswc2013/`.

2 System Overview

A general three-layer architecture of the Optique system is depicted in Figure 1 (*Left*). The current version of the system offers two main functionalities: to query/visualise data and install/maintain the ontology and the mappings. At the backend, the system also offers an efficient query processing mechanism.

Optique 1.0 allows to pose queries via a visual query formulation (VQF) interface, a SPARQL editor, or from a query catalog. VQF exploits reasoning in order to show both explicit and implicit domain knowledge to guide the formulation of the query.

Queries are executed by the Query Answering module based on Ontop system.[3] Ontop provides functionalities for rewriting SPARQL queries using the system's ontology and mappings, syntactic and semantic query optimisation, and query unfolding. Thus, high efficiency of query answering is guaranteed. Rewritten and unfolded queries are in SQL and they are executed over the NPD FactPages data, which is stored in a relational database. The query answers are converted into triples in order to confirm the format of the system's ontology, temporally stored in the system's triple store, and displayed to the user in a tabular way or on maps (using OpenStreetMap).

The installation and maintenance of the ontology and the mappings is done via the Ontology and Mapping Management component. Currently, this component includes two installation wizards: basic and advanced. In Figure 1 (*Right*) we depict workflows of the wizards. The basic wizard exploits the relational database metadata and automatically extracts an initial version of the ontology and direct mappings[4] to the ontology entities. The advanced wizard, unlike the basic one, requires the user intervention and an ontology vocabulary as input in order to (manually) create and edit R2RML mappings.[5] Both the basic and advanced wizards provide functionalities to align the bootstrapped ontology with a state of the art domain ontology and approximate the resulting ontology if it is outside the desired OWL 2 QL profile.[6] Alignment is performed using the ontology matching system LogMap,[7] which has shown to work well in practice and also includes mapping repair facilities.

Optique 1.0 is built on top of the Information Workbench[8] (IWB), a generic platform for semantic data management. The IWB provides a shared triple store for managing the assets of Optique 1.0, such as, ontologies, mappings, query logs, (excerpts of) query answers, database metadata, etc. The IWB also provides generic interfaces and APIs for semantic data management, e.g., ontology processing APIs. In addition to these backend data management capabilities, the IWB provides a flexible user interface which follows a semantic wiki approach, based on a rich, extensible pool of widgets for visualisation, interaction, mashup, and collaboration.

Finally, Optique 1.0 is customised for the NPD FactPages, which is a public, freely available dataset created to regulate and overlook the petroleum activities on the Norwegian Continental Shelf (NCS) and contains information collected from a wide range of activities on the NCS, e.g., operating companies, fields, discoveries, facilities, pipelines, and seismic surveys—both historic and current data. Its data has been converted and published as semantic web data [3], of which parts have been fed into the Optique 1.0 system.

---

[3] `http://ontop.inf.unibz.it/`

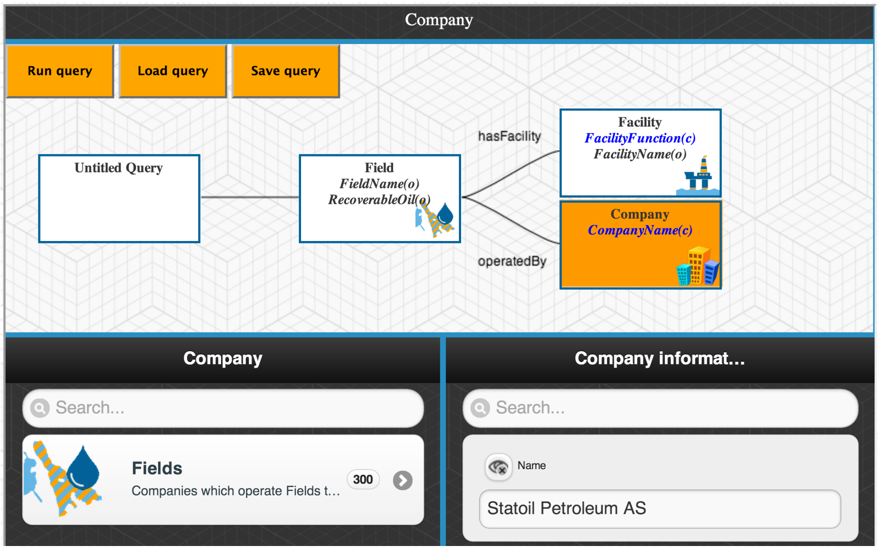
[4] `http://www.w3.org/TR/rdb-direct-mapping/`

[5] `http://www.w3.org/2001/sw/rdb2rdf/r2rml/`

[6] `http://www.w3.org/TR/owl2-profiles/`

[7] `http://code.google.com/p/logmap-matcher/`

[8] `http://www.fluidops.com/information-workbench/`



Name: Statoil Petroleum AS

Fig. 2. Optique 1.0 System, visual query formulation component

3 Demonstration Details

During the demonstration we will describe the NPD FactPages and present functionalities of the Optique 1.0 system, with the focus on the following aspects: query formulation and execution, and system installation. These aspects will be illustrated on the NPD FactPages data. For the query formulation we will stress our visual query formulation tool that currently supports construction of tree-shaped conjunctive SPARQL queries. The demonstrated queries will be from the oil industry domain. An example query is: *"Find all fields that are operated by 'Statoil Petroleum AS' and which have a facility that produces oil"*; it can be seen in the screenshot of the VQF in Figure 2. We will run queries and present results both in tables and maps, e.g., the location of "Fields" and "Oil facilities" will be displayed on maps. Regarding the system's installation, we will present both basic and advanced wizards and guide through their steps, that is, loading metadata, extraction of an ontology and mappings, alignment with the domain ontology, and approximation of the integrated ontology. We will also show how to edit extracted direct mappings and define new R2RML mappings.

References

1. M. Giese et al. "Scalable End-user Access to Big Data". In: *Big Data Computing*. Ed. by R. Akerkar. Chapman and Hall/CRC, 2013.
2. E. Kharlamov et al. "Optique: Towards OBDA Systems for Industry". In: *ESWC postproceedings volume: Best Workshop Papers*. 2013.
3. M. G. Skjæveland, E. H. Lian, and I. Horrocks. "Publishing the Norwegian Petroleum Directorate's FactPages as Semantic Web Data". In: *The Semantic Web – ISWC 2013*. Ed. by H. Alani et al. Vol. 8219. LNCS. 2013.

Hunting for Inconsistencies in Multilingual DBpedia with QAKiS

Elena Cabrio, Julien Cojan, Serena Villata, and Fabien Gandon

INRIA Sophia Antipolis, France
{firstname.lastname}@inria.fr

Abstract. QAKiS, a system for open domain Question Answering over linked data, allows to query DBpedia multilingual chapters with natural language questions. But since such chapters can contain different information w.r.t. the English version (e.g., more specificity on certain topics, or fill information gaps), *i)* different results can be obtained for the same query, and *ii)* the combination of these query results may lead to inconsistent information about the same topic. To reconcile information obtained by distributed SPARQL endpoints, an argumentation-based module is integrated into QAKiS to reason over inconsistent information sets, and to provide a unique and motivated answer to the user.

1 Introduction

In the Web of Data, the combination of the information items concerning a single real-world object coming from different data sources, e.g., the results of a single SPARQL query on different endpoints, may lead to an inconsistent results set, mining the overall quality of the data itself. In particular, this problem arises while querying DBpedia multilingual chapters, since different information can be provided for the same query (e.g. answers can be either identical, contradictory, or one can subsume the other). To reconcile information provided by multilingual DBpedia chapters to obtain a consistent results set, we embed an argumentation module in QAKiS, that *i)* detects the semantic relations linking each piece of information to the others returned by the different SPARQL endpoints, and *ii)* adopts *abstract bipolar argumentation theory* to reason over the inconsistencies among the answers, and to return a consistent (sub)set of them to the user.

An abstract bipolar argumentation framework (BAF) [2] represents a negative relation between elements called *arguments* through a binary *attack* relation, and a positive relation among arguments through a binary *support* relation. Argumentation semantics then allow to reason about the arguments and their relations to detect the set of *accepted* arguments, i.e., those considered as believable by an external evaluator with full knowledge of the BAF. However, such kind of crisp evaluation of the arguments is not suitable for the real life scenarios where a numerical value is required. This is why we adopt and extend the fuzzy labeling algorithm proposed in [3] to consider also the support relation in addition to the attack one.

The overall argumentation framework together with the acceptability degree of each argument is used to motivate to the user the answer the system returns.

2 Extending QAKiS to reason over inconsistent answers

QAKiS (Question Answering wiKiFramework-based System)[1] [1] addresses the task of QA over structured knowledge-bases (e.g., DBpedia), where the relevant information is expressed also in unstructured forms (e.g., Wikipedia pages). It implements a relation-based match for question interpretation, to convert the user question into a query language (e.g., SPARQL). More specifically, it makes use of relational patterns (automatically extracted from Wikipedia), that capture different ways to express a certain relation in a given language. QAKiS is composed of four main modules (Fig. 1): *i)* the **query generator** takes the user question as input, generates the typed questions, and the SPARQL queries from the retrieved patterns; *ii)* a **Named Entity (NE) Recognizer**; *iii)* the **pattern matcher** takes as input a typed question, and retrieves the patterns matching it with the highest similarity; and *iv)* the **SPARQL package** handles the queries to DBpedia. QAKiS targets questions containing a NE related to the answer through one ontological property, i.e., questions match a single pattern.

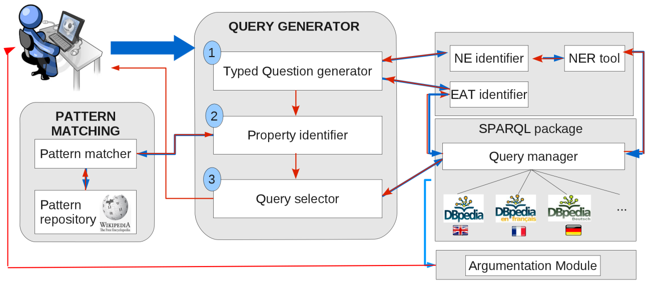


Fig. 1: QAKiS workflow

Given the answers retrieved by DBpedia multilingual endpoints for a SPARQL query, the argumentation module assigns a support or attack relation between the arguments (see Fig 2): *i)* **identity [assigned relation: support]**: if two endpoints provide identical answers (Fig. 2a-b where both French and English DBpedia SPARQL endpoints provide *Italy* as answer to *Where is the Colosseum located?*; *sameAs* links are used to recognize the translation of the same word in multilingual DBpedia). Arguments are merged into a unique one becoming highly acceptable as shared by several sources[2]; *i)* **subsumption [assigned relation: support]**, when one of the answers is more specific than the other, both in terms of spacial relation (Fig. 2d) and hyperonymy (Fig. 2c where *Gibson* is_a *Guitar*)[3]; *iii)* **conflict [assigned relation: attack]**, if the answers are different,

---

[1] `http://qakis.org/qakisArgumentation`

[2] The starting confidence score of this argument is calculated as the *arctangent* of the confidence scores of the endpoints providing such answer (max value = 1).

[3] External sources of semantic knowledge are exploited, e.g., GeoNames, YAGO.

and there is no subsumption (Fig. 2e-f where the locations of the Colosseum by Italian and English DBpedia are contradictory). When each endpoint provides a *list* of values as answer (e.g., DBpedia non-functional properties, Fig. 2g), QAKiS does not consider arguments of the same list as conflictual.

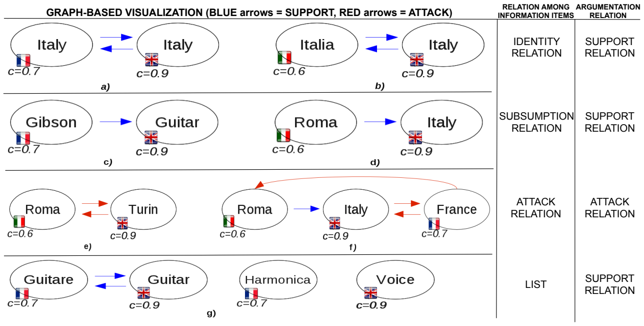
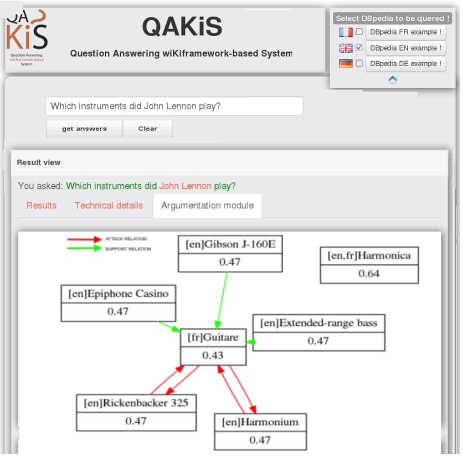


Fig. 2: Semantic relations and their mapping in argumentation

We assign an apriori confidence score to the endpoints according to their dimensions and solidity in terms of maintenance (other methods are under investigation). Starting from the obtained set of arguments and relations, the module calculates the arguments' acceptability degree (i.e., the arguments that will be proposed to the user as more reliable). We propose a bipolar fuzzy labeling algorithm where $\mathcal{A}$ is a fuzzy set of trustful arguments, and $\mathcal{A}(A) = \max_{s \in \mathrm{src}(A)} \tau_s$ is the membership degree of argument A in $\mathcal{A}$ given by the trust degree of the most reliable source offering argument A, where τ_s is the degree to which source $s \in \mathrm{src}(A)$ is evaluated as reliable. A bipolar fuzzy labeling is a total function $\alpha : \mathcal{A} \rightarrow [0,1]$. We say that α is a bipolar fuzzy labeling *iff*, for all arguments A, $\alpha(A) = \mathtt{avg}\{\min\{\mathcal{A}(A), 1 - \max_{B:B \xrightarrow{attack} A} \alpha(B)\}; \max_{C:C \xrightarrow{support} A} \alpha(C)\}$. $\alpha(A) = 0$ means that A is outright unacceptable, $\alpha(A) = 1$ means A is fully acceptable. All cases in-between provide the degree of the acceptability of the arguments which are considered accepted at the end, if they overcome a certain threshold. The result of the fuzzy labeling is the arguments confidence score.



☐ DBpedia FR example !
☑ DBpedia EN example !
☐ DBpedia DE example !

Which instruments did John Lennon play?

You asked: Which instruments did John Lennon play?

[en]Gibson J-160E 0.47
[en,fr]Harmonica 0.64
[en]Epiphone Casino 0.47
[en]Extended-range bass 0.47
[fr]Guitare 0.43
[en]Rickenbacker 325 0.47
[en]Harmonium 0.47

Fig. 3: QAKiS demo interface.

Figure 3 shows the QAKiS demo interface. The user can select the DBpedia chapter he wants to query besides English, i.e. French or German DBpedia (top right corner) [1]. Then the user can either write a question or select among a list of examples. QAKiS outputs *i)* the user question, *ii)* the generated typed question, *iii)* the pattern matched, *iv)* the generated SPARQL query, *v)* the answer, and *vi)* the graph of the answers by the different endpoints and their relations, together with their confidence score.

Since QAKiS currently targets only questions containing a NE related to the answer through one ontological property, we extracted from QALD-2[4] data the questions corresponding to such criterion, i.e. 58 questions, and we run them over QAKiS (querying English, German and French DBpedia endpoints). Since QALD-2 questions are created for English DBpedia, only in 25/58 cases there are at least two endpoints that provide an answer. We carried out two sets of experiments. In Experiment 1 (input: the answers obtained from the different DBpedia endpoints, manually creating the SPARQL query), performances of the argumentation module in identifying the arguments from the endpoints are F-meas. 0.97, in relation assignment are F-meas. 0.72. Errors in arguments identification are due to missing *SameAs* links in DBpedia: the algorithm does not merge translations of the same answer, and it considers them as different. Wrong relation assignments are mainly due to missing attacks among arguments.

Since QAKiS performances are about ∼50%, the results of Experiment 2 (submitting natural language questions to QAKiS) are obtained accordingly, F-meas 0.72 for argument identification and F-meas 0.55 for relation assignment (the argumentation module is biased by QAKiS mistakes). The average computation cost of the argumentation module is ∼5s for 1-answer, and ∼125s for n-answers questions. The complexity is quadratic, at least one SPARQL query is sent for each couple of answers. We are working on the algorithm optimization.

3 Future perspectives

Extensions are planned in several directions: *i)* to let the user assign the confidence degree to the information sources embedding this feature in the QAKiS interface; *ii)* extend the set of ontologies we consider to detect further relations (positive and negative) among the information items; *iii)* perform a user evaluation campaign to verify which kind of visualization is better usable.

References

1. Cabrio, E., Cojan, J., Gandon, F., , Hallili, A.: Querying multilingual DBpedia with QAKiS. In: Procs of ESWC 2013. Demo papers. (2013)
2. Cayrol, C., Lagasquie-Schiex, M.C.: Bipolarity in argumentation graphs: Towards a better understanding. In: Procs of SUM 2011, pp. 137–148. LNCS, v. 6929 (2011)
3. da Costa Pereira, C., Tettamanzi, A., Villata, S.: Changing one's mind: Erase or rewind? In: Procs of IJCAI2011. pp. 164–171. IJCAI/AAAI (2011)

---

[4] Question Answering Linked Data challenge: `http://bit.ly/QALD2`

Demo: Swip, a Semantic Web Interface using Patterns

Camille Pradel, Ollivier Haemmerlé, and Nathalie Hernandez

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

Abstract. Our purpose is to provide end-users with a means to query ontology based knowledge bases using natural language queries and thus hide the complexity of formulating a query expressed in a graph query language such as SPARQL. The main originality of our approach lies in the use of query patterns. Our contribution is materialized in a system named *SWIP*, standing for *Semantic Web Interface Using Patterns*. The demo will present use cases of this system.

1 Introduction

End-users need to access the huge amount of data available through the Internet. With the development of RDF triplestores and OWL ontologies, a need for interfacing SPARQL engines emerged, since it is impossible for an end-user to handle the complexity of the "schemata" of these pieces of knowledge. We think that the availability of voice recognition software which are becoming more and more popular, especially on smartphones, implies that we now have to work on the translation of NL queries into formal queries. The main hypothesis behind our work states that, in real applications, the submitted queries are variations of a few typical query families. We propose to guide the interpretation process by using predefined query patterns which represent these query families. The process benefits from the pre-established families of frequently expressed queries for which we know that real information needs exist.

In [1], we proposed a way of building queries expressed in terms of conceptual graphs from user queries composed of keywords. In [2] we extended the system in order to take into account relations expressed by the user between the keywords he/she used in his/her query and we introduced the pivot language allowing these relations to be expressed in a way inspired by keyword queries. In [3], we adapted our system to the Semantic Web languages instead of Conceptual Graphs. Such an adaptation was important for us in order to evaluate the interest of our approach on large and actual knowledge bases. The Swip system also participated in the first and third editions of the *Question Answering over Linked Data* (QALD) challenge. The results of this participation are detailed in [4].

This article gives a brief overview of our approach and presents its implementation which will be demonstrated.

2 Swip system overview

In the SWIP system, the query interpretation process is made of two main steps: the translation of the NL user query into a pivot query, and the formalization of this pivot query, respectively described in subsections 2.1 and 2.2.

2.1 From natural language to pivot query

The whole process of interpreting a natural language query is divided into two main steps, with an intermediate result, which is the user query translated into a new structure called the *pivot query*. This structure is half way between the NL query and the targeted formal query, and can be expressed through a language, called pivot language, which is formally defined in [3]. Briefly, this structure represents a query made of keywords and also expresses relations between those keywords. We use this pivot language in order to facilitate the implementation of multilingualism by means of a common intermediate format: a specific module of translation of NL to pivot has to be written for each different language, but the pivot query formalization step remains unchanged. This translation step is detailed in [4]. It consists of four stages.

The first stage aims at identifying in the NL query named entities corresponding to knowledge base resources; this allows these entities to be considered as a whole and prevents the parser from separating them in the next stage. Then, in the second stage, a dependency tree of the user NL query is processed by a dependency parser, taking into account the previously identified named entities. The third stage aims at identifying the query focus, i.e. the element of the query for which the user wants results (the element corresponding to the variable which will be attached to the SPARQL `SELECT` clause); SWIP is also able to detect *count queries* which ask for the number of resources fulfilling certain conditions and correspond in SPARQL to a `SELECT` query using a `COUNT` aggregate as a projection attribute, and *dichotomous (or boolean) queries* which allow only two answers (True or False / Yes or No), and are expressed in SPARQL with an `ASK` query. Finally, a set of predefined rules are applied to the dependency graph in order to obtain the elements of the pivot query and their relations.

2.2 From pivot to formal query

Formalizing pivot queries using query patterns was the first task we tackled and is extensively described in [2] and [3]. We briefly describe the structure of a query pattern and the process of this formalization.

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

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

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

3 Implementation and evaluation

A prototype of our approach was implemented in order to evaluate its effectiveness. It is available at `http://swip.univ-tlse2.fr/SwipWebClient`. It was implemented in Java and uses the MaltParser[1] for the dependency analysis of English user queries. The system performs the second main process step (translating from pivot to formal query) by exploiting a SPARQL server based on the ARQ[2] query processor, here configured to exploit LARQ[3], allowing the use of Apache Lucene[4] features, such as indexation and Lucene score (used to obtain the similarity score between strings).

Experiments were carried out on the evaluation framework proposed in task 1 of the QALD-3 challenge[5]. A detailed analysis of the results is available in [4]. The evaluation method was defined by the challenge organizers. It consists in calculating, for each test query, the precision, the recall and the F-measure of the SPARQL translation returned by the system, compared with handmade queries of a gold standard document. We participated in both subtasks proposed by the challenge organizers, one targeting the DBpedia[6] knowledge base and the other targeting an RDF export of Musicbrainz[7] based on the music ontology[8]. The quality of the results varies with the target KB.

---

[1] http://www.maltparser.org/

[2] http://openjena.org/ARQ/

[3] LARQ = Lucene + ARQ, see http://jena.sourceforge.net/ARQ/lucene-arq.html

[4] http://lucene.apache.org/

[5] http://greententacle.techfak.uni-bielefeld.de/~{}cunger/qald/index.php?x=task1&q=3

[6] http://dbpedia.org

[7] http://musicbrainz.org/

[8] http://musicontology.com/

On the Musicbrainz test dataset, we processed 33 of the 50 test queries. 24 were correctly interpreted, 2 were partially answered and the others failed. The average precision, recall and F-measure, calculated by the challenge organizers, are all equal to 0.51. We consider these results as quite good and very encouraging. However, results on the DBpedia dataset are more disappointing. We processed 21 of the 100 test queries, of which 14 were successful, 2 were partially answered and 5 were not correct. The average precision, recall and F-measure are all equal to 0.16.

The proposed demo will showcase a didactic user interface which displays results of intermediate steps. The target dataset will be Musicbrainz and the target language English. It will also be possible to visualize and edit query patterns through a control panel in order to influence the interpretation process.

4 Conclusion and future work

In this paper, we presented the approach we are designing to allow end users to query graph-based knowledge bases. This approach is implemented in the SWIP system and is mainly characterized by the use of query patterns in the interpretation of the user NL query. The setting up of the two main parts of the system process is nearly done and the first results are very encouraging.

We plan to extend our work in several directions: experimenting the ease of adaptation to different user languages, by participating to the multilingual task of the QALD challenge, and collaborating with IRSTEA (the French institute of ecology and agriculture) in order to build a real application framework concerning French queries on organic farming; experimenting methods to automate or assist the conception of query patterns; extending the query that can be processed by our system, for example by taking into account extensions of SPARQL 1.1, such as aggregates; exploring new leads allowing the approach to evolve and stick more to the data itself than to the ontology, in order to obtain better results on datasets from the Web of linked data, such as DBpedia.

References

1. Comparot, C., Haemmerlé, O., Hernandez, N.: An easy way of expressing conceptual graph queries from keywords and query patterns. In: ICCS. pp. 84–96 (2010)
2. Pradel, C., Haemmerlé, O., Hernandez, N.: Expressing conceptual graph queries from patterns: how to take into account the relations. In: Proceedings of the 19th International Conference on Conceptual Structures, ICCS'11, Lecture Notes in Artificial Intelligence # 6828. pp. 234–247. Springer, Derby, GB (July 2011)
3. Pradel, C., Haemmerlé, O., Hernandez, N.: A semantic web interface using patterns: The swip system (regular paper). In: Croitoru, M., Rudolph, S., Wilson, N., Howse, J., Corby, O. (eds.) IJCAI-GKR Workshop, Barcelona, Spain, 16/07/2011-16/07/2011. pp. 172–187. No. 7205 in LNAI, Springer, http://www.springerlink.com (mai 2012)
4. Pradel, C., Peyet, G., Haemmerlé, O., Hernandez, N.: Swip at qald-3: results, criticisms and lesson learned (working notes). In: CLEF 2013, Valencia, Spain, 23/09/2013-26/09/2013 (2013)

Using Ontologies to Identify Patients with Diabetes in Electronic Health Records

Hairong Yu, Siaw-Teng Liaw, Jane Taggart, and Alireza Rahimi Khorzoughi

School of Public Health & Community Medicine and Research Centre for Primary Health Care & Equity, Faculty of Medicine, University of New South Wales, Sydney, NSW 2052, Australia

{hairong.yu,siaw,j.taggart}@unsw.edu.au
alireza.rahimikhorzoughi@student.unsw.edu.au

Abstract. This paper describes a work in progress that explores the applicability of ontologies to solve problems in the medical domain. We investigate whether it is feasible to use ontologies and ontology-based data access (OBDA) to automate common clinical tasks faced by general practitioners (GPs), which are labor-intensive and error prone in terms of relevant information retrieved from electronic health records (EHRs). Our study aims to improve the selection of diabetes patients for clinical trials or medical research. The biggest impediment to automating such clinical tasks is the essential bridging of the semantic gaps between existing patient data in EHRs, such as reasons for visit, chronic conditions and diagnoses, pathology tests and prescriptions stored in general practice EHRs (GPEHR), and the ways which medical researchers or GPs interpret those records. Our current understanding is that automated identification of diabetes patients can be specified systematically as a solution supported by semantic retrieval. We detail the challenges to building a realistic case study, which consists of solving issues related to conceptualization of data and domain context, integration of different datasets, ontology creation based on the SNOMED CT-AU® standard, mapping between existing data and ontology, and the challenge of data fitness for research use. Our prototype is based on data which scale to thirteen years of approximately 100,000 anonymous patient records from four general practices in south western Sydney.

Keywords: Ontology, Diabetes Mellitus, Electronic Health Records, eHealth, Knowledgebase Management, Ontology_Based Database Access

1 Introduction

This paper reports on work that explores the applicability of ontologies for solutions in the health domain. In Australia, the main health applications of ontologies appear to be the SNOMED terminology services. We investigated the feasibility of the use of ontologies and OBDA to automate clinical tasks, such as identifying patients with specific diabetes mellitus (DM) phenotypes in EHRs contributing to the data repository of the electronic Practice Based Research Network (ePBRN). The ePBRN is used to conduct translational research on primary and integrated care, including tracking patients, managing chronic disease, and providing quality evidence-based care. The biggest barrier to automating the clinical task of identifying a patient with DM is the semantic gaps between patient data in EHRs, such as reasons for visit, diagnoses, pathology tests and prescriptions, and how these EHR data are interpreted. For example, in addition to a diagnostic label, DM can be implied by a blood glucose test with suggestive levels of diabetes, certain medications such as oral hypoglycaemics or insulin, or the use of DM supplies such as glucose diagnostic strips. By using ontologies, our experiments show that it is possible to automate this interpretation process and build a reusable conceptual infrastructure over diverse standards or experience or datasets. Currently most efforts at automation is only limited within individual clinics or in a physician-driven process or at data levels.

The SNOMED CT-AU®, the Australian extension to SNOMED CT® (Systematized Nomenclature Of Medicine Clinical Terms), is an ontology which formally defines classes of medical procedure, pharmaceutical or biologic product, and body structure and so on. The SNOMED CT-AU® Ontology (SCAO) is the reference terminology for EHRs in Australia. SCAO is available in Web Ontology Language (OWL) format from the Australian National E-Health Transition Authority (NEHTA). Our experiments showed that the integration of SNOMED CT-AU and the Diabetes Identification Ontology (DIO) based on ePBRN data to select patients with DM is well suited for our case study. Our key approach is that the automation of the process of identifying DM patients is an issue of semantic retrieval, i.e. selection criteria can be expressed as semantic queries, which are processed by a reasoner to retrieve explicit information on eligible patients from datasets and infer implicit knowledge from ontologies simultaneously.

The objective of this study is to assess the practicality and utility of ontologies in a real world environment. The technical challenges of conceptualization of data and domain context, ontology integration of different datasets or ontologies, mapping between existing datasets and ontologies, and finding solutions to ensure data fitness for clinical or research use will be described and discussed in the following sections.

2 Methodology

The architecture for this study comprises six parts separated by dashed lines as shown in Figure 1. Patient data were extracted from individual GPEHRs, e.g. Medical Director™[1] at each clinic by GRHANITE™[2]. The software provides a data repository over server called GRHANITE™ Databank, in our case the ePBRN repository operated by MS SQL Server™. The ABox, associated with instances of ontology classes or properties, is populated through ontopPro (formerly known as an OBDA plugin for Protégé[3]). Another primary component in our knowledgebase, the TBox, related to concep-

---

[1] http://www.hcn.com.au/Products/Medical+Director

[2] http://www.grhanite.com/

[3] http://protege.stanford.edu/

tual terminologies defined in ontologies, is built through Protégé, a popular open source ontology editor and knowledgebase framework.

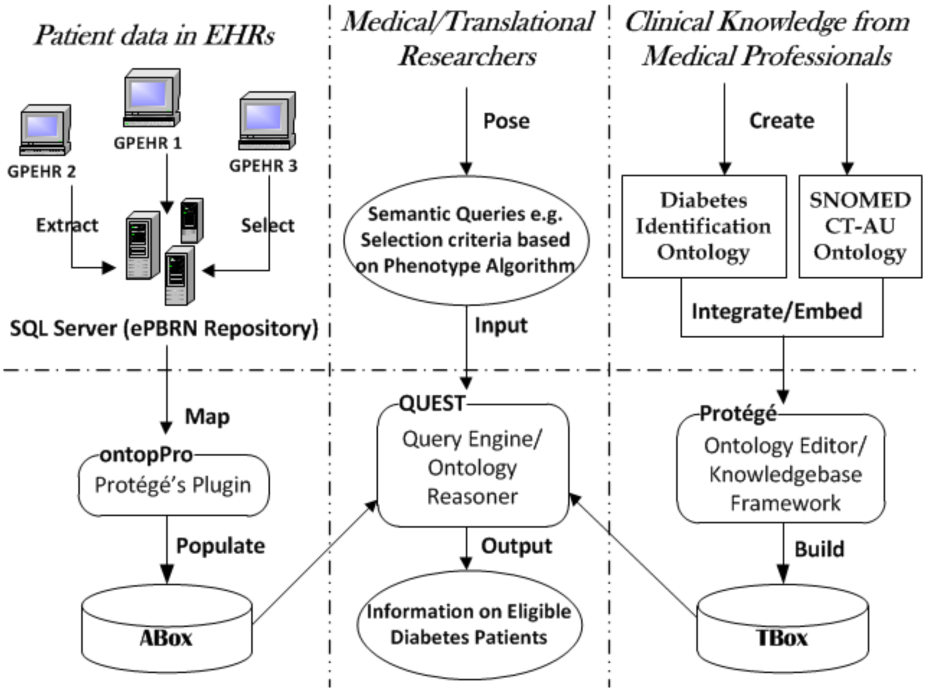

Patient data in EHRs
Medical/Translational Researchers
Clinical Knowledge from Medical Professionals
GPEHR 1
GPEHR 2
GPEHR 3
Extract
Select
SQL Server (ePBRN Repository)
Pose
Semantic Queries e.g. Selection criteria based on Phenotype Algorithm
Input
Create
Diabetes Identification Ontology
SNOMED CT-AU Ontology
Integrate/Embed
Map
ontopPro
Protégé's Plugin
Populate
ABox
QUEST
Query Engine/ Ontology Reasoner
Output
Information on Eligible Diabetes Patients
Protégé
Ontology Editor/ Knowledgebase Framework
Build
TBox

Fig. 1. ePBRN Diabetes Identification Case Study Solution Architecture

Clinical selection criteria are formulated as semantic queries in SPARQL Protocol and RDF Query Language (SPARQL). The SPARQL query engine QUEST[4] that comes with ontopPro[5] checks the queries against the knowledgebase to retrieve matched patients. A demonstration is given for each of six parts in Figure 1.

The first step in building this solution is creating the specific DIO in hierarchical conceptual modeling, based on the Australian National Guidelines for Type 2 Diabetes Mellitus (T2DM) and discussions with the research team and GPs participating in the ePBRN. The output of this first task is a formalized ontology which consists of 4 main classes Actor, Content, Mechanism and Impact and 68 subclasses with object/data properties. Some of them can be mapped to the SNOMED CT-AU Ontology (SCAO), which has more than 300,000 concepts.

Due to the small number of concepts captured by the DIO, the mapping can be operated manually. For example, T2DM is a Disease under the subclass of Problem which has a superclass Context in DIO. In the SCAO, T2DM is a disorder of glucose metabolism which is a subclass of Disease under the highest level concept of Clinical

---

[4] http://semanticweb.org/wiki/Quest/

[5] http://ontop.inf.unibz.it/

finding. Similarly, Actor class in DIO corresponds to Environment or Geographical location in SCAO. However the automation of integration of two ontologies can be complex for large terminologies.

Next we linked the server objects in SQL Server to integrate other heterogeneous datasets by T-SQL™. The SQL query results are mapped by ontopPro for ABox associated with relevant classes in ontologies. This meant that the schematic or semantic heterogeneity challenges faced were solved at either data or ontology level. The mapping mechanism supplied by ontopPro theoretically based on OBDA [1], provided a big advantage on populating class members, assigning property values, and incorporating schematic data in the ePBRN repository with semantic concepts in ontologies. The raw data in EHRs that contribute to the ePBRN repository are incomplete, incorrect and inconsistent (against external standards or internal logic perspectives). We used definitions of properties in DIO or mappings created in ontopPro to solve core data quality issues before preparation of semantic queries.

We then wrote semantic queries in SPARQL according to requirements from domain experts, and ran them through QUEST, the query engine and OWL reasoner. The query results are expected to identify DM patients and help clinicians to manage the cycle of care for the cohort. The SPARQL queries were validated using SQL over an artificial dataset of 100 patients schematically similar to the ePBRN dataset. The approach that we developed and tested on the artificial dataset will be scalable to the ePBRN repository of more than 100,000 patient records. Other use case scenarios, for example assisting researchers to conduct association and/or controlled studies will contribute to the validation of the architecture.

3 Discussion and Conclusion

We have briefly presented a feasibility study of the use of ontologies to detect patients with DM in real world EHRs. Using real patient datasets, we solved some engineering challenges around ontology creation and integration, bridging between ontologies and datasets, and data quality [2]. Apart from usability, interoperability and scalability aforementioned, other quality attributers are assessed closely for architecture evaluation for instance, modifiability with many facades/locations where data/data types are transferred in our solution, integrability and extensibility which are especially critical as several open source software components are used in our design.

References

1. M. Lenzerini, Data Integration: A Theoretical Perspective, Proc. of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS'02), pp. 233 – 246.
2. S.-T. Liaw, et al. Data quality and fitness for purpose of routinely collected data – a general practice case study from an electronic Practice-Based Research Network, in: AMIA Annual Symposium Proceedings, 2011:785–794.

Monitoring the Status of SPARQL Endpoints

Pierre-Yves Vandenbussche[1], Carlos Buil Aranda[2],
Aidan Hogan[3], and Jürgen Umbrich[1]

[1] Fujitsu (Ireland) Limited, Swords, Co. Dublin, Ireland
[2] Department of Computer Science, Pontificia Universidad Católica de Chile
[3] Digital Enterprise Research Institute, National University of Ireland, Galway

Abstract. We demo an online system that tracks the availability of over four-hundred public SPARQL endpoints and makes up-to-date results available to the public. Our demo currently focuses on how often an endpoint is online/offline, but we plan to extend the system to collect metrics about available meta-data descriptions, SPARQL features supported, and performance for generic queries.

1 Motivation

In previous work [2], we presented an analysis of the landscape of public SPARQL endpoints and asked the question: are these endpoints ready for action?[4] Taking the full list of 427 public endpoints from the CKAN/DataHub catalogue (as available at the time of writing), for each endpoints, we conducted a number of experiments to gauge the following four main aspects:

Discoverability: What kinds of meta-data descriptions are available about the endpoints and their content? How easy are these descriptions to find?

Interoperability: Which SPARQL (1.1) features does each endpoint support? Which features (or combinations of features) lead to exceptions?

Efficiency: How do the endpoints perform for answering generic forms of queries? How is cold-cache performance vs. warm-cache performance? What is the latency like over HTTP?

Availability: What are the average uptimes of the endpoints? How many endpoints are dying/have died? How many endpoints have high reliability?

Our results showed that about half of the endpoints listed on CKAN/-DataHub are now offline, that only a few endpoints make meta-data descriptions available about their content (VoID) or features supported (SPARQL 1.1 Service Descriptions) in easy-to-find locations, that there was mixed adoption

---

This work was supported by Fujitsu (Ireland) Ltd. & by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2). Carlos Buil-Aranda was supported by CONICYT/FONDECYT project No. 3130617.

[4] This work is accepted for the Experiments track of ISWC 2013 [2]. This demo paper rather focuses on our tool for making results available to the community.

of SPARQL and (recently standardised) SPARQL 1.1 features, that the performance of different endpoints over HTTP for generic queries could vary by orders of magnitude, and that less than one third of the endpoints had an average availability in the interval 99–100% (i.e., at least two-nines availability). We concluded that the usability of different public endpoints varies greatly.

We thus propose a system that tracks and collects metrics about public endpoints over time. Currently, our service tracks the hourly availability of endpoints, and we plan to extend it to collect weekly metrics about the available meta-data, supported features and performance of these endpoints, as well as other metrics that the community may wish to suggest.

In Section 2, we first discuss our current "SPARQL Endpoint Status" system, available online at `http://labs.mondeca.com/sparqlEndpointsStatus/`. Thereafter, in Section 3, we discuss our proposed extensions.

2 SPARQL Endpoint Status

Monitoring Availability The system automatically collects and updates a list of public SPARQL endpoints from the CKAN/DataHub catalogue. These endpoints are queried on an hourly basis using two alternative SPARQL queries:

```
ASK WHERE{ ?s ?p ?o . }
```

```
SELECT ?s WHERE{ ?s ?p ?o . } LIMIT 1
```

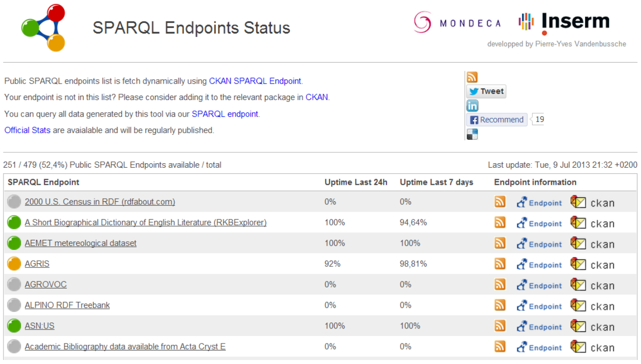
The `ASK` query on the left is issued first. If this query fails (from previous experience, we note that some endpoints do not support `ASK` [2, § 3]), we try the `SELECT` query on the right. Both queries are selected at they should be as cheap as possible for the endpoint to run: our goal is simply to check whether or not the endpoint is available for answering queries. If the endpoint returns a valid SPARQL response for either query, we then say that the endpoint is available at that timepoint. We also record the time taken for the query to execute.

At the time of writing, we have collected more than two million hourly pings across hundreds of endpoints over a period of more than two years. Detailed analysis of these availability results is available in [2, § 5].

User Interface We provide a user interface to browse and visualise the hourly results. The user interface supports two primary views.

The first view, exemplified in Figure 1, provides a full list of all the monitored endpoints, their availability in the past 24 hours (ratio of successful hourly queries in that period), and their availability in the past seven days. A green/yellow/red/gray icon indicates, resp., that the endpoint is operating normally/available but had problems in the past 24 hours/not available currently/not available once in the past 24 hours. As per the icons listed on the right of the screenshot, each endpoint is also associated with (1) an RSS feed to provide updates on availability information, (2) a link to the endpoint itself and (3) a link to the relevant CKAN/DataHub page for the dataset it relates to.

The second view provides details for a given endpoint. Figure 2 shows an example screenshot for the DBpedia endpoint. The graph on the left shows the



Public SPARQL endpoints list is fetch dynamically using CKAN SPARQL Endpoint.
Your endpoint is not in this list? Please consider adding it to the relevant package in CKAN.
You can query all data generated by this tool via our SPARQL endpoint.
Official Stats are avaialable and will be regularly published.

251 / 479 (52,4%) Public SPARQL Endpoints available / total
Last update: Tue, 9 Jul 2013 21:32 +0200

SPARQL Endpoint	Uptime Last 24h	Uptime Last 7 days	Endpoint information
2000 U.S. Census in RDF (rdfabout.com)	0%	0%	Endpoint ckan
A Short Biographical Dictionary of English Literature (RKBExplorer)	100%	94,64%	Endpoint ckan
AEMET metereological dataset	100%	100%	Endpoint ckan
AGRIS	92%	98,81%	Endpoint ckan
AGROVOC	0%	0%	Endpoint ckan
ALPINO RDF Treebank	0%	0%	Endpoint ckan
ASN:US	100%	100%	Endpoint ckan
Academic Bibliography data available from Acta Cryst E	0%	0%	Endpoint ckan

Fig. 1. Screenshot of current SPARQL Endpoint Status list

response times for the last 24 hourly pings to that endpoint. The graph on the right plots the 24 hour availability for each of the last seven days.

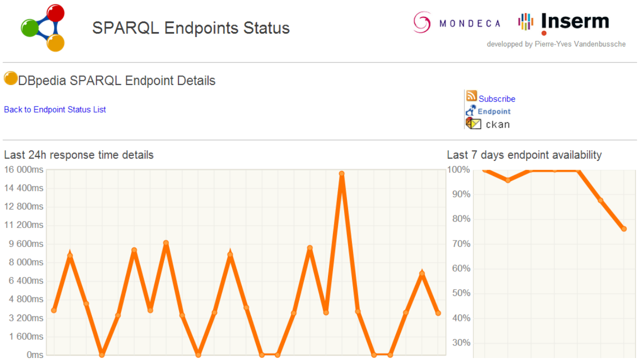

SPARQL Endpoints Status
MONDECA
Inserm
developped by Pierre-Yves Vandenbussche
DBpedia SPARQL Endpoint Details
Back to Endpoint Status List
Subscribe
Endpoint
ckan
Last 24h response time details
16 000ms
14 400ms
12 800ms
11 200ms
9 600ms
8 000ms
6 400ms
4 800ms
3 200ms
1 600ms
0ms
Last 7 days endpoint availability
100%
90%
80%
70%
60%
50%
40%
30%

Fig. 2. Screenshot of current SPARQL Endpoint Status detail view for DBpedia

RDF Meta-data Results of the hourly pings are exported as RDF. Figure 3 presents an example description. We reuse existing vocabularies as much as possible (VoID, dcterms, etc.) to describe each dataset, their related SPARQL endpoint, title and identifier, etc. To capture availability information, we designed a new vocabulary (no existing one handled this feature). The "endpoint status" vocabulary[5] (*ends*) allows the description of a status observation with the information of date, description (we are here reusing dcterms vocabulary), status availability and response time. All RDF data are then published in a SPARQL Endpoint available at: `http://labs.mondeca.com/endpoint/ends`.

---

[5] `http://labs.mondeca.com/vocab/endpointStatus/`

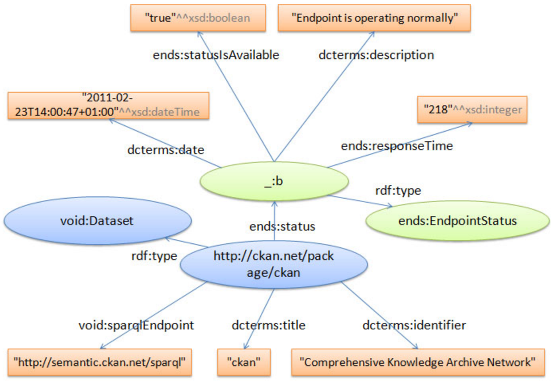


Fig. 3. Schema used to express SPARQL endpoint availability in RDF

3 Future Extensions

Our system currently captures endpoint availability and query latency. In line with the discussion of Section 1 and the methods of our experimental paper [2], we wish to extend our system to track more metrics about public endpoints. These would include: (1) what meta-data descriptions about each endpoint/-dataset are available and where (e.g., VoID, SPARQL 1.1 SD), (2) what query features each endpoint supports (e.g., SPARQL 1.1, full-text), (3) what performance can be expected for generic queries (atomic lookups, dump queries, controlled joins). Since the queries are more expensive to run, we propose running them on a weekly basis to not overburden endpoints. We would then extend our UI and RDF vocabulary to make these metrics available. We are very much open to suggestions/use-cases from the community for collecting further metrics. Furthermore, we are considering making a locally deployable version for clients to monitor endpoints of relevance to them.

Acknowledgements: *This paper was supported by Fujitsu (Ireland) Limited, and funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2). Carlos Buil-Aranda was supported by the CONICYT/FONDECYT project No. 3130617.*

References

1. K. Alexander, R. Cyganiak, M. Hausenblas, and J. Zhao. Describing linked datasets. In *LDOW*, 2009.
2. C. B. Aranda, A. Hogan, J. Umbrich, and P.-Y. Vandenbussche. SPARQL Web-Querying Infrastructure: Ready for Action? In *ISWC*. Springer (LNCS), 2013. (Accepted; to appear.).
3. G. T. Williams. SPARQL 1.1 Service Description. W3C Recommendation, March 2013.

GetThere: A Rural Passenger Information System Utilising Linked Data & Citizen Sensing

David Corsar, Peter Edwards, Chris Baillie, Milan Markovic,
Konstantinos Papangelis, and John Nelson

dot.rural Digital Economy Hub,
University of Aberdeen, Aberdeen, UK
`{dcorsar,p.edwards,c.baillie,m.markovic,k.papangelis,j.d.nelson}@abdn.ac.uk`
`http://www.dotrural.ac.uk`

Abstract. This demo paper describes a real-time passenger information system based on citizen sensing and linked data.

Keywords: provenance, data quality, citizen sensing, linked data, semantic infrastructure, transport

1 Introduction

Real-time passenger information (RTPI) systems provide details about public transport, allowing passengers to plan and make decisions regarding their journeys. Typical requirements for RTPI systems include: 1) listing available public transport services; 2) providing timetable (schedule) information for those services; 3) providing (real-time) vehicle locations; and 4) providing details of disruptions. However, few RTPI systems exist in rural areas for a variety of reasons, including a lack of infrastructure for obtaining and providing real-time information [7]. As part of the Informed Rural Passenger project[1], we are developing *GetThere*, an RTPI system for rural areas. The *GetThere* system consists of a smartphone app, supported by a semantic infrastructure that integrates data from multiple sources (including users). This system has been deployed in the Scottish Borders, UK in partnership with First Group.

This demonstration[2] will show a typical use of the *GetThere* app to view timetabled and real-time vehicle locations for a selected route, contribute vehicle locations while making a journey, report a disruption event, and assess the quality of real-time locations with and without the presence of disruption. The demo will utilises the datasets and services shown in Fig. 1.

2 Information Ecosystem

GetThere is supported by a semantic information ecosystem (Fig. 1) itself underpinned by a series of ontologies. Semantic web and linked data technologies are

---

[1] `http://www.dotrural.ac.uk/irp`

[2] A video of the demo is available at `http://www.gettherebus.com/iswcdemo/`

used for data representation and storage within the ecosystem as they provide an effective approach for large scale data integration [4]. Further, accessing and storing data via SPARQL endpoints allows storage to be handled by technologies appropriate for the characteristics of individual datasets; for example, using RDF streams or a database with a R2RML wrapper for high throughput data.

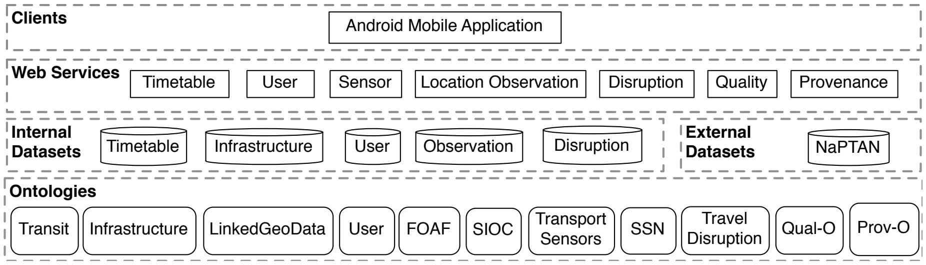


Fig. 1. Real-time passenger information ecosystem.

Details of public transport services and timetables are stored in the Timetable dataset[3] and represented by the Transit ontology[4]. This dataset is used by the Timetable Service to provide details of available transport services, timetable, and vehicle location information to the *GetThere* app. The Infrastructure dataset provides details of the road networks used by public transport vehicles. This data is extracted from openstreetmap.org, and is represented using the Infrastructure ontology[5] (which defines bus route maps) and the LinkedGeoData[6] ontology. NaPTAN[7] provides details of bus stops, including their IDs and locations.

The User ontology[8] and dataset describe user profiles using SIOC[9] and FOAF[10], a description of each user's msobile device(s), along with details of public transport journeys made while using the *GetThere* app. The Observation dataset uses the Transport Sensors ontology[11] which extends the W3C Semantic Sensor Network (SSN) ontology[12] to describe observations (e.g. of vehicle occupancy level, vehicle location) obtained from users of the *GetThere* app. The Sensor service provides an API for storing and retrieving sensor and observation

---

[3] Timetable information is received in the ATCO-CIF format (`http://www.travelinedata.org.uk/CIF/atco-cif-spec.pdf`); the RDF conversion program is available at `https://github.com/dcorsar/ecosystem.timetable`.

[4] `http://vocab.org/transit/terms/`

[5] `http://www.dotrural.ac.uk/irp/uploads/ontologies/infrastructure.owl`

[6] `http://linkedgeodata.org`

[7] `http://data.gov.uk/dataset/naptan`

[8] `http://www.dotrural.ac.uk/irp/uploads/ontologies/user.owl`

[9] `http://rdfs.org/sioc/spec/`

[10] `http://xmlns.com/foaf/spec/`

[11] `http://www.dotrural.ac.uk/irp/uploads/ontologies/sensors.owl`

[12] `http://www.w3.org/2005/Incubator/ssn/ssnx/ssn`

descriptions expressed using the SSN ontology; the Location Observation service handles real-time locations provided by app users. The Travel Disruption ontology describes different types of disruption, based on an analysis of existing travel disruption information sources [5]. Disruption reports from app users are managed and stored by the Disruption service and dataset.

Given the open nature of this data, issues such as data quality and trust naturally arise [6]. Examples range from malicious users and inaccurate devices to out-of-date information (e.g. timetables). As part of addressing these issues, the ecosystem features a service that can evaluate data quality. The quality ontology (Qual-O[13]), and its associated quality assessment service are discussed in detail elsewhere [1]. Briefly, the service is configured with a set of quality metrics encoded as SPARQL rules expressed against the relevant ontologies. These guide a SPIN reasoner [3] to perform quality assessment, producing quality scores which can be utilised by other services to filter low quality data.

Our current quality metrics are focused on real-time locations, and have been developed following several deployments of the system. They include: *Timeliness* - timely observations are under 1 minute old; *Accuracy* - accurate observations have a GPS error margin of less than 25 metres; *Relevance* - relevant observations are no further than 500 metres from the expected route of travel; *Availability* - observations with a high availability score have no more than a 1 minute delay between being created on the device and published by the ecosystem.

The provenance service uses the W3C Prov-O ontology[14] to maintain a record of the entities, agents, and activities involved in producing data within the ecosystem. Uses of provenance include: associating users with location observations generated by their mobile device, which can support detection of potentially malicious users; and recording dataset provenance to ensure the latest timetable information is provided to users [2].

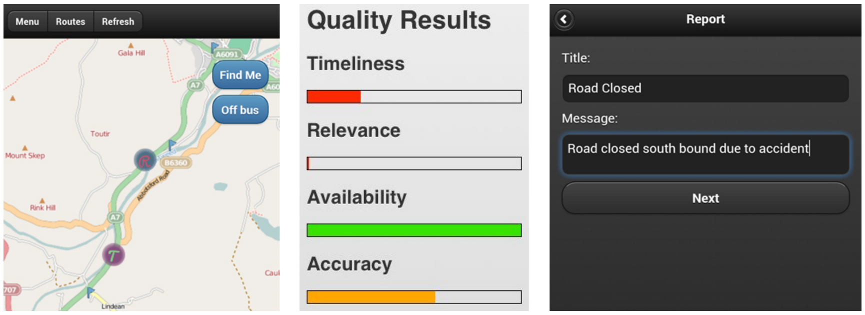
2.1 The *GetThere* Smartphone App

The ecosystem has been designed to support a range of applications through the creation of relevant application services. At present we have used the ecosystem to support the *GetThere* RTPI system, which is provided via an Android smartphone app (see Fig. 2). The app invokes the web services, which execute SPARQL queries against relevant datasets, process the results, and send a response to the app. Users are presented with a list of available bus routes; after selecting a route (and direction, either inbound or outbound), vehicle locations are displayed. These locations include both estimates based on the timetable and real-time locations obtained from other users on that route (Fig. 2, left screenshot). Bus stops along the route are also shown. The user can access timetable information for the previous and next arrivals at a particular stop. When the user boards the bus, they have the option of pressing a button to have their

---

[13] `http://sensornet.abdn.ac.uk/onts/Qual-O.ttl`

[14] `http://www.w3.org/TR/prov-o/`



Title: Road Closed

Message: Road closed south bound due to accident

Fig. 2. Screenshots of the *GetThere* app showing (left to right): vehicle locations; the results of invoking the quality assessment service; and creating a disruption report.

location uploaded to the ecosystem every minute. The uploaded location is then used as the vehicle's real-time location provided to other users.

Users can view quality assessment results for a real-time vehicle location by tapping its icon. We are working with users to determine an appropriate visualisation of quality results. Currently each assessed dimension is shown with a colour-coded bar representing its quality score (Fig. 2, centre screenshot).

Acknowledgements The research described here is supported by the award made by the RCUK Digital Economy programme to the dot.rural Digital Economy Hub; award reference: EP/G066051/1

References

1. C. Baillie, E. Edwards, P. Pignotti, and D. Corsar. Short paper: Assessing the quality of semantic sensor data. In *Proc. of The 6th International Workshop on Semantic Sensor Networks*, page to appear, October 2013.
2. D. Corsar, P. Edwards, N. Velaga, J. Nelson, and J. Pan. Exploring provenance in a linked data ecosystem. In P. Groth and J. Frew, editors, *Provenance and Annotation of Data and Processes*, volume 7525 of *LNCS*, pages 226–228. Springer Berlin Heidelberg, 2012.
3. C. Furber and M. Hepp. Swiqa - a semantic web information quality assessment framework. In *19th European Conference on Information Systems*, pages 922–933, 2011.
4. V. Lopez, S. Kotoulas, M. Sbodio, M. Stephenson, A. Gkoulalas-Divanis, and P. Aonghusa. Queriocity: A linked data platform for urban information management. In *The Semantic Web – ISWC 2012*, volume 7650 of *LNCS*, pages 148–163. Springer Berlin Heidelberg, 2012.
5. M. Markovic, P. Edwards, D. Corsar, and J. Pan. Demo: Managing the provenance of crowdsourced disruption reports. In *Provenance and Annotation of Data and Processes*, volume 7525 of *LNCS*, pages 209–213. Springer Berlin Heidelberg, 2012.
6. S. D. Ramchurn, T. D. Huynh, and N. R. Jennings. Trust in multiagent systems. *The Knowledge Engineering Review*, 19(1):1–25, 2004.
7. N. R. Velaga, M. Beecroft, J. D. Nelson, D. Corsar, and P. Edwards. Transport poverty meets the digital divide: accessibility and connectivity in rural communities. *Journal of Transport Geography*, 21(0):102 – 112, 2012.

DRETa: Extracting RDF from Wikitables

Emir Muñoz, Aidan Hogan, and Alessandra Mileo

Digital Enterprise Research Institute, National University of Ireland, Galway
{emir.munoz, aidan.hogan, alessandra.mileo}@deri.org

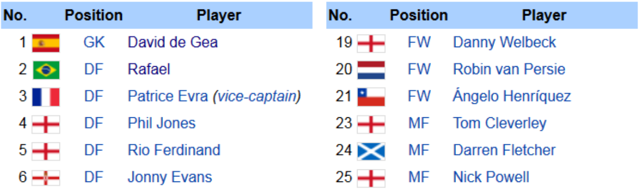
Abstract. Tables are widely used in Wikipedia articles to display relational information – they are inherently concise and information rich. However, aside from info-boxe s, there are no automatic methods to exploit the integrated content of these tables. We thus present DRETa: a tool that uses DBpedia as a reference knowledge-base to extract RDF triples from generic Wikipedia tables.

1 Introduction

Large amounts of data on the Web are presented in tables [3]. Interpreting and extracting knowledge from HTML Web tables is thus relevant for many areas, including: finance, public policy, user experience, health-care, and so forth. However, such tables are diverse in terms of representation, structure and vocabulary used; they often contain polysemous (or missing or otherwise vague) attribute labels, ambiguous free-text cell content and referents, cell spanning multiple rows and/or columns, split tables, obscured contextual validity, and so forth. Recovering the semantics of generic Web tables is thus extremely challenging.

Instead of interpreting generic Web tables, we have rather been focussing on (partially) interpreting the tables embedded in Wikipedia (henceforth "Wikitables"). In particular, we have created DRETa: a prototype for (semi-)automatically performing a best-effort extraction of RDF triples from Wikitables. Though many of the challenges remain the same, focussing on Wikitables has a number of distinct advantages over the more general Web table scenario: (1) Wikitable cells often contain links to Wikipedia articles that disambiguate the entities being talked about; (2) Wikitables contain a high ratio of rich encyclopaedic knowledge; (3) the context of a Wikitable can be mapped to the article in which it appears; (4) existing RDF knowledge-bases, that offer partial exports of Wikipedia content, can be used for reference and for mining legacy entity URIs and predicates. To maximise the precision of the triples extracted from Wikitables, DRETa exploits these unique advantages insofar as possible.

The DRETa system works by using a suitable reference knowledge-base—such as DBpedia [1], YAGO2 [4], Freebase [2], etc.—to extract triples from Wikitables. We only consider tables embedded in article bodies (`class=wikitables` in the HTML source), filtering info-boxes (already used by DBpedia and YAGO2) and tables-of-content. Our prototype system—available at http://deri-srvgal36.nuigalway.ie:8080/wikitables-demo-0.1.0/—currently uses DBpedia as the reference knowledge-base. Thus, the RDF triples



No.	Position	Player
1	GK	David de Gea
2	DF	Rafael
3	DF	Patrice Evra *(vice-captain)*
4	DF	Phil Jones
5	DF	Rio Ferdinand
6	DF	Jonny Evans
19	FW	Danny Welbeck
20	FW	Robin van Persie
21	FW	Ángelo Henríquez
23	MF	Tom Cleverley
24	MF	Darren Fletcher
25	MF	Nick Powell

Fig. 1: Split table of current Manchester United F.C. squad members (abridged from `http://en.wikipedia.org/wiki/Manchester_United_F.C.`).

extracted by DRETa from Wikipedia's tables use the same URIs as DBpedia to identify entities (subject/object URIs) and relations (predicate URIs). Our system can be seen as enriching the reference knowledge-base with additional facts found in tables using the legacy relations from the knowledge-base itself.

2 Triple Extraction: A Motivating Example

We sketch the extraction process by way of a real-world example. Figure 1 presents a Wikitable abridged from the "Manchester United F.C." Wikipedia article, containing relations between players, their shirt number, country and position. There are also relations between players and the entity described by the article (their current club is Manchester United F.C.).

Aside from the **No.** columns, the cells of the table contain hyperlinks to other articles in Wikipedia, including countries, football positions, and individual players. For example, the flags link to articles for the country; GK links to the article for Goalkeeper_(associated_football). These links provide unambiguous referents to Wikipedia entities, which can in turn be mapped directly to DBpedia entities and descriptions. Currently, DRETa focuses on the extraction of relations between cells containing wiki-links and does not consider plain-string values. For example, it would not try to extract player numbers from the table.

Previous works on extracting RDF from such tables (e.g., [5]) propose a vertical, column-centric approach, where columns are seen as referring to types and/or relations that can then be extended to all rows. We rather adopt a horizontal, row-centric approach and look at the pre-existing relations between entities on the same row. For example, if we find that the predicate `dbp:position` holds between `dbr:David_de_Gea` and `dbr:Goalkeeper_(association_football)` (GK), we can suggest that the relation holds from all entities in the **Player** column to all entities on the same row in the **Position** column. Similarly, we consider the article in which the table is found to be a protagonist for the table. If we find the predicate `dbo:team` and `dbp:currentclub` holds from `dbr:David_de_Gea` to `dbr:Manchester_United_F.C.`, we can propose that the same relations hold from all entities in the **textsfPlayer** column to the protagonist (the article entity).

Candidate triples that we extract are further associated with a number of features to help classify them as correct/incorrect, associating each triple with a confidence score. Details of the features are out-of-scope, but, for example, we hypothesise that the more rows a given relation holds for across entities in two fixed columns in the reference KB, the higher the likelihood that that relationship exists on all such rows. Other features, such as a match between the label of the candidate relation and a column header, can further strengthen confidence in the match. Using a selection of 750 random triples labelled by three judges, we employed offline various machine learning methods (SVM, Naïve Bayes, Bagging Decision Trees, Random Forest, Logistic) to train a range of binary classifiers that are then made available to the DRETa system for classifying (in)correct triples, and ultimately for ranking and filtering candidate triples at runtime.

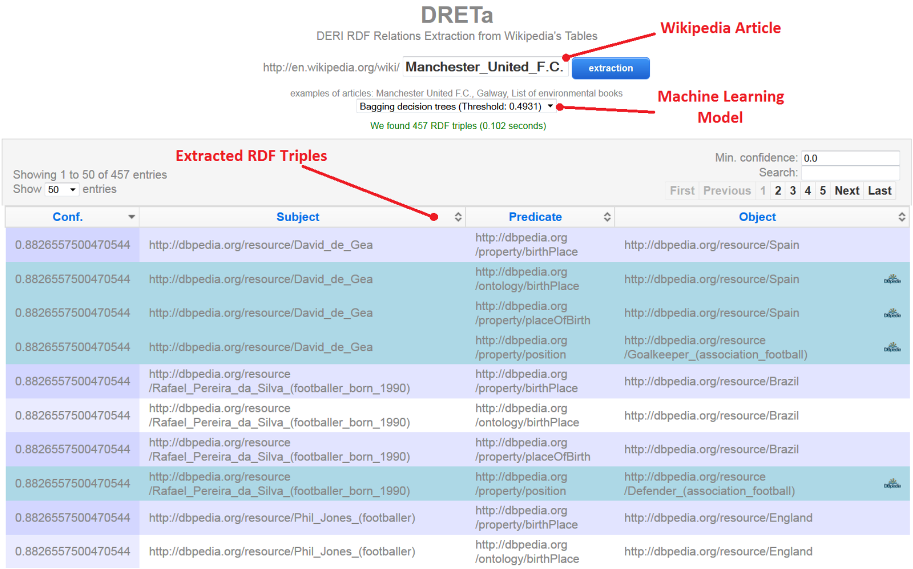
3 DRETa system description

The current DRETa prototype works with DBpedia 3.8 and Wikipedia article names (for auto-completion) as last updated in May 2013. The user submits a Wikipedia article title (with the help of auto-complete) and optionally selects a classifier. The extraction process is then as follows: *i)* the selected article is downloaded and cached in memory for future queries; *ii)* all the tables are extracted, repaired (as applicable), and filtered; *iii)* for each table, mappings from wiki-links to KB entities are executed; *iv)* candidate relations for pairs of resources are collected from the reference KB and candidate triples proposed; *v)* the selected classifier is run to rank triples by confidence, *vi)* each candidate triple is tested against the KB to determine if it is a pre-existing triple or not.

Following our motivating example, Figure 2 presents DRETa's results for the Wikipedia article "Manchester United F.C.". After ca. 11 secs. we get 457 RDF triples extracted from tables contained in that article as a result. Some of these triples already exist in DBpedia (rows with an DBpedia icon visible). Others are novel: from the top-10 extracted triples sorted by confidence, for example, we extract triples for the birth-places of the footballers Rafael da Silva (Brazil) and Phil Jones (England), which were not previously known to DBpedia.

4 Conclusion

In this paper, we have presented DRETa: a prototype system for extracting RDF triples from Wikipedia tables. The process maps entities in table cells to entities in a reference knowledge-base and then looks for potential relations that hold between entities on the same row across two given columns, or that hold between entities in a single column and the article entity. Triples are associated with a set of features that help classify correct/incorrect triples. A selection of machine learning methods are used offline to train classifiers, where these classifiers can then be used to rank the confidence of triples. The prototype is available online at `http://deri-srvgal36.nuigalway.ie:8080/wikitables-demo-0.1.0/`.



DRETa

DERI RDF Relations Extraction from Wikipedia's Tables

http://en.wikipedia.org/wiki/ Manchester_United_F.C.

examples of articles: Manchester United F.C., Galway, List of environmental books

Bagging decision trees (Threshold: 0.4931)

We found 457 RDF triples (0.102 seconds)

Showing 1 to 50 of 457 entries

Show 50 entries

Min. confidence: 0.0

Search:

Conf.	Subject	Predicate	Object
0.8826557500470544	http://dbpedia.org/resource/David_de_Gea	http://dbpedia.org/property/birthPlace	http://dbpedia.org/resource/Spain
0.8826557500470544	http://dbpedia.org/resource/David_de_Gea	http://dbpedia.org/ontology/birthPlace	http://dbpedia.org/resource/Spain
0.8826557500470544	http://dbpedia.org/resource/David_de_Gea	http://dbpedia.org/property/placeOfBirth	http://dbpedia.org/resource/Spain
0.8826557500470544	http://dbpedia.org/resource/David_de_Gea	http://dbpedia.org/property/position	http://dbpedia.org/resource/Goalkeeper_(association_football)
0.8826557500470544	http://dbpedia.org/resource/Rafael_Pereira_da_Silva_(footballer_born_1990)	http://dbpedia.org/property/birthPlace	http://dbpedia.org/resource/Brazil
0.8826557500470544	http://dbpedia.org/resource/Rafael_Pereira_da_Silva_(footballer_born_1990)	http://dbpedia.org/ontology/birthPlace	http://dbpedia.org/resource/Brazil
0.8826557500470544	http://dbpedia.org/resource/Rafael_Pereira_da_Silva_(footballer_born_1990)	http://dbpedia.org/property/placeOfBirth	http://dbpedia.org/resource/Brazil
0.8826557500470544	http://dbpedia.org/resource/Rafael_Pereira_da_Silva_(footballer_born_1990)	http://dbpedia.org/property/position	http://dbpedia.org/resource/Defender_(association_football)
0.8826557500470544	http://dbpedia.org/resource/Phil_Jones_(footballer)	http://dbpedia.org/property/birthPlace	http://dbpedia.org/resource/England
0.8826557500470544	http://dbpedia.org/resource/Phil_Jones_(footballer)	http://dbpedia.org/ontology/birthPlace	http://dbpedia.org/resource/England

Fig. 2: DRETa demo interface showing the top triples extracted from tables in the "Manchester United F.C." article with associated confidence scores.

We are currently investigating methods to perform a high-quality "bulk" triplification of all tables in English Wikipedia. We have used the architecture of DRETa to extract 22 million triples from over one million tables (all tables) in English Wikipedia. We have already estimated a 52% precision measure for the raw candidate triples extracted by our methods and our next steps are to evaluate the extent to which machine learning techniques and different classifiers improve this baseline precision by filtering incorrect triples at various thresholds, and we will identify a gold standard to be able to estimate recall. We also wish generalize our methods with YAGO2 and Freebase as reference knowledge-bases, towards a more ambitious generalisation that employs entity-recognition tools (instead of wiki-links) for processing generic Web tables.

References

1. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia – a crystallization point for the Web of Data. JWS 7(3), 154–165 (2009)
2. Bollacker, K.D., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Wang, J.T.L. (ed.) SIGMOD Conference. pp. 1247–1250. ACM (2008)
3. Cafarella, M.J., Halevy, A.Y., Wang, D.Z., Wu, E., Zhang, Y.: Webtables: exploring the power of tables on the web. PVLDB 1(1), 538–549 (2008)
4. Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. Artif. Intell. 194, 28–61 (2013)
5. Mulwad, V., Finin, T., Syed, Z., Joshi, A.: Using linked data to interpret tables. In: COLD Workshop (November 2010)

Enriching Concept Search across Semantic Web Ontologies

Chetana Gavankar[1,2,3], Vishwajeet Kumar[2],
Yuan-Fang Li[3], and Ganesh Ramakrishnan[2]

(1) IITB-Monash Research Academy, Mumbai, India
(2) IIT Bombay, Mumbai, India
(3) Monash University, Melbourne, Australia

Abstract. Semantic Web ontologies are fast-growing knowledge sources on the Web. Searching relevant concepts from this large repository is a challenging problem. The current Semantic Web search engines provide either (1) coarse-grained search over ontologies or (2) very fine-grained search over individuals. We believe searching and ranking concepts across ontologies provides an ideal granularity for certain tasks such as ontology population and web page annotation. Towards this objective, we propose a novel approach of indexing concepts using ontology axioms in an inverted file structure and ranking them using a dynamic ranking algorithm. Our proposed method is generic and domain-independent. A preliminary evaluation indicates that our proposed method is effective, outperforming the search function of BioPortal, a large and widely-used ontology repository.

Keywords: Semantic Web, Ontologies, Concept Search, Indexing

1 Motivation

The current breed of semantic web search engines can be broadly grouped into 2 categories: (1) those that search over ontologies, and (2) those that search over individual resources. The former may be too coarse-grained as a large ontology may contain hundreds of thousands or even millions of concepts. On the other hand, the latter approach may be too fine-grained – many resources may be relevant and returning them individually may not be the best approach. We describe an approach of retrieving relevant concepts from semantic web ontologies. We propose a novel technique of indexing concepts using axioms in ontologies. Our system supports semi-structured queries where names of concepts and relevant properties can be specified.

2 Related Work

Semantic search engines such as Sindice [1], Swoogle [2, 3], Falcon [4, 5], SWSE [6] provide semantic web search engine interface. They provide search over coarse-grained ontology level and fine-grained resources [7] on the semantic web. We provide search at concept level with middle level granularity. SchemEX [8] is stream based approach and tool for real time indexing and schema extraction of LOD data. Hu, Bo et. al [9] use information retieval tfidf for indexing the ontology documents. Semplore [10] use standard IR style indexing for semantic web content and textual information. In comparison we build index using context information around concept that makes it easy to search for relevant concept along with all its context information. The current work semantic web resources ranking is by adapting and modifying pagerank algorithm used in classical search engines. ReConRank [11], TripleRank[12] adapt Pagerank/HITS [13] algorithm for semantic web data. Our ranking function is parameterized using context features.

3 System Architecture

Our interface provides keyword query input as well as allows to select contextual information around concepts in an ontology corpus. Given a concept in an ontology, all its contextual features are indexed using an inverted file structure. Such features include the concept's label, ID, URI, synonyms, data and object properties used in axioms about the concept, sub classes, super classes, equivalent classes. This approach enriches concept search by disambiguating a concept from those with similar names. For example, if *heart* concept is searched in context of *diseases* using our approach, results related to diseases of heart will be ranked higher, while results in other contexts such as *functionality* will be ranked lower. We now explain the ranking algorithm based on contextual features.

Let α, β, γ represent weights of concept label, data properties ($i = 1$ to m) and object properties ($j = 1$ to n) of the concept respectively. Let δ represent weights of context features ($k = 1$ to t) like synonyms, provenance of the concept. The weights α, β, γ and δ are currently are assigned values based on heuristics. In future we plan to learn these weights using machine learning algorithms. The weight of concept c in the ontology corpus, denoted W_c, is calculated as follows:

$$W_c = \lambda.[\alpha + \beta.\sum_{i=1}^{m} i + \gamma.\sum_{j=1}^{n} j + \delta.\sum_{k=1}^{t} k] \quad (1)$$

$$\lambda = \begin{cases} 1 & \text{if exact match} \\ similarity(x, y) & \text{where } x \text{ and } y \text{ represent 2 strings} \end{cases} \quad (2)$$

4 Evaluation

For evaluation purposes we compare our system[1] with the search function on BioPortal,[2] a large and widely-used biomedical ontology repository. In our experiment a large portion of ontologies, 252 out of 348 in total, were downloaded from BioPortal and indexed. Together these ontologies contain more than 660,000 classes.

---

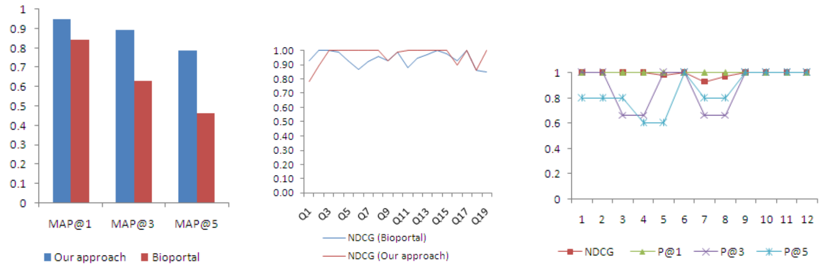
[1] Available at `http://qassist.cse.iitb.ac.in/LOD/`

[2] `http://bioportal.bioontology.org/`

Algorithm 1: Ranking Algorithm

```
Data: Query Tokens $Q = Q_c, Q_{d_1}, ..Q_{d_m}, Q_{o_1}, ...Q_{o_n}, Q_{f_1}, ..Q_{f_t}$, Concepts $C = C_1, C_2 ... C_n$
Result: Weight of Concept $W_c$
$\alpha \leftarrow 0$, $\beta \leftarrow 0$, $\gamma \leftarrow 0$, $\delta \leftarrow 0$, $W_c \leftarrow 0$;
foreach element $C_i \in C$ do
    if $sim(Q_c, label(C_i)) > 0$ then
        $\alpha \leftarrow \alpha + \lambda$
        foreach data property of $C$ do
            for $i$=1 to $m$ do
                if $sim(Q_{d_i}, dp(C_i)) > 0$ then
                    $\beta \leftarrow \beta + \lambda$
        foreach object property of $C$ do
            for $j$=1 to $n$ do
                if $sim(Q_{o_j}, op(C_m)) > 0$ then
                    $\gamma \leftarrow \gamma + \lambda$
        foreach context feature of $C$ do
            for $i$=$k$ to $t$ do
                if $sim(Q_{f_k}, feature(C_m)) > 0$ then
                    $\delta \leftarrow \delta + \lambda$
        $W_c = [\alpha + \beta + \gamma + \delta]$
```

Two metrics widely-used in information retrieval, normalized discounted cumulative gain (NDCG) and mean average precision (MAP) [14], were used to measure the effectiveness of our approach viz-a-viz BioPortal search across 20 queries. Figure 1 (a) and (b) depict MAP and DNCG results for queries that do not contain property information as contextual features. It can be seen from Figure 1 (a) that our system outperforms BioPortal for MAP. Figure 1 (b) shows that the NDCG values are comparable for the two systems. For queries that contain property information, BioPortal fails to return search results. The results for queries with property information in Figure 1 (c) depict high precision and NDCG values.



(a) without property infomation (b) without property infomation (c) with property infomation

Fig. 1. Preliminary evaluation results.

5 Conclusion and Future work

Semantic Web search is primarily divided into two types - one which allows keyword query capability and other which needs SPARQL query input. The latter gives exact results due to precise input queries. This requires user to have technical knowledge about writing a SPARQL query. We present an approach of searching for concepts using semistructured keyword queries that incorporates *contextual features* to improve precision. A preliminary evaluation and a comparison with BioPortal's search function shows the effectiveness of our system. In future we will investigate the incorporation of ontology reasoning to include implicit contextual features. Currently the ranking algorithm derives feature weights heuristically. Going ahead we will learn the weights using machine learning methods. In addition to enriched concept search, our further work will also include property search across ontologies.

References

1. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the open linked data. In: ISWC/ASWC. (2007) 552–565
2. Finin, T., Peng, Y., Scott, R., Joel, C., Joshi, S.A., Reddivari, P., Pan, R., Doshi, V., Ding, L.: Swoogle: A search and metadata engine for the semantic web. In: In Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management, ACM Press (2004) 652–659
3. Ding, L., Pan, R., Finin, T., Joshi, A., Peng, Y., Kolari, P.: Finding and ranking knowledge on the semantic web. In: Proceedings of the 4th International Semantic Web Conference. (2005) 156–170
4. Qu, Y., Cheng, G.: Falcons concept search: A practical search engine for web ontologies. IEEE Transactions on Systems, Man, and Cybernetics, Part A **41**(4) (2011) 810–816
5. Cheng, G., Ge, W., Qu, Y.: Falcons: searching and browsing entities on the semantic web. In: World Wide Web Conference Series. (2008) 1101–1102
6. Hogan, A., Harth, A., Umrich, J., Kinsella, S., Polleres, A., Decker, S.: Searching and browsing linked data with swse: the semantic web search engine. Web Semantics: Science, Services and Agents on the World Wide Web **9**(4) (2011)
7. Blanco, R., Mika, P., Vigna, S.: Effective and efficient entity search in rdf data. In: International Semantic Web Conference (1). (2011) 83–97
8. Konrath, M., Gottron, T., Staab, S., Scherp, A.: Schemex - efficient construction of a data catalogue by stream-based indexing of linked data. J. Web Sem. **16** (2012)
9. Hu, B., Croitoru, M., Dasmahapatra, S., Lewis, P., Shadbolt, N.: Indexing ontologies with semantics-enhanced keywords. In: Proceedings of the 4th international conference on Knowledge capture. K-CAP '07, ACM (2007) 119–126
10. Wang, H., Liu, Q., Penin, T., Fu, L., Zhang, L., Tran, T., Yu, Y., Pan, Y.: Semplore: A scalable IR approach to search the Web of Data. Journal of Web Semantics **7** (2009) 177–188
11. Hogan, A., Harth, A., Decker, S.: Reconrank: A scalable ranking method for semantic web data with context. In: 2nd Workshop on Scalable Semantic Web Knowledge Base Systems. (2006)
12. Franz, T., Schultz, A., Sizov, S., Staab, S.: Triplerank: Ranking semantic web data by tensor decomposition. In: International Semantic Web Conference. (2009) 213–228
13. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. Journal of The ACM **46** (1999) 604–632
14. Manning, C., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. An Introduction to Information Retrieval. Cambridge University Press (2008)

Semantic tools for improving software development in open source communities

Gregor Leban

Jožef Stefan Institute, Ljubljana, Slovenia
gregor.leban@ijs.si

Abstract. Software development communities use different communication channels such as mailing lists, forums and bug tracking systems. These channels are not integrated which makes finding information difficult and inefficient. As a result of the ALERT project we developed a system that is able to collect and annotate information from various communication channels and store it in a single knowledge base. Using the stored knowledge the system can provide users valuable functionalities such as semantic search, finding potential bug duplicates, custom notifications and issue recommendations.

Keywords: information extraction, semantic search, software development, open source, data integration

1 Introduction

Large open source communities frequently use several communication channels for information exchange. Detected bugs are reported on bug tracking systems such as Bugzilla and Mantis. Forums and mailing lists are used for exchanging general discussions and questions about various aspects of the developed software. Source code is shared and updated using source code management systems such as git and svn. Project documentation is often written using a platform such as wiki.

Due to use of various information channels these communities face different challenges. A common problem is finding information. In order to find an answer to a question, the user has to use different search interfaces of these channels to find information of interest. Additionally, since computer science terminology contains many synonyms and abbreviations the search has to be repeated several times using different keywords. Because of the lack of integration, duplicates of the same questions are frequently asked on different information channels. Although some user generated data contains structured information it is not available to the user. For example, a developer is not able to see who are the developers who in the past modified a particular method or class. Such functionality can be crucial in identifying who could be responsible for and therefore fix a particular bug. For large communities such as KDE or Eclipse, the amount of generated content can also be a problem. When hundreds of posts and bugs are reported each day it is time consuming for a user to sweep though the content and find the relevant information.

In order to alleviate such problems we developed a system called ALERT that will be described in the rest of the paper. We will start by describing the system architecture, followed by main features of the system and finish with a conclusion.

2 System architecture

In order to integrate information from different information sources we developed a set of sensors that are able to detect whenever new information is available in the source. Our sensors can import data from five different types of sources: bug tracking systems (BTS), mailing lists, forums, source code management systems (SCM) and wikis. For each data source we collect all available information. In case of a bug report, for example, we obtain the title and description of a bug as well as the author name, time, the assigned product and component information, status and resolution of the bug, etc. In case of a code commit we collect the comment of the commit, time and author as well as extract all the methods modified in the commit.

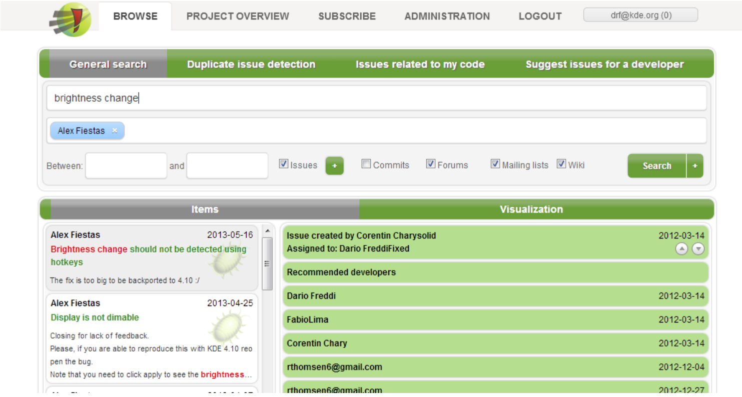
Since all information sources provide some user generated text we first send it to a text enrichment component that annotates the text using a custom built Annotation ontology. The ontology has 6,196 distinct concepts that contain 10,233 labels related to computer science. The details of ontology construction are described in [1]. After annotating the text we store the information from the sensors together with the annotations in a knowledge base.

After new data is stored, two components are notified about the change: the Recommendation service and the Event detector. The Recommendation service is responsible creating and maintaining an expertise profile for each user. The profile consists of code expertise (based on the methods created or modified by the user) and of topics expertise (based on the content the user is writing about). The Event detector is responsible for detecting if the new data matches any of the patterns that users provided as their interests. If it matches, then corresponding users are notified that new information relevant for them has been posted. Additional component of the ALERT system is also the search/visualization service which provides an interface for the user to query the knowledge base.

3 Core features of the ALERT system

Advanced search and visualization functionality

ALERT system provides an intuitive user interface (see Figure 1) where the user can find information by specifying a rich set of search conditions. The most commonly used is the keyword search. Using the Annotation ontology, the system is able to expand the search terms with their synonyms. In this way, searching for term "dialog" also returns posts containing terms "window" or "form". In the additional input box the user can specify constraints such as the author



brightness change

Alex Fiestas

Alex Fiestas 2013-05-16
Brightness change should not be detected using hotkeys
The fix is too big to be backported to 4.10 :/

Alex Fiestas 2013-04-25
Display is not dimable
Closing for lack of feedback.
Please, if you are able to reproduce this with KDE 4.10 reopen the bug.
Note that you need to click apply to see the brightness...

Issue created by Corentin Charysolid 2012-03-14
Assigned to: Dario FreddiFixed

Recommended developers

Dario Freddi	2012-03-14
FabioLima	2012-03-14
Corentin Chary	2012-03-14
rthomsen6@gmail.com	2012-12-04
rthomsen6@gmail.com	2012-12-27

Fig. 1. Screenshot of the search interface of ALERT system.

of the post, the product/component of the bug report, id of the bug report or the file/class/method name. The user can also uncheck individual sources if he wishes to ignore results from them and also set a specific time period of interest.

Search results are displayed in a list where each item contains the author, date and a short snippet of text. Selecting an item displays its full details on the right. Clicking an issue, for example, displays its available meta data, the description and all following comments. Clicking a commit, on the other hand, displays the tree of files, classes and methods modified in the commit. Results are also visualized and summarized using three visualizations based on the research done in [2]. The social graph shows a network of people involved in the results. There is an edge between two persons in the graph if one is responding to the others post (e.g., one person sends an email to the other). The timeline visualization shows the distribution of results over time. This allows one to quickly identify interesting patterns, such as very high/low activity. The last visualization is a tag cloud that provides a keyword summary of results.

Recommending open issues to developers

When bug reports are created, a stack trace with a detailed exception information is frequently provided. The trace contains names of the methods in the call path where the exception occurred. Since we are monitoring the source code developed in the project (using SCM sensors) we are able to automatically identify references of the known methods. By taking into account also the information about who changed individual methods, we can then suggest developers who can most likely fix a particular issue.

A developer can be recommended even in cases when no stack trace is provided. In such cases we can recommend developers based on the topics discussed in the bug report. For each developer we store a profile based on the content he wrote about. When we need a recommendation for a report we can suggest developers whose profile matches best with the content of the report.

Bug duplicate detection

A common problem on issue tracking systems is that users unknowingly create duplicates of the same issue. A bug triager needs to determine if a new report is describing a new issue or is a duplicate, before he can assign a developer to fix it. Since this is an error prone and time consuming task we wanted to make it easier. Given a bug report, the ALERT system can provide the user with a ranked list of other most content-wise similar bug reports. The similarity is computed using cosine similarity on bug reports represented as TF-IDF normalized vectors. Our experiments indicate that 60% of bug duplicates can be identified already by checking only the first 5 most similar reports.

Custom notifications

ALERT system allows the users to define custom patterns of interest and when they occur, the users are notified about it by an email or in a notification window inside the system. The user can, for example, choose to be notified when a post with X, Y or Z topic is published in some information channel, when more than X new bugs are reported in Y days, when the person X creates a new post, when a new bug is more than X% similar to an existing bug, etc. Using such notifications the user does not have to manually check for new relevant information – instead the content is automatically ”pushed” to him.

4 Conclusion

In this paper we presented a system called ALERT whose goal is to help software developing communities. The system collects data from various information channels used by the community, extracts available information from it and stores it in a knowledge base. Using the stored data, the system can provide the users with advanced functionalities, such as semantic search and visualization over all information sources, bug recommendation, bug duplicate detection and custom notifications. The developed system was tested in three open source communities (KDE, Linagora OW2 and Morfeo project) that provided very positive feedback. The demo of the system containing KDE data is currently running at http://aidemo.ijs.si/alert/.

5 Acknowledgements

This work was supported by the Slovenian Research Agency and ALERT (ICT-249119-STREP).

References

1. G. Leban, L. Dali, and I. Novalija. Enabling semantic search in open source communities. In *Proceedings of ESWC 2012, Heraklion, Crete, Greece*, 2012.
2. L. Stopar and G. Leban. Searching for information in software development projects using ALERT system. In *Proceedings of the 15th International Multiconference on Information Society IS-2012*, 2012.

Editing R2RML Mappings Made Easy.

Kunal Sengupta[1,2]*, Peter Haase[1], Michael Schmidt[1], and Pascal Hitzler[2]

[1] fluid Operations AG, Walldorf, Germany
[2] Wright State University, Dayton OH 45435, USA

Abstract. The new W3C standard R2RML[3] defines a language for expressing mappings from relational databases to RDF, allowing applications built on top of the W3C Semantic Technology stack to seamlessly integrate relational data. A major obstacle in using R2RML, though, is the creation and maintenance of mappings. In this demo, we present a novel R2RML mapping editor which provides a user interface to create and edit mappings interactively even for non-experts.

1 Introduction

The RDB to RDF mapping language (R2RML) has recently become a W3C standard for creating mappings from relational databases to RDF. This enables many semantic web applications to integrate easily with relational databases. Although very useful, we observe certain problems with the adoption of the new standard: (1) creating R2RML rules manually is a time consuming process, (2) even simple rules could be syntactically heavy in terms of usage of the R2RML vocabulary, and (3) a steep learning curve is involved in gaining expertise of this new language. With these issues in mind, we have developed an R2RML mapping editor that provides an intuitive interface to the users to create, edit and manage R2RML mappings. The editor is fully integrated into the Information Workbench [1], an Open Source platform for Linked Data application development.

2 R2RML by Example

A mapping rule in this language is referred to as a TriplesMap. Example 1 illustrates an R2RML mapping rule in Turtle[4] syntax. The R2RML mapping in this example has three components: The predicate `rr:logicalTable` points to a structure selecting a table, view, or SQL query from the relational database that is mapped into RDF; in this example, a custom SQL query is used to retrieve the columns id, gid and length from the table recording. The predicate `rr:subjectMap` points to a structure defining how rows from the logical table query results are transformed into possibly typed, subjects; in the example, we create subjects by instantiating the specified `rr:template` with the values from

---

[*] Work performed while at fluid Operations.

[3] See: http://www.w3.org/TR/r2rml/

[4] See `http://www.w3.org/TR/turtle/`.

the `{gid}` column, and specify `mo:recording` as the type of these subjects. The predicate `rr:predicateObjectMap` points to a structure defining property-value pairs for the subjects defined by the `rr:subjectMap`; in the example, for every subject we create a property `mo:duration`, pointing to a literal created by column `length` of the source database.

Example 1.

```
rr:logicalTable
  [rr:sqlQuery
     """SELECT id,  gid, length FROM  musicbrainz.recording
     WHERE musicbrainz.recording.length IS NOT NULL"""]
rr:subjectMap
  [rr:class mo:recording ;
   rr:template "http://musicbrainz.org/artist/{gid}\#\_"]
rr:predicateObjectMap
   [rr:objectMap
     [ rr:column "length" ;
       rr:datatype <http://www.w3.org/2001/XMLSchema\#float> ] ;
     rr:predicate  mo:duration].
```

As can be seen from the example, the syntax to denote the mapping is rather convoluted and requires deep technical knowledge of R2RML, its language constructs, syntactic details (such as quotation), and the R2RML vocabulary itself.

3 The Mapping Editor

The editor that we have developed in order to overcome these technicalities features an intuitive user interface which hides the R2RML vocabulary details from the user, provides access to relational metadata of the relational database for which the mappings are to be created, provides instant feedback where, at each step, the user has the option to preview the triples that would result from the mapping that is being created, and supports most of R2RML vocabulary except `rr:graph`, which we plan to include as we continuously evolve the editor. In the following we briefly describe the various steps of the editor's wizard workflow.

Step 1 **Datasource, Base URI Selection.** The first step in the wizard lets the user choose the datasource for which mappings shall be defined, i.e. by design the editor supports the maintenance of mappings over multiple relational data sources. As part of the datasource selection, the user may also choose a base URI for URI construction templates specified in the upcoming steps.

Step 2 **R2RML Rule Selection.** After datasource selection, the editor loads and displays previously saved mappings for the datasource. At this point the user may choose to edit an existing rule or add a new one. The editor also provides a filter using which the mappings could be looked up based on involved table names, concept names and property names.

Step 3 **Logical Table Selection.** In this step, the editor provides various options to choose a logical table. The logical table could be a database table, view or a custom SQL query written by the user. If the option to add a database table is selected, all the tables from the database schema are displayed. Options to preview the table's data and metadata are also available. On the other hand, if the option of SQL query is chosen then an SQL editor is shown where the user can type SQL queries and preview the results.

Step 4 **Subject Map Creation.** Once a logical table has been selected, the editor guides the user to a subject template creation step. By the click of a button, the user can carry over columns that should be used to generate the template. Additionally, classes (i.e., types) can be added to the subject by using the auto-complete field rdf:type, which displays all the classes that are present in the system. The editor is flexible enough such that multiple classes can be added. At this stage a preview of the generated triples can be accessed as well. This provides feedback to the user such that they can adjust the values in order to obtain the intended results.

Step 5 **Predicate-Object Maps Creation.** After the subject map creation, predicate-object maps can be added to the mapping. Again, the editor provides flexibility for adding as many predicate-object pairs as needed. Additionally, for object maps all types of terms defined in R2RML are supported, namely constant terms, column values, templates, and object references. The object reference option lets the user choose from the previously defined mappings. Furthermore, on choosing the appropriate rule the child and parent columns are prepopulated from the corresponding logical tables. Again, multiple join conditions can be added through the interface. The preview triples option is also available at this stage, again providing the user with valuable feedback. Fig. 1 depicts the screen corresponding to this step.

Step 6 **Textual Representation.** Finally, an intuitive textual representation of the mapping rule is displayed. At this point the user could go back to any previous step to modify the mapping or otherwise save the mapping.

The demo video: `https://www.youtube.com/watch?v=PfP2wmWw9-o` is available online, the video includes a very brief introduction to R2RML and motivation for an editor to create R2RML mappings. The remainder of the demonstration shows a use case of mapping MusicBrainz[5] database tables to the Music ontology.[6] In particular, we demonstrate two example mappings, where each step of the mapping creation process is explained and special features of the editor are highlighted.

Although tools like Karma [2], Ontop [3], Topbraid Composer[7], and Neon Toolkit[8] can be used for mapping relational and other datasources to RDF, the editor presented here provides the following advantages (1) It is the first editor to fully support R2RML mappings, (2) The preview feature is unique as the users get instant feedback and can modify the mappings iteratively, (3)

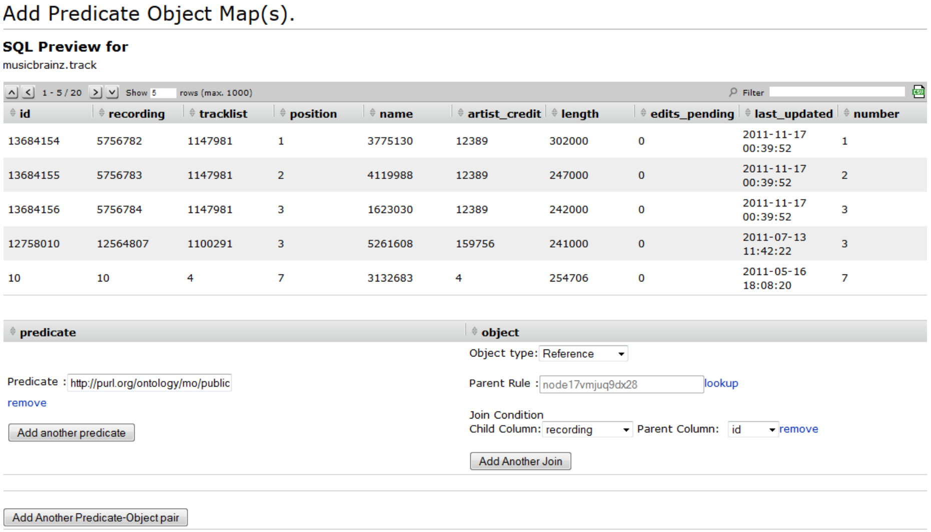
---

[5] MusicBrainz: https://wiki.musicbrainz.org/Next_Generation_Schema

[6] Music ontology: http://musicontology.com

[7] See `http://www.topquadrant.com/products/TB_Composer.html`

[8] See `http://neon-toolkit.org/wiki/Main_Page`



Add Predicate Object Map(s).

SQL Preview for
musicbrainz.track

id	recording	tracklist	position	name	artist_credit	length	edits_pending	last_updated	number
13684154	5756782	1147981	1	3775130	12389	302000	0	2011-11-17 00:39:52	1
13684155	5756783	1147981	2	4119988	12389	247000	0	2011-11-17 00:39:52	2
13684156	5756784	1147981	3	1623030	12389	242000	0	2011-11-17 00:39:52	3
12758010	12564807	1100291	3	5261608	159756	241000	0	2011-07-13 11:42:22	3
10	10	4	7	3132683	4	254706	0	2011-05-16 18:08:20	7

predicate

Predicate : http://purl.org/ontology/mo/public

object

Object type: Reference

Parent Rule : node17vmjuq9dx28

Join Condition
Child Column: recording Parent Column: id

Fig. 1. Screen for adding predicate-object maps

It provides easy management of R2RML mappings, (4) It provides a search utility, where users can find mappings based on table names, concept names and property names. Apart from these advantages, we are going to further improve the editor by adding features like automatic mapping suggestions and on click default mapping [4] generation.

References

1. Hartig, O., Harth, A., Sequeda, J. (eds.): Proceedings of the Second International Workshop on Consuming Linked Data (COLD2011), Bonn, Germany, October 23, 2011, CEUR Workshop Proceedings, vol. 782. CEUR-WS.org (2011)
2. Knoblock, C.A., Szekely, P.A., Ambite, J.L., Goel, A., Gupta, S., Lerman, K., Muslea, M., Taheriyan, M., Mallick, P.: Semi-automatically mapping structured sources into the semantic web. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, Ó., Presutti, V. (eds.) The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7295, pp. 375–390. Springer (2012)
3. Rodriguez-Muro, M., Calvanese, D.: -ontop- framework (2012), `http://obda.inf.unibz.it/protege-plugin/`
4. Sequeda, J., Arenas, M., Miranker, D.P.: On directly mapping relational databases to RDF and OWL. In: Mille, A., Gandon, F.L., Misselis, J., Rabinovich, M., Staab, S. (eds.) Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012. pp. 649–658. ACM (2012)

TRT - A Tripleset Recommendation Tool

Alexander Arturo Mera Caraballo[1], Bernardo Pereira Nunes[1], Giseli Rabello Lopes[1], Luiz André P. Paes Leme[2], Marco A. Casanova[1], Stefan Dietze[3]

[1] Department of Informatics, PUC-Rio, Rio de Janeiro/RJ – Brazil
`{acaraballo,bnunes,grlopes,casanova}@inf.puc-rio.br`
[2] Computer Science Institute, Fluminense Federal University, Niterói/RJ – Brazil
`{lapaesleme}@ic.uff.br`
[3] L3S Research Center, Leibniz University Hannover, Germany
`{dietze}@l3s.de`

Abstract. *According to the Linked Data principles, a tripleset should be interlinked with others to take advantage of existing knowledge. However, interlinking is a laborious task. Thus, users interlink their triplesets mostly with data hubs, such as DBpedia and Freebase, ignoring the more specific yet often even more promising triplesets. To alleviate this problem, this paper describes a tripleset interlinking recommendation tool based on link prediction techniques and evaluates the tool on a real-world tripleset repository.*

Key words: Linked Data, Recommender Systems, Social Networks

1 Introduction

A considerable number of triplesets, following the Linked Data principles, have already been published in a large number of areas, ranging from geographic to bibliographic data. This growth makes it difficult to choose which triplesets should be interlinked with a given tripleset. Thus, users interlink their triplesets mostly with data hubs, such as DBpedia and Freebase, ignoring the more specific triplesets which often contain particularly useful data. Furthermore, the metadata provided in data repositories such as the DataHub are typically not sufficient to help users choose the most suitable triplesets to interlink with.

To help alleviate this situation, we describe a tool for tripleset interlinking recommendation, based on previous work by the authors [1, 2]. More precisely, the tool addresses the *tripleset recommendation problem*, defined as follows: Given a tripleset t and a set of triplesets S, rank the triplesets in S based on the probability of interlinking t with them.

2 TRT - The Tripleset Recommendation Tool

Recommendation Procedure. A *tripleset* t is a set of RDF triples. A resource, identified by an RDF URI reference s, is *defined* in t iff s occurs as the subject of a triple in t.

Table 1: Local and quasi-local indices

Indice		**Equation**
Type	**Name**	
Local indices	Common Neighbors	$CN_{t,u} = \lvert C_t \cap C_u \rvert$
	Salton	$Salton_{t,u} = \frac{\lvert C_t \cap C_u \rvert}{\sqrt{C'_t . C'_u}}$
	Jaccard	$Jaccard_{t,u} = \frac{\lvert C_t \cap C_u \rvert}{\lvert C_t \cup C_u \rvert}$
	Sørensen	$S\o rensen_{t,u} = \frac{2. \lvert C_t \cap C_u \rvert}{C'_t + C'_u}$
	Hub Promoted index	$HPI_{t,u} = \frac{\lvert C_t \cap C_u \rvert}{min\{C'_t, C'_u\}}$
	Hub Depressed index	$HDI_{t,u} = \frac{\lvert C_t \cap C_u \rvert}{max\{C'_t, C'_u\}}$
	Leicht-Holme-Newman	$LHN_{t,u} = \frac{\lvert C_t \cap C_u \rvert}{C'_t . C'_u}$
	Preferencial Attachment	$PA_{t,u} = C'_t . C'_u$
	Adamic-Adar	$AA_{t,u} = \sum_{w \in C_t \cap C_u} \frac{1}{log \lvert C'_w \rvert}$
	Resource Allocation	$RA_{t,u} = \sum_{w \in C_t \cap C_u} \frac{1}{\lvert C'_w \rvert}$
Quasi-local indices	Local Path	$LP_{t,u} = A^2 + \varepsilon A^3$
	Local Random Walk	$LRW_{t,u}(s) = \frac{C'_t}{2\lvert C \rvert} . \pi_{t,u}(s) + \frac{C'_u}{2\lvert C \rvert} . \pi_{u,t}(s)$

Let t and u be two triplesets. A *link* from t to u is a triple of the form (s, p, o), where s is an RDF URI reference identifying a resource defined in t and o is an RDF URI reference identifying a resource defined in u; we also say that (s, p, o), *interlinks* s *and* o. We say that t *can be interlinked* with u iff it is possible to define links from t to u. A *Linked Data network* is a graph $G = (S, C)$ such that S is a set of triplesets and C contains an edge (t, u), called a *connection* from t to u, iff there is at least one link from t to u.

Our recommendation procedure analyses the Linked Data network in much the same way as a Social Network. The inputs of the procedure are: (i) a *Linked Data network* $G = (S, C)$; (ii) a *target tripleset* t not in S (intuitively the user wishes to define links from t to the triplesets in S); and (iii) a *target context* C_t for t consisting of one or more triplesets u in S (intuitively the user knows that t can be interlinked with u). The output is an order list L of triplesets in S, called a *ranking*. The triplesets in the ranking are ordered using link prediction techniques discussed in what follows.

Link prediction techniques. The procedure uses link prediction theory to estimate the likelihood of the existence of a link between triplesets. We focus on local and quasi-local indices to measure the structural similarity between triplesets [3] according to their link structure. Table 1 summarizes the indices the procedure implements, where:

- C_i is the context of i (triplesets that i points to), where i a specific tripleset;
- C'_i is the inverse context of i (triplesets that point to i), where i a specific tripleset;
- A^j is the number of different paths with length j connecting t and u;
- ε is a free parameter;
- $\pi_{t,u}(s)$ is the probability that a random walker starting on t locates u after s steps;
- C is the set of all edges of the Linked Data network G.

Description of the TRT Tool in Action. Briefly, suppose that the user is working on a tripleset t and wants to discover one or more triplesets u such that t can be interlinked with u. He then uses the tool to obtain recommendations.

The tool first builds the Linked Data network $G = (S, C)$ defined by the metadata stored in the DataHub repository.

Then, the user defines the rest of the input data the tool requires. He may define a target context C_t for t, consisting of one or more triplesets in S, in two different ways: (i) by providing a VoID descriptor V_t for t from which the tool extracts C_t by analysing the *void:linkset* declarations occuring in V_t; or (ii) by manually selecting triplesets from the categories the tool displays. Finally, the user chooses a similarity index from those shown on Table 1.

From this input data, the tool outputs a ranked list of triplesets, thereby helping reduce the effort required to find related triplesets for the interlinking process.

The tool can be accessed at `http://web.ccead.puc-rio.br:8080/Uncover/`.

3 Evaluation

The tool was evaluated using the DataHub repository, which contains more than 6,000 triplesets, with approximately 15 thousand links that connect only 711 of the available triplesets. The links across triplesets were used to rank and recommend triplesets for interlinking. The recommendation process was assessed using the 10-fold cross validation approach, where we randomly divided the observed links into 10 subsets used as recommendation subgraphs. Finally, the overall performance was computed in terms of the average of the performances in the testing partitions.

To evaluate the prediction indices, we used three standard metrics: Area Under the receiver operating characteristic Curve (AUC), Mean Average Precision (MAP) and Recall. Table 2 summarizes the results for different target context sizes (shown in the first column of the table). The entries corresponding to the highest results among the 12 indices are emphasized in boldface underlined. The reader may observe that the PA index achieved the highest AUC (ranging from 83.74% to 95.90% depending on the target context size). The PA index also obtained the best MAP (37.83%) for target contexts with very few triplesets, while the RA index turned out to be more precise (72.42%) for larger target contexts. Table 2 also shows the coverage results. The PA index obtained the highest recall (96.4%), regardless of the size of the target context.

Table 2: AUC, MAP and Recall of the local and quasi-local indices

AUC	CN	Salton	Jaccard	Sørensen	HPI	HDI	LHN	PA	AA	RA	LP	LRW
1	70.52	47.79	69.84	69.28	48.94	69.31	48.00	**83.74**	71.31	70.53	70.74	69.67
5	87.10	55.73	81.20	80.93	58.78	80.17	52.24	90.76	88.45	88.02	**92.70**	83.21
10	92.42	57.14	85.06	84.85	60.84	83.79	52.87	**92.81**	92.37	92.40	92.25	86.69
20	92.77	58.47	88.34	88.30	59.45	86.54	51.39	**94.33**	92.53	92.64	92.76	88.22
50	92.84	59.10	92.96	92.99	56.27	92.09	52.30	**95.90**	92.17	92.72	91.91	90.26
MAP	CN	Salton	Jaccard	Sørensen	HPI	HDI	LHN	PA	AA	RA	LP	LRW
1	18.17	14.49	16.30	14.73	17.08	15.00	14.80	**37.83**	18.06	17.80	18.46	15.57
5	49.48	25.07	21.80	20.36	35.14	19.20	18.38	48.26	**52.20**	51.48	58.23	26.05
10	63.49	30.99	30.40	28.71	41.81	24.41	19.44	52.62	63.43	**63.71**	62.63	31.91
20	71.20	34.22	44.37	43.56	38.14	34.14	17.90	53.97	71.46	**72.38**	70.59	34.66
50	71.13	27.73	69.49	70.55	20.64	66.14	15.92	47.30	70.99	**72.42**	67.51	39.03
Recall	CN	Salton	Jaccard	Sørensen	HPI	HDI	LHN	PA	AA	RA	LP	LRW
1	48.72	49.69	49.86	49.76	49.68	49.55	50.02	**96.40**	50.81	48.74	48.81	49.12
5	81.45	83.80	82.69	83.03	83.68	82.23	82.43	**98.45**	83.63	82.83	86.90	82.42
10	89.52	88.73	89.42	89.29	89.35	89.85	89.17	**98.74**	89.49	89.28	88.96	89.21
20	90.03	90.31	89.68	89.18	89.12	89.53	89.19	**99.80**	89.50	89.58	89.84	90.01
50	90.05	90.16	90.15	90.21	90.04	89.38	88.45	**99.56**	89.06	89.58	89.02	89.71

4 Conclusions

In this paper, we proposed the use of link prediction techniques to address the tripleset recommendation problem in the Linked Data domain and presented a tool that implements the techniques. The tool computes local and quasi-local indices to predict links between triplesets. The results showed that the tool performs better, with respect to both AUC and recall, when the PA index is adopted. In terms of MAP, the PA index should be adopted for smaller context sizes, while the RA index should be adopted for larger context sizes.

Acknowledgments. This work was partly supported by CNPq, under grants 160326/2012-5, 301497/2006-0, 475717/2011-2 and 57128/2009-9, by FAPERJ, under grants E-26/170028/2008 and E-26/103.070/2011.

References

1. Leme, L.A.P.P., Lopes, G.R., Nunes, B.P., Casanova, M.A., Dietze, S.: Identifying candidate datasets for data interlinking. In Daniel, F., Dolog, P., Li, Q., eds.: ICWE. Volume 7977 of Lecture Notes in Computer Science., Springer (2013) 354–366
2. Lopes, G.R., Leme, L.A.P.P., Nunes, B.P., Casanova, M.A., Dietze, S.: Recommending tripleset interlinking through a social network approach. In: Proceedings of WISE'13. (2013 (to appear))
3. Lü, L., Jin, C.H., Zhou, T.: Similarity index based on local paths for link prediction of complex networks. Physical Review E **80**(4) (2009) 046122

KbQAS: A Knowledge-based QA System

Dat Quoc Nguyen, Dai Quoc Nguyen, and Son Bao Pham

Faculty of Information Technology
University of Engineering and Technology
Vietnam National University, Hanoi
`{datnq, dainq, sonpb}@vnu.edu.vn`

Abstract We present the first ontology-based Vietnamese QA system KbQAS where a new knowledge acquisition approach for analyzing English and Vietnamese questions is integrated.

1 Introduction

Recent years have witnessed a new trend of building ontology-based question answering (QA) systems to make the use of semantic information in terms of semantic web. This demo paper introduces a knowledge-based QA system named KbQAS, the *first* ontology-based QA system for Vietnamese. The target domain is modeled as an ontology in our KbQAS system to leverage techniques and latest advances in the semantic web. Thus semantic markups can be used to add meta-information to provide more precise answers to complex questions expressed in natural language. This is an avenue that has not been actively explored for Vietnamese.

2 System overview

The architecture of our KbQAS system as shown in figure 1 contains a question analysis component where *a new knowledge acquisition approach* to systematically build knowledge bases of grammar rules for processing natural language questions over semantic annotations is integrated. The question analysis component takes the user question as an input and returns an intermediate element representing the question in a compact form. The answer retrieval component is responsible for making sense of the user query with respect to a target ontology using concept-matching techniques for a natural language phrase and elements in the ontology. It takes an intermediate representation produced by the question analysis component and an ontology as its input to generate answers.

Illustrative example. For demonstration[1] purpose, we exploit an ontology modeling the organizational structure of an university as referred in our first KbQAS version [1].

Consider the complex structure question of *"Liệt kê tất cả sinh viên học lớp K50 khoa học máy tính mà có quê ở Hà Nội?"* ("List all students studying in K50 computer science course, who have hometown in Hanoi?"): the Question analysis component determines that this question has a query structure of type *"And"* and two query-tuples (Normal, List, sinh viên$_{student}$, học$_{study}$, lớp K50 khoa học máy tính$_{K50\ computer\ science\ course}$, **?**[2]) and (Normal, List, sinh viên$_{student}$, có quê$_{has\ hometown}$, Hà Nội$_{Hanoi}$, **?**).

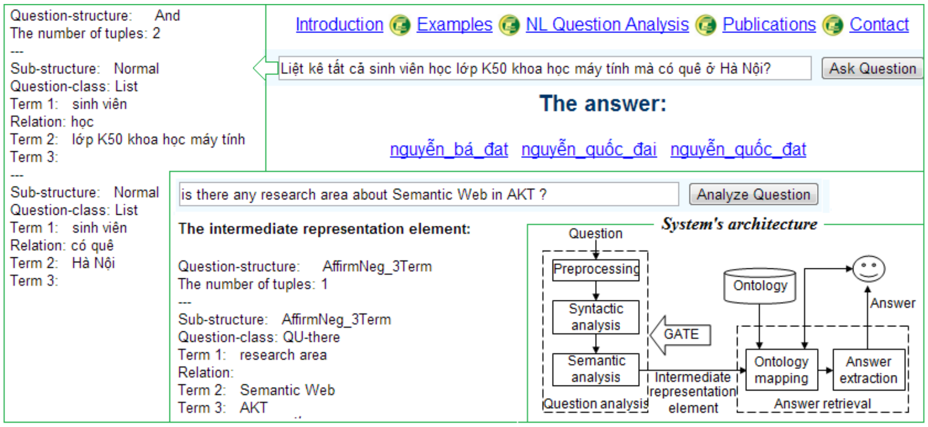
Query-tuples are then mapped to ontology-tuples by the Ontology mapping module in the Answer retrieval component: (sinh_viên$_{student}$, học$_{study}$, lớp_K50_khoa_học_

---

[1] KbQAS available in http://cntt.dyndns.info:8856/KbQAS/
Vietnamese question analysis demonstration at http://cntt.dyndns.info:8856/KbVnQA/
English question analysis demonstration available in http://cntt.dyndns.info:8856/KbEnQA/

[2] **?** indicates the missing element.

máy_tính$_{K50\_computer\_science\_course}$) and (sinh_viên$_{student}$, có_quê$_{has\_hometown}$, Hà_Nội$_{Hanoi}$). With each ontology-tuple, the Answer extraction module finds all satisfied instances in the target ontology before generating an answer presented in the figure 1 based on the question structure *"And"* and the question class *"List"*.



Question-structure: And
The number of tuples: 2

Sub-structure: Normal
Question-class: List
Term 1: sinh viên
Relation: học
Term 2: lớp K50 khoa học máy tính
Term 3:

Sub-structure: Normal
Question-class: List
Term 1: sinh viên
Relation: có quê
Term 2: Hà Nội
Term 3:

Liệt kê tất cả sinh viên học lớp K50 khoa học máy tính mà có quê ở Hà Nội?

The answer:

nguyễn_bá_đat nguyễn_quốc_đai nguyễn_quốc_đat

is there any research area about Semantic Web in AKT ?

The intermediate representation element:

Question-structure: AffirmNeg_3Term
The number of tuples: 1

Sub-structure: AffirmNeg_3Term
Question-class: QU-there
Term 1: research area
Relation:
Term 2: Semantic Web
Term 3: AKT

Figure 1: The system's architecture and illustrations of question analysis and question answering.

Intermediate representation of question. The intermediate representation used in our KbQAS system consists of a *question-structure* and one or more *query-tuple*s in the following format: *(sub-structure, question-class,* $Term_1$*, Relation,* $Term_2$*,* $Term_3$*)*. Simple questions only have one query-tuple and its question-structure is the query-tuple's sub-structure. More complex questions such as composite questions have several sub-questions, each sub-question is represented by a separate query-tuple, and the question-structure is to capture this composition attribute.

Question analysis component. The question analysis component contains three modules: preprocessing, syntactic analysis and semantic analysis. It makes the use of JAPE grammars in GATE framework [2] to specify regular expression patterns based on semantic annotations for question analysis. The preprocessing and syntactic modules are responsible for identifying noun phrases, question-phrases, and the relations among noun phrases or between noun phrase and question-phrase in the input questions. The semantic analysis module embodies **the key innovation** in the current KbQAS version. This semantic module utilizes the noun phrase, question-phrase and relation annotations created by the two preceding modules. It aims to specify the question-structure and to produce the query-tuples as the intermediate representation of the input question.

In the current semantic analysis module, we propose *a new knowledge acquisition approach* for analyzing natural language questions by applying Single Classification Ripple Down Rules (SCRDR) methodology [3] to acquire rules incrementally. A SCRDR knowledge base, where grammar rules over semantic annotations are structured in an exception structure and new rules are only added to correct errors of existing rules, is built to generate the intermediate representations of questions. This process is to create rules in a systematic manner to solve difficulties which appear in such most existing rule-based question analysis approaches as in Aqualog system [4] and the first KbQAS version [1] in managing the interaction between rules and keeping consistency among them. Moreover, our proposed approach enables ones to easily construct a new system or adapt an existing system to a new domain or a new language, thus a lot of time and effort of human experts can be saved. The experimental evaluation of our method for English and Vietnamese question analyses is detailed in our previous work [5].

Answer retrieval component. The detail description of this component can be found in the first KbQAS version [1]. In short, the task of its Ontology mapping module is to map terms and relations in the query-tuples to concepts, instances and relations in the target ontology. Then the Answer extraction module finds all instances associated to mapped instances and concepts, satisfying ontology relations. Depending on the question-structure and question-class, the best semantic answer will be returned.

Evaluation. The performance of the current KbQAS on a wide range of different Vietnamese questions is promising with accuracies of 84.1% and 82.4% accounted for the question analysis and answer retrieval components, respectively.

3 Demonstration: knowledge acquisition for question analysis

In this section, we only illustrate the process of systematically constructing a SCRDR knowledge base for analyzing English questions[3]. In demonstration session, however, we plan to present other illustrations of building English and Vietnamese knowledge bases for question analysis, and to provide other illustrative examples of the KbQAS.

The following exemplification shows how the knowledge base building process works. When we encounter the question *"who are the researchers in semantic web research area ?"* ([QuestionPhrase: who] [Relation: are the researchers in] [NounPhrase: semantic web research area]). Supposed we start with an empty knowledge base, the fired rule (i.e. last satisfied rule) is the default rule[4] that gives empty conclusion. This can be corrected by adding the following exception rule to the knowledge base:

Rule: R1
(({QuestionPhrase}):QPhrase ({Relation}):Rel ({NounPhrase}):NPhrase):left
−−→ :left.RDR1_ = {category1 = "UnknTerm"}
, :QPhrase.RDR1_QP = {} , :Rel.RDR1_Rel = {} , :NPhrase.RDR1_NP = {}

Conclusion: question-structure *"UnknTerm"* and query-tuple (RDR1_.category1, RDR1_QP.QuestionPhrase.category, ?, RDR1_Rel, RDR1_NP, ?).

If the condition of rule R1 matches whole input question, a new annotation RDR1_ will be created to entirely cover the input question and new annotations RDR1_QP, RDR1_Rel and RDR1_NP will also be generated for covering sub-phrases of the input question. Once rule **R1** is fired, the matched input question is deemed to have a query-tuple with *sub-structure* taking the value of the feature *"category1"* of RDR1_ annotation, *question-class* taking the value of the feature *"category"* of QuestionPhrase annotation surrounding the same span as RDR1_QP annotation. Besides, the query-tuple's *Relation* is the string covered by RDR1_Rel, $Term_2$ is the string surrounded by RDR1_NP while $Term_1$ and $Term_3$ are missing.

---

[3] We utilized JAPE grammars employed in AquaLog [4] for detecting the noun-phrase, question-phrase, and relation annotations in English questions. We also reused question-class definitions and took question examples of Aqualog for building the SCRDR knowledge base.

[4] A rule is composed of a condition part and a conclusion part. A condition is a regular expression pattern over semantic annotations using JAPE grammars. The conclusion contains the question structure and the tuples corresponding to the intermediate representation where each element in the tuple is specified by a newly posted annotations from matching the rule's condition. The default rule typically contains a trivial condition which is always satisfied.

Using rule R1, the knowledge base returns a correct intermediate representation of question-structure *"UnknTerm"* and query-tuple (UnknTerm, QU-who-what, ?, researchers, semantic web research area, ?) for the encountered question.

When it comes to the question *"How many researchers work on AKT project?"* ([RDR1_: [RDR1_QP: how many researchers] [RDR1_Rel: work on] [RDR1_NP: AKT project]]), rule R1 is the fired rule. However, rule R1 gives a wrong conclusion of question-structure *"UnknTerm"* and query-tuple (UnknTerm, QU-howmany, ?, work, AKT project, ?). We can add the following exception rule R4 to correct the conclusion of rule R1 by using constrains via an additional condition:

Rule: R4

({RDR1_}):left --→ :left.RDR4_ = {category1 = "Normal"}

Condition: RDR1_QP.hasAnno == QuestionPhrase.kind == ListWhichHMany

Conclusion: question-structure *"Normal"* and query-tuple (RDR4_.category1, RDR1_QP.QuestionPhrase.category, RDR1_QP, RDR1_Rel, RDR1_NP, ?).

The additional condition of rule R4 matches a RDR1_QP annotation which has a QuestionPhrase annotation covering their substring with *"ListWhichHMany"* as the value of its feature *kind*. The extra annotation constraint *hasAnno* requires that the text covered by the annotation (e.g. RDR1_QP) must contain the specified annotation (e.g. QuestionPhrase). Rule R4 generates the correct output consisting of question-structure *"Normal"* and query-tuple (Normal, QU-howmany, researchers, work, AKT project, ?).

Turning to the question *"which projects are about ontologies and the semantic web?"* ([RDR4_: [RDR1_QP: which projects] [RDR1_Rel: are about] [RDR1_NP: ontologies]] [And: and] [NounPhrase: the semantic web]), it is satisfied by rule R4, nevertheless rule R4 results to an incorrect intermediate representation as RDR4_ annotation only covers a part of the question. The following exception rule R37 is added to rectify the conclusion of the rule R4:

Rule: R37

({RDR4_}{And}({NounPhrase}):NPhrase):left
--→ :left.RDR37_ = {category1 = "UnknRel", category2 = "UnknRel"}
, :NPhrase.RDR37_NP = { }

Condition: RDR1_Rel.hasAnno == Relation.category == Rel-Auxiliary

Conclusion: question-structure *"And"* and two query-tuples (RDR37_.category1, RDR1_QP.QuestionPhrase.category, RDR1_QP, ?, RDR1_NP, ?) and (RDR37_.category2, RDR1_QP.QuestionPhrase.category, RDR1_QP, ?, RDR37_NP, ?).

Rule R37 enables to return a correct intermediate representation for the question with question-structure *"And"* and query-tuples (UnknRel, QU-whichClass, projects, ?, ontologies, ?) and (UnknRel, QU-whichClass, projects, ?, semantic web, ?).

References

1. Nguyen, D.Q., Nguyen, D.Q., Pham, S.B.: A Vietnamese Question Answering System. In: Proc. of KSE'09, IEEE CS (2009) 26–32
2. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In: Proc. of ACL'02
3. Richards, D.: Two decades of ripple down rules research. Knowledge Engineering Review **24**(2) (2009) 159–184
4. Lopez, V., Uren, V., Motta, E., Pasin, M.: AquaLog: An ontology-driven question answering system for organizational semantic intranets. Web Semantics **5**(2) (2007) 72–105
5. Nguyen, D.Q., Nguyen, D.Q., Pham, S.B.: Systematic Knowledge Acquisition for Question Analysis. In: Proc. of RANLP 2011. (September 2011) 406–412

Generating structured Profiles of Linked Data Graphs

Besnik Fetahu[1], Stefan Dietze[1], Bernardo Pereira Nunes[1,3], Davide Taibi[2] and Marco Antonio Casanova[3]

[1] L3S Research Center, Leibniz University Hanover, Germany
{fetahu, nunes, dietze}@L3S.de
[2] Italian National Research Council, Institute for Educational Technologies, Italy
davide.taibi@itd.cnr.it
[3] Department of Informatics - PUC-Rio - Rio de Janeiro, RJ - Brazil
{bnunes, casanova}@inf.puc-rio.br

Abstract. While there exists an increasingly large number of Linked Data, metadata about the content covered by individual datasets is sparse. In this paper, we introduce a processing pipeline to automatically assess, annotate and index available linked datasets. Given a minimal description of a dataset from the DataHub, the process produces a structured RDF-based description that includes information about its main topics. Additionally, the generated descriptions embed datasets into an interlinked graph of datasets based on shared topic vocabularies. We adopt and integrate techniques for Named Entity Recognition and automated data validation, providing a consistent workflow for dataset profiling and annotation. Finally, we validate the results obtained with our tool.

Keywords: Linked Data, Annotation, Datasets, Metadata

1 Introduction

The emergence of the Web of Data, in particularly Linked Data [1], has led to a vast amount of data being available on the Web. The DataHub[1], which serves as the central registry for open Web data, currently contains over 6000 datasets, 338 of which are (at the time of writing) part of the Linked Open Data group[2].

While datasets are highly heterogeneous with respect to represented resource types, currentness, quality or topic coverage, only brief and insufficient structured information about datasets are available. In the case of DataHub, only simple tags, few structured metadata about the size, endpoints or used schemas and a brief textual descriptions are available. This causes significant problems for data consumers (e.g. educational service providers or developers) to identify useful and trust-worthy data for different scenarios.

Nevertheless, earlier works address related issues [2, 3], such as schema alignment and extraction of shared resource annotations across datasets. However, they do not yet facilitate the extraction of reliable dataset metadata with respect

---

[1] `http://www.datahub.io`

[2] `http://datahub.io/group/lodcloud`

to represented topics. In order to address these limitations, we present an approach that automatically and incrementally indexes datasets by interlinking and annotating arbitrary datasets with relevant topics in the form of DBpedia entities and categories. By incrementally computing topic relevance scores for individual datasets, we gradually create a knowledge base of dataset meta-information. To improve scalability the process exploits representative sample sets of resources. Moreover, to ensure high annotation accuracy a semi-automated evaluation approach is proposed.

2 Semi-Automatic Dataset Annotation

Our dataset profiling platform automatically extracts top-ranked topic annotations (DBpedia categories) and captures these together with a relevance score for each dataset description. All dataset descriptions are captured using the VoID schema[3].

2.1 Entity Recognition

The analysis of sampled resources for a set of datasets consists of an annotation process using Named Entity Recognition (NER) and disambiguation tools (DBpedia Spotlight[4]). From each resource we extract the textual content assigned to the following properties: {`rdfs:label`, `rdfs:comment`, `teach:courseTitle`, `teach:courseDescription`, `skos:prefLabel`, `dcterms:description`, `dcterms:alternative`, `dcterms:title`, `bibo:abstract`, `bibo:body`, `cnrb:titolo`, `cnrd:descrizione`, `foaf:name`, `rdf:value`}; and perform contextual, that is resource-wise, NER. This establishes a common descriptive layer of top-ranked entities for each dataset extracted from DBpedia.

As the NER process can pose a bottleneck, we introduce an *incremental annotation* extraction process to alleviate this issue. This process avoids annotating resources similar to previously annotated ones by reusing already obtained annotations. Thus, for a predefined threshold similarity τ, from a pool of existing annotations $\mathcal{A}$, we assign an annotation to a resource if the similarity (resource-annotation) computed by the Jaccard's index is above threshold τ:

$$\forall a \in \mathcal{A}: \; J(r, a) = \frac{|r \cap a|}{|r \cup a|} \tag{1}$$

where $a \in \mathcal{A}$ represents already extracted annotations, while r is a resource instance which is analysed using the *incremental annotation* process.

2.2 Category Annotation

From the extracted annotations (DBpedia entities) $\mathcal{A}$, we analyse the set of assigned categories for each annotation. Such information is extracted from the DBpedia graph via the property `dcterms:subject` representing the topic covered by an entity. Furthermore, we leverage the hierarchical category organisation (as defined by SKOS schema: `skos:broader` and `skos:related`) assigned to entities within DBpedia.

---

[3] `http://www.w3.org/TR/void/`

[4] `http://spotlight.dbpedia.org`

However, such information extracted about categories is only useful when ranked according to their relevance for each dataset. Hence, we compute a normalised *relevance score* for each category assigned to a dataset by taking into (i) entities assigned to a category intra- and inter-datasets; and (ii) number of entities assigned to a dataset and over all datasets, see Equation 2:

$$score(t) = \frac{\Phi(t, D)}{\Phi(\cdot, D)} + \frac{\Phi(t, \cdot)}{\Phi(\cdot, \cdot)}, \ \ \forall t \in \mathcal{T} \wedge D \in \mathcal{D} \tag{2}$$

where $\Phi(\cdot, \cdot)$ represents the number of entities associated with a topic t and for a dataset D, in case of void arguments, it outputs the number of entities in a dataset or over all datasets.

2.3 Automated Annotation Validation & Filtering Approach

Validation and filtering of extracted annotations is necessary, due to noise inherited from NER&NED results. The approach we propose for filtering out noisy annotations takes into account the contextual support given for an annotation from the resource instance it is extracted from. Therefore, we compute a *confidence score* which measures the similarity between an annotation and a resource using Jaccard's index similar to Equation 1, based on values extracted from properties `dbpedia-owl:abstract` and `rdfs:comment`, and the set of analysed properties listed in Section 2.1, respectively.

Whereas, in the validation phase we consider only entities that have a *confidence score* above some pre-define threshold and use human evaluators to assess the relevance of an extracted annotation with respect to the resource context.

3 Results and Evaluation

Our current implementation focuses on educationally relevant datasets as collected in a dedicated group on the DataHub[5] from which we selected a subset of 17 datasets based on their accessibility. Our topic annotation used representative, randomly selected samples of resources from each datasets, with approximately 100 instances for each resource type. Steps included NER, category extraction and threshold-based filtering using our *relevance & confidence scores*.

From the extracted categories based on the resulting annotations, we incorporated only the top-*50* categories being the most representative ones for a dataset based on the computed *normalised-score*. Results obtained from this processing are stored as part of a VoID[6]-based dataset catalog currently being provided as part of the LinkedUp project[7]; a catalogue providing access to such extensive information can be accessed under the following url[8].

The evaluation of annotation accuracy was measured based on two datasets: (a) annotation accuracy without any filtering (see Section 2.3); and (b) annotation accuracy after filtering, where only annotations with scores above some

---

[5] `http://datahub.io/groups/linkededucation`

[6] `http://www.w3.org/TR/void/`

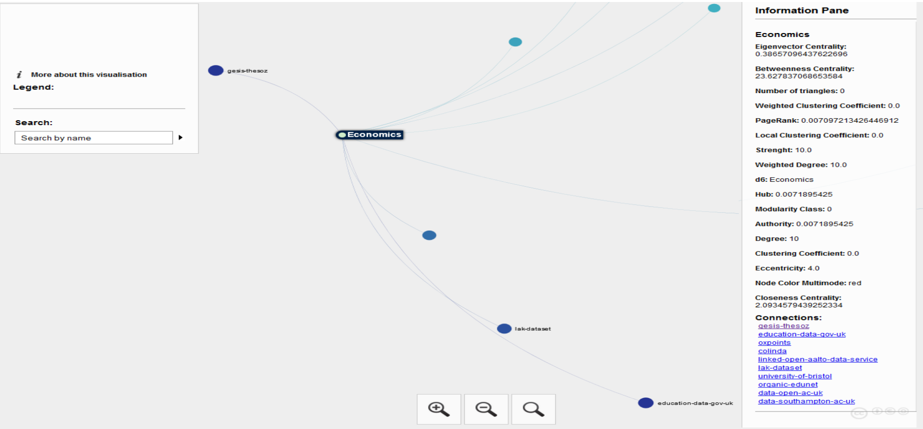
[7] `http://www.linkedup-project.eu`

[8] `http://data.linkededucation.org`

threshold (in our case ≥ 0.15) are considered. The accuracy was measured for **1000** extracted annotations, picked randomly from $\mathcal{A}$. For (a) the accuracy was **71**%, whereas for (b) after filtering annotations below threshold $\tau \geq 0.15$. We observed an increase in accuracy of almost +**10**%.

Our demo application[9] focuses mainly on representation, profiling and search functionalities of the analysed datasets based on the structured descriptions. Figure 1 shows a screenshot of the exploratory search functionality of datasets using extracted annotations and categories. The user interface provides the following:

- Exploratory search of datasets based on extracted annotations & categories
- Interlinking of datasets based on most representative categories
- List of ranked categories for each dataset



Economics

Eigenvector Centrality: 0.38657096437622696

Betweenness Centrality: 23.627837068653584

Number of triangles: 0

Weighted Clustering Coefficient: 0.0

PageRank: 0.007097213426446912

Local Clustering Coefficient: 0.0

Strenght: 10.0

Weighted Degree: 10.0

d6: Economics

Hub: 0.0071895425

Modularity Class: 0

Authority: 0.0071895425

Degree: 10

Clustering Coefficient: 0.0

Eccentricity: 4.0

Node Color Multimode: red

Closeness Centrality: 2.0934579439252334

Connections:
gesis-thesoz
education-data-gov-uk
oxpoints
colinda
linked-open-aalto-data-service
lak-dataset
university-of-bristol
organic-edunet
data-open-ac-uk
data-southampton-ac-uk

Fig. 1. Screenshot of the profiling of Linked Data demo, with an example category interlinking different datasets shown on the right hand side panel.

4 Future Work

Our current processing pipeline is able to extract topic annotations for arbitrary Linked Data with only minimal manual intervention. Having applied it to a small subset of available datasets, our future work aims at the automatic profiling of all available LOD datasets, towards providing a more descriptive catalog of Linked Datasets.

Acknowledgements. This work was partly funded by the LinkedUp (GA No:317620) and DURAARK (GA No:600908) projects under the FP7 programme of the European Commission.

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. d'Aquin, A. Adamou, and S. Dietze. Assessing the educational linked data landscape. In *WebSci*, pages 43–46. ACM, 2013.
3. D. Taibi, B. Fetahu, and S. Dietze. Towards integration of web data into a coherent educational data graph. In *WWW (Companion Volume)*, pages 419–424, 2013.

---

[9] `http://l3s.de/~fetahu/iswc_demo/`

ActiveRaUL: A Web form-based User Interface to create and maintain RDF data

Anila Sahar Butt[1,2], Armin Haller[1], Shepherd Liu[1], and Lexing Xie[2]

[1] CSIRO ICT Centre, `firstname.lastname@csiro.au`
[2] Australian National University, `firstname.lastname@anu.edu.au`

Abstract. With the advent of Linked Data the amount of automatically generated machine-readable data on the Web, often obtained by means of mapping relational data to RDF, has risen significantly. However, *manually created*, *quality-assured* and *crowd-sourced* data based on ontologies is not available in the quantities that would realise the full potential of the semantic Web. One of the barriers for semantic Web novices to create machine-readable data, is the lack of easy-to-use Web publishing tools that separate the schema modelling from the data creation. In this demonstration we present ActiveRaUL, a Web service that supports the automatic generation of Web form-based user interfaces from any input ontology. The resulting Web forms are unique in supporting users, inexperienced in semantic Web technologies, to create and maintain RDF data modelled according to an ontology. We report on a use case based on the Sensor Network Ontology that supports the viability of our approach.

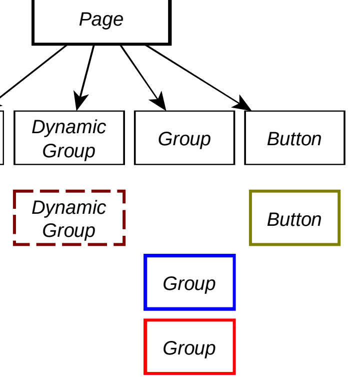
1 Introduction

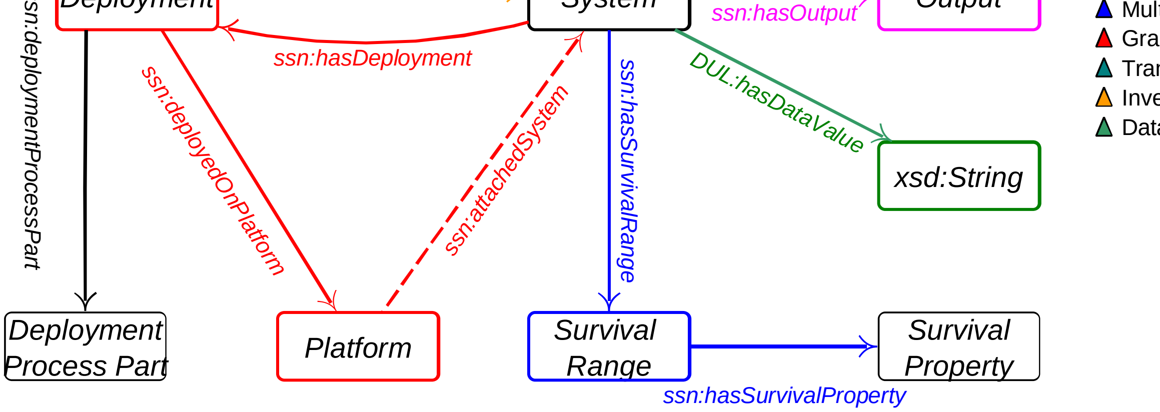
The tools of choice for creating quality-assured ontology instances (the so-called *ABox*) are still ontology editors such as WebProtégé [4]. However, creating the *ABox* in an ontology editor requires some degree of understanding of RDF(s) and OWL since the user has to define to which class an individual belongs to and what are the permissible relationships between individuals. To address this issue, some Web publishing tools on top of Wikis, Microblogs or Content Management systems have been developed (e.g. the work discussed in [1], [5] and [3]) that allow a user to exclusively create ontology instances. However, they are mostly developed for a specific domain (i.e. specific ontologies) and often do not strictly follow OWL semantics and consequently allow the creation of a logically inconsistent ABox.

In this demonstration we will showcase ActiveRaUL [6], a Web service that operates on a model defined according to the RDFa User Interface Language (RaUL)[3] that consists of two parts, [1.] a *form model* describing the structure of a Web form with different types of form controls, such as *Textboxes*, *Radiobuttons*, *Listboxes* etc., and their associated operations (`CREATE`, `READ`, `UPDATE` or `DELETE`) and [2.] a *data model* defining the structure of the exchanged data as RDF statements which are referenced from the *form model* via a data binding mechanism. The ActiveRaUL Web service also provides functionality to automatically generate a Web form-based user interface according to the RaUL ontology from arbitrary ontologies. We argue that the resulting user interfaces are easier-to-use for a semantic Web novice to create

---

[3] See `http://purl.org/NET/raul#`





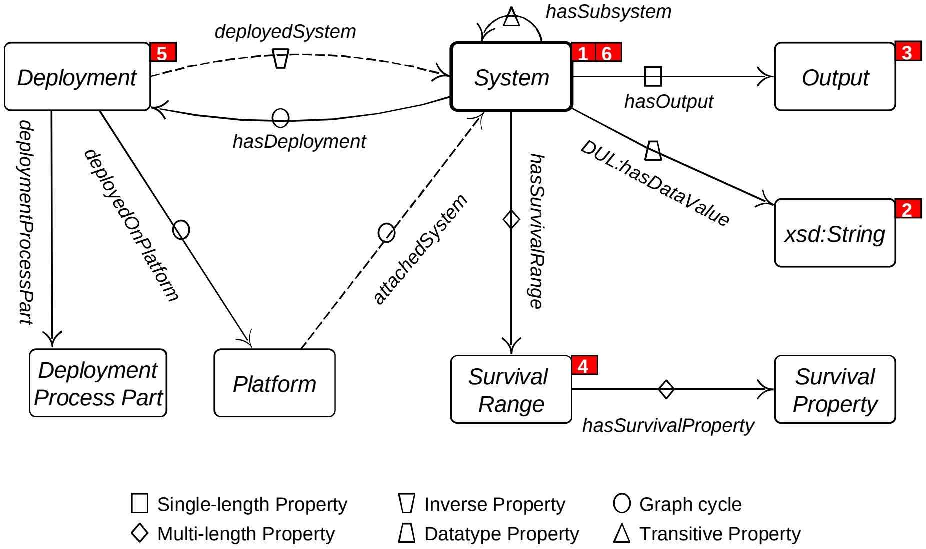
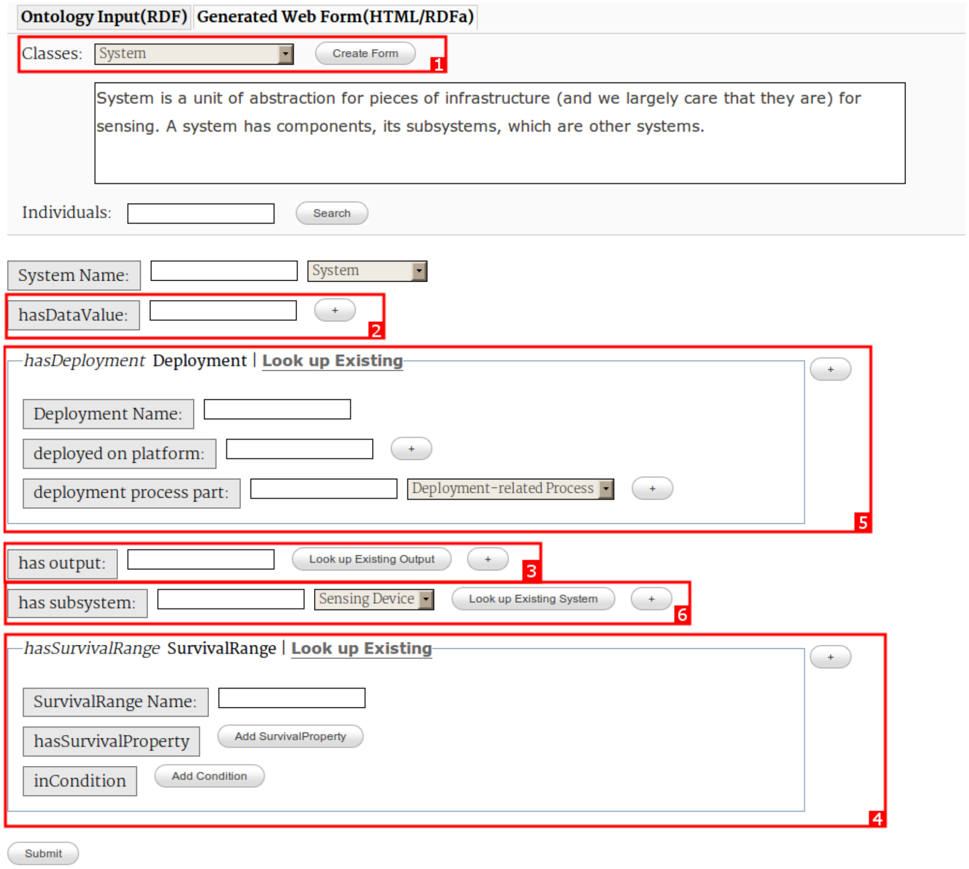


Fig. 1. Sub-graph structure for the System class in the SSN ontology

To relieve a Web developer from manually defining a Web form model according to the RaUL ontology, we have extended the ActiveRaUL service with a deployment endpoint that upon invocation generates RaUL Web forms from an arbitrary user submitted ontology. The biggest challenge in automatically creating such Web forms from an ontology is the mismatch between the graph nature of RDF and the tree structure of a Web form. In the algorithm implemented in ActiveRaUL we distinguish six different types of sub-graphs occurring in ontologies and introduce decision controls to map these sub-graphs to useable web forms. We will demonstrate these different types of mapping on a use-case based on the the Semantic Sensor Network (SSN) ontology that can be used to describe the capabilities of sensors, the measurement processes used and the resultant observations. Figure 1 shows the "System" class of the SSN ontology and its relations to other classes, whereas Figure 2 shows a screenshot of a generated Web form by ActiveRaUL of the "System" class. The numbers 1–6 in both figures indicate the six different types of sub-graphs we distinguish in the algorithm (see Figure 1) and how they are displayed in the Web form (see Figure 2).



Classes: System

System is a unit of abstraction for pieces of infrastructure (and we largely care that they are) for sensing. A system has components, its subsystems, which are other systems.

Fig. 2. Screenshot of ActiveRaUL generated Web form for the `ssn:System` class

3 Evaluation

We compared ActiveRaUL to the widely used state-of-the-art ontology editing tool, WebProtégé. The demonstration deployment of ActiveRaUL set up for the user study already pre-loading the SSN ontology is available at: `http://www.activeraul.org/demo/index.html` From the university deployment example defined by the SSN working group we extracted three test cases, each with a number of tasks. We asked users to model these test cases in WebProtégé and ActiveRaUL. For evaluating the two systems we considered three usability metrics, (1) the effectiveness of the system in supporting the user to complete the task measured by the accuracy of the resulting models; (2) the efficiency of the users in using the system measured by the time they spent on completing a task and (3) a user's subjective reactions using the system measured by the widely-used System Usability Scale (SUS) [2]. In the following we briefly outline the results of our user study. These results are based on the performance and feedback of twelve participants: five of which, based on their self-assessment, were categorised into the *semantics experienced user group*, and seven categorised into the *semantics inexperienced user group*.

Table 1. Overall accuracy in completing test cases in WebProtégé and ActiveRaUL

	WebProtégé	ActiveRaUL
Exp. Users	82.05%	91.03%
Inexp. Users	76.92%	87.91%
All Users	82.05%	91.03%

Accuracy: Table 1 shows the overall accuracy over the three test cases which shows that the participants performed better in ActiveRaUL, managing to create 91% correct triples compared to 82% in WebProtégé. For ActiveRaUL, the accuracy of the participants was already very high in the first test case, even though no participant has ever used the system before. This confirms our hypothesis that a Web form-based user interface is familiar enough to computer literate users to create RDF data correctly, even if the participants are inexperienced in semantic Web technologies.

Table 2. Average times (mm:ss) to complete test cases in WebProtégé and ActiveRaUL

	Test Case 1		Test Case 2		Test Case 3	
	WebProtégé	ActiveRaUL	WebProtégé	ActiveRaUL	WebProtégé	ActiveRaUL
Exp. Users	3:46	1:50	5:13	1:50	10:24	4:01
Inexp. Users	4:05	2:17	5:23	1:29	11:26	4:08
All Users	3:57	2:05	5:19	1:38	11:00	4:05

Efficiency: Table 2 shows the average times participants required to complete a test case. Both participant groups, inexperienced and experienced, were significantly faster (between 27% and 56% faster) completing the test cases in ActiveRaUL compared to WebProtégé.

Usability: After completion of the three test cases in both systems we asked the participants to rate their subjective reactions on the usability of the systems on a five-point Likert scale as required by the SUS methodology. SUS yields a single number representing a composite measure of the usability of a system with scores in the range from 0 to 100, 100 being the best score. Overall ActiveRaUL scored 72.1 out of 100 points compared with 32.5 for WebProtégé, indicating that the participants found ActiveRaUL easier to use than WebProtégé for the creation of ontology instances.

Concluding, our user study proved that ActiveRaUL is indeed easier, more effective and more efficient to use *for the creation of RDF data* than the state-of-the-art ontology editing tool.

References

1. J. Baumeister, J. Reutelshoefer, F. Puppe. KnowWE: a Semantic Wiki for knowledge engineering. *Applied Intelligence*, 35:323–344, 2011.
2. J. Brooke. SUS - A quick and dirty usability scale. In P. W. Jordan, B. Thomas, B. A. Weermeester, A. L. McClelland, editors, *Usability Evaluation in Industry.* Taylor and Francis, London, 1996.
3. S. Corlosquet, R. Delbru, T. Clark, A. Polleres, S. Decker. Produce and Consume Linked Data with Drupal! In *Proceedings of ISWC*, pages 763–778, 2009.
4. T. Tudorache, C. Nyulas, N. F. Noy, M. A. Musen. WebProtégé: A Collaborative Ontology Editor and Knowledge Acquisition Tool for the Web. *Semantic Web* 4(1), 2013.
5. A. Passant, J. G. Breslin, S. Decker. Open, distributed and semantic microblogging with smob. In *Proceedings of ICWE 2010*, pages 494–497, 2010.
6. A. Haller, T. Groza, and F. Rosenberg. Interacting with Linked Data via Semantically Annotated Widgets. In *Proceedings of JIST*, pages 300–317, 2011.

XLore: A Large-scale English-Chinese Bilingual Knowledge Graph

Zhigang Wang†, Juanzi Li†, Zhichun Wang‡, Shuangjie Li†, Mingyang Li†,
Dongsheng Zhang†, Yao Shi†, Yongbin Liu†, Peng Zhang†, and Jie Tang†

† DCST, Tsinghua University, P.R. China
{wzhigang, ljz, lsj, lmy, zds, sy, lyb, zp,
tangjie}@keg.cs.tsinghua.edu.cn
‡ CIST, Beijing Normal University, P.R. China
zcwang@bnu.edu.cn

Abstract. Current Wikipedia-based multilingual knowledge bases still suffer the following problems: (i) the scarcity of non-English knowledge, (ii) the noise in the semantic relations and (iii) the limited coverage of equivalent cross-lingual entities. In this demo, we present a large-scale bilingual knowledge graph named XLore, which has adequately solved the above problems.

1 Introduction

Multilingual knowledge bases are important for the globalization of knowledge sharing. Knowledge bases such as DBpedia[1], YAGO[2], and BabelNet[3] are mainly built upon the multilingual Wikipedia. Some problems are to be addressed: (i) The imbalanced sizes of different Wikipedia language versions lead to the highly imbalanced knowledge distribution in different languages. Knowledge encoded in non-English languages is much less than those in English. (ii) The inconsistency of the large category system in Wikipedia causes incorrect semantic relations between concepts that are defined based on categories. For example, "Wikipedia-books-on-people is the `subCategoryOf` People" will lead to the wrong "Wikipedia-books-on-people is `subClassOf` People" in DBpedia's SKOS schema. (iii) Integrated by directly using cross-lingual links in Wikipedia, the amount of integrated multilingual knowledge totally depends on these existing cross-lingual links.

In this demo, we present an English-Chinese bilingual knowledge graph named XLore to adequately solve the above problems. We use much larger heterogenous online wikis to enrich the Chinese knowledge, utilize a classification-based method to correctly semantify the wikis' category systems, and employ a cross-lingual knowledge linking approach to find new cross-lingual links between entities. Besides, we use a cross-lingual structured knowledge extraction method to enrich the semantic relations.

---

[1] `http://dbpedia.org/`

[2] `http://www.mpi-inf.mpg.de/yago-naga/yago/`

[3] `http://lcl.uniroma1.it/babelnet/`

To the best of our knowledge, XLore is the first large-scale cross-lingual knowledge graph with balanced amount of Chinese-English knowledge. XLore gives a new way for building such a knowledge graph across any two languages.

2 Approach

As shown in Figure 1, the building of XLore contains three stages: (1) *Data Preprocessing*: First we collect and clean the data sets from four online wikis, namely English Wikipedia, Chinese Wikipedia, Baidu Baike and Hudong Baike. (2) *Knowledge Graph Building*: Next, we learn the cross-lingual ontology as follows: semantify the online wikis to predict correct semantic relations, conduct cross-lingual knowledge linking to integrate the heterogenous wikis together, and extract the structured knowledge to enrich more relations in the graph. (3) *Knowledge Query*: Finally, we construct an online system for knowledge acquisition.

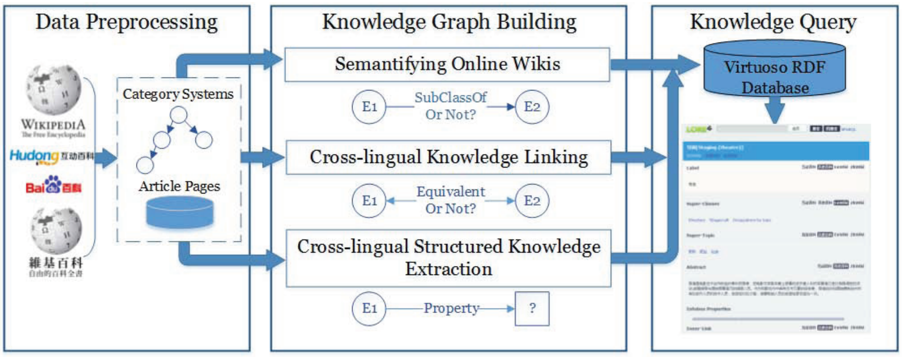


Fig. 1. Overview

Semantifying Online Wikis To semantify the online wikis, we are to predict the correct `subClassOf` and `instanceOf` relations between two entities. We view both the correct `subClassOf` and the `instanceOf` relations as `is-a` relations. Table 1 shows some examples about the semantic relations.

Table 1. Examples of Semantic Relations

Entity 1	Relation	Entity 2	Right or Wrong
European Microstates	`instanceOf`	Microstates	Right
European Microstates	`instanceOf`	Europe	Wrong
教育人物(Educational Person)	`subClassOf`	人物(Person)	Right
教育人物(Educational Person)	`subClassOf`	教育(Education)	Wrong

Formally, we learn two series of functions g_1 (for English) and g_2 (for Chinese) to predict the probabilities to be an `is-a` relation between two entities. We define some literal and structural features and train the Logistic Regression models. The most important features are the head words' singular/plural forms of English entities and the substring relationship between the labels of Chinese entities. By iteratively expanding the training data sets, both the functions achieve over 90.48% F1-score. To keep the semantic relatedness, we treat the incorrect relations as the `subTopicOf` relations and import these relations into the RDF database too.

Cross-lingual Knowledge Linking via Concept Annotation To integrate the knowledge in different languages, we proposed learning based approaches for linking equivalent entities in different languages [1][2]. Several features are defined based on the link structures in wikis to assess the similarities between two different entities. Then learning models are trained based on the already known cross-lingual links in Wikipedia, which afterward predict new equivalent entity pairs. In order to find desired number of new cross-lingual links, we use concept annotation to enrich the inner links within wikis, which improves the knowledge linking approach considerably. The knowledge linking process as a whole can execute iteratively, resulting in large number of new cross-lingual links.

Cross-lingual Structured Knowledge Extraction To enrich kinds of relations in XLore, we apply our cross-lingual knowledge extraction framework named WikiCiKE to complete the missing infoboxes [3]. WikiCiKE is based on the hypothesis: one can use the rich auxiliary (e.g. English) information to assist the target (e.g. Chinese) infobox extraction. We treat this task as a transfer learning-based binary classification problem. Given an attribute in the target wiki, WikiCiKE automatically generates the cross-lingual training data and learns the extractor using TrAdaBoost model. Finally, WikiCiKE uses the learned extractor to extract the missing value from the unstructured article texts. Our experiments in [3] demonstrate that WikiCiKE significantly outperforms the monolingual knowledge extraction method and the translation-based method.

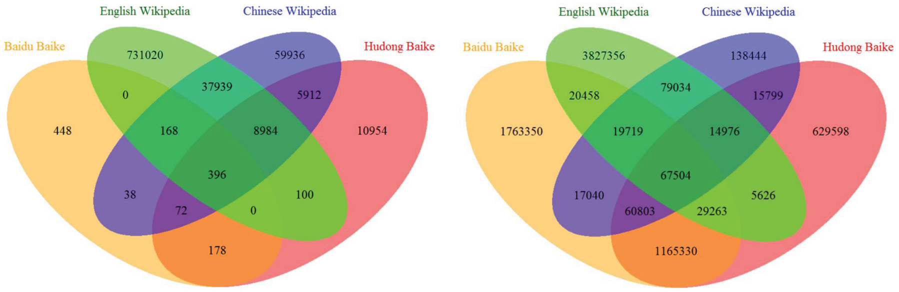
3 XLore System

We construct a unified knowledge graph in the form of RDF and use the OpenLink Virtuoso server[4] for systematical data management. Using the proposed approach, XLore harvests 856,146 classes , 71,596 properties and 7,854,301 instances across English and Chinese. Figure 2 gives a brief statistics the number of linked entities from different online wikis.

We also deploy an online system to illustrate our XLore. As shown in Figure 3, the system supports the keyword-based or SPARQL queries, gives the statistical information, offers visualization demonstrations, etc. A live demonstration of the system can be found at `http://www.youtube.com/watch?v=QKA-RYFfztA`. We invite the readers to try our XLore prototype at `http://xlore.org`.

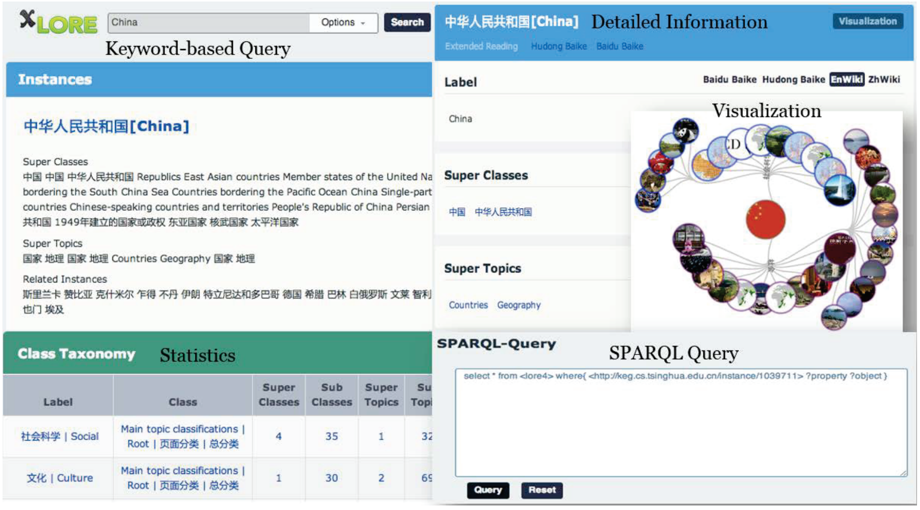
---

[4] `http://virtuoso.openlinksw.com/`



(a) *Number of Linked Concepts* (b) *Number of Linked Instances*

Fig. 2. Statistics of the Linked Entities.



中华人民共和国[China]

Super Classes
中国 中国 中华人民共和国 Republics East Asian countries Member states of the United Na
bordering the South China Sea Countries bordering the Pacific Ocean China Single-part
countries Chinese-speaking countries and territories People's Republic of China Persian
共和国 1949年建立的国家或政权 东亚国家 核武国家 太平洋国家

Super Topics
国家 地理 国家 地理 Countries Geography 国家 地理

Related Instances
斯里兰卡 赞比亚 克什米尔 乍得 不丹 伊朗 特立尼达和多巴哥 德国 希腊 巴林 白俄罗斯 文莱 智利
也门 埃及

Label	Class	Super Classes	Sub Classes	Super Topics	Su Topi
社会科学 | Social	Main topic classifications | Root | 页面分类 | 总分类	4	35	1	32
文化 | Culture	Main topic classifications | Root | 页面分类 | 总分类	1	30	2	69

```
select * from <lore4> where{ <http://keg.cs.tsinghua.edu.cn/instance/1039711> ?property ?object }
```

Fig. 3. Interface of XLore System

References

1. Wang, Z., Li, J., Wang, Z., Tang, J.: Cross-lingual knowledge linking across wiki knowledge bases. WWW'12
2. Wang, Z., Li, J., Tang, J.: Boosting cross-lingual knowledge linking via concept annotation. IJCAI'13
3. Wang, Z., Li, Z., Li, J., Tang, J., Z.Pan, J.: Transfer learning based cross-lingual knowledge extraction for wikipedia. ACL'13

Git2PROV: Exposing Version Control System Content as W3C PROV

Tom De Nies[1], Sara Magliacane[2], Ruben Verborgh[1], Sam Coppens[1], Paul Groth[2], Erik Mannens[1], and Rik Van de Walle[1]

[1] Ghent University - iMinds - Multimedia Lab
`{tom.denies,ruben.verborgh,sam.coppens,`
`erik.mannens, rik.vandewalle}@ugent.be`
[2] VU University Amsterdam
`{s.magliacane,p.t.groth}@vu.nl`

Abstract. Data provenance is defined as information about entities, activities and people producing or modifying a piece of data. On the Web, the interchange of standardized provenance of (linked) data is an essential step towards establishing trust [2]. One mechanism to track (part of) the provenance of data, is through the use of version control systems (VCS), such as Git. These systems are widely used to facilitate collaboration primarily for both code and data. Here, we describe a system to expose the provenance stored in VCS in a new standard Web-native format: W3C PROV [4]. This enables the easy publication of VCS provenance on the Web and subsequent integration with other systems that make use of PROV. The system is exposed as a RESTful Web service, which allows integration into user-friendly tools, such as browser plugins.

1 Introduction

Version control systems (VCS) have a long history in computing. The first such system was the Source Code Control System, developed in 1972 [10]. Nowadays, VCS are widely popular and becoming more so with the advent of cloud-based services, such as Github[3] and Bitbucket[4], that both simplify the management of the VCS and expose their information through Web interfaces. For example, Github has over 3 million users and maintains over 6 million repositories.[5]

Versioning of data is an aspect of *provenance*: information about entities, activities and people producing or modifying a piece of data. Provenance is critical in contexts ranging from scientific reproducibility to journalism. Furthermore, a number of sub-disciplines of computer science, including databases, distributed systems and the Web have been addressing issues related to provenance [8]. Provenance is particularly important to the Semantic Web community because

---

[3] `http://www.github.com`

[4] `http://www.bitbucket.com`

[5] See http://thenextweb.com/insider/2013/04/11/code-sharing-site-github-turns-five-and-hits-3-5-million-users-6-million-repositories/

of the need to ascertain the trust of data originating from multiple interlinked sources [5]. Because of the importance of provenance to the Web, the W3C produced the PROV recommendations for the interchange of provenance on the Web [4]. PROV was recently released and already has over 60 implementations and is in use by several Linked Data datasets.

Thus, the aim of the system described in this paper is to enable the provenance within VCS to be exposed in a *Web-native and interoperable format*. This provenance can then be consumed by other PROV enabled tools. Indeed, given that the software used to create many Linked Data sets is available through public version control systems[6], we believe that the tool can be used to enrich the provenance of many of these datasets.

The rest of this paper is organized as follows. We briefly discuss related work and present a mapping from the Git version control system to PROV. This is followed by a description of the implementation and demonstration of our system. In particular, our demo illustrates how the resulting information can be consumed by other PROV enabled systems. The demonstration (live and video) is available at the following URI: `http://git2prov.org`.

2 Related work

This paper is part of a greater effort to create more interoperability among different platforms that track provenance information, by mapping them to a standard interlingua such as [4]. Two relevant examples are the Dublin Core Mapping to PROV [1], and the mapping of the revision history of Wikipedia to OPM (an ancestor of PROV) [9].

Currently, the most commonly used VCS is Git, an example of a distributed version control system. Due to brevity we will omit a detailed explanation of Git, and refer to [6] for an overview.

3 A Mapping from VCS to PROV

Our mapping, shown in Fig. 1, was created by identifying whether the data could represent information from one or more broad classes of provenance information. The three classes we used are identified below. For each class, we describe how provenance can be expressed using concepts from the PROV Data Model [7].

- Dependency - a dependency between two objects expressed as the relationship between two prov:Entity objects using prov:wasDerivedFrom and prov:specializationOf , e.g. a file f_c was derived from another previous file f_{c-1}, both are a specialization of a certain file f;
- Activities - a process expressed as a prov:Activity that connects two prov:Entity objects, expressed through prov:used and prov:wasGeneratedBy relations, e.g. a commit c uses a file f_{c-1} and generates a file f_c;

---

[6] e.g. `https://github.com/dbpedia` or `https://github.com/jimmccusker/twc-healthdata`

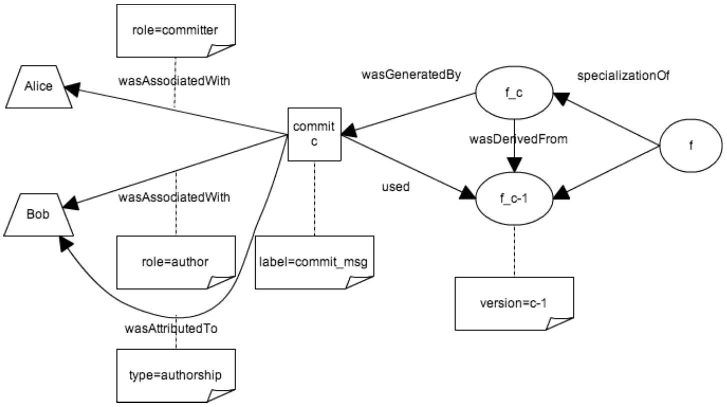


Fig. 1. Mapping of Git operations to PROV concepts. Note that the Activity *Start* and *End* concepts of PROV are not depicted, and correspond to, respectively, the author time and the commit time of each commit.

– Attribution - attribution information expressed as the prov:Agent that created the Entity using the prov:wasAttributedTo and prov:wasAssociatedWith relations, modeling the two potentially distinct roles of a author and a committer.

These classes reflect the three use-case perspectives on provenance identified by the W3C Provenance Primer [3]: object-oriented, process-oriented and agent-oriented.

4 Implementation

Because we want our conversion tool to be as flexible as possible, we chose to build a RESTful git2prov Web Service for this purpose, using the Node.js[7] framework. The service is available at the following URI: `http://git2prov.org/git2prov?gituri=<your_git_uri>`.

The only required input for this service is a URL **giturl** that refers to a git repository. In this first proof-of-concept implementation, only openly accessible repositories are supported. However, adding support for secure repositories is part of our future work. In addition to **giturl**, the service accepts a number of optional parameters, with the default value in bold:

serialization *(possible values: [**PROV-N**, PROV-O, PROV-JSON, PROV-XML])* This parameter is used to specify the desired PROV serialization.

shortHashes *(possible values: [**false**, true])* This parameter forces the service to use the short commit hash in the exported provenance, to increase readability for human users

---

[7] http://nodejs.org

ignore *(possible values: a provenance relation)* This parameter is used to filter the specified relation from the converted provenance.

Note that each provenance document generated by the git2prov service includes a link to the complete document (without any restricting parameters).

Upon receiving a request, the service *clones* the git repository to a temporary location, and performs a *git log* command on it. The output of this log is then *mapped* to PROV as described in Sect. 3, and written to the HTTP response in the requested serialization. To demonstrate our service, we wrote a small web application, available at `http://git2prov.org`. A video is also available at `http://vimeo.com/70980809`.

5 Conclusions and Future Work

We believe that systems such as Git2PROV have the potential to become an important enabler of the widespread interchange of standardized provenance. With our proof-of-concept implementation, we have shown that it is certainly feasible to build a lightweight RESTful Web service to convert versioning systems into PROV. In future work, we aim to improve our work by including more semantic annotations in combination with the provenance to allow further reasoning over it, with the prospect of deriving trust assessments.

Acknowledgements These research activities were funded by Ghent University, iMinds, the IWT Flanders, the FWO-Flanders, and the European Union, as well as the Data2Semantics project in the Dutch national program COMMIT.

References

[1] Daniel Garijo, K.E.: Dublin core to prov mapping - w3c working group note
[2] De Nies, T., Coppens, S., Verborgh, R., Vander Sande, M., Mannens, E., Van de Walle, R., Michaelides, D., Moreau, L.: Easy Access to Provenance: an Essential Step Towards Trust on the Web. In: METHOD 2013, Kyoto, Japan (2013)
[3] Gil, Y., Miles, S., et al.: PROV Model Primer. W3C Working Group Note (2013)
[4] Groth, P., Moreau, L.: PROV-Overview: An Overview of the PROV Family of Documents. W3C Working Group Note (2013)
[5] Groth, P., Gil, Y.: Editorial - using provenance in the semantic web. Web Semantics: Science, Services and Agents on the World Wide Web 9(2) (2011), `http://www.websemanticsjournal.org/index.php/ps/article/view/196`
[6] Loeliger, J.: Version Control with Git: Powerful Tools and Techniques for Collaborative Software Development. O'Reilly Media, Inc. (2009)
[7] Moreau, L., Missier, P., (Eds.) et al: PROV-DM: The PROV Data Model. W3C Recommendation (2013)
[8] Moreau, L.: The foundations for provenance on the web. Foundations and Trends in Web Science 2(2–3), 99–241 (2010)
[9] Orlandi, F., Passant, A.: Modelling provenance of DBpedia resources using wikipedia contributions. Web Semantics pp. 149 – 164 (2011)
[10] Rochkind, M.J.: The source code control system. Software Engineering, IEEE Transactions on (4), 364–370 (1975)

Publishing Data from the Smithsonian American Art Museum as Linked Open Data*

Craig A. Knoblock[1], Pedro Szekely[1], Shubham Gupta[1], Animesh Manglik[1],
Ruben Verborgh[2], Fengyu Yang[3], and Rik Van de Walle[2]

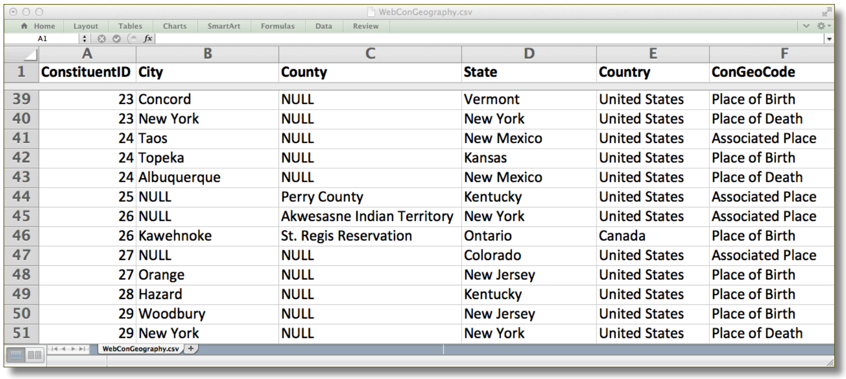
[1] University of Southern California
Information Sciences Institute and Department of Computer Science, USA
[2] Multimedia Lab – Ghent University – iMinds, Belgium
[3] Nanchang Hangkong University, Nanchang, China
{knoblock,pszekely,shubhamg}@isi.edu, manglik@usc.edu,
{ruben.verborgh,rik.vandewalle}@ugent.be, frueyang@gmail.com

Abstract. Museums around the world have built databases with metadata about millions of objects, the people who created them, and the entities they represent. This data is stored in proprietary databases and is not readily available for use. Recently, museums embraced the Semantic Web as a means to make this data available to the world, but the experience so far shows that publishing museum data to the Linked Data cloud is difficult: the databases are large and complex, the information is richly structured and varies from museum to museum, and it is difficult to link the data to other datasets. We have been collaborating with the Smithsonian American Art Museum to create a set of tools that allow museums and other cultural heritage institutions to publish their data as Linked Open Data. In this demonstration we show the end-to-end process of starting with the original source data, modeling the data with respect to a ontology of cultural heritage data, linking the data to DBpedia, and then publishing the information as Linked Data.

Recently there have been a number of efforts to publish metadata about the objects in museums as Linked Open Data (LOD). Some notable efforts include the Euopeana project [2], which published data on 1,500 of Europe's museums, libraries, and archives, the Amsterdam Museum [1], which published data on 73,000 objects, and the LODAC Museum [4], which published data from 114 museums in Japan. Despite the many recent efforts, there are still significant challenges in publishing data about artwork to the Linked Data cloud. In particular, the past work requires a user to manually convert their data into RDF and there has been almost no work on actually linking the cultural heritage data across sources. The contributions of this work are that we developed a very quick and efficient method for mapping museum data to a cultural heritage ontology and we created tools for linking and validating the links to other sources [5].

---

* This research was funded by the Smithsonian American Art Museum. A demonstration video is available at: http://youtu.be/1Vaytr09H1w



	A	B	C	D	E	F
1	ConstituentID	City	County	State	Country	ConGeoCode
39	23	Concord	NULL	Vermont	United States	Place of Birth
40	23	New York	NULL	New York	United States	Place of Death
41	24	Taos	NULL	New Mexico	United States	Associated Place
42	24	Topeka	NULL	Kansas	United States	Place of Birth
43	24	Albuquerque	NULL	New Mexico	United States	Place of Death
44	25	NULL	Perry County	Kentucky	United States	Associated Place
45	26	NULL	Akwesasne Indian Territory	New York	United States	Associated Place
46	26	Kawehnoke	St. Regis Reservation	Ontario	Canada	Place of Birth
47	27	NULL	NULL	Colorado	United States	Associated Place
48	27	Orange	NULL	New Jersey	United States	Place of Birth
49	28	Hazard	NULL	Kentucky	United States	Place of Birth
50	29	Woodbury	NULL	New Jersey	United States	Place of Birth
51	29	New York	NULL	New York	United States	Place of Death

Fig. 1. Source data from the Smithsonian

Datasets. In this demonstration we will use datasets extracted from Smithsonian American Art Museum collection management database. Figure 1 shows a sample of raw data about artists. One of the challenges in mapping this particular table to RDF is that the meaning of each row is encoded in the last column of the table, which shows whether the data specifies the place of birth, place of death, or an associated place. Each row must be mapped to a different property in the ontology.

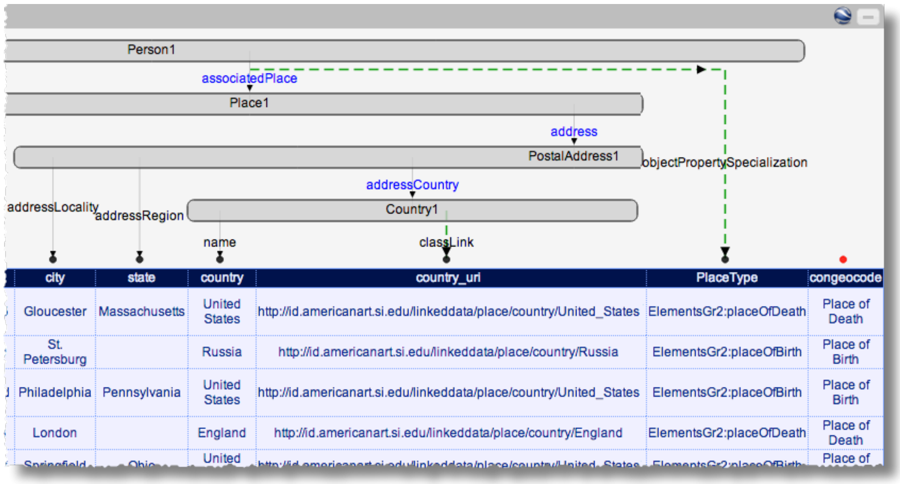
Mapping the Data to RDF. The first step in the process is to model the raw data using an ontology of cultural heritage data. We developed a tool called Karma [3], which semi-automatically builds a semantic description of a data source using machine learning techniques. Karma makes it possible to quickly model the dataset shown in Figure 1. The resulting model, shown in Figure 2, illustrates several capabilities needed for real datasets, such as customizing the URL generation and generating different properties for different rows in a table. After modeling the data, users can ask Karma to publish the model as R2RML and publish the data as RDF.

Linking to External Datasets. Once the RDF is published, the next step is to link the information about entities, including artists and locations, to the corresponding entities in other sources. In particular we link artists to people in DBpedia and locations to the corresponding locations in Geonames. The name fields of the artists, possibly augmented with other identifying information such as birth year, is sent to a reconciliation service that has a fuzzy index of DBpedia information. Figure 3 shows a user invoking the reconciliation service on the Person class, and it shows the resulting links and scores for each match to DBpedia. The interface also allows a user to verify each of the links and to drill down into the individual data sources to determine whether the links are correct.

Publishing the Linked Data Once the links have been verified, the information is then published and made available as Linked Open Data. We created a user-friendly version of Pubby[4] that describes the content and shows the images

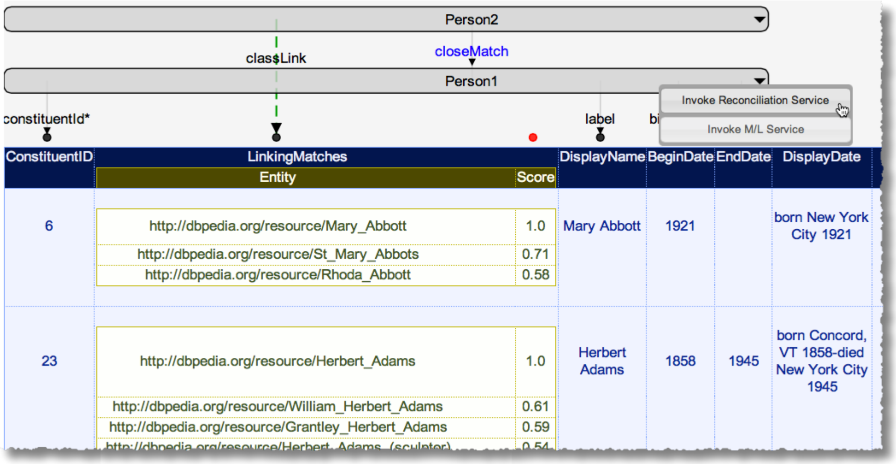
---

[4] `http://wifo5-03.informatik.uni-mannheim.de/pubby/`



city	state	country	country_uri	PlaceType	congeocode
Gloucester	Massachusetts	United States	http://id.americanart.si.edu/linkeddata/place/country/United_States	ElementsGr2:placeOfDeath	Place of Death
St. Petersburg		Russia	http://id.americanart.si.edu/linkeddata/place/country/Russia	ElementsGr2:placeOfBirth	Place of Birth
Philadelphia	Pennsylvania	United States	http://id.americanart.si.edu/linkeddata/place/country/United_States	ElementsGr2:placeOfBirth	Place of Birth
London		England	http://id.americanart.si.edu/linkeddata/place/country/England	ElementsGr2:placeOfDeath	Place of Death
Springfield	Ohio	United	http://id.americanart.si.edu/linkeddata/place/country/United_States	ElementsGr2:placeOfBirth	Place of

Fig. 2. Source model interactively generated using Karma

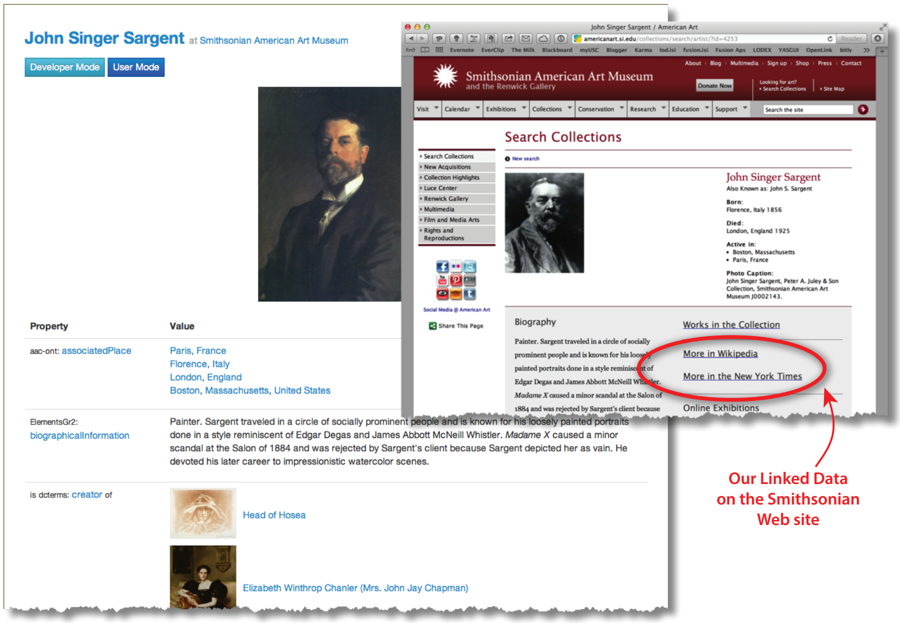


ConstituentID	LinkingMatches		DisplayName	BeginDate	EndDate	DisplayDate
	Entity	Score				
6	http://dbpedia.org/resource/Mary_Abbott	1.0	Mary Abbott	1921		born New York City 1921
	http://dbpedia.org/resource/St_Mary_Abbots	0.71				
	http://dbpedia.org/resource/Rhoda_Abbott	0.58				
23	http://dbpedia.org/resource/Herbert_Adams	1.0	Herbert Adams	1858	1945	born Concord, VT 1858-died New York City 1945
	http://dbpedia.org/resource/William_Herbert_Adams	0.61				
	http://dbpedia.org/resource/Grantley_Herbert_Adams	0.59				
	http://dbpedia.org/resource/Herbert_Adams_(sculptor)	0.54				

Fig. 3. Screen shot of the interface for linking the museum data to DBPedia

and labels directly on the page to make the resulting linked data more readable. Figure 4 shows a screen shot of the published data from the same data source shown in Figure 1.

Using the Linked Data Once the linking is complete, the Linked Data can also be used to augment other sources of data. In our project with the Smithsonian, they found that one use of the Linked Data is to augment their current Web pages with the additional information available from Wikipedia and the New York Times, which follows directly from the linking of their data about artists



John Singer Sargent at Smithsonian American Art Museum

Property	Value
aac-ont: associatedPlace	Paris, France Florence, Italy London, England Boston, Massachusetts, United States
ElementsGr2: biographicalInformation	Painter. Sargent traveled in a circle of socially prominent people and is known for his loosely painted portraits done in a style reminiscent of Edgar Degas and James Abbott McNeill Whistler. *Madame X* caused a minor scandal at the Salon of 1884 and was rejected by Sargent's client because Sargent depicted her as vain. He devoted his later career to impressionistic watercolor scenes.
is dcterms: creator of	Head of Hosea Elizabeth Winthrop Chanler (Mrs. John Jay Chapman)

Smithsonian American Art Museum
and the Renwick Gallery

Search Collections

John Singer Sargent
Also Known as: John S. Sargent

Born:
Florence, Italy 1856

Died:
London, England 1925

Active in:
- Boston, Massachusetts
- Paris, France

Photo Caption:
John Singer Sargent, Peter A. Juley & Son Collection, Smithsonian American Art Museum J0002143.

Biography

Painter. Sargent traveled in a circle of socially prominent people and is known for his loosely painted portraits done in a style reminiscent of Edgar Degas and James Abbott McNeill Whistler. *Madame X* caused a minor scandal at the Salon of 1884 and was rejected by Sargent's client because

Works in the Collection

More in Wikipedia

More in the New York Times

Online Exhibitions

Our Linked Data
on the Smithsonian
Web site

Fig. 4. Linked data viewed in a user-friendly version of Pubby and page of the Smithsonian Web site that includes the Linked Data

to DBpedia. Figure 4 shows screen shots of our user friendly version of Pubby and the Smithsonian American Art Museum's Web site with the additional links generated directly from the Linked Data that we produced for them. The Linked Data can also be used to curate the Smithsonian's own database, create virtual online museums, or create new applications that build on the Linked Data.

References

1. Boer, V., Wielemaker, J., Gent, J., Hildebrand, M., Isaac, A., Ossenbruggen, J., Schreiber, G.: Supporting Linked Data Production for Cultural Heritage Institutes: The Amsterdam Museum Case Study. *Lecture Notes in Computer Science*, pp. 733–747. Springer Berlin Heidelberg (2012).
2. Haslhofer, B., Isaac, A.: The Europeana Linked Open Data Pilot. *Proceedings of the International Conference on Dublin Core and Metadata Applications*, (2011)
3. Knoblock, C., Szekely, P., Ambite, J.L., Goel, A., Gupta, S., Lerman, K., Muslea, M., Taheriyan, M., Mallick, P.: Semi-Automatically Mapping Structured Sources into the Semantic Web. *Proceedings of the 9th Extended Semantic Web Conference* (2012)
4. Matsumura, F., Kobayashi, I., Kato, F., Kamura, T., Ohmukai, I., Takeda, H.: Producing and Consuming Linked Open Data on Art with a Local Community. *Proceedings of the Third International Workshop on Consuming Linked Data* (2012)
5. Szekely, P., Knoblock, C.A., Yang, F., Zhu, X., Fink, E., Allen, R., Goodlander, G.: Connecting the Smithsonian American Art Museum to the Linked Data Cloud. *Proceedings of the 10th Extended Semantic Web Conference* (2013)

GRAPHIUM: Visualizing Performance of Graph and RDF Engines on Linked Data

Alejandro Flores, Guillermo Palma, Maria-Esther Vidal, Domingo De Abreu, Valeria Pestana, José Piñero, Jonathan Queipo, José Sánchez

Universidad Simón Bolívar, Caracas, Venezuela
{aflores,gpalma,mvidal,dabreu,vpestana,jpinero,jqueipo,jsanchez}@ldc.usb.ve

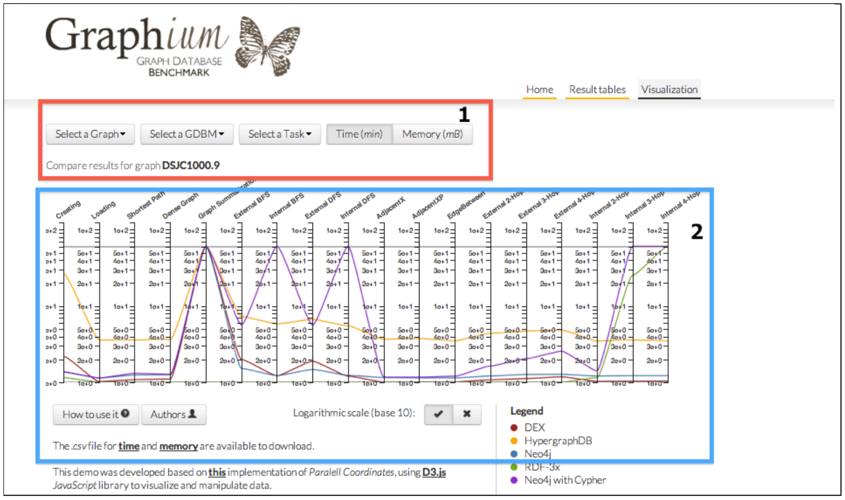
Abstract. We present GRAPHIUM a tool to visualize trends and patterns in the performance of existing graph and RDF engines. We will demonstrate GRAPHIUM and attendees will be able to observe and analyze the performance exhibited by Neo4j, DEX, HypergraphDB and RDF-3x when core graph-based and mining tasks are run against a variety of benchmarks of graphs of diverse characteristics.

1 Introduction

Graphs are commonly used to represent linked data, and several efficient algorithms have been proposed to consume and mine graphs. For example, Saha et al. [8] and Thor et al. [9] have defined densest subgraphs and graph summarization techniques to mine linked datasets and identify patterns between concepts and links. Further, algorithms for pattern matching, graph traversal, and graph reachability have been extensively studied in the literature [1]. The majority of these algorithms are computationally complex, and rely on main-memory structures to efficiently solve core graph tasks. Additionally, different engines have been developed to manage, store and query graph databases (e.g., Neo4j [7], DEX [4], HypergraphDB [2], RDF-3x [6]). Each graph database engine implements particular structures and usually relies on indices to speed up execution time; additionally, some engines make available APIs comprised of methods to solve core graph-based tasks. Although existing graph and RDF engines could be used to store linked data, mined and consumed by existing graph algorithms, there is no clear understanding of how these algorithms may behave on these engines. We present GRAPHIUM a visualization tool that exploits different graphical representations to report on the results of evaluating Neo4j, DEX, HypergraphDB and RDF-3x on a variety of benchmarks of graphs and graph-based tasks. Visualization techniques used in GRAPHIUM facilitate the understanding of trends and patterns between the performance exhibited by these engines during the execution of tasks of reachability, traversal, adjacency, pattern matching, densest graph, and graph summarization on a variety of graphs of different density and size. During the demonstration attendees will go through the visualization of different patterns that will allow them to uncover the properties and limitations of existing graph engines, as well as to reach conclusions about which engine is more appropriate for a given task. Demo is available at `http://graphium.ldc.usb.ve/demo/`.

2 The GRAPHIUM architecture

GRAPHIUM is built on top of a catalog that keeps experimental results collected during the evaluation of existing graph database and RDF engines against a variety of benchmarks. GRAPHIUM exploits visualization services implemented by the D3.js JavaScript library[1]. Figure 1 shows GRAPHIUM GUI. In the area enclosed in red rectangle number 1, a user can select to analyze: *i*) a particular graph, e.g., the dense graph DSJC1000.9; *ii*) an engine, e.g., Neo4j, DEX; and *iii*) a particular task, e.g., reachability. Results are visualized in the area enclosed by the blue rectangle number 2; GRAPHIUM exploits visualization capabilities of the Parallel Coordinates[2] to illustrate patterns and trends in the performance of each engine.



Compare results for graph **DSJC1000.9**

The *.csv* file for **time** and **memory** are available to download.

This demo was developed based on **this** implementation of *Paralell Coordinates*, using **D3.js** *JavaScript* library to visualize and manipulate data.

Fig. 1. The GRAPHIUM GUI for Neo4j, DEX, HypergraphDB and RDF-3x. 1-Selection Area: Graphs, GDBMs, Tasks, and Metrics can be selected. 2-Visualization Area: visualization ranges and scales can be chosen; explanation of how create and remove visualization ranges is presented.

3 Demonstration of Use Cases

We consider a benchmark of six graphs: DSJC1000.1, DSJC1000.5, DSJC1000.9, USA-road-d.NY, USA-road-d.FLA, and Berlin10M. The family of DSJC1000.X graphs were randomly generated using the techniques proposed by Johnson et al. [3] as instances to solve the graph coloring problem[3]; all these graphs have 1,000 nodes, and the graph density varies from 0.1 to 0.9. Instances of USA-road-d.NY and USA-road-d.FLA correspond to the New York City and Florida State road networks that were part of the 9th DIMACS Implementation Challenge - Shortest Paths[4]. Finally, Berlin10M was generated with The Berlin SPARQL Benchmark[5]. The goal of the demonstration is to visualize trends and patterns that can be found in the performance of Neo4j, DEX, HypergraphDB, and RDF-3x, where performance is measured in terms of execution time (elapsed time in msecs.), main-memory required to execute the graph-task (measured in KB), and secondary-memory needed to store the internal representation of

---

[1] `http://d3js.org/`

[2] `http://mbostock.github.io/d3/talk/20111116/iris-parallel.html`

[3] https://sites.google.com/site/graphcoloring/vertex-coloring

[4] http://www.dis.uniroma1.it/challenge9/download.shtml

[5] http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/

the graph (measured in MB). Experiments were run on a Sun Fire X4100 M2 machine with two AMD Opteron 2218 processors with 16GB RAM, running a 64-bit Linux CentOS 5.5. All tests were executed in cold cache, i.e., we cleared the cache before running each task by performing the command `sh -c "sync ; echo 3 > /proc/sys/vm/drop_caches"`. Additionally, the machine was dedicated exclusively to run these experiments. The evaluated graph-based tasks are the following:
Graph Creation: creates and stores internal representation of a graph.
Adjacency: checks node/edge adjacencies.
Reachability: traverses a graph following different strategies: *Breadth-first search* (BFS) and *Depth-first search* (DFS). Additional, *k-hops* retrieves sets of nodes such that there is a path of length k from a given start node. External implementations rely on basic adjacency methods, while internal implementations use API methods provided by the engines to solve the task.
Pattern matching: solves subgraph isomorphisms. It was evaluated as the result of traversing the graphs and finding the subgraphs that meet the given patterns; we call this implementation `internal`. Additionally, pattern matching tasks were specified as SPARQL and Cypher queries and evaluated in RDF-3x and Neo4j, respectively.
Densest subgraph: given a graph $G = (V, E)$ this tasks is to find a bipartite subgraph BSG between subsets S and T of V, such that, that BSG maximizes the density, i.e., $d(S,T) = \frac{|E(S,T)|}{\sqrt{(|S||T|)}}$ where *E(S,T)* is the set of edges going from S to T. The evaluated algorithm corresponds to the one proposed by Saha et al.[8]; our implementation exploits node/edge adjacency API methods of the engines.
Graph summarization: given a graph $G = (V, E)$ this tasks is to find a compact representation of G or aggregate graph SG comprised of hyper-nodes, hyper-edges, and corrections. Hyper-nodes correspond to sets of nodes in G, while a hyper-edge connects two hyper-nodes and represents set of edges between all pairs of nodes in the two hyper-nodes. The set of corrections corresponds to additions or deletions of edges represented in the hyper-edges of SG and that are either not present in G (deletions) or that are not presented in the hyper-edge but that were in G. We evaluate the performance of the greedy algorithm proposed by Navlakha et al.[5] on Neo4j, DEX and HypergraphDB; our implementation exploits node/edge adjacency API methods of the engines.

We will demonstrate the following use cases:
Effects of graph characteristics on the performance of the graph and RDF engines. Graphs are characterized by density, number of edges and nodes, and label distribution. Attendees will be able to choose between diverse graphs, and analyze the performance (time and memory) of the different engines in all the studied graph-based tasks. First, time required to create the internal representation of a graph is affected by both the density of the graph and the number of edges in any engine. Additionally, we will be able to observe that even RDF-3x outperforms the rest of the engines in pattern matching, its performance is impacted whenever the graph is dense. Further, graph density, size and number of labels affect the performance of both graph summarization and densest subgraph in all the engines. Nevertheless, graph summarization seems to be more impacted by the graph density and the number of labels than for the size of the graph. Contrary, densest subgraph is more influenced by the size of the graphs.
Effects of the techniques implemented by a given engine in the performance of the

graph-based tasks in different graphs. Attendees will observe that RDF-3x exhibits the best performance during graph creation and adjacency tasks (expressed as SPARQL queries); in case of *k-hops*, RDF-3x also outperforms the rest of the engines, except in the case of dense graphs. DEX seems to overcome the rest of the engines when the graphs are dense, while Neo4j exhibits better performance in sparse graphs whenever they have a large number of labels, e.g., USA-road-d.NY and USA-road-d.FLA.

Impact of a given tasks in the performance of the graph and RDF engines. We show the impact that a given task can have in the performance of an engine. For example, during graph creation RDF-3x can exploit main-memory data structures, B+-tree indices and internal representation of a graph, and exhibits the best performance. Similarly, because RDF-3x implements optimization and execution techniques that exploit the properties of a graph internal representation; thus, the best implementation of this task seems to be on top of RDF-3x. During the evaluation of *k-hops*, DEX and Neo4j are competitive. For traversals, internal implementations are able to exploit the properties of the data structures and indices implemented by each engine as well as the methods exported in their APIs; in both BFS and DFS, Neo4j and DEX exhibit a similar performance. Finally, when the mining tasks of densest subgraph and graph summarization are considered, DEX performs quite well in mining tasks if graphs are dense, while Neo4j has better performance in sparse graphs with a large number of labels.

4 Conclusions

GRAPHIUM allows to visualize patterns in the performance of graph and RDF engines when they are executed against different benchmarks of graphs and tasks. Different configurations will be analyzed allowing the attendees to understand the graph characteristics and tasks that benefit the performance of existing graph and RDF engines.

References

1. C. C. Aggarwal and H. Wang. Graph data management and mining: A survey of algorithms and applications. In *Managing and Mining Graph Data*, pages 13–68. 2010.
2. B. Iordanov. Hypergraphdb: A generalized graph database. In *WAIM Workshops*, pages 25–36, 2010.
3. D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation; part ii, graph coloring and number partitioning. *Operations research*, 39(3):378–406, 1991.
4. N. Martínez-Bazan, V. Muntés-Mulero, S. Gómez-Villamor, J. Nin, M.-A. Sánchez-Martínez, and J.-L. Larriba-Pey. Dex: high-performance exploration on large graphs for information retrieval. In *CIKM*, pages 573–582, 2007.
5. S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *ACM SIGMOD*, pages 419–432. ACM, 2008.
6. T. Neumann and G. Weikum. x-rdf-3x: Fast querying, high update rates, and consistency for rdf databases. *PVLDB*, 3(1):256–263, 2010.
7. I. Robinson, J. Webber, and E. Eifrem. *Graph Databases*. O'Reilly Media, 2013.
8. B. Saha, A. Hoch, S. Khuller, L. Raschid, and X.-N. Zhang. Dense subgraphs with restrictions and applications to gene annotation graphs. In *RECOMB*, pages 456–472, 2010.
9. A. Thor, P. Anderson, L. Raschid, S. Navlakha, B. Saha, S. Khuller, and X.-N. Zhang. Link prediction for annotation graphs using graph summarization. In *ISWC*, pages 714–729, 2011.

SILURIAN: a Sparql vIsuaLizer for UndeRstanding querIes And federatioNs

Simón Castillo, Guillermo Palma, Maria-Esther Vidal

Universidad Simón Bolívar, Caracas, Venezuela
{scastillo, gpalma, mvidal}@ldc.usb.ve

Abstract. SPARQL federated queries can be affected by both characteristics of the query and datasets in the federation. We present SILURIAN a Sparql visualizer for understanding queries and federations. SILURIAN visualizes SPARQL queries and, thus, it allows the analysis and understanding of a query complexity with respect to relevant endpoints and shapes of the possible plans.

1 Introduction

Over the past decade, the number of datasets in the Linking Open Data cloud has exploded as well as the number of SPARQL endpoints. As more linked data becomes available, applications from different domains are frequently developed, and queries that require gathering data from several endpoints are more likely everyday. So far several approaches have addressed the problem of executing federated SPARQL queries on the Web of Data [1, 2, 4]. For example, FedX [4] is a rule-based system able to generate left-linear plans comprised of subqueries that can be exclusively answered by existing endpoints (*Exclusive Groups (EG)*); ANAPSID [1] resorts to source descriptions to determine all the triple patterns that can be executed on the same endpoints and that can be grouped as star-shaped queries; finally, SPLENDID [2] exploits statistics during source selection and query planning to identify the subqueries that will be executed to gather the query answers. Performance of SPARQL queries against these federated engines can be affected by diverse parameters, e.g., number of triple patterns in the query, number of endpoints that can answer a triple pattern, and shape of the query. Analyzing a query and the federation where this query is going to be executed provides the basis not only to understand the performance of a given federated query engine, but also can be useful during query benchmarking. We present SILURIAN a Sparql visualizer for understanding queries and federations. We will demonstrate SILURIAN; attendees will be able to visualize SPARQL queries and understand complexity of both federations and possible plans. The demo is published at `http://choroni.ldc.usb.ve/silurian`.

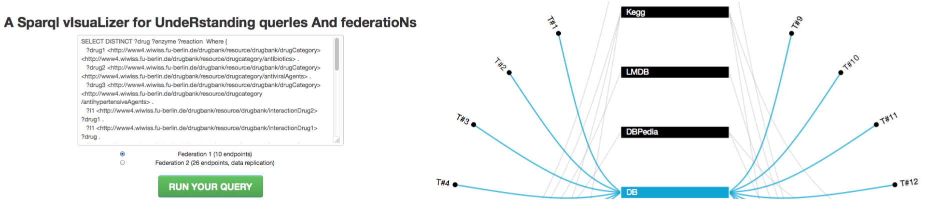
2 The SILURIAN architecture

SILURIAN is built on top of existing federated engines to visualize plans generated by the engines for a given query and federation of endpoints. In this first version, SILURIAN was built on top of ANAPSID[1], and exploits visualization services implemented by the D3.js JavaScript library[1]. Figures 1(a), (b), (c), and (d) show SILURIAN

---

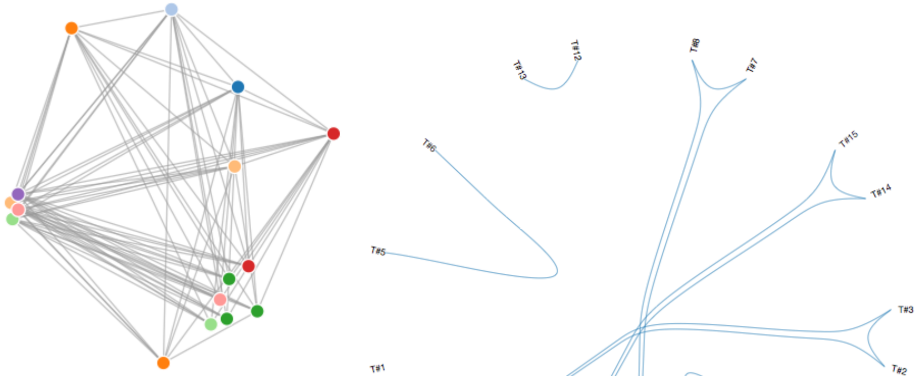
[1] `http://d3js.org/`

snapshots; users will be able to introduce their own SPARQL queries and select the federation (Figure 1(a)). Different type of plots will be used to illustrate the properties of queries and federations. Figure 1(b) uses a `Concept Network Browser` plot[2] to illustrate the endpoints that can answer the triple patterns in a query. Figure 1(c) uses `Force-Directed Graph`[3] to visualize a join graph of the input SPARQL query. Each node in the graph represents a triple pattern in the query; an edge between two nodes exists if the corresponding triple patterns do not share a join variable or there is no endpoint in the federation that can answer both triple patterns. Finally, Figure 1(d) relies on a `Hierarchical Edge Bundling`[4] to visualize the decomposition of a query into subqueries of triple patterns; nodes correspond to triple patterns while edges connect triple patterns in the same subquery of the decomposition.


A Sparql vIsuaLizer for UndeRstanding querIes And federatioNs
● Federation 1 (10 endpoints)
○ Federation 2 (26 endpoints, data replication)
RUN YOUR QUERY
Kegg
LMDB
DBPedia
DB
T#1
T#2
T#3
T#4
T#9
T#10
T#11
T#12

```
SELECT DISTINCT ?drug ?enzyme ?reaction  Where {
  ?drug1 <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/drugCategory>
<http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugcategory/antibiotics> .
  ?drug2 <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/drugCategory>
<http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugcategory/antiviralAgents> .
  ?drug3 <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/drugCategory>
<http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugcategory
/antihypertensiveAgents> .
  ?I1 <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/interactionDrug2>
?drug1 .
  ?I1 <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/interactionDrug1>
?drug .
```

(a) SILURIAN Data Entry (b) Triple Patterns Per Endpoint



(c) Join Graph (d) Query Decomposition

Fig. 1. The SILURIAN snapshots for two Federations of Endpoints on FedBench data collections.

3 Demonstration of Use Cases

We motivate our work by observing how the performance of existing federation engines can be affected during the execution of SPARQL queries with triple patterns bound to

---

[2] `http://www.findtheconversation.com/concept-map`

[3] `http://bl.ocks.org/mbostock/4062045`

[4] `http://mbostock.github.io/d3/talk/20111116/bundle.html`

predicates of general vocabularies such as RDFS or OWL. These vocabulary terms may occur in almost all data sources, e.g., `rdf:type`, `owl:sameAs`, or `rdfs:seeAlso`; we denominate these terms *general predicates*. We designed a set of three queries q_j $(j = 0...2)$, where q_{i+1} is comprised of more triple patterns bound to general predicates than q_i. First, q_0 retrieves the Kegg compound identifier and among their drugs, those that have a substrate that is an enzyme. Next, q_1 selects drugs that meet q_0 and their `owl:sameAs` link to Drugbank; and finally, q_2 checks that these drugs are also drugs in the DBpedia ontology. Triple patterns bound to *general predicates* are highlighted.

```
q0 Select * WHERE {?d drugbank:keggCompoundId ?c. ?e bio2rdf-kegg:xSubstrate ?c.
                   ?e rdf:type bio2rdf-kegg:Enzyme }
q1 Select * WHERE {?d drugbank:keggCompoundId ?c. ?e bio2rdf-kegg:xSubstrate ?c.
                   ?e rdf:type bio2rdf-kegg:Enzyme.?d owl:sameAs ?d1 .}
q2 Select * WHERE {?d drugbank:keggCompoundId ?c. ?e bio2rdf-kegg:xSubstrate ?c.
                   ?e rdf:type bio2rdf-kegg:Enzyme.?d owl:sameAs ?d1 .
                   ?d1 rdf:type dbpedia-owl:Drug .}
```

An experiment was set up in order to evaluate the performance of different federated SPARQL query engines: FedX, SPLENDID, and ANAPSID. Queries q_0, q_1, and q_2 were executed on 26 Virtuoso endpoints that locally access the FedBench collections[5], November 2011. Each collection was assigned to one Virtuoso endpoint, except *Geonames* and DBpedia that were fragmented to impact on the performance of the query decomposition techniques. *Geonames* was horizontally partitioned into eleven fragments and each fragment was assigned to a different endpoint. Additionally, each of the DBpedia files was made available through a different SPARQL endpoint, i.e., DBpedia was vertically partitioned. This study was executed on a Linux Mint machine with an Intel Pentium Core 2 Duo E7500 2.93GHz 8GB RAM 1333MHz DDR3. We could observe that the performance of all these engines is deteriorated as the number of triple patterns on general predicates increases. Based on these results, we formulated the following research questions: *1*) is the observed behavior due to limitations of these federation engines?, or *2*) is this behavior caused by the properties of these queries?. We will visualize the characteristics of queries and federations that provide evidences to answer our research questions. FedBench 10 collections: DBpedia, NY Times, Geonames, KEGG, ChEBI, Drugbank, Jamendo, LinkedMDB, SW Dog Food, and SP2B-10M, were integrated into two federations of endpoints. $\mathtt{Fed}_1$ comprises the previously explained 26 Virtuoso[6] endpoints, and $\mathtt{Fed}_2$ is composed of 10 endpoints, one per FedBench collection. In both federations, Virtuoso timeout was set up to 300 secs. or 100,000 tuples. Different criteria to decompose SPARQL queries into subqueries answerable by existing endpoints will be demonstrated; e.g., Exclusive Groups (EG) [2, 4] , Star-Shaped Group Single endpoint selection (SSGS), and Star-Shaped Group Multiple endpoint selection (SSGM) [3]. We will demonstrate the following use cases:

Effects of number of triple patterns bound to general predicates. We will demonstrate that in queries as the ones presented in the previous example, almost all the endpoints in the federation can instantiate variables in the triple patterns of the query. Particularly, triple patterns bound to `owl:sameAs` could be answerable for 24 out of 26

---

[5] `http://fedbench.fluidops.net`

[6] `http://virtuoso.openlinksw.com/`, November 2011.

endpoints of `Fed`$_1$ and all the endpoints of `Fed`$_2$. Federated engines may have to consider all these endpoints to produce a complete answer of the query.

Effects of the number of triple patterns and shape of the query. Attendees will observe that in queries will a large number of triple patterns that comprised star-shaped or chain-shaped subqueries, the space of possible plans of the query may exponentially explode. For example, we will show queries comprise of 46 triple patterns which can be decomposed into 9 star-shaped subqueries, which could not be executed in any of existing federated engines in less than 30 minutes. These queries may constitute challenges for federation engines and should be included in future benchmarks.

Effects of the data fragmentation and replication. The aim of this use case is to show the effects of data fragmentation and replication in the complexity of SPARQL federated queries. In federation `Fed`$_1$, data in Geonames is horizontally partitioned while DBpedia is vertically fragmented. Attendees will observe that the number of relevant endpoints increases according to fragments of data are made available from different endpoints of a federation. For example, in queries with triple patterns bound to predicates in Geonames the number of relevant endpoints is larger in `Fed`$_1$ than in `Fed`$_2$. Because Geonames data is horizontally partitioned, many of the relevant data may not actually provide the instantiations of the variables required to execute the query. Thus, federated engines have to either contact all the endpoints to decide which one can execute the corresponding subqueries or simply pay the price of executing the subquery in all of them, and the execution of these queries can be costly. These queries may be challenging for existing federation engines and should be included in future benchmarks.

4 Conclusions

SILURIAN visualizes SPARQL federated queries as well as the properties of the federations that may impact on the complexity of these queries. Particularly, SILURIAN helps to understand why data fragmentation and replication among different endpoints, shape of the queries and the type of predicates in the triple patterns, may affect the performance of a federated query engine. Because main sources of query complexity can be analyzed, SILURIAN provides the basis for understanding the behavior of existing engines and may help during the design of benchmarks to evaluate these engines.

References

1. M. Acosta, M.-E. Vidal, T. Lampo, J. Castillo, and E. Ruckhaus. Anapsid: an adaptive query processing engine for sparql endpoints. In *ISWC*, pages 18–34, 2011.
2. O. Görlitz and S. Staab. SPLENDID: SPARQL Endpoint Federation Exploiting VOID Descriptions. In *COLD*, Bonn, Germany, 2011.
3. G. Montoya, M.-E. Vidal, and M. Acosta. A heuristic-based approach for planning federated sparql queries. In *COLD*, 2012.
4. A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt. Fedx: Optimization techniques for federated query processing on linked data. In *ISWC*, pages 601–616, 2011.

Modeling and Reasoning Upon Facebook Privacy Settings

Mathieu d'Aquin and Keerthi Thomas

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

Abstract. Understanding the way information is propagated and made visible on Facebook is a difficult task. The privacy settings and the rules that apply to individual items are reasonably straightforward. However, for the user to track all of the information that needs to be integrated and the inferences that can be made on their posts is complex, to the extent that it is almost impossible for any individual to achieve. In this demonstration, we investigate the use of knowledge modeling and reasoning techniques (including basic ontological representation, rules and epistemic logics) to make these inferences explicit to the user.

1 Introduction

The notion of social translucence (as defined in [4]) concerns the design of systems with a social process component, to achieve coherent behaviours from the user(s) through making such behaviours visible and understandable to them. This notion is especially relevant in relation to privacy, where the principles of visibility, awareness and accountability promoted by social translucence are used to enable a coherent and informed behaviour from the users with respect to the distribution and propagation of their personal information. This idea is well illustrated in the notion of "Privacy Mirrors", i.e., systems that integrate the necessary tools of "awareness and control to understand and shape the behaviour of the system" [6].

While these notions might appear to naturally apply to social networking systems such as Facebook[1], their privacy settings and of the mechanisms for information sharing they implement are only deceptively simple: for an individual user to keep track of all the necessary elements to understand what information others might have access to, and what inferences they might derive from it, is actually too complex to be achieved. For example, while individual photos have specific privacy scopes, the tagging, comments, likes, geographical information attached to them can make much more information about the user available to much more people than the user might intend, without the user's ability to understand the full scope of the implications of such sharing and tagging.

In this demonstration, we show how a privacy mirror for Facebook can be implemented using knowledge modeling and reasoning techniques, to make explicit to the user some of the inferences that can be made out of information available about them on the social platform. We use basic ontology modeling, rules and a simplification of the basic concepts of epistemic logics.

---

[1] http://facebook.com

2 Information from Facebook

In this demonstration, to simplify the discussion, we focus on information about photos, especially the ones (explicitly) referring to the user. However, the basic notions and approach described apply similarly to other types of information. The basic concepts extracted using the Facebook Graph API[2] concern people (users), photos, comments, places and dates. Individuals (variables and constants) therefore represent instances of these concepts. Predicates represent relationships. For example, users can be friends with each-other ($friend(bob, alice)$), a photo can be at a place ($photoAt(photo1, segovia)$), at a certain date ($date(photo1, 08-07-2013)$) and include some users ($onPhoto(photo1, bob)$). Finally, any post including photos have a privacy scope which could be $everyone$, $friendoffriend$, $allfriends$, $custom$ (e.g., $scope(photo1, allfriends)$).

3 Basic ontological modeling and reasoning

From the explicit data extracted from Facebook, basic information can be inferred using ontology-based mechanisms. For example, including range and domain information associated with the predicates mentioned above can help identifying types of objects (e.g., $friend(bob, alice)$ implies that $person(bob)$ and $person(alice)$). Similarly, using constructs available in OWL 2, the $friend$ predicate can be declared to be reciprocal (as it is in Facebook): $friend(bob, alice)$ implies $friend(alice, bob)$.

Property hierarchies can also be used to introduce intermediary predicates, more abstract than the notions explicitly available in Facebook. For example, declaring $friend$ as a sub-property of $know$ (so that $friend(bob, alice)$ implies $know(bob, alice)$). The same mechanisms, combined with the property composition construct available in OWL 2, can be used to represent much more complex inferences (e.g., that if two people are on the same photo, they know each other). However, for convenience, we choose to use rules (which can also be used for other types of inferences not feasible with basic OWL constructs) for such complex implications.

4 Rule-based inference

As mentioned above, some more complex inferences need to be represented that are not conveniently achieved with ontological constructs. This includes information such that being on a picture, geotagged with a certain place, implies that the user was at that place ($wasIn(Per, Pl)$:- $onPhoto(Pic, Per), photoAt(Pic, Pl)$) or that two users on the same photo know each other ($know(Per1, Per2)$:- $onPhoto(Pic, Per1), onPhoto(Pic, Per2)$), and possibly that they were at the same place.

---

[2] https://developers.facebook.com/docs/reference/api/

5 Epistemic inference

The mechanisms described above make it possible for the model to explicitly make the inferences possible from the information being shared. However, the important aspect here is not only which inferences can be made, but who can make them. To address this, we use notions from epistemic logics [5]. Indeed, epistemic logics are a type of logic that allows one to express not only statements about the world, but also about the way the world is perceived or known by agents in the world. In such a logic, a statement of the form $\mathbf{K}_a\ \alpha$ indicates that the agent a 'knows' the statement α to be true. Basic properties, such as the one of self reflection (i.e., $\mathbf{K}_a\ \alpha \rightarrow \mathbf{K}_a\mathbf{K}_a\ \alpha$) and rules can be used to reason upon the knowledge agents have of some information.

This framework, combined with information about the privacy settings of Facebook, allows us to express information regarding which user might have access to what item of information. Straightforwardly for example, information on who knows about a photo can be derived from the privacy scope of the photo (e.g., $\mathbf{K}_a\, photo(Pic)$:- $author(Pic, Per), scope(Pic, allfriends), friend(Per, a)$). More complex mechanisms are also represented using this type of rules however, for example that the friends of somebody tagged in a photo would know about the photo, or that knowing about a photo implies knowing all the information attached to a photo and the possible inferences that can be made from them (e.g., that somebody was in a certain place with somebody else).

6 Implementation

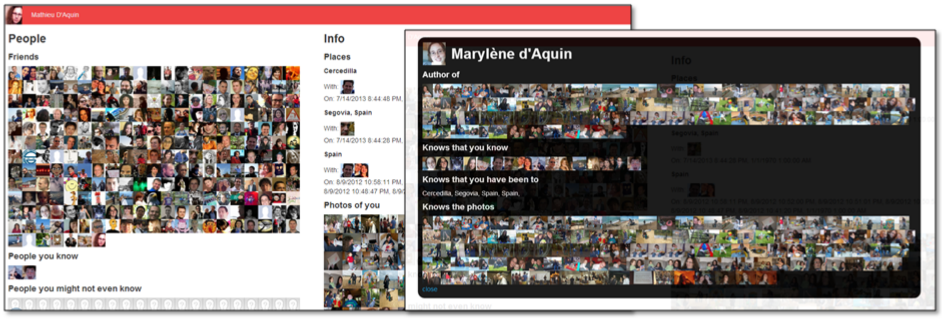

Mathieu D'Aquin
People
Friends
People you know
People you might not even know
Info
Places
Cercedilla
With:
On: 7/14/2013 8:44:48 PM,
Segovia, Spain
With:
On: 7/14/2013 8:44:28 PM,
Spain
With:
On: 8/9/2012 10:58:11 PM,
8/9/2012 10:48:47 PM, 8/9
Photos of you
Marylène d'Aquin
Author of
Knows that you know
Knows that you have been to
Cercedilla, Segovia, Spain, Spain,
Knows the photos
close

Fig. 1. Screenshots of the system making explicit privacy inferences in Facebook.

The implementation of the system showing to a user the inferences that can be made from information sharing items concerning them (currently focusing on photos) and by who is a Web-based interface built in PHP and Javascript, that allows the user to connect to their Facebook account and extract the relevant information. The knowledge representation and reasoning mechanisms described above is delegated to a Prolog-based API, carrying out the ontological reasoning (through a basic mapping between OWL and Prolog), the rule-based reasoning and a simplified implementation of epistemic rules described in the previous sections.

As shown in the screenshots of Figure 1, the system displays the inferred information to the user: 1- the people they are friend with, the ones they know (without being friends) and the people the user might not know, but who might have access to some of their information; 2- the photos depicting the user; 3- the places where the user have been (who with and on what date). Clicking on a person (as shown on Figure 1) displays information about items shared by this user, as well as the information they know about the logged-in user. Clicking on an item displays the information that can be inferred from this item, and the people who might have access to these inferences.

7 Conclusion

Our goal in this demonstration is to show that knowledge modeling and reasoning techniques can support the notion of privacy mirrors, in systems where the privacy implications of information sharing are complex and difficult for a user to keep track of. In the demonstration, participants will be able to connect the system to their own Facebook account, to check whether the results are surprising, concerning or on the contrary, just reassuring (which are the types of reactions we uncovered in another study [3]). In terms of future work, besides completing and validating the modeling of Facebook's privacy mechanisms (which can be a complex task), one of the interesting research directions is to integrate the model of Facebook with other sources of personal information sharing (using for example techniques described in some of our previous works, e.g., [1, 2]). The other direction we plan to investigate is the use of more sophisticated knowledge representation techniques to deal with the complexity of online social situations, including uncertainty and different levels of epistemic knowledge of information (e.g., having access to information vs. having surely seen a piece of information).

References

1. M. dAquin, S. Elahi, and E. Motta. Personal monitoring of web information exchange: Towards web lifelogging. *Web Science*, 2010.
2. M. d'Aquin, S. Elahi, and E. Motta. Semantic technologies to support the user-centric analysis of activity data. In *Social Data on the Web (SDoW) workshop at ISWC*, 2011.
3. M. d'Aquin and K. Thomas. Consumer activity data: Usages and challenges. Knowledge Media Institute, Tech. Report kmi-12-03, 2012.
4. T. Erickson and W. A. Kellogg. Social translucence: an approach to designing systems that support social processes. *ACM transactions on computer-human interaction (TOCHI)*, 7(1):59–83, 2000.
5. J.-J. Ch. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science*. Cambridge University Press, 2004.
6. E. D. Nguyen, D. H. ; Mynatt. Understanding and shaping socio-technical ubiquitous computing systems. GVU Technical Report;GIT-GVU-02-16, 2002.

Do it your own (DIY) Jeopardy Question Answering System

André Freitas and Edward Curry

Digital Enterprise Research Institute (DERI)
National University of Ireland, Galway

1 Motivation

The evolution and maturity of semantic technologies techniques and frameworks are bringing functionalities which were once considered academic or prototypical into real-life applications. Products such as *IBM Watson* [1] and *Siri* are examples of applications which are heavily leveraged on state-of-the-art semantic technologies. These systems provide a synthesis of the functionalities which are available for general applications today such as: natural language search and queries over large-scale data, semantic flexibility and integration between structured and unstructured resources. The success of these projects in demonstrating the potential of existing technologies lies on the fact that they bring into a single system approaches from Natural Language Processing (NLP), Semantic Web (SW), Information Retrieval (IR) and Databases.

This work demonstrates *Treo*, a framework which converges elements from NLP, IR, SW and Databases, to create a semantic search engine and question answering (QA) system for heterogeneous data. *Jeopardy* and *Question Answering* queries over open domain structured and unstructured data are used to demonstrate the approach. In this work, *Treo* is extended to cope with unstructured text in addition to structured data. The setup of the framework is done in 3 steps and can be adapted to other datasets in a simple DIY process.

2 Treo: Querying structured & unstructured data

Treo supports free natural language queries over both structured and unstructured data. To enable *semantic flexibility* and *vocabulary independency* in the query process, a principled *distributional-compositional semantic model* is used to build a distributional structured vector space model ($\tau - Space$) [2]. *Distributional semantics* focuses on the *automatic construction* of a semantic model based on the statistical distribution of co-located words in large-scale corpora. The distributional semantics component of the model, supports a semantic approximation between query and dataset terms: operations in the $\tau - Space$ are mapped to *semantic relatedness* operations using the distributional model as a commonsense knowledge base [2]. The automatic creation of distributional semantic models supports the *transportability* of the approach to other datasets and languages, not requiring the manual creation effort of ontologies (Treo does not rely on ontology-based reasoning for semantic approximation).

In addition to queries over structured data, this work extends the query mechanism for searching entities in unstructured text. Both structured and unstructured data are linked in an entity-centric semantic index (Figure 1 (B)). The elements of the query processing approach are depicted in Figure 1 (A).

Two different query processing strategies are used:

- Query processing over structured data: In the *query pre-processing* phase, the natural language query is analyzed by the *Interpreter* component, where a set of *query triple patterns and features* are detected in the user query. The second phase consists of the *vocabulary independent query processing approach* which defines a sequence of search and data transformation operations over the structured data graph embedded in the $\tau - Space$ [2], targeting the maximization of the semantic matching with the query. The *Query Planner* generates the sequence of *semantic search, navigation and transformation operations* over the graph data, which defines the *query processing plan*, based on a set of *query features* which are determined in the pre-processing phase. The third phase consists in the execution of the query processing plan operations over the $\tau - Space$ index.

- Query processing over structured & unstructured data: In case the query is not addressed by the available structured data, the query can be processed against both structured data and unstructured text in the entity-centric index. The query pre-processing approach for this query type consists on the detection of the *query focus* by the application of POS Tag based rules and by the detection and resolution of *named entities* in the query. The *query plan* consists of the composition of keyword-search operations over the text segments associated with entities, distributional search operations over structured data, and keyword search over associated entities. A *ranking function* weights the results of all operations, also taking into account the *cardinality* for each entity (number of associated entities, facts and text segments). The initial top-20 entity results are re-ranked based on the computation of the *distributional semantic relatedness scores* between the *query focus phrase* and the associated *entity types.*

3 DIY Setup Process

The setup of the Treo platform for a new dataset consists in the creation of a *semantic index* for both structured and unstructured data, which requires three steps:

1. *Construction of the distributional semantic model:* Consists on the use of a large-scale reference corpora to build the distributional semantic reference model [2]. In this demonstration Wikipedia 2006 is used as the reference corpus and Explicit Semantic Analysis (ESA) is the distributional semantic model.
2. *Semantic indexing of structured data:* Consists in the indexing of structured data using the distributional semantic reference model [2]. The framework

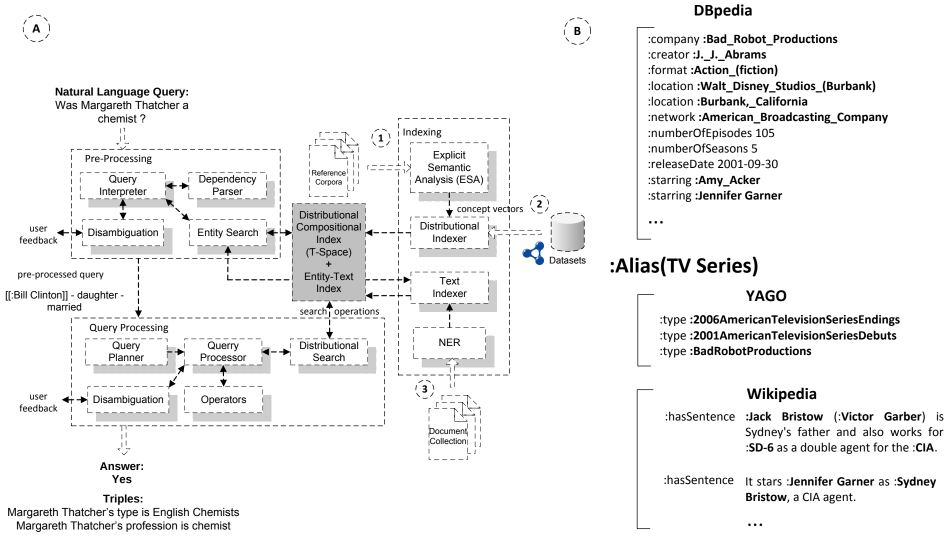

A
B
Natural Language Query:
Was Margareth Thatcher a chemist ?
Pre-Processing
Query Interpreter
Dependency Parser
Disambiguation
Entity Search
user feedback
pre-processed query
[[:Bill Clinton]] - daughter - married
Reference Corpora
1
Indexing
Explicit Semantic Analysis (ESA)
concept vectors
2
Distributional Compositional Index (T-Space) + Entity-Text Index
Distributional Indexer
Datasets
Text Indexer
search operations
NER
3
Document Collection
Query Processing
Query Planner
Querry Processor
Distributional Search
Disambiguation
Operators
user feedback
Answer:
Yes
Triples:
Margareth Thatcher's type is English Chemists
Margareth Thatcher's profession is chemist
DBpedia
:company :Bad_Robot_Productions
:creator :J._J._Abrams
:format :Action_(fiction)
:location :Walt_Disney_Studios_(Burbank)
:location :Burbank,_California
:network :American_Broadcasting_Company
:numberOfEpisodes 105
:numberOfSeasons 5
:releaseDate 2001-09-30
:starring :Amy_Acker
:starring :Jennifer Garner
...
:Alias(TV Series)
YAGO
:type :2006AmericanTelevisionSeriesEndings
:type :2001AmericanTelevisionSeriesDebuts
:type :BadRobotProductions
Wikipedia
:hasSentence :Jack Bristow (:Victor Garber) is Sydney's father and also works for :SD-6 as a double agent for the :CIA.
:hasSentence It stars :Jennifer Garner as :Sydney Bristow, a CIA agent.
...

Fig. 1: (A) Semantic indexing and query processing architecture. (B) Entity-centric representation of structured and unstructured data.

takes as input data any dataset following an Entity-Attribute-Value (EAV) format. DBpedia 3.7 and YAGO are used as the demonstration datasets.

3. *Unstructured data entity-centric indexing:* This step takes as input a text collection, recognizes the named entities based on the structured data previously indexed, aligning it with the indexed structured data. The demonstration uses Wikipedia 2013 as the test collection.

The steps are executed by calling one script, which takes as input the three types of resources (reference corpora, structured datasets and unstructured texts). After the setup, natural language queries can be executed against the structured and unstructured data indexes. Figure 1 shows the components of the Treo architecture (A) and an example of the entity-centric linking between structured and unstructured data (B).

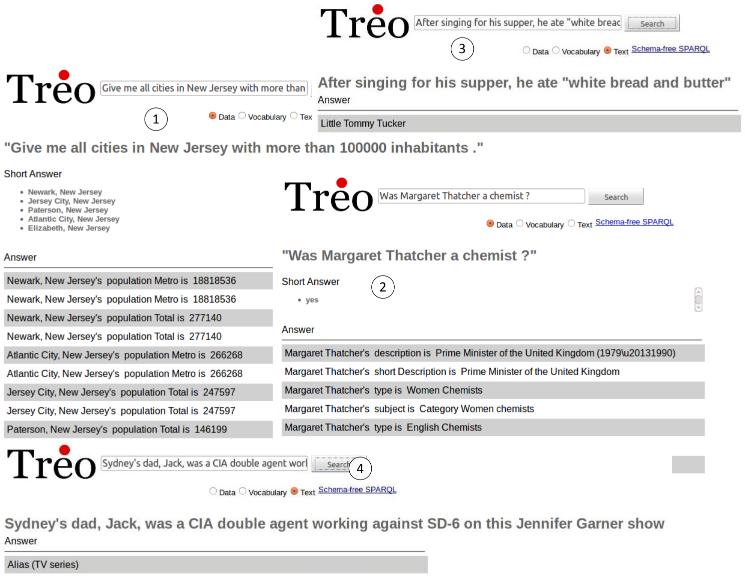
4 Demonstration

The system is demonstrated over the open-domain *DBpedia 3.7/YAGO* RDF datasets and Wikipedia 2013 text data. The RDF datasets consist of 128,071,259 triples (17GB) loaded into the Treo index for structured data. A set of natural language queries from the *Jeopardy challenge*[1] and from the *Question Answering over Linked Data* challenge[2] are used to demonstrate the system. In the demonstration, users input free natural language queries and the system returns two

---

[1] http://j-archive.com/

[2] QALD-1, `http://www.sc.cit-ec.uni-bielefeld.de/qald-1`, 2011


Treo
Search
③
○ Data ○ Vocabulary ● Text
Schema-free SPARQL
Treo
①
● Data ○ Vocabulary ○ Tex
Treo
Search
● Data ○ Vocabulary ○ Text
Schema-free SPARQL
②
Treo
Search
④
○ Data ○ Vocabulary ● Text
Schema-free SPARQL

After singing for his supper, he ate "white breac

After singing for his supper, he ate "white bread and butter"

Answer

Little Tommy Tucker

Give me all cities in New Jersey with more than

"Give me all cities in New Jersey with more than 100000 inhabitants ."

Short Answer

- Newark, New Jersey
- Jersey City, New Jersey
- Paterson, New Jersey
- Atlantic City, New Jersey
- Elizabeth, New Jersey

Answer

Newark, New Jersey's population Metro is 18818536
Newark, New Jersey's population Metro is 18818536
Newark, New Jersey's population Total is 277140
Newark, New Jersey's population Total is 277140
Atlantic City, New Jersey's population Metro is 266268
Atlantic City, New Jersey's population Metro is 266268
Jersey City, New Jersey's population Total is 247597
Jersey City, New Jersey's population Total is 247597
Paterson, New Jersey's population Total is 146199

Was Margaret Thatcher a chemist ?

"Was Margaret Thatcher a chemist ?"

Short Answer

- yes

Answer

Margaret Thatcher's description is Prime Minister of the United Kingdom (1979\u20131990)
Margaret Thatcher's short Description is Prime Minister of the United Kingdom
Margaret Thatcher's type is Women Chemists
Margaret Thatcher's subject is Category Women chemists
Margaret Thatcher's type is English Chemists

Sydney's dad, Jack, was a CIA double agent worl

Sydney's dad, Jack, was a CIA double agent working against SD-6 on this Jennifer Garner show

Answer

Alias (TV series)

Fig. 2: Example queries: (1,2) Queries over structured data (3,4) Jeopardy queries over structured and unstructured data.

types of results: (i) a list of highly related triples or (ii) post-processed results, depending on the query type.

Figure 2 (2) shows the output of a query over the structured data index for the query *'Was Margaret Thatcher a chemist?'*. In addition to the post-processed answer, which provides a direct (QA-style) answer for the query, the mechanism shows the justification for the answer with the supporting triples. Figure 2 (1) shows a query over structured data with a complex query plan (*'Which cities in New Jersey have more than 10000 inhabitants?'*). Figure 2 (3) and (4) show examples of Jeopardy queries, which typically provide a natural language description of a named entity or concept (for example: *'Sydney's dad, Jack, was a CIA double agent working against SD-6 on this Jennifer Garner show'*). Further examples can be found online[3].

Acknowledgments. This work was funded by SFI Ireland (SFI/08/CE/I1380).

References

1. D. Ferrucci et al., Building Watson: An Overview of the DeepQA Project, AI Magazine, 2010.
2. A. Freitas, E. Curry, J. G. Oliveira, S. O'Riain, A Distributional Structured Semantic Space for Querying RDF Graph Data. International Journal of Semantic Computing (IJSC), 2012.

---

[3] http://treo.deri.ie/ISWC2013Demo

A Machine Reader for the Semantic Web

Aldo Gangemi[12], Francesco Draicchio[1], Valentina Presutti[1]
, Andrea Giovanni Nuzzolese[13], and Diego Reforgiato[1]

[1] STLab-ISTC Consiglio Nazionale delle Ricerche, Rome, Italy.
[2] LIPN, Université Paris13-CNRS-SorbonneCité, France
[3] Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy.

Abstract. FRED is a machine reading tool for converting text into internally well-connected and quality linked-data-ready ontologies in web-service-acceptable time. It implements a novel approach for ontology design from natural language sentences, combining Discourse Representation Theory (DRT), linguistic frame semantics, and Ontology Design Patterns (ODP). The current version of the tool includes Earmark-based markup, and enrichment with word sense disambiguation (WSD) and named entity resolution (NER) off-the-shelf components.

1 Introduction

The problem of knowledge extraction (KE) from text is still insufficiently addressed from a semantic web (SW) perspective. Being able to automatically produce quality linked data and ontologies from natural language text would be a breakthrough as it would enable the development of applications that automatically produce machine-readable information from Web content as soon as it is edited and published by generic Web users. A rather detailed landscape analysis of the currently available tools for KE, and their exploitation for SW basic tasks is presented in [4]: it shows the substantial lacking of tools for creating RDF graphs that are connected enough to perform application tasks such as event extraction, fact detection, story mining, etc.

FRED[4] [5] is an exception, since it is intended to produce semantic data and ontologies with a quality closer to what is expected at least from average linked datasets and vocabularies: FRED candidates as a deep version of a machine reader [3] for the Semantic Web. The following requirements have inspired the design of FRED: (i) ability to capture accurate semantic structures (i.e. compliant to formal semantics); (ii) representing complex relations (i.e. n-ary, multigrade relations); (iii) exploitation of sophisticated lexical resources (e.g. VerbNet, FrameNet); (iv) no need of large-size domain-specific text corpora and training sessions (i.e. we address open information extraction); (v) minimal time of computation; (vi) ability to map natural language to RDF/OWL representations; (vii) ability to link the extracted knowledge to both lexical linked data and linked datasets (for maximal interoperability).

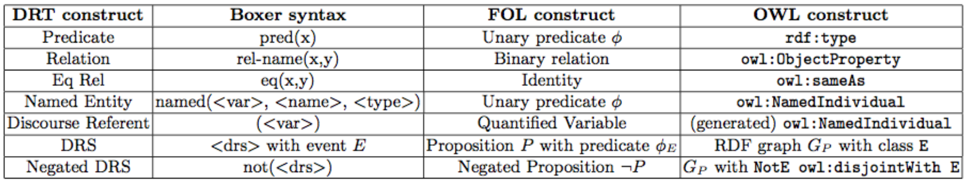
---

[4] `http://wit.istc.cnr.it/stlab-tools/fred`

Related works and comparison to other tools for knowledge extraction are detailed in [5], and [4], where FRED seems to outperform (though on a limited test) the other tools in sophisticated tasks such as relation and factoid extraction, frame detection, and taxonomy induction.

2 FRED at work

In this section we present an overview of the system and a scenario that shows the output resulting from FRED. [2] shows that detecting the most appropriate frames from the input text leads to improve the design quality of the resulting ontology because frames can be directly mapped to an important variety of ontology design patterns based on *n-ary* relations. On the above consideration, FRED makes use of Boxer [1], a deep semantic parser based on categorial grammar and Discourse Representation Theory (DRT), which generates formal semantic representation of text through an event (neo-Davidsonian) semantics.



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

Fig. 1. A sample subset of quality ontology production heuristics.

Boxer frame-based approach supports FRED in automatically design an ontology by following good modeling practices based on ontology patterns. However, Boxer tranforms natural language to a logical form compliant with DRT (substantially a variety of first-order logic) which differs a lot from RDF or OWL, and the heuristics that it implements for interpreting a natural language and transforming it to a DRT-based structure can be sometimes awkward when directly translated to ontologies for the SW, because it obeys pure FOL-oriented design style. For this reason, Boxer DRT-based output is transformed by FRED to OWL/RDF ontologies by means of a set of heuristics, some of them are showed in Figure 1. Figure 2 depicts the main components of FRED: *Communication*: exposes APIs for querying the system; *Refactoring*: transforms Boxer output[5] into a convenient data structure to be passed to the *Reenginering* component; *Reengineering*: applies a collection of ad-hoc mapping rules and heuristics for producing logically consistent OWL/RDF like triples.

In addition, FRED architecture is open to be easily integrated with other components that exploit the text span markup specification supported its current version, i.e. Earmark [6]. This solution, which is similar to architectures such as

---

[5] Boxer is an external component.

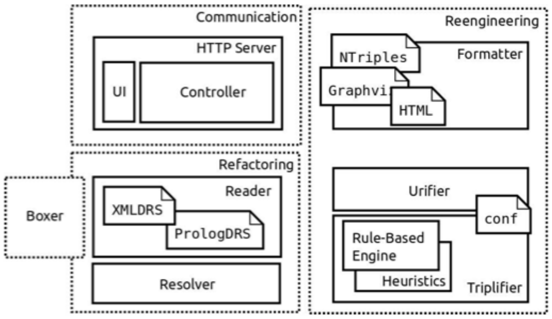


Fig. 2. Block architecture and workflow

NIF and NERD, makes it trivial to augment FRED graphs with off-the-shelf components for e.g. NER, WSD, etc. Our demo, available online[6], allows a user to enter any text (or select one from the list of examples provided), and by simply clicking the *Read It!* button, to receive a RDF/OWL representation of it[7]. Some features can be customized, e.g., type of output, NER or WSD activation, tense-relation between events activation, etc. By default, the output is in the form of Graphviz-like graphs (showing only a core subset of triples), to allow human users to quickly check the OWL/RDF representation.

FRED output consists in RDF triples including either fixed properties: `rdf:type, rdfs:subClassOf, owl:sameAs, dul:associateWith, owl:equivalentTo`, Earmark properties, thematic roles for extracted events, etc., or customized properties produced by the automatic (machine) reading of the sentence.

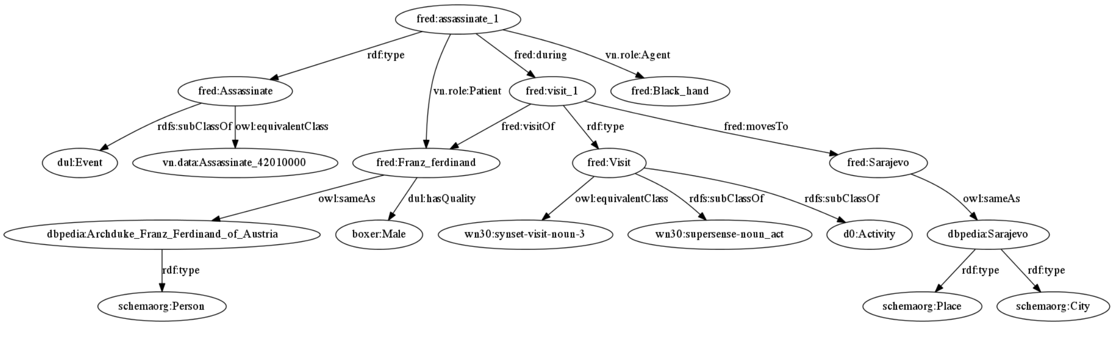


Fig. 3. FRED output for the sentence *“The Black Hand assassinated Franz Ferdinand during his visit to Sarajevo.”*

---

[6] http://wit.istc.cnr.it/stlab-tools/fred

[7] A graphical output is provided for human users

As an example, consider the sentence *"The Black Hand assassinated Franz Ferdinand during his visit to Sarajevo."* Figure 3 shows FRED output for this sentence: an instance of `dul:Event`, `fred:assassinate_1`, is used to represent the assassination of Franz Ferdinand. It is typed as `fred:Assassinate`, which is disambiguated by the VerbNet frame `vn.data:Assassinate_42010000`. Such an event involves the individual `fred:Black_hand` as agent, and the individual `fred:Franz_ferdinand` (who is recognized and resolved as the same individual as `dbpedia:Archduke_Franz_Ferdinand_of_Austria`) as patient. Furthermore, the RDF graph expresses that such an event happened `fred:during` `fred:visit_1`. FRED heuristically assigns a type `fred:Visit` to `fred:visit_1`; such a type is disambiguated by means of alignments to WordNet, which in turn is aligned to other ontologies, so that FRED can infer e.g., that `fred:Visit` is a `d0:Activity`. The location of the visit is also identified and correctly resolved to `dbpedia:Sarajevo`. Notice that FRED assigns types to all identified individuals either by using classes from existing ontologies e.g., DOLCE, Schema.org, etc., when the entity can be resolved e.g., as a DBpedia entity, or by creating new classes based on the terms used in the input text and disambiguating them on WordNet.

FRED also supports more sophisticated constructs, e.g. propositional referents, called *situations*, full-fledged negation on events or situations, and basic modalities over events.

3 Conclusion

We have presented FRED, a machine reader for the Semantic Web, which automatically extracts rich and connected knowledge from text, represents it as OWL/RDF, and links it to other resources: VerbNet, FrameNet, WordNet, DBpedia, foundational ontologies, etc. The current research is on mainly evaluating it on vertical tasks, and extending its internal components for multilinguality.

References

1. Johan Bos. Wide-Coverage Semantic Analysis with Boxer. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing*, pages 277–286. College Publications, 2008.
2. Bonaventura Coppola, Aldo Gangemi, Alfio Massimiliano Gliozzo, Davide Picca, and Valentina Presutti. Frame detection over the semantic web. In Lora Aroyo et al., editor, *ESWC*, volume 5554 of *LNCS*, pages 126–142. Springer, 2009.
3. Oren Etzioni, Michele Banko, and Michael Cafarella. Machine reading. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, 2006.
4. Aldo Gangemi. A comparison of knowledge extraction tools for the semantic web. In *Proceedings of ESWC2013*. Springer, 2013.
5. Valentina Presutti, Francesco Draicchio, and Aldo Gangemi. Knowledge extraction based on discourse representation theory and linguistic frames. In *EKAW: Knowledge Engineering and Knowledge Management that matters*. Springer, 2012.
6. Peroni S., Gangemi A., and Vitali F. Dealing with markup semantics. In *Proceedings of the 7th International Conference on Semantic Systems, Graz, Austria (i-Semantics2011)*. ACM, 2011.

ONTOMS2: an Efficient and Scalable ONTOlogy Management System with an Incremental Reasoning

Min-Joong Lee[1], Jong-Ryul Lee[1], Sangyeon Kim[1], Myung-Jae Park[1], and Chin-Wan Chung[2]

Department of Computer Science , KAIST, Daejeon, Republic of Korea
`{mjlee,jrlee,sangyeon,jpark}@islab.kaist.ac.kr`[1] `chungcw@kaist.edu`[2]

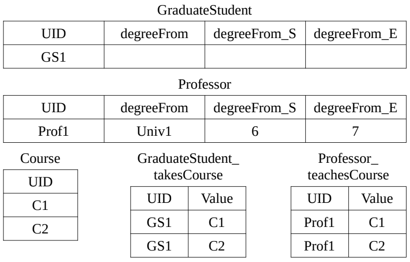
Abstract. We present ONTOMS2, an efficient and scalable ONTOlogy Management System with an incremental reasoning. ONTOMS2 stores an OWL document and processes OWL-QL and SPARQL queries. Especially, ONTOMS2 supports SPARQL Update queries with an incremental instance reasoning of inverseOf, symmetric and transitive properties.

1 Introduction

In order to efficiently manage ontology data, various ontology data management systems [4–6] are proposed based on RDBMS. However, existing ontology data management systems do not support updates incrementally. To efficiently manage such large sized ontology data, we need a way of incremental updating for ontology data. We propose an incremental update strategy to efficiently handle a large amount of ontology data and the frequent updates of such ontology data. Our incremental update strategy provides an insertion and deletion based on SPARQL Update with the support of some important semantics of ontologies such as inverseOf, symmetric, and transitive. We apply our incremental update strategy to ONTOMS which is an efficient and scalable ONTOlogy Management System proposed in [6].

ONTOMS efficiently manages large sized OWL data based on RDBMS and using OWL-QL. It stores OWL data into a class based relational schema to increase the query processing performance. Figure 1 describes an example of OWL data stored into relational tables in ONTOMS. Unlike other approaches, ONTOMS generates a class based relational schema, where one relation is created for each class. Each class relation contains associated properties as its attributes. For more details of ONTOMS, please refer [6].

We denote the new version of ONTOMS where our incremental update strategy and SPARQL processor are applied as ONTOMS2. We designed the architecture of ONTOMS2 as depicted in Figure 2. The core modules of ONTOMS2 are OWL Data Storage Module, Instance Inference Module, SPARQL Module, and OWL-QL Module. To apply our incremental update strategy to ONTOMS, we create SPARQL Module in Query Processing Module and modified Instance Inference Module for the incremental reasoning. OWL Data Storage Module obtains class and property hierarchy information from OWL Reasoner, Pellet, and



GraduateStudent

UID	degreeFrom	degreeFrom_S	degreeFrom_E
GS1			

Professor

UID	degreeFrom	degreeFrom_S	degreeFrom_E
Prof1	Univ1	6	7

Course

UID
C1
C2

GraduateStudent_takesCourse

UID	Value
GS1	C1
GS1	C2

Professor_teachesCourse

UID	Value
Prof1	C1
Prof1	C2

Fig. 1. Relational Tables in ONTOMS

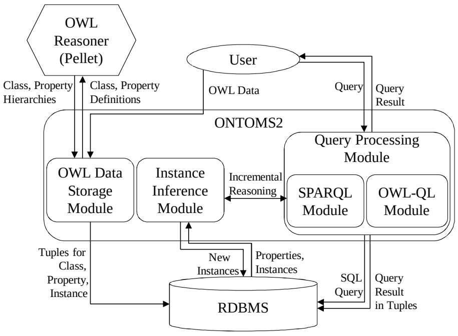


Fig. 2. The Architecture of ONTOMS2

generates relationship information among classes and properties when parsing the given OWL data. Once OWL data is stored, Instance Inference Module performs instance reasoning with properties which are collected along with their values from RDBMS, and stores newly generated instances into RDBMS. The details of instance reasoning for initial OWL data can be found at Section 6 in [6]. The Query Processing Module processes OWL-QL query and SPARQL query. When a SPARQL update query is given, SPARQL Module processes it through Instance Inference Module for the incremental reasoning.

2 Ontology Data Update

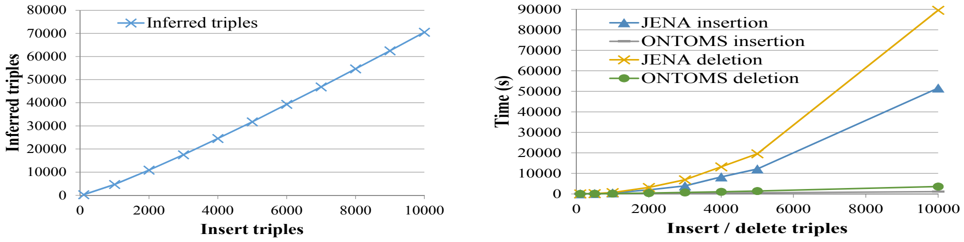
SPARQL Update There are several query languages for OWL such as OWL-QL and SQWRL . However, all of them support a read-only(select) query only. The previous version [6] of ONTOMS2 uses a OWL-QL to retrieve instances. We enhanced ONTOMS to have an ability to update ontology data by adding the SPARQL processor in addition to the OWL-QL processor. This is because OWL is developed as a vocabulary extension of RDF, SPARQL is a de facto query language for RDF, and recently W3C published a recommendation [2] for the SPARQL update.

According to the SPARQL Update recommendation proposed by W3C, SPARQL Update provides three operations related to update : INSERT DATA, DELETE DATA, and DELETE/INSERT. The INSERT DATA operation is to add new triples into the ontology data while the DELETE DATA operation is to remove triples from the ontology data. Lastly, the DELETE/INSERT operation is to remove triples and add new triples into the ontology data with the WHERE clause. ONTOMS2 supports all INSERT DATA, DELETE DATA, and DELETE/INSERT operations. Due to the space limitation, we omitted the detailed syntax. Please refer Chapter 3 SPARQL 1.1 Update Language in [2] for the detailed syntax.

Incremental Reasoning Only some of existing ontology data management systems support the update for ontology data. Even though there are some existing systems with the update feature, the instance reasoning for the updates has to be conducted from the scratch due to the constraints of the system while ONTOMS supports an incremental reasoning. Thus, the instance reasoning of the existing systems for each update is performed by processing all the stored triples. Eventually, it degrades the update processing performance.

OWL defines several types of properties. However, only inverseOf, symmetric and transitive properties may generate new facts(triples). The incremental reasoning for the inverseOf and symmetric properties can be done in a straight forward way. For the insertion and deletion of transitive property triples, we adapt the SQL-based transitive closure maintenance algorithm presented in [3] to effectively maintain the transitive properties. We cut down the step for finding truly new tuples among the generated tuples by unifying the transitive closure table and multi-value class property table. This reduces costly table join operations and it also reduces a storage size.

3 Experiments



(a) The number of inferred triples

(b) The update processing time

Fig. 3. Experimental results

The most important enhanced point of ONTOMS2 over ONTOMS is that ONTOMS2 supports an efficient ontology data update with an incremental reasoning. Therefore, we focused on the incremental reasoning performance when ontology data is updated. We compare ONTOMS2 with JENA which is one of the most popular ontology data management systems. JENA provides the SPARQL update feature through ARQ. However, ARQ only supports the instance reasoning with non-update(SELECT) queries[1]. Therefore, we implemented a simple SPARQL update interface for JANA which uses OntModel. Among reasoners in JENA, *OWL_MEM_MICRO_RULE_INF* is used.

Note that, in JENA, the instance reasoning should be re-run from the scratch for each update query and the update processing time for JENA does not contain the time for storing the results of the instance reasoning into the relational tables while ONTOMS2 does the incremental reasoning and the update processing time for ONTOMS2 contains the storing time. Our experiments were performed on 2.27GHz Intel Xeon with 12GB of main memory. We implemented ONTOMS2 using MS SQL Server 2008 and Java. For JENA [1], Jena-2.6.4 version is used.

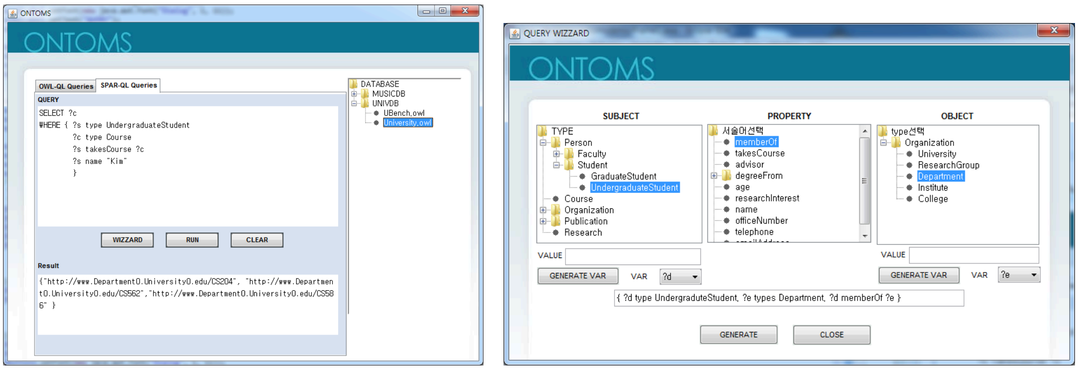
Figure 3(a) shows the number of triples to be inferred for various sizes of insertion triples(facts). All fact triples have same transitive property. New triples to be inserted as a facts are generated such that their subjects and objects are randomly selected from subjects and objects of previously inserted fact triples.

---

[1] ARQ supports a SPARQL update query on basic Model only, not on OntModel or InfModel. However, JENA uses OntModel or InfModel for the reasoning.

In JENA, the instance reasoning for each insertion is conducted from the scratch while the instance reasoning of ONTOMS2 is incrementally conducted. As a result, ONTOMS2 outperforms JENA for the insertion and deletion due to the incremental update strategy. The results are as shown in Figure 3(b).

4 Demo



```
SELECT ?c
₩HERE { ?s type UndergraduateStudent
        ?c type Course
        ?s takesCourse ?c
        ?s name "Kim"
        }
```

```
{"http://www.Department0.University0.edu/CS204", "http://www.Departmen
t0.University0.edu/CS562","http://www.Department0.University0.edu/CS58
6" }
```

```
{ ?d type UndergraduteStudent, ?e types Department, ?d memberOf ?e }
```

Fig. 4. Query interface and Query wizard of ONTOMS2

During the demonstration, we will illustrate how our system handles SPARQL queries over ontology data such as retrieving instances of classes or properties, inserting/removing instances which satisfy a specific condition with an incremental reasoning. Our system also contains a GUI-based wizard to help a user to create a SPARQL query easily. The screen shots of ONTOMS2 are depicted in Figure 4 and the recorded video can be found at our demo page (http://islab.kaist.ac.kr/ONTOMS2/).

Acknowledgments This work was supported by the National Research Foundation of Korea grant funded by the Korean government (MSIP) (No. NRF-2009-0081365).

References

1. Jena - a semantic web framework for java. http://jena.sourceforge.net/index.html/.
2. Sparql 1.1 update, w3c recommendation 2013. http://www.w3.org/TR/sparql11-update/.
3. G. Dong, L. Libkin, J. Su, and L. Wong. Maintaining the transitive closure of graphs in sql. *Int. J. Information Technology*, 5:46–78, 1999.
4. S. Kang, J. Shim, and S. goo Lee. Tridex: A lightweight triple index for relational database-based semantic web data management. *Expert Syst. Appl.*, 40(9):3421–3431, 2013.
5. Z. Pan, X. Zhang, and J. Heflin. Dldb2: A scalable multi-perspective semantic web repository. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, volume 1, pages 489–495. IEEE, 2008.
6. M.-J. Park, J. Lee, C.-H. Lee, J. Lin, O. Serres, and C.-W. Chung. An efficient and scalable management of ontology. In *Proceeding of DASFAA '07*, pages 975–980, 2007.

SPACE: SPARQL Index for Efficient Autocompletion

Kasjen Kramer, Renata Dividino, and Gerd Gröner

WeST, University of Koblenz-Landau, Germany

Abstract. Querying Linked Data means to pose queries on various data sources without information about the data and the schema of the data. This demo shows SPACE, a tool to support autocompletion for SPARQL queries. It takes as input SPARQL query logs and builds an index structure for efficient and fast computation of query suggestions. To demonstrate SPACE, we use available query logs from the USEWOD Data Challenge 2013.

1 Introduction

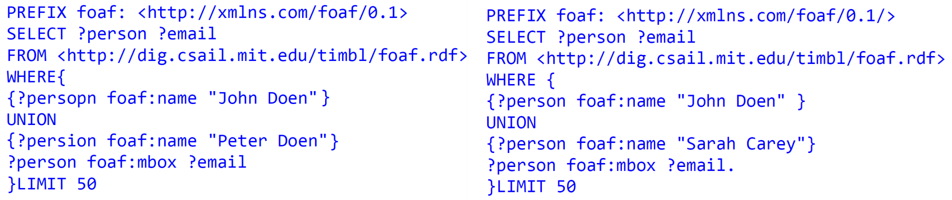
The linked data cloud is mainly accessible in two ways: SPARQL queries and direct RDF requests. Querying linked data on the Web using SPARQL is different to querying (relational) databases. First, linked data connect various data sources with heterogeneous data. Second, the schema and type statements are often unknown to the user or even missing in the data source. Thus, querying linked data means to pose queries on various data sources without information about the vocabulary and structure of the data.

In order to assist users when writing SPARQL queries, we show SPACE, a tool to support autocompletion of SPARQL queries. Autocompletion enables users to write queries fast. Existing approaches for query writing assistance make use of RDF datasets to extract query suggestions or they extract data source descriptions [1, 5] and/or relationships [2, 3]. Instead, we explore query logs available from SPARQL endpoints. We argue that previously executed queries are valuable information sources about the underlying data structure and schema of data sources. These queries reveal how resources are related and they reflect the user interests on resources and their relationships.

Our tool aims at enhancing usability of SPARQL query writing by providing suggestions of different possible query formulations to the user. Besides this, it enables fast and efficient computation of new suggestions. The computation of suggestions relies on an index structure for SPARQL queries. The SPACE index structure incorporates the structure and composition of graph patterns in SPARQL queries.

2 The SPACE Data Structure for Indexing SPARQL Queries

When a user writes a SPARQL query, SPACE aims to find the most similar queries available in the query logs in order to build new suggestions. A SPARQL query is a tuple defined as $Q = (A, V, G, P, M)$, where A is the set of prefix declarations (Line 1 in Fig. 1(a)), V is the output form (Line 2 in Fig. 1(a)). G refers to the RDF graph(s) being queried (Line 3 in Fig. 1(a)), P is a graph pattern (Line 5-8 in Fig. 1(a)) and M are query modifiers (Line 9 in Fig. 1(a)). In this work, we focus only on graph patterns. In that view, a query is composed only by its P. The core of SPACE is its index structure. The index structure is a graph, representing a set of SPARQL queries.



```
PREFIX foaf: <http://xmlns.com/foaf/0.1>
SELECT ?person ?email
FROM <http://dig.csail.mit.edu/timbl/foaf.rdf>
WHERE{
{?persopn foaf:name "John Doen"}
UNION
{?persion foaf:name "Peter Doen"}
?person foaf:mbox ?email
}LIMIT 50
```

(a) It returns the email of the persons named John Doen and Peter Doen

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?person ?email
FROM <http://dig.csail.mit.edu/timbl/foaf.rdf>
WHERE {
{?person foaf:name "John Doen" }
UNION
{?person foaf:name "Sarah Carey"}
?person foaf:mbox ?email.
}LIMIT 50
```

(b) It returns the email of the persons named John Doen and Sarah Carey

Fig. 1. SPARQL Queries: Toy scenario

Definition 1 (SPACE Index). *The SPACE index $\mathcal{I}$ is a hierarchical index in form of a directed acyclic ordered graph $\mathcal{I} = (\mathcal{V}, \mathcal{E})$. Each vertex $v \in \mathcal{V}$ is associated with a level $l(v) \in 0, \ldots n-1$. Each edge $(v, v') \in \mathcal{E}$ leads to a vertex at a higher level $(l(v) < l(v'))$. The partial relation $\subseteq$ (set-inclusion) on the set $\mathcal{V}$ defines $v \subseteq v'$ for all $(v, v') \in \mathcal{E}$.*

According to Def. 1, the index structure has the following shape:

1. The vertices at the highest index level $(n-1)$ are represented by elements of the (pairwise disjoint) infinite sets I, B, L and V (IRIs, Blank nodes, literals and variables). Additionally, they represent the binary operators AND, UNION, OPT, FILTER, and GRAPH used to combine graph patterns. These vertices have only incoming edges.
2. The vertices from index level $n-2$ until index level 1 represent graph patterns, according to the recursive definition in [4]. A triple pattern is a graph pattern of the form $(I \cup L \cup V) \times (I \cup V) \times (I \cup L \cup V)$. If P1 and P2 are graph patterns then (P1 AND P2), (P1 OPT P2), and (P1 UNION P2) are also graph patterns. Given a SPARQL built-in condition R, then (P1 FILTER R) is a graph pattern. Finally, given a G $\in I$ or $\in V$, then (G GRAPH P) is a graph pattern.
3. The vertices from index level 1 represent SPARQL queries. Each query is composed by one graph pattern.
4. The (single) vertex, at the (lowest) level 0 (also called root vertex) represents a set of queries, e.g., all queries of a query log. This vertex has only outgoing edges.

Please note that, the number of graph patterns in the queries determine the height of the index tree. To illustrate our approach, Fig. 1(a) and Fig: 1(b) present two SPARQL queries. The first query searches for the email of the persons named John Doen and Peter Doen. The second one searches for the email of the persons named John Doen and Saray Carey. The SPACE index structure is shown in Fig. 2. The IRIs *foaf:name*, *foaf:mbox*, the literals *'John Coen'*, *'Peter Coen'* and *'Sarah Carey'*, the variables *?person* and *?email*, as well as the operator *AND* and *UNION* are represented by the nodes at the last level. The graph patterns are represented in the levels above. For instance, the triple pattern $t1$ is composed by the nodes *?person*, *foaf:name* and *'John Coen'*.

The process of searching for suggestions is done by sub-graph matching. Whenever there is a match of the query written by the user in the index graph, the tool is able to provide suggestions. The suggestions are ordered regarding to a popularity score. The popularity score represents the frequency of an element in the queries of the dataset.

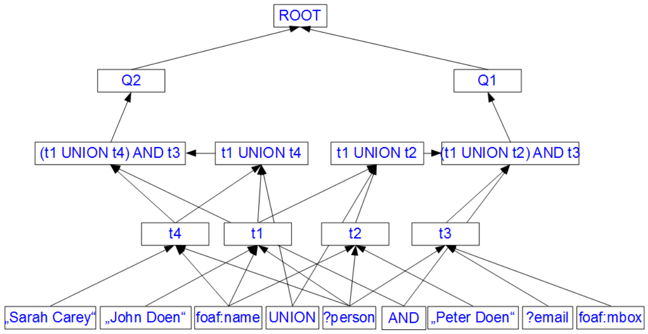


Fig. 2. SPACE Index of our example

When the user start writing a query, up the first symbols he writes, he gets some suggestions. For instance, if the user starts with the symbols ⟨, then the tools suggest all possible URIs found in the graph' nodes. If the user writes ”*?*”, the tool searches for all the nodes representing a variable in the index. Given a variable, the possible follow-up suggestions are the predicates nodes, in our case, the predicates in ”foaf:name” and ”foaf:mbox”, since they are the only predicates connected to variable nodes in the index graph. The tool only suggests (parts of) already observed queries. The more the user writes the smaller is the region where the query may be located in the index graph. Therefore, the longer the written query is, the more precise are the suggestions. The tool searches for the most similar queries in the query log in a bottom-up matching manner. For speeding up, an extra index for prefixes and namespaces is built. Please note that, nodes representing variables are not named and can be seen as just placeholders. The substitution method is used to check the equality of graph patterns.

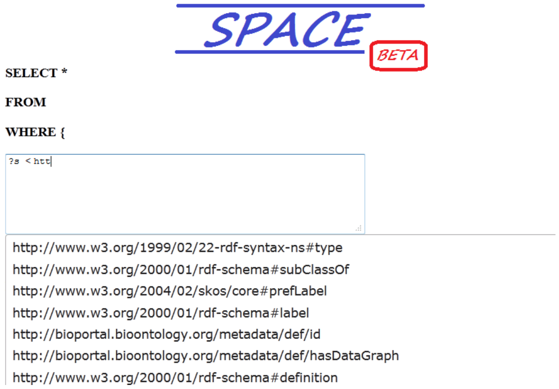
3 SPACE in Use

Dataset: To demonstrate our tool, we collect queries from available query logs from the USEWOD Data Challenge 2013 [1] posted to SPARQL endpoints. In particular, the logs are from the two following sources: Open-BioMed.org.uk and BioPortal. The Open-BioMed.org.uk service offers gene expression search for Drosophila research, as well as drug discovery for the Alzheimer's disease. BioPortal provides access to commonly used biomedical ontologies.
Tool demonstration: The SPACE[2] is a web application tool written in Java and is based on the Jena framework. Its client-side is implemented in Javascript code. Fig. 3 shows a screenshot of the tool. SPACE is composed of two parts: (1) the not-editable part, representing an incomplete SELECT query and (2) the editable part, representing the graph pattern of the query. Suggestions are given to the user starting from the moment the user writes something in this field. The autocompletion functionality includes suggestion of IRIs such as classes and properties, of literals, of variables, of SPARQL binary operators as well as of namespaces and prefixes.

---

[1] USEWOD Data Challenge: `data.semanticweb.org/usewod/2013/challenge.html`

[2] SPACE: `http://west.uni-koblenz.de/Research/systems/SPACE`



SELECT *

FROM

WHERE {

```
?s <htt
```

http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/2000/01/rdf-schema#subClassOf
http://www.w3.org/2004/02/skos/core#prefLabel
http://www.w3.org/2000/01/rdf-schema#label
http://bioportal.bioontology.org/metadata/def/id
http://bioportal.bioontology.org/metadata/def/hasDataGraph
http://www.w3.org/2000/01/rdf-schema#definition

Fig. 3. SPACE screenshot

4 Conclusion and Outlook

In this paper, we have shown SPACE, a SPARQL editor that assists users when writing SPARQL queries. The tool is based on a hierarchical index structure of SPARQL queries, which enables fast computation of the most similar queries that are available in the query logs in order to generate new suggestions. So far, we have focused on the suggestions of query patterns.

We plan to proceed this research into three directions: (1) incorporate all operators of SPARQL and apply optimizations, (2) combine with approaches based on dataset statistics to improve recommendation, (3) conduct a user evaluation to get feedback from users to estimate which suggestion is intuitive with respect to human feeling.

References

1. S. Campinas, T. E. Perry, D. Ceccarelli, R. Delbru, and G.. Tummarello. Introducing RDF Graph Summary with application to Assisted SPARQL Formulation. In *DEXA*, 2012.
2. T. Gottron, A. Scherp, B. Krayer, and A. Peters. Lodatio: Using a schema-level index to support users in finding relevant sources of linked data. In *K-CAP'13*, 2013.
3. M. Jarrar and M. D. Dikaiakos. A Query Formulation Language for the Data Web. *IEEE Trans. on Knowledge and Data Engineering*, 24(5):783–798, 2012.
4. J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of sparql. *ACM Trans. Database Syst.*, 34(3):16:1–16:45, September 2009.
5. M. Zviedris and Barzdins G. ViziQuer: A Tool to Explore and Query SPARQL Endpoints. In *ESWC*, volume 6644 of *LNCS*, pages 441–445. Springer, 2011.

SemantEco Annotator

Patrice Seyed[1,2], Katherine Chastain[2], Brendan Ashby[2], Yue Liu[2],
Timothy Lebo[2], Evan Patton[2], and Deborah McGuinness[2]

[1]DataONE, University of New Mexico, Albuquerque, NM 87131
[2]Tetherless World Constellation, Rensselaer Polytechnic Institute, Troy, NY 12180, USA
{seyeda2, chastk, ashbyb, liuy18, lebot, pattoe, dlm}@rpi.edu

Abstract. Generating useful RDF linked data is not a straightforward process for scientists using today's tools. In this paper we introduce the SemantEco Annotator, a semantic web application that leverages community-based vocabularies and ontologies during the translation process itself to ease the process of drawing out implicit relationships in tabular data so that they may be immediately available for use within the LOD cloud. Our goal for the SemantEco Annotator is to make advanced RDF translation techniques available to the layperson.

1 Introduction

Scientists generating datasets of tabular data make choices about how they record that data. These decisions are informed by many factors, including how scientists understand and analyze the data. Often, information that can be gleaned from a table alone is limited by the initial input strategy, including what is appropriate for the table headers (i.e., attributes) and cell values, and the conventions for structure and terminology. There may be implicit information apparent only to the creator, leading to challenges for uniformly interpreting data generated by different researchers. Linked data formats based on web standards such as RDF provide an avenue for addressing this problem, allowing the data to be more explicitly represented in statements uniquely typing entities and relationships described in data using common vocabularies. Unfortunately, the process for translating source data into linked data has its own barriers, as software tools allowing scientists that are inexperienced with linked data to perform translations are lacking.

In our demonstration we introduce SemantEco Annotator, a semantic web application for translating tabular data into RDF for immediate use in the LOD cloud. The annotator serves as a frontend to the *csv2rdf4lod* [1] conversion tool that uses an RDF metadata vocabulary to specify a variety of powerful techniques for translating data from CSV to RDF. We see the SemantEco Annotator as an application that makes translation more accessible to the layperson, and which can act as a module in a larger semantic workbench. We created our initial implementation for our SemantEco environment, which uses linked data to present water quality data in a visual and interactive manner [2]. In the next section we describe other RDF translation tools, following which we highlight the primary translation techniques the annotator currently enables, and finally we provide some insight into how the application was developed, and our future work.

2 Related Work

There are existing tools both for converting tabular data to RDF and for semantic annotation. RDF Refine[1] is an extension to OpenRefine where conversion to linked data requires that a user 1) define a "skeleton" for the final RDF as a tree structure, 2) maps table columns to nodes of the tree. Our tool combines these steps, providing a simpler interface by keeping all of the interactions for RDF mapping situated with the original data table. The users also retain the flexibility to define or change the schema as they go along. RDF Refine also allows certain types of conversions through "transposition" that requires the user to change the structure of the original tabular data in order to perform the conversion. Our approach enables similar modifications while maintaining the original structure of the CSV file.

Anzo Express[2] by Cambridge Semantics enables semantic annotation of Excel spreadsheets, and also provides various other functions, including an OWL ontology editor. Our tool focuses on re-use of existing ontologies and community-standard vocabularies. In contrast to both RDF Refine and Anzo Express, *csv2rdf4lod* has the added functionality of generating provenance statements using popular vocabularies[3] that enable tracking the original dataset through each of the stages from retrieval, through conversion, and its publication as dump files or into a SPARQL endpoint.

3 Introducing the SemantEco Annotator

To illustrate the primary translation techniques available through the annotator, we consider data from the Darrin Freshwater Institute that tracks the quality of lake water in New York State's Adirondack region. The techniques are 1) row and cell-based translation; 2) implicit and explicit bundling; and 3) leveraging OWL ontology classes, properties, and individuals to formulate new RDF statements. Each of these techniques is demonstrated and discussed in detail in our video linked at https://github.com/apseyed/SemantEcoAnnotator/wiki.

3.1 Row and Cell-Based Translation

A simple RDF translation of a table generates a triple for each non-empty cell value, where the subject, predicate, and object URIs are formed from the row, column, and cell value, respectively. The subject URI is either generated from a convention or

---

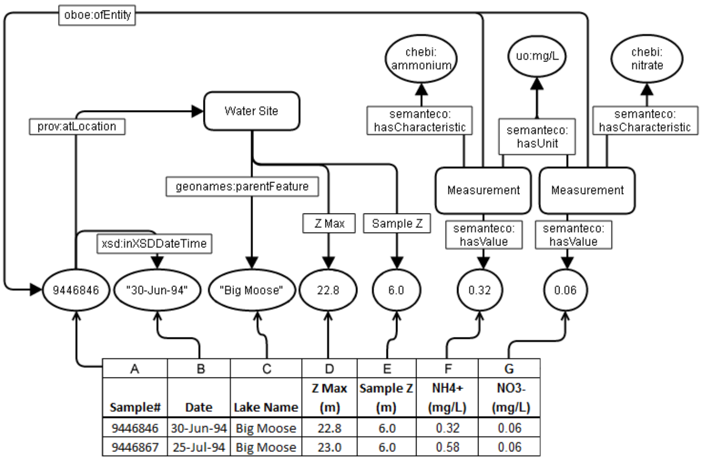
[1] http://refine.deri.ie/

[2] http://www.cambridgesemantics.com/products/anzo-express

[3] http://prefix.cc/void,dcterms,prov

constructed from a column's cell value. This default technique as *row-based translation*, since every triple generated from a row shares the same subject.

In more complex tabular arrangements, each row may instead contain many subjects that are described across multiple columns, and are related direct or indirectly. The technique for this pattern is *cell-based translation*. Figure 1 illustrates a scenario where it is useful for columns F and G, which contain measurements of NH4+ and NO3, respectively, for a water sample. Direct or indirect features of the measurements are contained in other cells in the same row, and relate back to the sample. Columns are designated for cell-based translation via a context menu. Translating the indirect relationships requires *bundling*, which we discuss in the next section.



A	B	C	D	E	F	G
Sample#	Date	Lake Name	Z Max (m)	Sample Z (m)	NH4+ (mg/L)	NO3- (mg/L)
9446846	30-Jun-94	Big Moose	22.8	6.0	0.32	0.06
9446867	25-Jul-94	Big Moose	23.0	6.0	0.58	0.06

Fig. 1. An excerpt of water quality data from Darrin Freshwater Institute including information about the sample (*Cols. A-E*) and two of its measurements (*Cols. F,G*). Directed arcs above the table illustrate relationships between what is described in columns, which our annotator helps a user describe and ultimately generates for the user via translation into RDF.

3.2 Bundling

Sometimes a column represents a description of an entity identifiable in another column. The Date (col. B) describes when the Sample (col. A) was taken; the user can capture this relationship by creating an *explicit bundle*. In contrast, columns may describe an unspecified, or implicit set of entity not captured by another. For the water sample data, the Lake Name (col. C), the depth to the bottom of the lake (col. D, Z Max), and the depth of sample collection (col. E, Sample Z) together represent a sample location. The user can represent the implicit entity by bundling the columns that represent it together, in an implicit bundle. Bundled columns are visually "pushed down" into a new row in the header to show they are now grouped, and subordinate aspects of the entity they are bundled into. After the bundling selection is complete, the user has a choice of one of the existing column identifiers for explicit from a drop-down list, or declaring it an implicit bundle.

For the water measurement columns (F, G) designated for cell-based conversion, there is vital unit information in the header. We provide a *subject annotation* feature to directly assert out-of-band triples about the measurement units. Annotations are created with the context menu and accept dragged properties and classes to generate further triples. We illustrate these steps in our demonstration video.

3.3 Enhancing RDF Translation with OWL Ontologies

In connection with the aforementioned translation techniques, we provide mechanisms for using OWL ontologies. The right side of the interface includes a catalog of ontologies to choose from, which in turn populates trees with choices of the ontologies' classes, properties, and datatypes, for adding semantics to the resultant RDF data. If an object, data, or annotation property node is dragged from the tree interface into a column, the triples generated for that column will use the given predicate. Dragging a class node into the column will type the object of the triple as an instance of the given class. We also employ restrictions to selections based on previous choices for a column. This is highlighted in our demonstration video, through use of various scientific-observation ontologies such as OBO-E[4].

4 Discussion and Future Work

Once enhancements are committed RDF files for the enhancement parameters and the RDF produced by *csv2rdf4lod* are downloadable for use as LOD. In the future we will leverage logical restrictions from ontologies to guide the user to constructs that are most appropriate for subsequent enhancements. We also plan to extend the ontology menu to load any ontology listed in catalogs such as BioPortal[3], as well as via URI. While the annotator enables translation of data into RDF using existing ontologies and vocabularies, other components can enable mappings of terms within data packages to ontology concepts. This environment would essentially enable generation of semantically enriched data as linked data to enable better search capabilities.

References

1. Lebo, T., Erickson, J.S., Ding, L., Graves, A., Williams, G.T., DiFranzo, D., Li, X., Michaelis, J., Zheng, J., Flores, J., Shangguan, Z., McGuinness, D.L., and Hendler, J. Producing and Using Linked Open Government Data in the TWC LOGD Portal. In Linking Government Data, pages 51–72. New York, 2011. 10.1007/978-1-4614-1767-5_3
2. Seyed, A. P., Lebo, T., Patton E., McCusker J., and McGuinness D. L. SemantEco: A Next-Generation Web Observatory. 1st International Web Observatory Workshop. 22nd International World Wide Web Conference, Rio de Janeiro, May 13-17, 2013.

---

[3] https://semtools.ecoinformatics.org/oboe

[4] http://bioportal.bioontology.org/ontologies

A user interface to build interactive visualizations for the semantic web

Miguel Ceriani, Paolo Bottoni, and Simona Valentini

Sapienza, University of Rome, Italy
ceriani@di.uniroma1.it, bottoni@di.uniroma1.it, simo.valentini@hotmail.it

Abstract. While the web of linked data gets increasingly richer in size and complexity, its use is still constrained by the lack of applications consuming this data. We propose a Web-based tool to build and execute complex applications to transform, integrate and visualize Semantic Web data. Applications are composed as pipelines of a few basic components and completely based on Semantic Web standards, including SPARQL Construct for data transformation and SPARQL Update for state transition. The main novelty of the approach lays in the support to interaction, through the availability of user interface event streams as pipeline inputs.

1 Introduction

A number of tools, libraries and frameworks exist for manipulating and visualizing Semantic Web data. The RDF applications built with these tools are usually written in a host (imperative) programming language, possibly using (declarative) Semantic Web languages (SPARQL, OWL, rule languages) at specific steps of the process. The combination of different approaches, though effective from a programming point of view, makes it difficult to a user to grasp the overall process, at the same time making the reuse of parts of the process harder.

We propose an entirely functional approach in which an application is designed building a pipeline composed of a set of operators on RDF graphs. A Web-based application supports the developer in visually building the pipelines, which can then be used as components of other pipelines to achieve complex applications through a modular approach, under complete user control.

The application generated from a pipeline is also Web-based and is responsive to relevant user interface events, thanks to automatically generated event management and AJAX [1] client/server interaction.

After discussing related work in Sect. 2, Sect. 3 describes the pipeline editor, while Sect. 4 presents the software architecture and Sect. 5 outlines the presented demo Finally, Sect. 6 discusses conclusions and future work.

2 Related Work

During the last few years, two pipeline languages, DERI Pipes [1] and SPARQLMotion [2], have emerged as the state of the art for defining RDF trans-

---

[1] Asynchronous JavaScript And XML

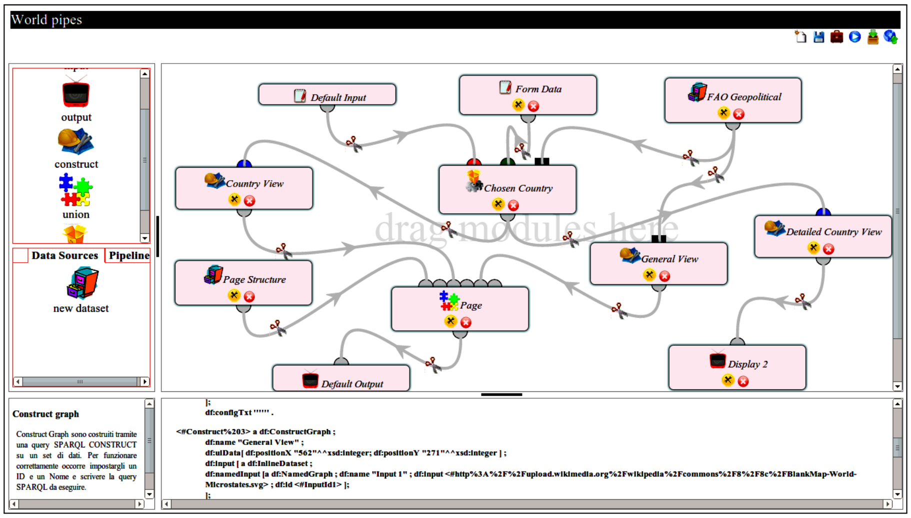
formations, offering a high level interface to execute operations on linked data, providing a set of basic operators on RDF graphs to build the pipelines, which are then typically executed in the context of a batch or Web application, in the latter case in response to GET or POST requests. Similarly, *networked graphs* [3] propose a view-based approach, where an RDF-based syntax is used to define new graphs in terms of queries on existing graphs, possibly using recursion.

The Visualbox [4] and Callimachus [5] systems have been proposed explicitly for linked data visualization. In their two-step model/view approach, SPARQL queries select data and a template language generates the (XML) visualization.

We leverage these approaches to give developers the possibility of specifying event-based interactive applications through pipelines composing query-based operators on graphs. As the User Interface state and events can be used as inputs of the queries, programmers can build complex visualization scenarios.

3 Dataflow Editor

The editor is contained in a Web page (see Fig. 1) providing tools to create and modify dataflows saved as RDF graphs on a Graph Store [6] and composed of:



Construct graph

Construct Graph sono costruiti tramite una query SPARQL CONSTRUCT su un set di dati. Per funzionare correttamente occorre impostargli un ID e un Nome e scrivere la query SPARQL da eseguire.

```
        ];
        df:configTxt """" .

<#Construct%203> a df:ConstructGraph ;
        df:name "General View" ;
        df:uiData[ df:positionX "562"^^xsd:integer; df:positionY "271"^^xsd:integer ] ;
        df:input [ a df:InlineDataset ;
        df:namedInput [a df:NamedGraph ; df:name "Input 1" ; df:input <#http%3A%2F%2Fupload.wikimedia.org%2Fwikipedia%2Fcommons%2F8%2F8c%2FBlankMap-World-
        Microstates.svg> ; df:id <#InputId1> ];
        ];
```

Fig. 1. A screenshot of the web-based editor

- a *component panel*(left), where components represent language operators;
- the *data sources tab* (left), allowing reference to RDF data sources already known to the system or the creation of new ones;
- the *pipelines tab* (left), allowing developers to use, view or modify other pipelines created by the user or any pipeline available online (in which case it would not be modifiable);

- the *editor area* (center), where the pipeline is built by dragging, linking and configuring the desired components;
- the *command panel* (upper right), contains some buttons for operations related to the whole pipeline, e.g. saving it or executing it;
- the *helper area* (bottom-left corner), for contextual help on components;
- the *source code area* (bottom), showing the RDF graph for the dataflow.

3.1 Components

A pipeline is a side-effect-free dataflow programming module, taking as input an RDF Dataset and returning another RDF Dataset. Each component is identified by a `Name` (visualized in the interface) and an `ID` (used as fragment identifier to programmatically identify the component). The available components are:

- the *default input graph* and the *(named) input graphs*
- the *default output graph* and the *(named) output graphs*
- the *union graphs* generated by merging the RDF triples of a set of graphs;
- the *construct graphs* generated by executing a SPARQL 1.1 Construct [7] query against a set of graphs;
- the *updatable graphs*, whose content is incrementally modified during an execution of the pipeline by executing a SPARQL 1.1 Update [8] request against a set of graphs each time one of these graphs changes;
- existing *pipelines* which can be used as components in the current pipeline.

A pipeline can be designed just for reuse by other pipelines. If a pipeline has to be executed (i.e. it is a *top level* pipeline), its default output graph must comply with an *XML DOM Ontology*[2] describing the XML DOM in RDF. It will represent a HTML or SVG document, to be rendered by the user interface. Its default input graph will receive the DOM Events generated in the user interface, described with a *DOM Events Ontology*[3].

4 Software Architecture

The main blocks of the application are the *editor*, the *pipeline repository* and the *dataflow engine*. The editor is a rich Web application with its client side logic coded in HTML+CSS+JavaScript. The pipeline repository is an instance of Graph Store that must be located in the same host of the editor. The dataflow engine is a Java based (using Apache Jena [9]) web application that maintains the state of each running pipeline instance; when a new instance is launched (e.g., from the editor) the engine initializes the pipeline and returns to the client its output along with a piece of JavaScript logic to report the handled events back to the server; each time an event is fired on the client, the dataflow engine is notified and answers with the changes that have to be executed on client content.

---

[2] `http://www.swows.org/2013/07/xml-dom`

[3] `http://www.swows.org/2013/07/xml-dom-events`

On the client side, any *modern* browser with JavaScript support is sufficient to use both the editor and the generated application. The software is free and available on line[4].

5 Demonstration

We will demonstrate how to build a data visualization through the creation of a pipeline. Static content (e.g., an SVG world map) will be referenced as Datasource and composed with dynamic content built from RDF Datasources (e.g., the FAO Geopolitical Ontology) using some chained Construct queries (e.g., to color the map based on some statistic); to add interaction the Default Input of the pipeline (user interface events) will be connected to an Updatable Graph storing the application state (e.g., the selected statistic)[5].

6 Conclusions and Future Work

We have presented a user interface to build pipelines which specify transformations of RDF graphs in order to build data visualization applications. The RDF pipeline language is based on existing standards (such as SPARQL) and is unique in having been designed for interactive applications and thus able to react to graph modification events.

We are a proposing this system as a proof-of-concept and as a test bed for experimentation in RDF programming with a dataflow approach. We want to leverage this experimentation to build higher level interfaces, designed also for usage by non-expert users, as a way to flexibly interact with linked data.

References

1. Le-Phuoc, D., Polleres, A., Hauswirth, M., Tummarello, G., Morbidoni, C.: Rapid prototyping of semantic mash-ups through semantic web pipes. In: Proc. WWW '09, ACM (2009) 581–590
2. Knublauch, H., et al.: SPARQLMotion Specifications (2010) sparqlmotion.org.
3. Schenk, S., Staab, S.: Networked graphs: a declarative mechanism for SPARQL rules, SPARQL views and RDF data integration on the web. In: Proc. WWW '08, ACM (2008) 585–594
4. Graves, A.: Creation of visualizations based on linked data. In: Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics, ACM (2013) 41
5. Battle, S., Wood, D., Leigh, J., Ruth, L.: The Callimachus Project: RDFa as a Web Template Language. In: COLD. (2012)
6. Ogbuji, C.: Sparql 1.1 Graph Store HTTP Protocol. W3C REC 21 March 2013
7. Harris, S., et al.: SPARQL 1.1 Query Language. W3C REC 21 March 2013
8. Schenk, S., Gearon, P., et al.: SPARQL 1.1 Update. W3C REC 21 March 2013
9. McBride, B.: Jena: a semantic Web toolkit. Internet Computing, IEEE **6**(6) (Nov/Dec 2002) 55–59

---

[4] `http://www.swows.org/`

[5] A clip of this example is available at `http://www.swows.org/?q=ISWC2013`.

Coordinating Social Care and Healthcare using Semantic Web Technologies

Spyros Kotoulas, Vanessa Lopez, Martin Stephenson, Pierpaolo Tommasi,
Wei Jia Shen, Gang Hu, Marco Luca Sbodio, Veli Bicer, Anastasios Kementsietsidis,
M. Mustafa Rafique, Jason Ellis, Thomas Erickson, Kavitha Srinivas,
Kevin McAuliffe, Guo Tong Xie, and Pol Mac Aonghusa

IBM Research

1 Introduction

Healthcare and Social Care are unique domains in terms of cultural importance, economic magnitude and complexity. On a cultural level, the level of advancement of a society is often measured in terms of protection of the less able. In economic terms, for 2009, total expenditure on healthcare in the United States was 2.6 trillion USD or 17.4% of the GDP[1]. Total expenditure on social care was 2.98 trillion USD or 19.90% of the GDP[2]. In terms of US Federal government expenditure, social security, medicare and medicaid amount to 45% of total spending. In terms of complexity, organizations that are involved in providing social and medical care are numerous and span a very wide domain. For example, AHIP, the trade association of health insurers numbers some 1300 members[3]; the number of hospitals registered with the American Hospital Association is 5724[4] and the number of homeless shelters surpasses 4000[5]. In addition, medical information is vastly complex: Nuance reports that LinkBase®[6] contains more than 1 million concepts. Social care depends on information from a very broad domain, ranging from criminal records to housing.

Coordinating social care and health care has been identified both as a major pain point and a significant opportunity in modern health and social systems [1]. Several studies have shown that costs can be contained and outcomes improved with a more holistic approach to care [2]. As a simple motivating example, consider an individual quartered in inappropriate housing while suffering from a relatively minor health issue, aggravated by the housing condition. As a result, the given individual frequently resorts to visiting emergency rooms, resulting in significant cost to the healthcare system and a less effective treatment. By itself, the housing situation does not warrant state intervention. Nevertheless, resolving it would dramatically improve the health situation, resulting in a better quality-of-life for the individual and lower costs for the health system.

---

[1] http://dx.doi.org/10.1787/888932523215

[2] http://www.oecd.org/els/social/expenditure

[3] http://www.ahip.org

[4] http://www.aha.org/research/rc/stat-studies/fast-facts.shtml, retrieved 19/04/2013

[5] http://www.shelterlistings.org/

[6] http://www.nuance.com/for-healthcare/resources/clinical-language-understanding/ontology/index.htm

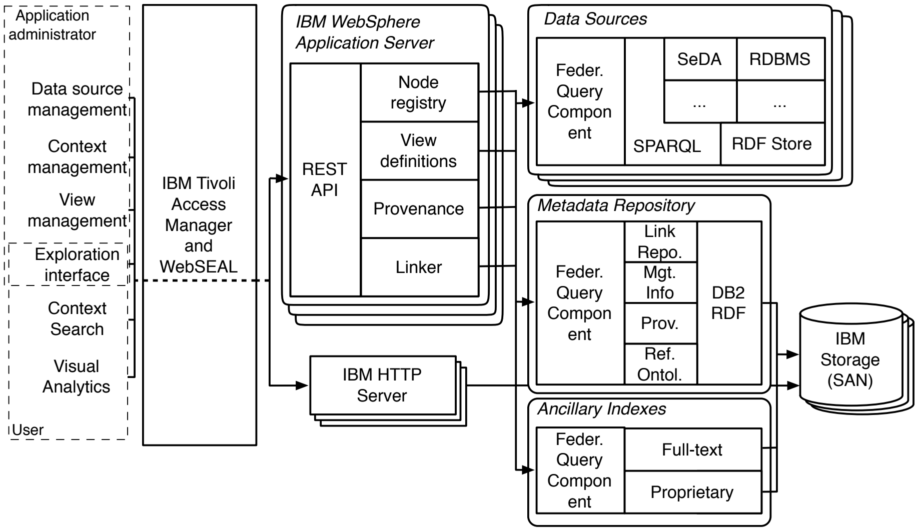


Fig. 1: System architecture

Even in this simple example, the challenges presented are significant: *How do we access information in disparate systems, storing vastly heterogeneous information on various infrastructures? How do we cope with policy constraints disallowing replication or centralization of data? How do we abstract from the information and representation complexity?*

In this paper, we propose a novel technical solution to augment applications with cross-domain context, in the domain of Social Care and Healthcare based on business rules and contextual exploration. We claim that Semantic Technologies can uniquely address these problems because: (a) The distributed nature of RDF allows access to integrated information across silos. (b) Explicit and global semantics allow us to ground business rules across systems. (c) The distributed and incremental data integration paradigm advocated by linked data can help coping with the complexity of the data.

We present a demonstrator of a system that supports two key use-cases for this domain: (a) Displaying a view of the combined needs across several dimensions for a given person and people in their social context, based on a set of business rules. This allows a social/health worker to quickly assess the situation of an individual. From a knowledge management perspective, it requires grounding a set of business rules across several ontologies and instance data in several data sources. (b) Exploration of the context to surface information not directly covered by the business rules. Given the heterogeneity of the domain, the user will most likely need additional information around a given individual. Our demonstrator uses the business rules as a navigational aid to explore the semi-structured information.

2 Approach

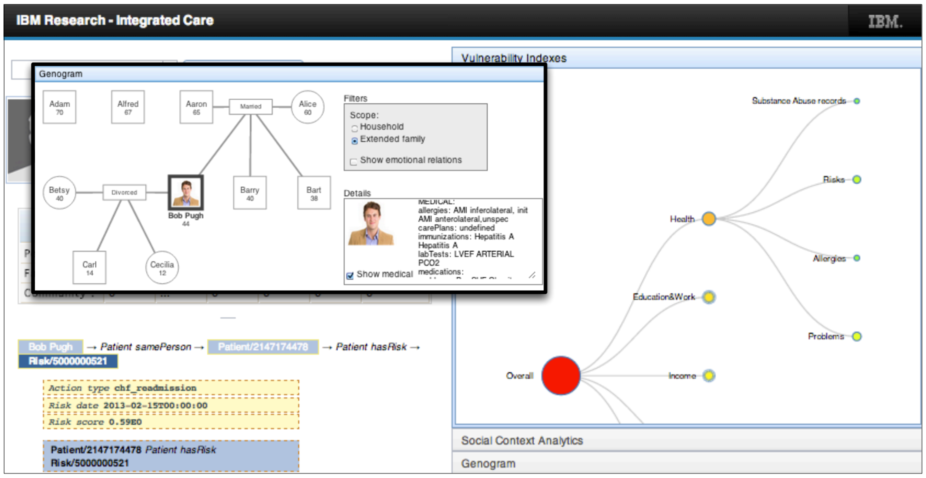
Key Performance Indicators (KPIs) are used to ground business rules to data, offering a tree-based view over the factors that contributed to a given KPIs and the weight of each factor (influence) to the global vulnerability score, helping us understand relations across different needs. For example, being homeless (or living under poor housing conditions) is an aggravating factor for health. These views can be applied to an individual, family members, or socially/geographically organized groups. Each node in a KPI is associated to two SPARQL queries. The first one is to calculate the score of a given contributing factor (if present) obtained from a given data source(s) and with a given weight. The second one is a CONSTRUCT query to retrieve the set of triples providing additional context in the ontology(-ies) associated to the data that contributed to the score, as well as the justification on the values from which this KPI factor was derived. The score of a KPI node is the sum of its own score and that of its children. For both types of queries, rather than being tied to a specific model, we abstract from the particular representation using a set of query patterns.

An enterprise architecture supporting our approach is shown in Fig. 1. Due to space restrictions, we describe only the components necessary to understand the basic operation of the system. Web-facing services use a set of REST services, implemented on a custom application running on IBM WebSphere Application Server. The main components for these services are the *Node registry*, which tracks nodes in the *Federated Query Engine*, the *View definitions*, that are used to project information out of the graph model for use by analytics widgets and UI elements. *Data Sources* are exposed as virtual RDF, using SeDA, an IBM technology to execute R2RML mappings. The virtual RDF Data Sources, the Metadata Repository and the Ancillary Indexes are accessed through the *Federated Query Engine*, providing transparent access to the distributed information. All core components in this architecture can be clustered, for high availability and performance.

3 Deployment

We have internally deployed a proof of concept based on the above architecture, integrating a set of IBM solutions for clinical and social program information: **IBM software Patient Care and Insights** provides data driven population analysis to support patient centered care processes. It integrates and analyzes the full breadth of patient information sourced from multiple systems and different care providers. It stores three categories of data: extracted patient medical history called clinical summary; medical data analytics results from an analytics component called care insights and personalized electronic care plans. **IBM Cúram** is a business and technology solution to help social program organizations provide optimal outcomes for citizens, satisfy increasing demand, and lower costs for organizations. In connection to this paper, the information of interest mainly regards social relationships, known problems concerning employment, substance abuse, participation in social assistance programs and information concerning housing, education and safety.

Figure 2 shows some UI components from our proof of concept. Since our approach is meant to be deployed as part of a existing application, in order to augment them with information from other systems, we have opted to focus on the context that can be retrieved, rather than trying to replicate the enterprise application: (a) *Genogram, Fig. 2, floating frame on top-left.* We have adapted the genogram visualization [3] to



MEDICAL:
allergies: AMI inferolateral, init
AMI anterolateral,unspec
carePlans: undefined
immunizations: Hepatitis A
Hepatitis A
labTests: LVEF ARTERIAL
PCO2
medications:

Bob Pugh → *Patient samePerson* → Patient/2147174478 → *Patient hasRisk* →
Risk/5000000521

Action type chf_readmission
Risk date 2013-02-15T00:00:00
Risk score 0.59E0

Patient/2147174478 *Patient hasRisk*
Risk/5000000521

Fig. 2: Screenshot

explore the family environment of a person and associated problems. (b) *Hierarchical KPI, Fig. 2, right.* The tree allows the user to explore the vulnerabilities of a person using information coming from several sources. The KPIs themselves are tree structures. Clicking on a node brings up a contextual exploration view. (c) *Contextual exploration, Fig. 2, bottom-left.* The user is able to investigate information related to a node in the KPI tree based on a graph exploration interface. In addition to elements shown in the figure, our proof of concept supports exploration and analysis based on the spatial component and family relations.

From internal feedback, the main strong points of our approach lie in the ability to consume data from heterogeneous sources without complicated data warehouses, associated ETL processes and setting up related infrastructures, although it remains to be seen whether the tooling required for a semantic approach will reach the sophistication of what is currently found in the enterprise domain. In addition, tabular or tree-like visualizations are strongly preferred to graphs. Future work lies in better-informed data exploration, mining the RDF graphs to identify meaningful relationships and data inconsistency checking across silos.

References

1. Rigby, M., Hill, P., Koch, S., Keeling, D.: Social care informatics as an essential part of holistic health care: A call for action. I. J. Medical Informatics **80**(8) (2011) 544–554
2. Peikes, D., Chen, A., Schore, J., Brown, R.: Effects of care coordination on hospitalization, quality of care, and health care expenditures among medicare beneficiaries. JAMA: the journal of the American Medical Association **301**(6) (2009) 603–618
3. Jolly, W., Froom, J., Rosen, M., et al.: The genogram. The Journal of family practice **10**(2) (1980) 251

A Distributed Reasoning Platform to Preserve Energy in Wireless Sensor Networks

Femke Ongenae[1], Stijn Verstichel[1], Maarten Wijnants[2], and Filip De Turck[1]

[1] Department of Information Technology (INTEC), Ghent University - iMinds,
Gaston Crommenlaan 8 bus 201, B-9050 Ghent, Belgium
`Femke.Ongenae@intec.ugent.be`
[2] Expertise centre for Digital Media (EDM), Hasselt University - iMinds,
Wetenschapspark 2, 3590 Diepenbeek, Belgium
`Maarten.Wijnants@uhasselt.be`

Abstract. A distributed reasoning platform is presented to reduce the energy consumption of Wireless Sensor Networks (WSNs) offering geospatial services by minimizing the amount of wireless communication. It combines local, rule-based reasoning on the sensors and gateways with global, ontology-based reasoning on the back-end servers. The Semantic Sensor Network (SNN) Ontology was extended to model the WSN energy consumption. One exemplary prototype is presented, namely the Garbage Bin Tampering Monitor (GBTM).

1 Introduction

The GreenWeCan [1] project investigates a "green" wireless city access network infrastructure able to offer geospatial services by aggregating data from multiple sources, in a scalable and cost-effective way, and minimizing energy consumption as well as the human exposure to electromagnetic radiation. The machines in a WSN range from heavily resource-constrained sensors to powerful back-end servers. These WSNs often use a hierarchical approach with a sink that interconnects the sensors and the back-end. Power management is important in WSNs. Sensors are often battery-operated, so their autonomy must be maximized. As radio transmissions needed for communication are costly operations [2], it is often beneficial to carry out as much processing as possible on the node itself.

Therefore, a distributed reasoning platform (Section 2) was utilized. Rule-based reasoning on the sensors allows for conclusions concerning measured variables to be drawn locally. A back-end ontology-based reasoning mechanism, which has a complete overview of the senor data being produced, can influence the behavior of the WSN nodes. Section 3 describes the ontology, which is used to model and reason on the sensor knowledge to reduce energy consumption.

The use of proven standard reasoning mechanisms in WSNs is still premature. However, the reduction in energy consumption by a reduced transmission rate, compared to the extra power needed for such processes, should result in a positive balance. Moreover, using standard reasoning algorithms, instead of proprietary ones, makes the approach more generic and facilitates reusability. A prototype was developed to demonstrate these advantages (Section 4).

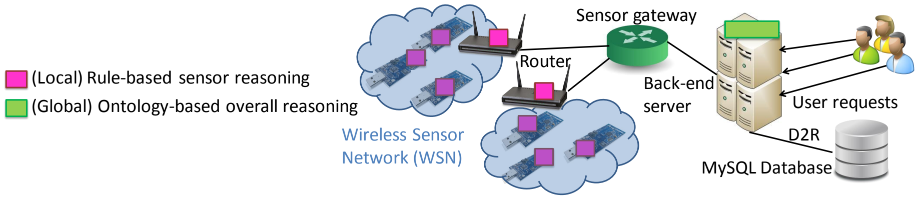

(Local) Rule-based sensor reasoning
(Global) Ontology-based overall reasoning
Wireless Sensor Network (WSN)
Router
Sensor gateway
Back-end server
User requests
D2R
MySQL Database

Fig. 1. Reasoning Architecture of the WSN

2 Reasoning architecture

As shown in Figure 1, information requested by the users is gathered from the sensors, which measure environmental parameters and pre-process them by performing rule-based reasoning. As such, less data is transmitted to the back-end. The complexity of the local reasoning can be adapted to the sensor's capabilities, e.g., battery, to optimize energy consumption. Moreover, the local reasoning is able to monitor the sensor's inner workings, e.g., CPU usage, in order to detect problems that influence energy consumption.

The pre-processed data is forwarded to the back-end via a gateway, optionally multi-hopping over routers. The sinks perform local, rule-based reasoning, e.g., to avoid retransmissions and preserve energy, the network load is monitored.

The back-end maintains an ontology to model the knowledge about the WSN and its observations. Static information, e.g., sensor specifications, is gathered from a database using D2R[3]. The received sensor data is integrated into this ontology to answer user requests and optimize the overall energy consumption.

Using an ontology ensures reusability and adaptability. Should new types of sensors be deployed, their semantic description and measurements can be mapped on the existing ontology. Moreover, by making the ontology publicly available as well as the data and conclusions corresponding to the run-time situation of the WSN, new applications can be created by anyone persuing a new usage and easy integration of this information.

The ontology can also be used to define the local, rule-based reasoning algorithms. The developed *Reasoning Sensor App Generator* generates sensor application code based on an XML-based application description. This description specifies a rule set and a template. The first contains the reasoning logic, which is executed each time the sensor wakes up. The second contains the code needed to run the reasoning logic on hardware.

3 A SSN Ontology extension modeling energy usage

The W3C Semantic Sensor Network Incubator group has developed the SSN Ontology[4] for modeling sensor devices and their capabilities, systems and processes. Based on brainstorms with GreenWeCan partners, i.e., OneAccess and Bausch Datacom, the requirements for the modeling concepts were defined. These were mapped on the SSN Ontology and some extensions were made. The relations

---

[3] `http://d2rq.org/d2r-server`

[4] `http://www.w3.org/2005/Incubator/ssn/ssnx/ssn`

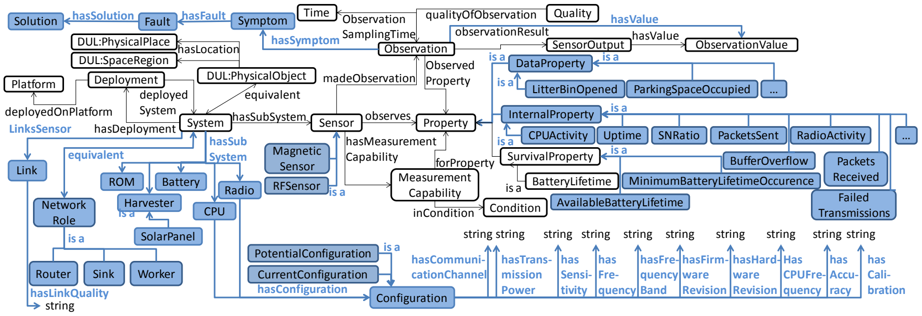


Fig. 2. The SSN Ontology extended (indicated in blue) to model energy consumption

between the sensor configurations, networks and applications and the influence on the energy consumption were also derived.

As shown in Figure 2, the SSN Ontology allows to model sensors, their observations and measurement capabilities. To avoid error propagation and retransmissions, the SNN was extended with concepts to make the quality of the observations explicit as they can be imprecise, ambiguous or erroneous. `Symptoms` define rules, which allow detecting specific phenomena in the observations. Axioms are defined that reclassify these `Symptoms` as `Faults` and `Solutions`.

The SSN `Property` concept models the type of metrics that can be observed. The `Data-` and `InternalProperty` subclasses are added to group the application-relevant observations monitored and the hardware-specific properties internally measured by the sensor. Figure 2 shows some internal properties to minimize energy consumption, e.g., sleeping schemes and radio settings can be adjusted to avoid buffer overflows and thus the amount of retransmissions.

The type of a sensor indicates which local reasoning techniques can be adopted. The `Sensor` concept in the SSN Ontology is annotated with a reference towards SensorML[5]. This specification can be used to reflect all the sensor's details.

`Battery`, `Harvester`, `ROM`, `CPU` and `Radio` concepts are introduced as these influence the reasoning complexity that can be used. The SSN Ontology already defines the `Battery LifeTime` property. Some other battery properties were added. The `Current-` and `Potential Configurations` of the radio and CPU are also modelled. The first models the currently used values for the characteristics, while the second represents the combination of values that can potentially be used together. Similarly, new sensors can be modeled, as shown in Figure 2 for the `Magnetic Sensor` and its settings, e.g. `Sensitivity`.

The location of the sensors can influence the energy consumption. The SSN Ontology models the WSN's deployment. To represent the physical locations the SSN Ontology aligns with the DOLCE Ultra Lite Ontology[6]. These concepts are preceded by the *DUL* namespace in Figure 2. `Link` and `NetworkRole` concepts are introduced to represent the network components used to interconnect the

---

[5] `http://www.opengeospatial.org/standards/sensorml`

[6] `http://www.loa.istc.cnr.it/ontologies/DUL.owl`

nodes and the role each node plays. Characteristics can be attached to the links, e.g., `LinkQuality`, which influences packet drops and retransmissions.

Finally, the context in which the WSN operates plays a role. Therefore, the ontology is linked to existing ones, e.g., the OWL Time ontology and DBPedia[7].

4 Garbage Bin Tampering Monitor Prototype

The GBTM monitors garbage bins in Ghent, which are used for small-scale litter disposal and are equipped with a sensor to detect the opening and closing of their cover. The cover can only be removed by a special-purpose key. Any other manipulations are illegitimate. A web-based interface[8] allows personnel to consult the observations, either as raw data, as an alligned table clustering data per bin or on a map. Anomalies are highlighted by combining the observations with external data, e.g., garbage collection timetables. The views also allow optimizing garbage collection routes and timetables.

Rule-based sensor reasoning The garbage bins are equipped with a magnet-activated reed switch, which stores a type, i.e., open or close, and timestamp in the sensor's ROM when a hardware interrupt occurs. To reduce the number of transmissions, rule-based reasoning accumulates the sensor readings during a configurable time interval, after which they are transported in bulk to a database on the back-end server, which exposes them via a D2R-based RESTful interface.

Ontology-based back-end reasoning The sensors issue their measurements once per time interval to the back-end. The *Next-Wake-Up-Time* configuration parameter determines the timepoint at which this happens. It is preferably avoided that sensors wake up at the same time as this increases the amount of retransmissions, particularly in single-hop topologies, due to collisions. Therefore, if such a situation is discovered, the back-end reasoner will use the WSN ontology to recalculcate a dephazed next wake-up time scheme.

Rule-based gateway reasoning When the gateway receives a request, it first checks if the required up-to-date info is available in its cache. If it is, the cached data is sent to back-end to reduce the amount of data transmitted and thus the energy consumption. If not, the measurements are retrieved from the sensors. Determining the time after which data in the cache should be refreshed is difficult. Applications preferably use the most recent measurements. However, they need to comply with legislation concerning how much Radio Frequency communication is used, e.g., 6 minutes per hour for a 169 MHz radio. Therefore, the gateway monitors the duty cycle and adapts its caching strategy accordingly.

References

1. M. Wijnants, et al.: An eco-friendly hybrid urban computing network combining community-based wireless LAN access and wireless sensor networking. In: Proc. of GreenCom. (2012) 410-417
2. P. De Mil, et al.: Design and implementation of a generic energy-harvesting framework applied to the evaluation of a large-scale electronic shelf-labeling wireless sensor network. EURASIP JWCN (2010) 12

---

[7] `http://www.w3.org/TR/owl-time/` & `http://wiki.dbpedia.org`

[8] `http://mediasharing2.edm.uhasselt.be/greenwecan_v3/php/gwc_usecase_gbtm.php`

SexTant: Visualizing Time-Evolving Linked Geospatial Data *

Konstantina Bereta[1], Charalampos Nikolaou[1], Manos Karpathiotakis[2], Kostis Kyzirakos[1], and Manolis Koubarakis[1]

[1] National and Kapodistrian University of Athens, Greece
[2] École Polytechnique Fédérale de Lausanne, Switzerland

Abstract. We present SexTant, a Web-based system for the visualization and exploration of time-evolving linked geospatial data and the creation, sharing, and collaborative editing of "temporally-enriched" thematic maps which are produced by combining different sources of such data.

1 Introduction and Motivation

Linked geospatial data has recently received attention as researchers and practitioners have started tapping the wealth of geospatial information available in the archives of various national cartographic agencies and making it available on the Web as linked data [2]. As a result, in the last few years, the Web of data is being rapidly populated with geospatial information. As the real-world entities represented in linked geospatial datasets evolve over time, the datasets themselves get updated and both the *spatial* and the *temporal dimension* of data become significant for users.

In the demo paper [4] we presented Sextant[3], a tool that enables the visualization and exploration of the spatial dimension of linked geospatial data. Sextant enables map creation and sharing, as well as the visualization and exploration of data by evaluating GeoSPARQL queries on SPARQL endpoints. In this way rich thematic maps can be created by layering information coming from the evaluation of GeoSPARQL queries. Sextant is based on standards defined by the Open Geospatial Consortium (OGC), thus it is interoperable with other well-known GIS and Web tools such as Google Earth.

In this demo paper we turn our attention to the *temporal dimension* of linked geospatial data and present a new version of Sextant that we now rename *SexTant* (the capital "T" in the new name emphasizes the time dimension). SexTant extends the functionalities of the earlier tool by visualizing the temporal dimension of data having a spatial extent simultaneously on a map and a timeline. The new capabilities of SexTant build on the temporal features of the data model

---

* This work was supported by the EU FP7 project TELEIOS (257662), the Greek NSRF project SWeFS (180), and the EU project Optique (318338).

[3] See `http://wikipedia.org/wiki/Sextant` for the explanation of the name.

stRDF, the query language stSPARQL, and their efficient implementation in the geospatial RDF store Strabon [1]. stRDF and stSPARQL go beyond the OGC standard GeoSPARQL by allowing the representation and querying of linked geospatial data *that changes over time* [1,3]. SexTant (and this demo paper) extends the research presented in [1] by demonstrating how graphs defined in stRDF/stSPARQL can be explored and visualized.

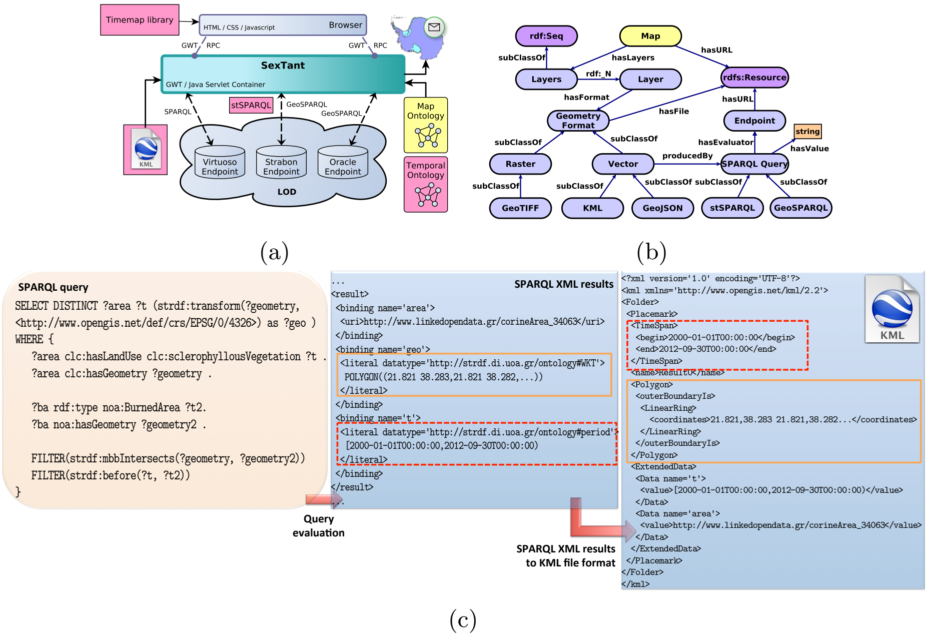
In related work, the French project GEOPEUPLE is also studying the modeling and visualization of spatiotemporal data. As an example, the demo available at `http://www.rotefabrik.free.fr/geopeuple/en/onglets-33038.html` visualizes the evolution of administrative regions in France over the time.

2 New Functionalities of SexTant

SexTant extends the architecture of our earlier system presented in [4] and shown in Fig. 1a as follows (new and modified components are highlighted with pink boxes). First, apart from the the map ontology used by the earlier system and shown in Fig. 1b, SexTant employs the temporal ontology dictated by stRDF and stSPARQL for the modeling of valid time [1]. This ontology enables the introduction of *user-defined time* and *valid time of a triple* in stRDF data. Times are modelled as *instants* or *intervals* and are represented using values of the datatypes `xsd:dateTime` and `strf:period` respectively. Second, one can now use all the temporal features of stSPARQL to query linked spatiotemporal data encoded in stRDF. In this way the full capabilities of endpoints using the spatiotemporal RDF store Strabon can be exploited. Third, the module that translates the results of stSPARQL queries from XML to KML format has been extended so that the temporal primitives of stRDF that we mentioned above are translated into the respective temporal primitives of the KML standard. An example of this transformation is provided in Fig. 1c. Last, SexTant builds on the Timemap Javascript library (`https://code.google.com/p/timemap/`) for visualizing "temporally-enriched" KML files. This enables the visualization of geospatial features with associated temporal information on a map and a timeline simultaneously. Timemap has been transparently integrated in the implementation of the earlier system which is based on the Google Web Toolkit framework.

3 Demonstration Overview

The demonstration of the spatio-temporal features of SexTant will be based on a real scenario in which an Earth Observation (EO) scientist studies the changes in the land cover of an area and assesses the damage caused by fires. This scenario is very common in the EO domain, where data is constantly produced by satellite sensors and is associated with metadata containing, among others, temporal attributes, such as the time that an image was acquired. Satellite acquisitions are utilized in related applications such as the CORINE Land Cover



SPARQL query

```
SELECT DISTINCT ?area ?t (strdf:transform(?geometry,
<http://www.opengis.net/def/crs/EPSG/0/4326>) as ?geo )
WHERE {
   ?area clc:hasLandUse clc:sclerophyllousVegetation ?t .
   ?area clc:hasGeometry ?geometry .

   ?ba rdf:type noa:BurnedArea ?t2.
   ?ba noa:hasGeometry ?geometry2 .

   FILTER(strdf:mbbIntersects(?geometry, ?geometry2))
   FILTER(strdf:before(?t, ?t2))
}
```

SPARQL XML results

```
...
<result>
 <binding name='area'>
  <uri>http://www.linkedopendata.gr/corineArea_34063</uri>
 </binding>
 <binding name='geo'>
  <literal datatype='http://strdf.di.uoa.gr/ontology#WKT'>
   POLYGON((21.821 38.283,21.821 38.282,...))
  </literal>
 </binding>
 <binding name='t'>
  <literal datatype='http://strdf.di.uoa.gr/ontology#period'>
   [2000-01-01T00:00:00,2012-09-30T00:00:00)
  </literal>
 </binding>
</result>
...
```

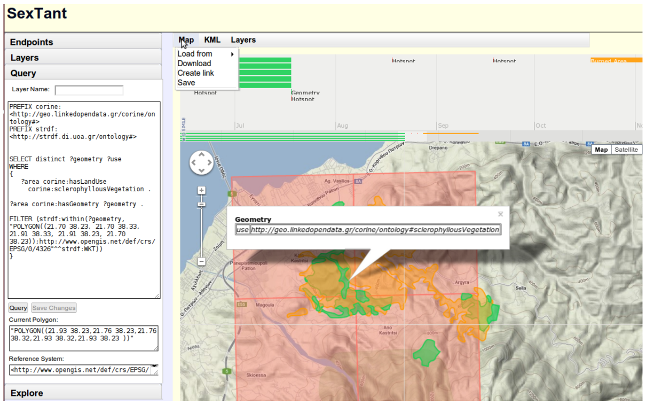
```
<?xml version='1.0' encoding='UTF-8'?>
<kml xmlns='http://www.opengis.net/kml/2.2'>
<Folder>
 <Placemark>
  <TimeSpan>
   <begin>2000-01-01T00:00:00</begin>
   <end>2012-09-30T00:00:00</end>
  </TimeSpan>
  <name>Result0</name>
  <Polygon>
   <outerBoundaryIs>
    <LinearRing>
      <coordinates>21.821,38.283 21.821,38.282...</coordinates>
    </LinearRing>
   </outerBoundaryIs>
  </Polygon>
  <ExtendedData>
   <Data name='t'>
    <value>[2000-01-01T00:00:00,2012-09-30T00:00:00)</value>
   </Data>
   <Data name='area'>
    <value>http://www.linkedopendata.gr/corineArea_34063</value>
   </Data>
  </ExtendedData>
 </Placemark>
</Folder>
</kml>
```

Fig. 1: (a) SexTant overview and (b) Map ontology (c) Translation of SPARQL XML results to KML

programme operated by the European Environment Agency that makes available as a cartographic product the land cover of European areas over time.

To achieve the goal of our scenario, we will combine information derived from the following datasets that were produced within the project TELEIOS (`http://www.earthobservatory.eu/`): the CORINE Land Cover dataset of year 2000, a Fire Hotspots dataset that provides information about fire hotspots in Greece, and a Burned Areas dataset that provides detailed information about areas of Greece that have been affected by fires during a recent fire season. The EO scientist will use SexTant to visualize the results of stSPARQL queries that use several thematic, spatial and temporal criteria so that she will be able to derive *implicit links* among the involved datasets due to their spatial and temporal correlation.

In our scenario, first we will visualize on a map the areas that have been classified as “sclerophyllous vegetation” according to the CORINE Land Cover dataset of year 2000. The valid time of the triples that encode information about these areas will be projected on the timeline. Next, a new layer that visualizes the hotspots that have been identified during the fire season will be displayed on the map while the timeline will display the time when the hotspots were detected. Then, a new layer that depicts the areas that were burned during the forest fires of 2012 will be overlayed on the map and the timeline. The resulting map will display to the EO scientist the schlerophyllous forests that got burnt by



```
PREFIX corine:
<http://geo.linkedopendata.gr/corine/on
tology#>
PREFIX strdf:
<http://strdf.di.uoa.gr/ontology#>


SELECT distinct ?geometry ?use
WHERE
{
   ?area corine:hasLandUse
     corine:sclerophyllousVegetation .

?area corine:hasGeometry ?geometry .

FILTER (strdf:within(?geometry,
"POLYGON((21.70 38.23, 21.70 38.33,
21.91 38.33, 21.91 38.23, 21.70
38.23));http://www.opengis.net/def/crs/
EPSG/0/4326"^^strdf:WKT))
}
```

Current Polygon: "POLYGON((21.93 38.23,21.76 38.23,21.76 38.32,21.93 38.32,21.93 38.23))"

Reference System: <http://www.opengis.net/def/crs/EPSG/

use: http://geo.linkedopendata.gr/corine/ontology#sclerophyllousVegetation

Fig. 2: A screenshot from SexTant depicting the evolution of the land cover

the forest fires of 2012 along with a preview of the evolution of the forest fires as they were detected by satellites so that she can assess the severity of the damage caused by fires. A similar procedure will be used in order to discover implicit links among the datasets enriched with provenance information, e.g., discover the cause of changes in the land cover of areas as represented in CORINE through the visualization and overlay of the other datasets. Such implicit links can later on be asserted to enrich all datasets.

Layers that contain solely geospatial information will be retrieved by evaluating a GeoSPARQL query on a Strabon, Oracle, Parliament, or Virtuoso endpoint, while layers that contain spatial and temporal information will be retrieved by evaluating an stSPARQL query on Strabon. The reason for this choice is that stSPARQL is the only language that provides the spatial and temporal primitives that are needed for this scenario, while Strabon is currently the only "temporally-enabled" geospatial RDF store as we have discussed in [1]. In this respect our demo will also serve to showcase the new functionalities of the system Strabon as presented in [1]. A video demonstration of SexTant that follows the scenario described in this section is available at `http://strabon.di.uoa.gr/sexTant/sexTant-demo-full-version.ogv`.

References

1. Bereta, K., Smeros, P., Koubarakis, M.: Representation and querying of valid time of triples in linked geospatial data. In: ESWC. LNCS, vol. 7882, pp. 259–274 (2013)
2. Koubarakis, M., Karpathiotakis, M., Kyzirakos, K., Nikolaou, C., Sioutis, M.: Data Models and Query Languages for Linked Geospatial Data. LNCS, vol. 7487, pp. 290–328. Springer (2012)
3. Kyzirakos, K., Karpathiotakis, M., Koubarakis, M.: Strabon: A Semantic Geospatial DBMS. In: ISWC. LNCS, vol. 7649, pp. 295–311. Springer (2012)
4. Nikolaou, C., Dogani, K., Kyzirakos, K., Koubarakis, M.: Sextant: Browsing and Mapping the Ocean of Linked Geospatial Data. ESWC 2013. Demo paper

Part II: Posters

Query Suggestion by Concept Instantiation

Jack Wei Sun[1], Franky[1], Kenny Q. Zhu[1], and Haixun Wang[2]

[1] ADAPT-Lab, Shanghai Jiao Tong University
[2] Google Inc.

Abstract. A class of search queries which contain abstract concepts are studied in this paper. These queries cannot be correctly interpreted by traditional keyword-based search engines. This paper presents a simple framework that detects and instantiates the abstract concepts by their concrete entities or meanings to produce alternate queries that yield better search results. [3]

Keywords: Query Suggestion, Concept-based Queries, Search Logs

1 Introduction

The quality of search results largely depend on the specificity of the keywords they put in search engine, for modern search engine are mostly keyword-based. When user queries do not general good results, search engine may suggest a list of alternate queries for the user to select and re-submit.

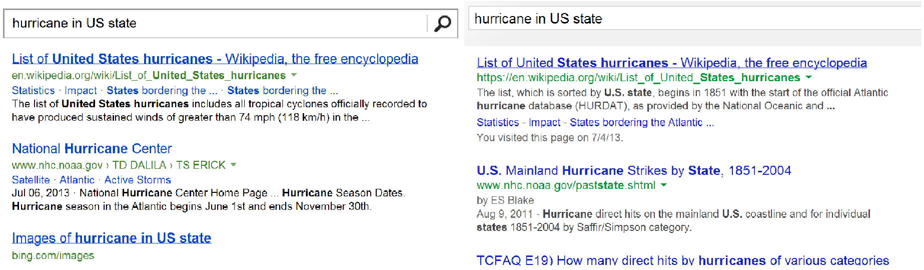
Traditionally, the function of query suggestion is implemented by keyword-based methods and partially depends on the information from search log [1, 4, 5, 10], which includes click-through data, session data, etc. With the rapid development of Internet, researchers are increasingly interested in semantic view of the web and do a lot of work on semantic relation extraction from search log [2] and concept search [6, 8]. There are also other attempts on sematic query suggestion without search log [3].

This paper focuses on a class of difficult queries which include abstract concepts, e.g., "*hurricane in US state*". The likely intention of this query is to look for a specific instance of a hurricane that happened to a US state, e.g., Katrina in Louisiana. Here both *hurricane* and *state* are used as abstract concepts which contain many specific entities. The reason for such an abstract query may be because the user has forgotten the name of the hurricane or the state.

Search engines today return pages *about* these abstract concepts, and that usually means a list of huricanes in the US history for the above example (see Figure 1). It takes the user at least a few more clicks and scanning through the pages to find the information needed. Our objectives is to return a list of suggested queries such as: *Katrina in Lousiana*, *Sandy in Connecticut*, *Dolly in Texas.*

---

[3] Kenny Q. Zhu (corresponding author) is partially supported by NSFC grants 61100050 and 61033002.



hurricane in US state

List of United States hurricanes - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/List_of_United_States_hurricanes
Statistics · Impact · States bordering the ... · States bordering the ...
The list of United States hurricanes includes all tropical cyclones officially recorded to have produced sustained winds of greater than 74 mph (118 km/h) in the ...

National Hurricane Center
www.nhc.noaa.gov › TD DALILA › TS ERICK
Satellite · Atlantic · Active Storms
Jul 06, 2013 · National Hurricane Center Home Page ... Hurricane Season Dates. Hurricane season in the Atlantic begins June 1st and ends November 30th.

Images of hurricane in US state
bing.com/images

hurricane in US state

List of United States hurricanes - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/List_of_United_States_hurricanes
The list, which is sorted by U.S. state, begins in 1851 with the start of the official Atlantic hurricane database (HURDAT), as provided by the National Oceanic and ...
Statistics - Impact - States bordering the Atlantic ...
You visited this page on 7/4/13.

U.S. Mainland Hurricane Strikes by State, 1851-2004
www.nhc.noaa.gov/paststate.shtml
by ES Blake
Aug 9, 2011 - Hurricane direct hits on the mainland U.S. coastline and for individual states 1851-2004 by Saffir/Simpson category.

TCFAQ E19) How many direct hits by hurricanes of various categories

Fig. 1. Search Result of Concept-based Query in Bing and Google

To this end, our main approach utilizes a comprehensive, probabilistic taxonomy and a search log with access statistics (click-through rate, etc.). The probabilistic taxonomy, called Probase[9], provides large number of concepts and their instances in is-a relations (e.g. katrina *isa* hurricane) as well as the *typicality* of an instance belonging to a concept. For example, a robin is a *typical* bird, but an ostrich is not. Given a user query, we use this taxonomy to detect high level concepts in it and translate them into likely instances to produce a new query, and then return the queries from the query log which are closest to the newly transformed query. Next, we present the prototype system in some detail and some preliminary experimental results.

2 The Prototype System

Our system is divided into two parts, an offline part which creates an index on all historical search queries from the search log and an online runtime system which retrieves a number of relevant queries to an input concept-based query and ranks them according to a scoring function. The architecture of the system is shown in Figure 2. Next, we briefly discuss the two key components in the architecture.

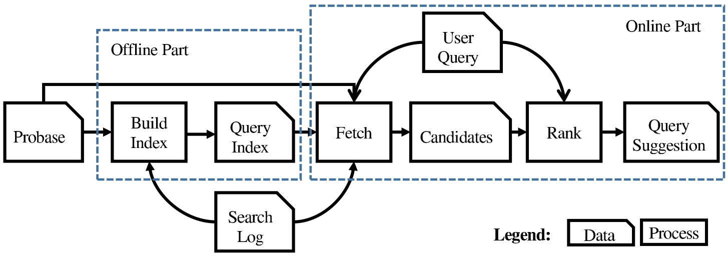


Fig. 2. Architecture of System

2.1 Build Index

First, we parse each historical query in the search log and identify all noun-phrase terms which exist as entities in Probase. Entities are those terms which appear as the instance in at least one is-a pair in the taxonomy, e.g., *katrina*, *louisina*, etc. Then, we conceptualize these instances into their most likely concepts by considering the neighboring instances in the same query, using the technique by Song et al.[7, 8]. For example, query "*katrina* victims from *texas*" maybe conceptualized into "[hurricane] victims from [state]". Finally, we build an index of the search log using the concepts as keys.

2.2 Rank Candidates

Given a query, our system first parses the query and recognizes the concepts in it and then fetch a number of historical queries which are indexed by these concepts as candidate suggestions. Ranking the candidate is the main challenge in this work. We apply a hybrid method to calculate the ranking score. This score takes three factors into consideration, i.e. semantic similarity, context similarity and quality of suggestion query.

The context similarity, $CScore$, is measured by the *edit distance* between the input query and the candidate in terms of the number of insertion, deletion or replacement of words. Matched instances in the candidate and the matched concepts in the input queries which represent semantic information are removed before calculating the edit distance.

The semantic similarity between a candidate query and an input query is the combined distance between the instances in the candidate and the corresponding concept in the input. The distance between an instance and a concept is measured by the *typicality* score between these two terms in Probase. Because *typicality* score is a value between 0 and 1 and can be dense in some range (around 0.01 in practice), we take logistic function on the typical value to expand the range into something more distinguishable. We also use a scalar to make it linear comparable with $CScore$ according to statistics. That is,

$$SScore(t) = \beta \times (-\frac{1}{1 + e^{-\alpha \times t}} + 1)$$

where t is the *typicality* value and α is the factor to locate the dense range and β is a scalar factor. Both factors are learned from some training examples.

We also utilize the click-through rate from the search log, in order to measure the quality or effectiveness of candidate suggestion. In this paper, we use a simple measure defined as

$$QScore(q) = \frac{\text{Num\_clicks(q)}}{\text{Frequency(q)}}$$

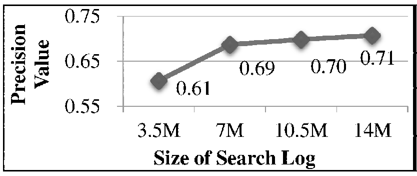
where q is a candidate suggestion query. Besides the click-through rate, the quality score can be extended to include other information from search log.

Currently, the overall score (the lower, the better) is defined as:

$$OverallScore = [CScore + SScore] + e^{-QScore}$$

3 Preliminary Result

We take 1/10 of a 6-month worth of Bing search log as our experimental data source. To evaluate the system, we arbitrarily select 10 concept-based queries related to the events happening during the 6-month time span. All suggested queries from the system are given to 3 human judges who would grade the suggestion on the scale of 1-5, 1 being the least helpful and 5 being the most helpful. The averaged normalized precision score of of these results with different size of search log and the suggestions of two example queries are shown in Figure 3. The results show that having more search log is helpful but the effect saturates at some point. Complete data set as well as evaluation results can be found at `http://adapt.seiee.sjtu.edu.cn/~jack/query/`.



extreme sports in american cities	**diseases found in domesticated animals**
skiing in chicago	tetanus in dog
snowboarding in washington	cirrhosis in horses
sky diving in seattle	herpes in cattle

Fig. 3. Average Precision and Examples of the System

References

1. Baeza-Yates, R., Hurtado, C., Mendoza, M.: Query recommendation using query logs in search engines. In: EDBT 2004 Workshops. pp. 588–596. Springer (2005)
2. Baeza-Yates, R., Tiberi, A.: Extracting semantic relations from query logs. In: Proceedings of ACM SIGKDD. pp. 76–85. ACM (2007)
3. Bhatia, S., Majumdar, D., Mitra, P.: Query suggestions in the absence of query logs. In: SIGIR. pp. 795–804 (2011)
4. Cao, H., Jiang, D., Pei, J., He, Q., Liao, Z., Chen, E., Li, H.: Context-aware query suggestion by mining click-through and session data. In: Proceedings of the 14th ACM SIGKDD. pp. 875–883. ACM (2008)
5. Dupret, G., Mendoza, M.: Recommending better queries from click-through data. In: String Processing and Information Retrieval. pp. 41–44. Springer (2005)
6. Giunchiglia, F., Kharkevich, U., Zaihrayeu, I.: Concept search: Semantics enabled syntactic search. Semantic Search p. 109 (2008)
7. Song, Y., Wang, H., Wang, Z., Li, H., Chen, W.: Short text conceptualization using a probabilistic knowledgebase. In: IJCAI (2011)
8. Wang, Y., Li, H., Wang, H., Zhu, K.Q.: Concept-based web search. In: Conceptual Modeling, pp. 449–462. Springer (2012)
9. Wu, W., Li, H., Wang, H., Zhu, K.Q.: Probase: A probabilistic taxonomy for text understanding. In: Proceedings of ACM SIGMOD. pp. 481–492. ACM (2012)
10. Zhang, Z., Nasraoui, O.: Mining search engine query logs for query recommendation. In: Proceedings of the 15th WWW. pp. 1039–1040. ACM (2006)

A Protein Annotation Framework Empowered with Semantic Reasoning

Jemma X. Wu, Edmond J. Breen, Xiaomin Song,
Brett Cooke, and Mark P. Molloy

Australian Proteome Analysis Facility, Maquarie University, Sydney, Australia
`{jwu,ebreen,xsong,bcooke,mmolly}@proteome.org.au`

Abstract. This paper presents an association discovery framework for proteins based on semantic annotations from biomedical literatures. An automatic ontology-based annotation method is used to create a semantic protein annotation knowledge base. A semantic reasoning service enables realisation reasoning on original annotations to infer more accurate associations. A case study on protein-disease association discovery on a real-world colorectal cancer dataset is presented.

Keywords: Protein annotation, bioinformatics, semantic reasoning

1 Introduction

To bridge the gap between the biomedical science and bioinformatics, many biomedical ontologies have been created in the past few years. Ontology-based semantic annotation for biomedical entities are of interest to both biomedical researchers and general public. Meanwhile, the biomedical domain has a large and fast-growing amount of literature resources, among which MedLine[1] is the primary publication repository for biomedical research. Ontology-based biomedical text annotation has shown promising progress and several tools have been successfully developed and evaluated in biomedical text mining problems[2, 5, 4]. However, these generic text-based biomedical annotation tools only provide concept level annotations. The ability to do protein-oriented semantic annotation will greatly benefit the proteomics research by enabling easy protein association discovery. Also, traditional text-based annotation tools tend to create excessive annotations and some tools expand the raw annotations by using semantic reasoning[3]. Inferring of the most informative and accurate annotations will be very valuable to efficient and accurate association discovery.

This paper proposes an integrated high performance framework that leveraging protein annotations and semantic reasoning to an informative protein-biomedical concepts association Knowledge Base(KB). Starting from a list of proteins, the system automatically retrieves a pool of MedLine citations and annotates the proteins using pre-defined biomedical ontologies. A realisation reasoning service is applied to infer more accurate protein association information. In our preliminary study, the focus is on the discovery of potential *protein-disease associations*. A case study on discovering protein-disease associations for a real-world colorectal cancer tissue protein dataset is presented.

---

[1] http://www.ncbi.nlm.nih.gov/pubmed

2 SPRAM: A Semantic PRotein Annotation framework based on MedLine

We propose a *Semantic PRotein Annotation method based on MedLine* (SPRAM) which produces semantically inferred protein annotations based on biomedical literatures and ontologies (Fig.1). *SPRAM* starts with a list of proteins of interest. A list of mapped MedLine publication IDs (PMIDs) for each candidate protein is produced by searching the UniProt[2] and Entrez Gene[3] databases. A parallel process runs to execute PubMed article fetching, semantic annotation and realisation reasoning based on biomedical ontologies, such as Human Disease Ontology (HDO[4]), for the pool of candidate PMIDs. The MedLine citations retrieved by the PubMed's Eutil service are further filtered by the co-occurrence of protein names. We choose the BioPortal's annotator service[5] with no semantic expanding as our annotator. The raw annotated concepts are post-processed by a realisation reasoning service to generate a set of clean and accurate protein annotations and inserted into the knowledge base. Biologists can then issue semantic queries to retrieve all proteins which are associated with one or more concepts in the ontology.

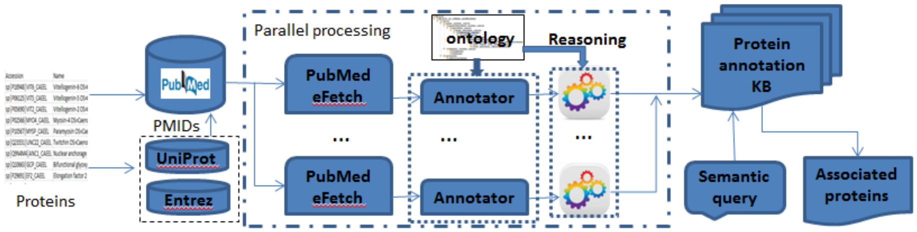


Fig. 1. The framework of Semantic Protein Annotation method based on MedLine.

3 Semantic reasoning for protein annotations

In biomedical ontology annotations, very often an instance is annotated with multiple classes with subclass relationships in the ontology. To the best of our knowledge, existing biomedical annotation tools with semantic reasoning functionalities only do semantic expanding[3]. There has been no prior work on drilling down the annotations to most specific concepts by using the semantic reasoning. Despite, in many cases, the most specific classes of a protein, can more accurately represent their biomedical categorical information. For example, the traditional protein Gene Ontology analysis that shows the distribution of biological process or molecular functions nearly always bias towards the top-level classes in the ontology[1].

---

[2] http://www.uniprot.org/

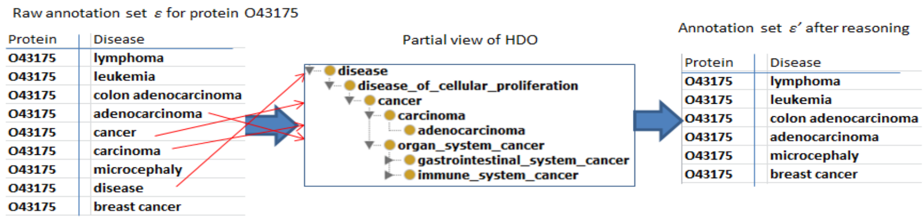
[3] http://www.ncbi.nlm.nih.gov/gene

[4] http://disease-ontology.org/

[5] http://www.bioontology.org/wiki/index.php/Annotator_Web_service

We developed a specialised realisation reasoning service for dynamically generated annotations. Different to the traditional Description Logic most specific concept reasoning, our algorithm works on a dynamic set of annotations on the fly instead of assertions in a static KB. Only the most specific annotations will be stored in the KB. The algorithm takes a set of semantic protein annotations, ϵ, and an ontology, $\mathcal{O}$, that ϵ is based on. A most specific class set, $\epsilon^{'}$, is initialised to be an empty set, $\emptyset$. For each class $t \in \epsilon$ for each protein, find all subclasses of t in $\mathcal{O}$, i.e., $\{C_i\}$ where $C_i \sqsubseteq t \in \mathcal{O}$. Class t is added to $\epsilon^{'}$ if $\{C_i\} \cap \epsilon = \emptyset$, i.e., t is the most specific annotation in ϵ given ontology $\mathcal{O}$. The algorithm outputs the most specific class set $\epsilon^{'}$ which will be inserted into the nascent KB.

Fig.2 shows an example of the effect of applying realisation reasoning on the disease annotations for a protein with a UniProt ID "O43175". Class "disease", "cancer" and "carcinoma" in the original annotation set are all realised to "adenocarinoma" because the last concept is subsumed by those three concepts and it is also in the original annotation set. This is important because it more accurately represents the biomedical categorical information and reduces complexity.



Raw annotation set ε for protein O43175

Protein	Disease
O43175	lymphoma
O43175	leukemia
O43175	colon adenocarcinoma
O43175	adenocarcinoma
O43175	cancer
O43175	carcinoma
O43175	microcephaly
O43175	disease
O43175	breast cancer

Annotation set ε' after reasoning

Protein	Disease
O43175	lymphoma
O43175	leukemia
O43175	colon adenocarcinoma
O43175	adenocarcinoma
O43175	microcephaly
O43175	breast cancer

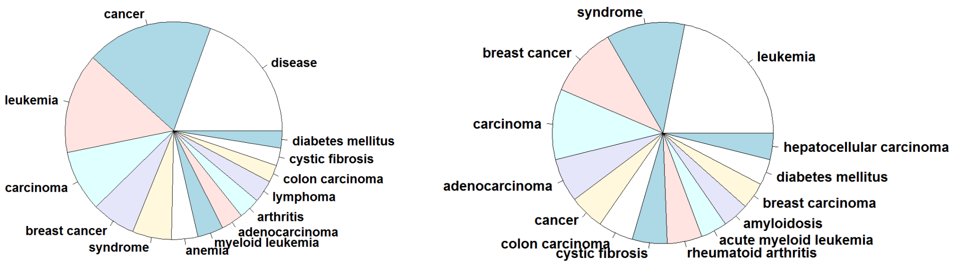
Fig. 2. An example showing the change to the protein disease annotations after realisation reasoning. Left: the original disease annotation. Middle: a simplified partial view of Human Disease Ontology. Right: the disease annotation after realisation reasoning.

4 Case study: discovery of proteins-disease associations for colorectal cancer tissues

Jankova et al. found 45 up-regulated proteins in colorectal cancer tissues by using experimental protein iTRAQ analysis[1]. The biologists would like to know what diseases are related to these proteins and if the associations to the colorectal cancer have been discovered before.

To help biologists achieve these goals, we take these 45 proteins and use our *SPRAM* workflow to assist discovering the potential diseases associated with these proteins. The result was: 1080 MedLine citations, 354 diseases associations based on HDO, that was reduced to 241 unique associations after realisation reasoning. *SPRAM* returns a set of protein-disease association to the biologists. That includes also the source reference titles and URLs. The biologist can then use this result to help validate these associations easily by tracing back to the references.

Fig.3 shows the changes of the distribution of the diseases associated with these 45 proteins before and after realisation reasoning. The distribution after realisation reasoning represents more accurate and sensible information. For example, the top distributed concept, disease, was removed and the next, cancer, was greatly reduced, thereby producing a clearer set of associations.



(a) DOA distribution before reasoning

(b) DOA distribution after reasoning

Fig. 3. The comparison of the Disease Ontology Annotation (DOA) distributions before and after realisation reasoning for the 45 up-regulated colorectal cancer proteins

To find proteins reported as colorectal cancer related, the biologist issues a query using the concept, "colorectal cancer". The semantic reasoning service rewrites this query into a union of this concept and all of its subclasses. The result shows that 6 proteins (CEA, NNE, HSP 84, NPM, 3-PGDH and UEV-1) reported in the literature as being related to colorectal cancer.

5 Conclusion

This paper proposes an automatic protein-oriented association discovery framework based on semantic annotations from literature. A semantic reasoning service provides realisation reasoning. We demonstrate the usage of our system on protein-disease association discovery using a real-world colorectal cancer protein dataset. In upcoming work, focus will be given to a ranking model of protein associations and customisable selection of protein-PMID mappings.

References

1. L. Jankova, C. Chan, C. Fung, X. Song, S. Kwun, M. Cowley, W. Kaplan, O. Dent, E. Bokey, P. Chapuis, M. Baker, G. Robertson, S. Clarke, and M.P. Molloy. Proteomic comparison of colorectal tumours and non-neoplastic mucosa from paired patient samples using iTRAQ mass spectrometry. *Mol Biosyst*, 7(11), 2011.
2. R. Jelier, M. Schuemie, A. Veldhoven, L. Dorssers, G. Jenster, and J. Kors. Anni 2.0: a multipurpose text-mining tool for the life sciences. *Gen biology*, 9(6), 2008.
3. Clement Jonquet, Nigam H. Shah, and Mark A. Musen. The open biomedical annotator. In *AMIA-TBI'09*, pages 56–60, 2009.
4. H. Lpez-Fernndez, M. Reboiro-Jato, D. Glez-Pea, F. Aparicio, D. Gachet, M. Buenaga, and F. Fdez-Riverola. Bioannote: A software platform for annotating biomedical documents with application in medical learning environments. *Computer Methods and Programs in Biomedicine*, 111(1):139 – 147, 2013.
5. Mariana Neves and Ulf Leser. A survey on annotation tools for the biomedical literature. *Brief Bioinform*, 18, December 2012.

Denoting Data in the Grounded Annotation Framework

Marieke van Erp[1], Antske Fokkens[1], Piek Vossen[1], Sara Tonelli[2], Willem Robert van Hage[3], Luciano Serafini[2], Rachele Sprugnoli[2], and Jesper Hoeksema[1]

[1] VU University Amsterdam
{`marieke.van.erp,antske.fokkens,piek.vossen,j.e.hoeksema`}`@vu.nl`
[2] Fondazione Bruno Kessler {`satonelli,serafini,sprugnoli`}`@fbk.eu`
[3] SynerScope B.V. `willem.van.hage@synerscope.com`

Abstract. Semantic web applications are integrating data from more and more different types of sources about events. However, most data annotation frameworks do not translate well to semantic web. We describe the grounded annotation framework (GAF), a two-layered framework that aims to build a bridge between *mentions* of events in a data source such as a text document and their formal representation as *instances*. By choosing a two-layered approach, neither the mention layer, nor the semantic layer needs to compromise on what can be represented. We demonstrate the strengths of GAF in flexibility and reasoning through a use case on earthquakes in Southeast Asia.

1 Introduction

Semantic web applications are ingesting data from more and more different sources such as output from natural language processing applications, sensor data, videos or financial transactions. Each of these domains has their own data annotation practices which first need to be reconciled with semantic web standards. One issue with integrating information from different sources is that representation formats tend to look at their domain in isolation, making it difficult to integrate information that comes from other domains.

The Grounded Annotation Framework (GAF) [1] aims at addressing this problem by distinguishing instance *mentions* which can be domain specific from *instances* conform to domain independent semantic web standards. In this manner, we can integrate information for example extracted by NLP tools or from sensor data in a formal context which can be shared by different applications and over which we can perform reasoning. This paper addresses the advantages of using GAF from the point of view of users of Linked Data.

We will describe GAF in Section 2, present an example in Section 3 and conclude with pointers for future work in Section 4.

2 The Grounded Annotation Framework

The main property of GAF is that it distinguishes *instances* from *instance mentions*. A *mention* is the act of referring to an object where an *instance* is the object itself. The relation between instances and mentions is defined by *gaf:denotedBy*, which is the only new predicate GAF introduces. Different resources (or even the same resource) may refer to an instance in different ways and each of these references may have properties of its own. This is quite common in natural language, where authors tend to alternate terms to refer to the same object for stylistic reasons, but it can also play a role in other sources of information. If, for instance, a sensor displays a measured temperature, this displayed value has properties of its own that are clearly not properties of the value that was measured, such as the instrument that was used to measure it and its error rate. In the remainder of this contribution, we will illustrate GAF through the example of presenting instances in the Simple Event Model (SEM) [2] and mentions in the TERENCE Annotation Format (TAF) [3] which represents linguistic properties.

SEM is a model to express *who* did *what*, *where*, and *when*. It is not the only RDF model to describe events but as SEM is not tied to a any domain and is among the most flexible, we chose this model as the core of our semantic layer. It should be noted however that, in principle any RDF schema can be integrated into GAF. TAF is designed to annotate coreference relations between event mentions as well as participants, locations and temporal expressions, which covers the kind of information also represented in SEM. TAF has the additional advantage that it already distinguishes between *instances* and *instance mentions* for participants and locations. We use a slightly adapted variant of TAF that extends this distinction to events and temporal expressions as described in [1]. We chose TAF as it is based on the ISO-TimeML standard and fits our event use-case, however, any representation format can be used in GAF.

The *gaf:denotedBy* relations is used to link events represented in SEM to specific mentions represented in TAF. If a linguistic analysis identifies a syntactic relation between an event mention and the mention of a person, we can derive that this person is an Actor of the event in SEM according to the analysis of a specific text. Mentions thus play an important role in modelling **provenance** of information. To model provenance we use the PROV-O ontology [4] as it is compatible with our RDF representation and is recommended by W3C for provenance modelling. When we represent alternative views in SEM, these views are linked to the mentions they were derived from. This leads us to the original source and hence information in who expressed which view.

Creating GAF Annotations

GAF annotations can be created both by starting from the linguistic layer and the semantic layer. When starting from text for the mention layer, first TAF annotations are added to the text using the Celct Annotation Tool [5], which are then translated to SEM relations using a conversion script. Instances extracted from a particular source (for example a document) are grouped into named graphs, to which provenance information is added. We use manually defined rules for mapping TAF to SEM, but plan to use machine learning in the future.

When starting from the semantic layer, events and event properties are linked to textual mentions. We are currently working towards an annotation environment based on CROMER [6], which will allow the user to switch easily between the linguistic and semantic layers.

3 Examples

The example sentences shown in Figure 1 both contain information about the 2004 Indian Ocean Earthquake and Tsunami. The articles disagree on the cause of the earthquake; where Bloomberg ascribes it to moving tectonic plates, Veteran's Today sees a stealth attack submarine as the likely cause. Figure 2 shows that these two declarations can co-exist within the GAF representation of the earthquake. It is up to the application or user accessing the information to interpret the fact that there is a contradiction and for example select only particular sources for further processing. GAF provides the glue to connect non-semantic web data to semantic web representation formats. The *rdfs:isDefinedBy* relation at the top of Figure 2 shows how RDF predicates can be used to link GAF representations to external resources such as the Linked Open Data cloud.[1]

"Indonesia lies in a zone where the Indo-Australian, Eurasian, Philippine and Pacific plates meet and occasionally shift, causing earthquakes and sometimes generating tsunamis. There have been hundreds of earthquakes in Indonesia since a 9.1 temblor in 2004 caused a tsunami that swept across the Indian Ocean, devastating coastal communities and leaving more than 220,000 people dead in Indonesia, Sri Lanka, India, Thailand and other countries."
(Bloomberg, 2009-01-07 01:55 EST)

"...were most concerned about the cause, scope, and consequences of the December 26, 2004 Indian Ocean tsunamis because they were far bigger and more destructive than they had anticipated. More important, it had no clear alibi that their most likely source of the disaster, the Multi-Mission Platform of the new stealth attack submarine, the USS Jimmy Carter, had not been the culprit."
(Veteran's Today, 2011-10-02)

Fig. 1. Sample sentences mentioning the December 2004 Indonesian earthquake

4 Conclusions and Future Work

We have presented GAF, a grounded annotation framework for integrating information from various sources. We have shown its flexibility in representing contradicting information from different textual sources.

We are currently developing an annotation tool that allows users to easily switch between linguistic and semantic annotation layers. After which we plan to develop tools supporting easy integration of other types of information, such as data from the Linked Open Data cloud, video metadata or sensor data.

---

[1] http://groundedannotationframework.org/ provides full examples and the GAF definition.

Acknowledgements
This research is supported by the European Unions 7th Framework Programme via the NewsReader Project (ICT-316404) and by the BiographyNet project, funded by the Netherlands eScience Center (http://esciencecenter.nl/).

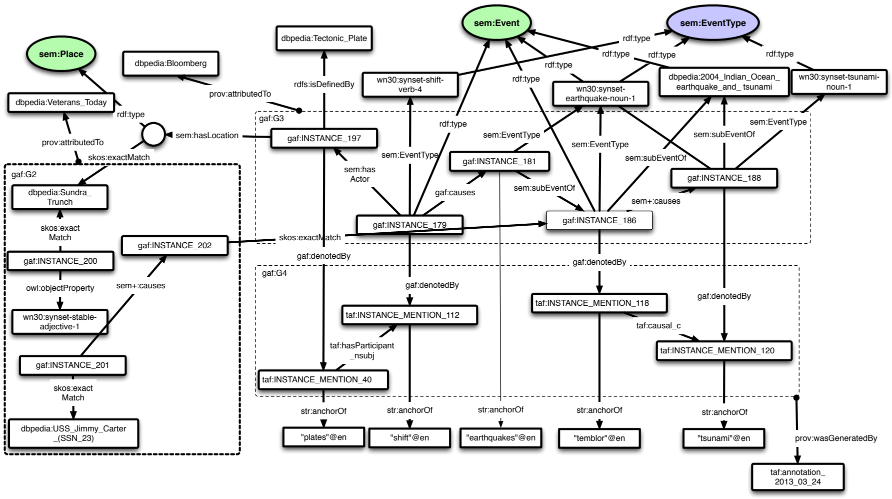


Fig. 2. GAF representation of Earthquake example

References

1. Fokkens, A., van Erp, M., Vossen, P., Tonelli, S., van Hage, W.R., Serafini, L., Sprugnoli, R., Hoeksema, J.: GAF: A grounded annotation framework for events. In: Proceedings of the first Workshop on Events: Definition, Dectection, Coreference and Representation, Atlanta, USA (2013)
2. Van Hage, W.R., Malaisé, V., Segers, R., Hollink, L., Schreiber, G.: Design and use of the simple event model (SEM). Journal of Web Semantics (2011)
3. Moens, M.F., Kolomiyets, O., Pianta, E., Tonelli, S., Bethard, S.: D3.1: State-of-the-art and design of novel annotation languages and technologies: Updated version. Technical report, TERENCE project-ICT FP7 Programme-ICT-2010-25410 (2011)
4. Moreau, L., Missier, P., Belhajjame, K., B'Far, R., Cheney, J., Coppens, S., Cresswell, S., Gil, Y., Groth, P., Klyne, G., Lebo, T., McCusker, J., Miles, S., Myers, J., Sahoo, S., Tilmes, C.: PROV-DM: The PROV Data Model. Technical report, W3C (2012)
5. Bartalesi Lenzi, V., Moretti, G., Sprugnoli, R.: CAT: the CELCT Annotation Tool. In: Proceedings of LREC 2012. (2012)
6. Bentivogli, L., Girardi, C., Pianta, E.: Creating a Gold Standard for Person Cross-Document Coreference Resolution in Italian News. In: Workshop on Resources and Evaluation for Identity Matching, Entity Resolution and Entity Management. (2008)

On the Semantics of R2RML and its Relationship with the Direct Mapping

Juan F. Sequeda

Department of Computer Science, University of Texas at Austin
`jsequeda@cs.utexas.edu`

Abstract. The W3C Relational Database to RDF (RDB2RDF) standards are positioned to bridge the gap between Relational Databases and the Semantic Web. The standards consist of two interrelated and complementary specifications: Direct Mapping of Relational Data to RDF and R2RML: RDB to RDF Mapping Language. In this paper we present initial results on the formal study of the R2RML mapping language by defining its semantics using Datalog. We prove that there are a total of 57 distinct Datalog rules which can be used to generate RDF triples from a relational table. Additionally, we provide insights on the relationship between R2RML and Direct Mapping.

1 Introduction

To live up to the promise of web-scale data integration, the Semantic Web will have to include the content of existing relational databases. In September 2012, two interrelated and complementary W3C standards, Direct Mapping [1] and R2RML [2], were standarized. R2RML is a mapping language which allows users to manually define mappings. Direct Mapping is the default and automatic way to translate relational databases into RDF without any input from a user, which can be represented in R2RML. The Direct Mapping has been well studied. The W3C specifications present the denotational and datalog-based semantics of the Direct Mapping [1]. Additionally, the W3C Direct Mapping has been augmented to include a direct mapping from the relational schema to OWL in order to prove fundamental correctness properties [3]. However, to the best of our knowledge, there has not been a thorough study of the R2RML language and its relationship with the Direct Mapping. As a matter of fact, the semantics of R2RML have not even been formally defined.

Our methodology is to study the problem of mapping relational databases to RDF from two different perspectives: logical and physical. Inspired by the relational query processing workflow, we propose the following workflow. Mappings are expressed in a declarative language: *R2RML*. A parser can translate the mapping into a logical expression: the *logical mapping*. A logical mapping can be rewritten into other equivalent logical mappings which may be better for performance. A logical mapping is translated into a executable program, the *physical mapping*. A physical mapping can then be implemented and optimized in different ways. This paper address the first part of the proposed workflow and presents initial results on the formal study of R2RML by define its semantics using Datalog. We prove that there are a total of 57 distinct Datalog rules which can be used to generate RDF triples from a relational table. Additionally, we provide insight on the relationship between R2RML and Direct Mapping. It is our hypothesis that a core subset of R2RML has the same expressive power as the Direct Mapping, if views are allowed as input.

2 R2RML

R2RML is a language for expressing mappings from a relational database to RDF. The input of an R2RML mapping $\mathcal{M}$ is a relational schema $\mathbf{R}$ and an instance I of $\mathbf{R}$. The output is an RDF graph. Consider a schema of a database with of tables `EMP(EMPNO,ENAME,DID)` and `DEPT(DEPTNO, DNAME, LOC)`. Moreover, we have the following constraints about the schema of the university: `EMPNO` is the primary key of `EMP`, `DEPTNO` is the primary key of `DEPT` and `DID` is a foreign key in `EMP` that references attribute `DEPTNO` in `DEPT`. An R2RML mapping is represented as an RDF graph itself and also has an associated RDFS schema[1]. For readibility, the RDF Turtle syntax is the recommended syntax to write R2RML mappings.

Example 1. An R2RML Mapping for the example database.

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns#>.

<#TriplesMap1>
 rr:logicalTable [ rr:tableName "emp" ];
 rr:subjectMap [ rr:template "http://ex.com/employee/{empno}";
                 rr:class ex:Employee; ];
 rr:predicateObjectMap [ rr:predicate ex:name;
                         rr:objectMap [ rr:column "ename" ]; ];
 rr:predicateObjectMap [ rr:predicate ex:department;
                         rr:objectMap [
                          rr:parentTriplesMap <#TriplesMap2>;
                            rr:joinCondition [
                               rr:child "dept";
                               rr:parent "deptno"; ]; ]; ].
<#TriplesMap2>
 rr:logicalTable [ rr:tableName "dept" ];
 rr:subjectMap [ rr:template "http://ex.com/dept/{deptno}";
                 rr:class ex:Department; ];
 rr:predicateObjectMap [ rr:predicate ex:name;
                         rr:objectMap [ rr:column "DNAME" ]; ].
```

The Datalog rules defining the R2RML semantics can be found at `http://www.cs.utexas.edu/~jsequeda/r2rml`. We briefly explain the components of R2RML with the running example. An R2RML mapping consists of a set of Triple Maps. In the example, there are two Triples Map: `<#TriplesMap1>` and `<#TriplesMap2>`. Each TripleMaps consists of exactly one LogicalTable, exactly one SubjectMap and a set (which may be empty) of Predicate-Object Maps. A LogicalTable is either an existing table/view in the database or a SQL query (known also as an R2RML view). In `<#TriplesMap1>`, the Logical Tables is the table name `"emp"`. A SubjectMap generates an RDF term for the subject and optionally an rdf:type statement. A PredicateObjectMap is a pair of PredicateMap and ObjectMap which generates the RDF terms for the predicate and object respectively of a triple, that is associated to the subject generated by the SubjectMap. An ObjectMap can also be a Referencing Object Map which allows using the subjects of another Triples Map as an object. Since both Triples Maps may be based on different logical tables, it may require a join between the logical tables.

---

[1] http://www.w3.org/ns/r2rml

SubjectMap, PredicateMap and ObjectMap are all Term Maps which is a function that generates an RDF term from the database. There are three ways of creating an RDF term, hence three types of Term Maps: 1) Constant-valued Term Map, 2) Column-valued Term Map, and 3) Template-valued Term Map. In `<#TriplesMap1>`, the SubjectMap is a Template-valued Term Map because it generates an IRI from a template and the value of the `"empno"` column. The PredicateMap is a Constant-valued Term Map because it generates a constant IRI: `ex:name`. The ObjectMap is a Column-valued Term Map because it generates a Literal from the value of the `"ename"` column.

There are three ways of generating RDF triples. *Table Triples* are triples describing an instance of a given class if a Subject Map has a `rr:class` associated to it. In this case, the subject is one of the three possible Term Maps, the predicate is `rdf:type` and the object is the given class. There are three possible rules to generate Table Triples. *Local Triples* are triples that are generated exclusively from a single logical table. Given that there are three ways of generating and RDF term for each the subject, predicate and object, there are 27 different possible cases to generating triples, hence 27 different rules. *Reference Triples* are triples that are generated for Referencing Object Maps, through a join conditions to another table Similar to Local Triples, there are also 27 distinct rules to generate triples. Figure 1 depicts al the possible 27 combinations.

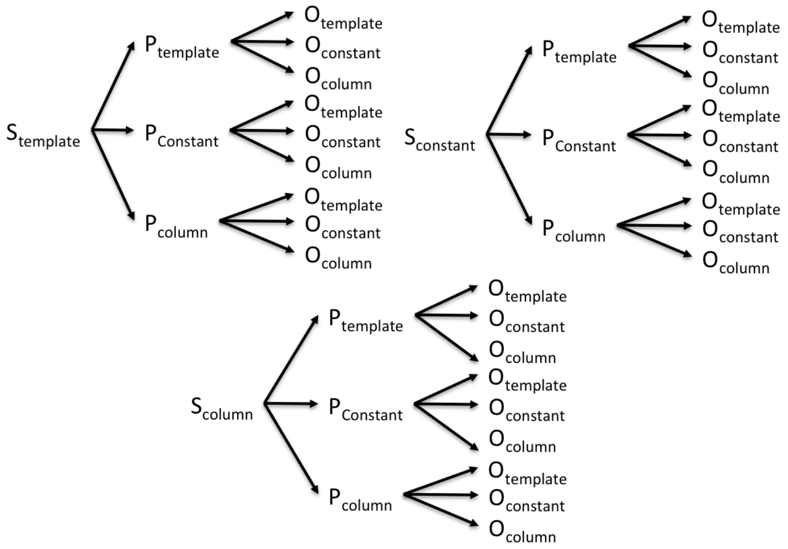


Fig. 1. The tree describes the space of 27 possible way of generating RDF Triples

Given that there are 3 rules to generate Table Triples, 27 rules to generate Local Triples and 27 rules to generate Reference Triples, the following theorem holds.

Theorem 1. *The total number of distinct Datalog rules which can be used to generate RDF triples from a table name is 57.*

3 Relationship with Direct Mapping

We now focus on a simple yet important subset of R2RML which we call $R2RML_{core}$ which consists of only three rules (out of the 57), one rule each for generating Table, Local and References Triples. In these rules, subjects are template-valued term maps, predicates are constant-valued term maps and objects are column-valued term maps. The following is an example of the a Datalog rule to generate Local Triples:

$$\begin{aligned}
\textsc{Triple}(S, P, O) \leftarrow\ & \textsc{SubjectMap}(TM, SID), \textsc{PredicateObjectMap}(TM, POID),\\
& \textsc{SubjectTemplateValueTermMap}(SID, T, S),\\
& \textsc{PredicateConstantValueTermMap}(POID, P),\\
& \textsc{ObjectColumnValueTermMap}(POID, T, O)
\end{aligned}$$

The motivation of $R2RML_{core}$ is two-fold. First, the W3C RDB2RDF Test Cases [5] consist of mainly subject template-valued term maps and predicate constant-valued term maps. Second, the anecdotal experience of the the author, who has implemented Ultrawrap [4], a system that supports R2RML, shows that subjects are usually template-valued term maps and predicates are constant-valued term maps.

Our hypothesis is that $R2RML_{core}$ is as expressive as the Direct Mapping, if views are allowed as input. We first would need to show how any mapping in $R2RML_{core}$ can be expressed in $\mathcal{DM}_{views}$. This means, that views and constraints would need to be created from the R2RML mapping. Additionally, there would need to be a mechanism to defining templates for IRIs. Subsequently, we would need to show how any $\mathcal{DM}_{views}$ mapping can be expressed as $R2RML_{core}$.

4 Conclusions

To the best of our knowledge, this is the first work presenting initial results on the formal study of the R2RML mapping language by defining its semantics using Datalog. and studying the relationship between R2RML and Direct Mapping. This is ongoing work where we are now considering additional features of R2RML such as SQL queries, languages and datatypes. We believe it is important to understand R2RML in order to know how better support users and build tools.

References

1. M. Arenas, A. Bertails, E. Prud'hommeaux, and J. Sequeda. Direct mapping of relational data to RDF. W3C Recomendation 27 September 2012, `http://www.w3.org/TR/rdb-direct-mapping/`.
2. S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF mapping language. W3C Recomendation 27 September 2012, `http://www.w3.org/TR/r2rml/`.
3. J. Sequeda, M. Arenas, and D. P. Miranker. On directly mapping relational databases to rdf and owl. In *WWW*, pages 649–658, 2012.
4. J. Sequeda and D. P. Miranker. Ultrawrap: Sparql execution on relational data. *To appear in Journal of Web Semantics*, 2013.
5. B. Villazon-Terrazas and M. Hausenblas. R2RML and direct mapping test cases. W3C Working Group Note 14 August 2012, `http://www.w3.org/TR/rdb2rdf-test-cases/`.

An FCA Framework for Knowledge Discovery in SPARQL Query Answers

Melisachew Wudage Chekol and Amedeo Napoli

LORIA (INRIA, CNRS, and Université de Lorraine), France
{melisachew.chekol,amedeo.napoli}@inria.fr

Abstract. Formal concept analysis (FCA) is used for knowledge discovery within data. In FCA, concept lattices are very good tools for classification and organization of data. Hence, they can also be used to visualize the answers of a SPARQL query instead of the usual answer formats such as: RDF/XML, JSON, CSV, and HTML. Consequently, in this work, we apply FCA to reveal and visualize hidden relations within SPARQL query answers by means of concept lattices.

1 Introduction

Recently, the amount of semantically rich data available on the Web has grown considerably. Since the conception of linked data publishing principles, over 295 linked (open) datasets (LOD) have been produced[1]. A reasonable number of these datasets provide endpoints for accessing (querying) their contents. Querying is mainly done through the W3C recommended query language SPARQL. The answers of SPARQL queries have often the following formats: RDF/XML, JSON, CSV, text, TSV, Java Script, XML, Spreadsheet, Ntriples, and HTML. It might be interesting to analyse, mine, and then visualize hidden relations within the answers. These tasks can be carried out using Formal Concept Analysis (FCA).

FCA is used for knowledge discovery within data represented by means of objects and their attributes [3]. Concept lattices can reveal hidden relations within data and can be used for organizing, classifying, and even mining data. A survey of the benefits of FCA to semantic web (SW) and vice versa has been proposed in [6] (in particular ontology completion [1]). Additionally, studies in [2] and [4] are based on FCA for managing SW data. The former provides an entry point to linked data using questions in a way that can be navigated. It gives a transformation of an RDF graph into a formal context where the subject of an RDF triple becomes the object, a composition of the predicate and object of the triple becomes an attribute. The latter obliges the user to specify variables corresponding to objects and attributes of a context. These variables are used to create a SPARQL query which is used to extract content from linked data in order to build the formal context. Following this line, we propose a way to

---

[1] `http://linkeddata.org/`

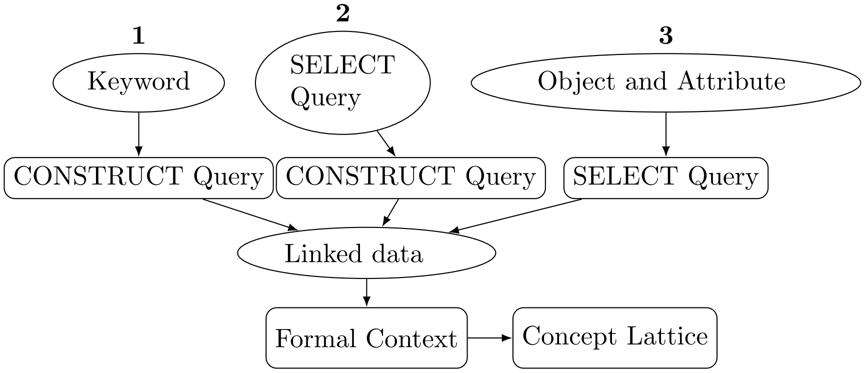


Fig. 1: Architecture of SPARQL answers organization.

organize Semantic Web data, and more precisely, the organization of SPARQL query answers by means of concept lattices. As a result, the user is able to visualize, navigate and classify the answers w.r.t. their context. For that, we propose the architecture depicted in Figure 1, based on three components which are discussed below:

1. *Keyword search:* In this component, a keyword (for instance, “14 juillet”) is used to search (to find information regarding this word) a specified dataset. To do so, a URI produced from the keyword is sent to the dataset to check its existence. When this is the case, a CONSTRUCT query containing the keyword is directed to the endpoint of the dataset. The answers are collected and organized to create a formal context as explained in the next section.
2. *SELECT Query:* This component builds a formal context out of the answers of a SPARQL query. SPARQL queries are converted into CONSTRUCT queries to form RDF graphs from the answers. Again, this will be illustrated in the next section.
3. *Variables corresponding to objects and attributes of a formal context:* This component enables the user to precisely specify the objects and attributes of the formal context. Out of which a SPARQL query is formed and sent to a chosen SPARQL endpoint. The answers of the query are collected to build a formal context. From that, a concept lattice is constructed. This is the approach considered by the authors in [4]. An example is proposed in the next section.

In each case, the objective is to build a formal context and then to build the associated concept lattice.

2 Proposal

A formal context represents data using objects and their attributes. Formally, it is a triple $K = (G, M, I)$ where G is a set of objects, M is a set of attributes, and $I \subseteq G \times M$ is a binary relation. A derivation operator ($'$) is used to compute *formal concepts* of a context. Given a set of objects A, a derivation operator $'$ computes the maximal set of attributes shared by objects in A and is denoted by A' (this is done dually with set of attributes B). A formal concept is a pair (A, B) where $A' = B$ and $B' = A$. A set of formal concepts ordered with the set inclusion relation form a *concept lattice* [3].

Definition 1 (RDF as a Formal Context). *Given an RDF graph G and a transformation function σ, a formal context is obtained from G as follows:*

- *If $\langle o_1, \text{rdf : type}, C \rangle \in G$, then $\sigma(\langle o_1, \text{rdf : type}, C \rangle) =$*

	C
o_1	x

- *If $\langle o_1, R, o_2 \rangle \in G$, then $\sigma(\langle o_1, R, o_2 \rangle) =$*

	$\exists R.\top$	$\exists R^{-}.\top$
o_1	x	
o_2		x

We consider a core fragment of RDFS called ρdf [5] which contains the minimal vocabulary, $\rho\text{df} = \{$`sp`, `sc`, `type`, `dom`, `range`$\}$, where `sp` denotes the *subproperty* relation, `sc` is *subclass*, and `dom` stands for *domain*. This fragment was proven to be minimal and well-behaved in [5]. Its semantics corresponds to that of full RDFS. Triples containing schema information are transformed as:

1. If $\langle C_1, \text{rdfs : subClassOf}, C_2 \rangle \in G$, then $\sigma(\langle C_1, \text{rdfs : subClassOf}, C_2 \rangle) = \forall o \in G.\ (o, C_1) \in I \Rightarrow (o, C_2) \in I$.
2. If $\langle R_1, \text{rdfs : subPropertyOf}, R_2 \rangle \in G$, then $\sigma(\langle R_1, \text{rdfs : subPropertyOf}, R_2 \rangle) = \forall o_1, o_2 \in G.\ (o_1, \exists R_1.\top) \in I \Rightarrow (o_1, \exists R_2.\top) \in I$.
3. If $\langle R, \text{rdfs : domain}, C \rangle \in G$, then $\sigma(\langle R, \text{rdfs : domain}, C \rangle) = \forall o_1 \in G.\ (o_1, \exists R.\top) \in I \Rightarrow (o_1, C) \in I$.
4. If $\langle R, \text{rdfs : range}, C \rangle \in G$, then $\sigma(\langle R, \text{rdfs : range}, C \rangle) = \forall o_2 \in G.\ (o_1, \exists R^{-}.\top) \in I \Rightarrow (o_2, C) \in I$.

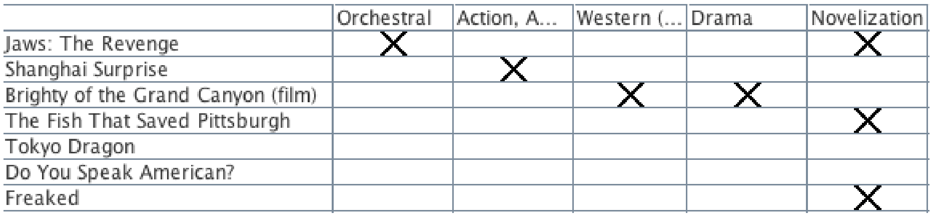
The users above are able to build a formal context from an RDF graph or a set of SPARQL query answers. Then, there are several algorithms that can compute the concept lattice associated with a formal context and that can be used in our framework.

Example: consider a SPARQL query that selects film titles (as objects of the formal context) and genres (as attributes) from DBpedia to populate a formal context. Consequently, this formal context is shown in Figure 2.

A possible concept lattice obtained from the formal context associated with the query answers is depicted in Figure 3. Now we have a classification of the results of the query w.r.t. a given topic or constraint. Additionally, this is exactly the same thing as if we were querying the Web with Google and here we have a classification of the answers w.r.t. a user constraints.

3 Conclusion

SPARQL query answers are provided in different formats (RDF/XML, CSV, JSON, TTL, and others), which do not reveal hidden semantics in the answers.



	Orchestral	Action, A...	Western (...	Drama	Novelization
Jaws: The Revenge	X				X
Shanghai Surprise		X			
Brighty of the Grand Canyon (film)			X	X	
The Fish That Saved Pittsburgh					X
Tokyo Dragon					
Do You Speak American?					
Freaked					X

Fig. 2: A part of the formal context associated with the query.

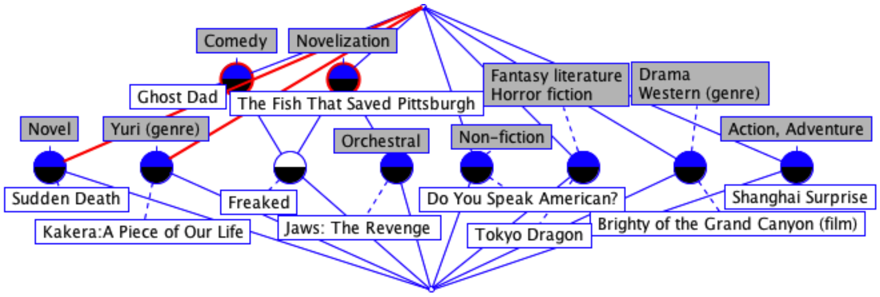


Fig. 3: Concept lattice of DBpedia movies and genres.

Concept lattices are useful in this regard. In this work, we used concept lattices to hierarchically organize and analyse the content of query answers.

This is an ongoing work and we are currently implementing the procedure. We should investigate how well it scales, given the size of SPARQL query answers over linked data. Overall, this work shows some of the benefits of FCA that can be provided to the semantic web.

References

1. Baader, F., Ganter, B., Sertkaya, B., Sattler, U.: Completing description logic knowledge bases using formal concept analysis. In: Proc. of IJCAI. vol. 7, pp. 230–235 (2007)
2. d'Aquin, M., Motta, E.: Extracting relevant questions to an RDF dataset using formal concept analysis. In: Proceedings of the sixth international conference on Knowledge capture. pp. 121–128. ACM (2011)
3. Ganter, B., Wille, R.: Formal Concept Analysis. Springer, Berlin (1999)
4. Kirchberg, M., Leonardi, E., Tan, Y.S., Link, S., Ko, R.K., Lee, B.S.: Formal concept discovery in semantic web data. In: ICFCA. pp. 164–179. Springer-Verlag (2012)
5. Muñoz, S., Pérez, J., Gutierrez, C.: Minimal deductive systems for RDF. In: The Semantic Web: Research and Applications, pp. 53–67. Springer (2007)
6. Sertkaya, B.: A survey on how description logic ontologies benefit from FCA. In: CLA. vol. 672, pp. 2–21 (2010)

A Study on the Correspondence between FCA and $\mathcal{ELI}$ Ontologies

Melisachew Wudage Chekol and Amedeo Napoli

LORIA (INRIA, CNRS, and Université de Lorraine), France
{melisachew.chekol,amedeo.napoli}@inria.fr

Abstract. The description logic $\mathcal{EL}$ has been used to support ontology design in various domains, and especially in biology and medecine. $\mathcal{EL}$ is known for its efficient reasoning and query answering capabilities. By contrast, ontology design and query answering can be supported and guided within an FCA framework. Accordingly, in this paper, we propose a formal transformation of $\mathcal{ELI}$ (an extension of $\mathcal{EL}$ with *inverse roles*) ontologies into an FCA framework, i.e. $K_{\mathcal{ELI}}$, and we provide a formal characterization of this transformation. Then we show that SPARQL query answering over $\mathcal{ELI}$ ontologies can be reduced to lattice query answering over $K_{\mathcal{ELI}}$ concept lattices. This simplifies the query answering task and shows that some basic semantic web tasks can be improved when considered from an FCA perspective.

1 Introduction

Knowledge discovery in data represented by means of objects and their attributes can be done using formal concept analysis (FCA) [5]. Concept lattices can reveal hidden relations within data and can be used for organizing and classifying data. A survey of the benefits of applying FCA to Semantic Web (SW) and vice versa has been proposed in [8]. As mentioned in that paper, a few of these benefits ranges from knowledge discovery, ontology completion, to computing subsumption hierarchy of least common subsumers. Additionally, studies in [3] and [7] are based on FCA for managing SW data while finite models of description logics (as $\mathcal{EL}$) are explored in [1]. All these studies propose methods for analysing SW data within FCA. Nevertheless, none of them offers a practical way of representing SW data within a formal context which is the basic data structure for FCA. We deem it necessary to provide a mathematically founded method to formalize the representation and the analysis of SW data based on FCA.

In this work, we focus on $\mathcal{ELI}$ (an extension of $\mathcal{EL}$ with inverse roles) ontologies. $\mathcal{EL}$ is one of OWL 2 profiles (OWL 2 $\mathcal{EL}$) which is mainly used for designing large biomedical ontologies such as SNOMED-CT[1] and the NCI thesaurus[2]. These two ontologies have large concept hierarchies that can be queried

---

[1] `http://www.ihtsdo.org/snomed-ct/`

[2] `http://ncit.nci.nih.gov/`

with SPARQL. However, including inferred data in query answering requires either a reasoner to infer all implicit information or query rewriting using property paths (that enable navigation in a hierarchy) [6]. The latter obliges the user to know the nuts and bolts of SPARQL. To overcome these difficulties, we reduce SPARQL query answering in $\mathcal{ELI}$ ontologies into query answering in concept lattices along with the transformation of the queried ontology into a formal context. Then, the resulting concept lattice provides support for query answering (but this does not replace SPARQL) and also for visualization and navigation of relations within SW data.

Overall, we work towards (i) a formal characterization of the transformation of ontologies into a formal context, (ii) translating the difficulty of SPARQL query answering over ontologies into query answering over concept lattices, and finally (iii) providing organization of SPARQL query answers with concept lattices.

2 Transforming $\mathcal{ELI}$ Ontologies into Formal Contexts

A formal context represents data using objects and their attributes. Formally, it is a triple $K = (G, M, I)$ where G is a set of objects, M is a set of attributes, and $I \subseteq G \times M$ is a binary relation. A derivation operator ($'$) is used to compute *formal concepts* of a context. Given a set of objects A, a derivation operator $'$ computes the maximal set of attributes shared by objects in A and is denoted by A' (this is done dually with set of attributes B). A formal concept is a pair (A, B) where $A' = B$ and $B' = A$. A set of formal concepts ordered with the set inclusion relation form a *concept lattice* [5].

One difficulty of transforming DL ontologies into formal contexts is mainly due to the fact that while DL languages are based on the *open world assumption (OWA)*, FCA relies on the *closed world assumption (CWA)*. The former permits to specify only known data whereas the later demands that all data should be explicitly specified. To slightly close the gap between these two worlds, we provide a transformation that maintains a DL semantics into an FCA setting.

To transform an $\mathcal{ELI}$ ontology $\mathrm{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ into a formal context $K = (G, M, I)$, the schema axioms in the TBox become background implications [4]. Then, individuals in the ABox correspond to objects in G, class names in the ABox and TBox yield attributes in M, and ABox assertions create relations between objects and attributes $I \subseteq G \times M$. Here, we consider acyclic TBoxes to avoid that class names become objects in a context. The following table gives a summary of the correspondence.

$\mathcal{ELI}$ $\mathrm{O} = \langle \mathcal{T}, \mathcal{A} \rangle$	FCA formal context $K_{\mathcal{O}} = (G, M, I)$ + background implications $\mathcal{L}$
$\mathcal{T} = \{C \sqsubseteq D\}$	$\{C \rightarrow D\} \in \mathcal{L}$
$\mathcal{A} = \{C(a),$	$a \in G$, $C \in M$, and $(a, C) \in I$
$R(a, b)\}$	$a, b \in G$, $\exists R.\top, \exists R^{-}.\top \in M$, $(a, \exists R.\top) \in I$, and $(b, \exists R^{-}.\top) \in I$

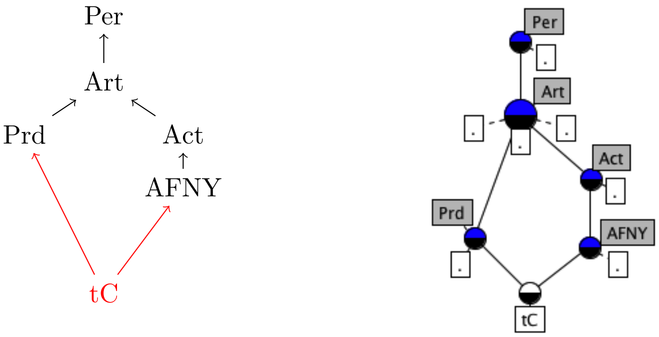


Fig. 1: The ontology in Example 1 and its associated concept lattice.

Example 1. Consider the transformation of the following $\mathcal{ELI}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ into a formal context $K_{\mathcal{O}}$ and its background implications $\mathcal{L}$.

$$\mathcal{T} = \{\text{ActorsFromNewYork (AFNY)} \sqsubseteq \text{Actor (Act)}, \\ \text{FilmProducer (Prd)} \sqsubseteq \text{Artist (Art)}, \text{Actor} \sqsubseteq \text{Artist}, \text{Artist} \sqsubseteq \text{Person (Per)}\}$$
$$\mathcal{A} = \{\text{tomCruise (tC)} \in \text{ActorsFromNewYork}\}$$

$K_{\mathcal{O}}$	AFNY	Prd	Act	Art	Per
tC	x				

$$\mathcal{L} = \{\ \text{AFNY} \rightarrow \text{Act},\ \text{Prd} \rightarrow \text{Art}, \\ \text{Act} \rightarrow \text{Art},\ \text{Art} \rightarrow \text{Per}\ \}$$

Construction of a concept lattice: In [4], an algorithm to construct a concept lattice from a formal context w.r.t. background implications is provided. This technique is employed here and the concept lattice associated with the formal context and background implications of Example 1 is depicted in Figure 1.

This is a simple example but SNOMED-CT and NCIT are much larger than that but not more complex. Let us how a concept lattice based on an $\mathcal{ELI}$ ontology can be queried.

3 SPARQL query answering over ontologies vs query answering over concept lattices

SPARQL query answering over $\mathcal{ELI}$ ontologies can be considered from the point of view of query answering over $K_{\mathcal{ELI}}$ concept lattices. Querying concept lattices amounts to fetching the objects given a set of attributes as query constants and to fetch the attributes given a set of objects as query constants or terms [2]. Query terms can be connected using the logical operators: *and*, *union*, and *set difference* to form a complex term.

SPARQL query answering over ontologies can be done in two main ways: (1) *Materialization* amounts to computing all implicit data before evaluating the query. This can be done by using a DL reasoner. (2) *Query rewriting* amounts to converting a query into another one using property paths and schema axioms.

Query answering over $\mathcal{ELI}$ ontologies with SPARQL appears to be harder than query answering over $K_{\mathcal{ELI}}$ concept lattices. SPARQL requires expensive tasks such as materialization and query rewriting but its expressive power is better than lattice querying. By contrast, lattice querying is practically sufficient to retrieve instances for $\mathcal{ELI}$ ontologies as shown by the following example.

Example 2. Let us consider the evaluation of the SPARQL query Q on the ontology $\mathcal{O}$ (in Figure 1) and its materialization $\mathcal{O}'$. Q = select all objects, elements who are artists = `SELECT ?x WHERE {?x a Artist .}`. Under simple entailment evaluation of a SPARQL query, the answer of Q over $\mathcal{O}$ is empty. To get non-empty answers for Q, one can evaluate Q over the materialization of $\mathcal{O}$ that we call $\mathcal{O}'$, where $Q(\mathcal{O}') = \{\text{tomCruise}\}$. Another way is to rewrite Q into Q' = `SELECT ?x WHERE {?x a/rdfs:subClassOf* Artist .}`. Q' selects all instances of Artist and that of its subclasses by navigating through the subclass relation. Then, the evaluation of $Q'(\mathcal{O})$ returns $\{\text{tomCruise}\}$.

By contrast, Q is converted into a lattice query as $q(x) = (x, Artist)$. The evaluation of this query over a concept lattice $K_{\mathcal{O}}$ obtained from $\mathcal{O}$ (Figure 1) is $Q'(K_{\mathcal{O}}) = \{\text{tomCruise}\}$, as it is sufficient to return all objects which are instances of Artist or any of its subconcept.

4 Discussion

It can be convenient to use FCA as a guideline for designing and querying $\mathcal{ELI}$ ontologies. In addition, FCA provides visualization and navigation capabilities. The present work does not apply to all ontologies but seems to be well suited to $\mathcal{ELI}$ ontologies. We plan to extend and experiment the proposed approach, especially with real-world and large datasets.

References

1. Baader, F., Distel, F.: Exploring finite models in the description logic EL gfp. In: ICFCA. pp. 146–161. Springer (2009)
2. Carpineto, C., Romano, G.: Concept data analysis: Theory and applications. Wiley (2004)
3. d'Aquin, M., Motta, E.: Extracting relevant questions to an RDF dataset using formal concept analysis. In: Proceedings of the sixth international conference on Knowledge capture. pp. 121–128. ACM (2011)
4. Ganter, B.: Attribute exploration with background knowledge. Theoretical Computer Science 217(2), 215 – 233 (1999)
5. Ganter, B., Wille, R.: Formal Concept Analysis. Springer, Berlin (1999)
6. Glimm, B.: Using SPARQL with RDFS and OWL entailment. Reasoning Web. Semantic Technologies for the Web of Data pp. 137–201 (2011)
7. Kirchberg, M., Leonardi, E., Tan, Y.S., Link, S., Ko, R.K., Lee, B.S.: Formal concept discovery in semantic web data. In: ICFCA. pp. 164–179. Springer-Verlag (2012)
8. Sertkaya, B.: A survey on how description logic ontologies benefit from FCA. In: CLA. vol. 672, pp. 2–21 (2010)

Towards Disambiguating Web Tables

Stefan Zwicklbauer, Christoph Einsiedler, Michael Granitzer, and
Christin Seifert

University of Passau
Innstrae 33a, 94032 Passau, Germany
{stefan.zwicklbauer,christin.seifert,michael.granitzer}@uni-passau.de

Abstract. Web tables comprise a rich source of factual information. However, without semantic annotation of the tables' content the information is not usable for automatic integration and search. We propose a methodology to annotate table headers with semantic type information based on the content of column's cells. In our experiments on 50 tables we achieved an F_1 value of 0.55, where the accuracy greatly varies depending on the used ontology. Moreover, we found that for 94% of maximal F_1 score only 20 cells (37%) need to be considered on average. Results suggest that for table disambiguation the choice of the ontology needs to be considered and the data input size can be reduced.

Keywords: Disambiguation, Semantic Enrichment, Table Annotation

1 Introduction

Tables on web sites or in scientific papers represent a valuable source of information for the human reader. For machines the story is different: Although tables are a structural representation of knowledge, the information itself is meaningless to machines – unless it is enriched with semantic information. The Semantic Web, and specifically the Linked Open Data initiative provide means for representing any kind of knowledge semantically. If tables were enriched semantically a variety of new applications could evolve, as is the idea of Google Fusion Tables [1] where the annotation is done by humans. Tables from different sources could be automatically aggregated, compared and used to generate new insights.

In this paper we move one step towards fully-automatic semantic table annotation. We propose a simple algorithm to annotate table headers with semantic types based on the types of all cells in that column. Further, we investigate the influence of the number of cells on the accuracy of the header-type inference.

2 Related Work

Current approaches in table annotation pursue a collective approach, i.e., the annotation model encompasses entities, types and relation between types. The underlying assumption is that columns and the cells in a column have some relation in common which is modeled in the semantic knowledge base. Limaye et al. [2] use a probabilistic graphical model to collectively annotate types (column headers), entities (cells), and semantic relations between types. They achieved an F_1 score of 0.56 on the Wiki table data set. The authors observed failures in the annotation if the corresponding correct links between entity types were not represented in the knowledge base. A recent paper employs a web search approach for entity classification [3] using the cell label as search query for a web search engine and applied text classification on the search results. Venetis et al. [4] annotate tables with class information crawled from the web based on a isA database mined with regular expressions. This web-crawled data base has a wider coverage than any modeled ontological database, but sufferes from more noise. The authors observe that using a hand-crafted ontology has a higher precision, but a high coverage is desired for their application of table search. Our approach differs in the following: First, we do not assume any relation of columns, or more specifically we do not assume that these relations are modeled in the knowledge base. Second, the only restriction we employ on the entity type is their availability in the knowledge base.

3 Approach

The general assumption behind our annotation algorithm is that the cells of a table column belong to one supertype, which we want to infer. We make no assumptions of interrelationships between columns, i.e. all columns are treated separately. Further, we assume that the tables do not have merged cells.

Let l_i $1 < i \leq n$ be the labels of non-header cells i and $E_i = \{e_i^k\}$ is the set of all possible semantic meanings of label l_i. The set T_i^k is the set of all type labels assigned to entity e_i^k. The annotation of table headers is performed in three steps (compare Figure 1):

Step 1 – Cell entity annotation: For each cell label l_i we derive a list of k possible entity candidates E_i using a search-based disambiguation method [5]. We set $k = 10$ in our experiments.

Step 2 – Entity-type resolution: For each entity candidate e_i^k in E_i a set of types is retrieved by following the `rdf:type` and `dcterms:subject` relations yielding the set of types T_i^k.

Step 3 – Type aggregation: The types assigned to the table header are the t types that occur most frequently in the set of all types of all cells $\bigcup_i \bigcup_k T_i^k$. We set $t = 1$ in our experiments, e.g. only use the most frequent type as result.

4 Experiments

Data Set. In our experiments we used dbpedia as knowledge base with type relations `rdf:type` and `dcterms:subject` from the Dublin Core Metadata Ontology[1]. We evaluated our approach on 50 tables extracted from Wikipedia pages

---

[1] `http://dublincore.org/documents/dcmi-terms/`

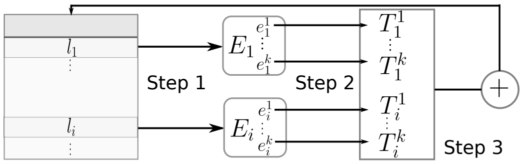


Fig. 1. Annotation process. 1) Cell labels are disambiguated to entity candidates. 2) Types of entity candidates are determined. 3) Type information is aggregated to determine the header type candidates.

Table 1. Performance for different cell annotation methods and type vocabularies. Reporting macro-averaged precision π, recall ρ, F_1.

Vocabulary	π^M	ρ^M	F_1^M
Rdf-Type	0.24	0.22	0.23
DublinCore	0.59	0.51	0.55
Rdf-Type + DublinCore	0.64	0.27	0.38

including all tables mentioned in Limaye et al. [2]. We removed all columns containing numbers or complete sentences.[2] The number of columns amounts to 132, the number of rows varies from 10 to 232 (average 54.1). We manually annotated the tables' columns yielding a total of 329 type annotations, 169 `rdf:type` and 160 `dcterms:subject`, averaging to 2.49 annotations per column header.

Overall Performance. We assessed the overall performance on the complete data set. Table 1 shows the results for three different type vocabularies (using Rdf-Type relations only, using DublinCore subjects only, or using both). In terms of precision the combined vocabulary performs best (0.64), however only slightly better than using DublinCore subjects only (0.59), whereas Rdf-type annotations are worst (0.24). For the combined approach, F_1 is low due to the low recall, which is because we have more correct results in the ground-truth but consider only the best result in the evaluation.

Table Length. In a second experiment we assessed the influence of the number of cells on the accuracy of table header disambiguation. From all 192 columns we randomly selected k cells for the cell-entity annotation step and assessed the header disambiguation accuracy using the DublinCore vocabulary. We repeated the experiment 10 times with different randomly selected cells for each $k \in \{1, 2, ..7, 8, 10, 12, 15, 20\}$. Figure 2 shows precision, recall and F_1 measure averaged over all runs. As expected for small numbers of cells the performance increases significantly when adding one more cell (e.g. from 3 to 4 cells the F_1 measure increases from 0.27 to 0.35 a growth of 26%). For larger numbers of cells there is less information gain by adding one more cell resulting in smaller increases in performance (all below 10%). Using 20 cells results in F_1 of 0.514, which is 94% of the F_1 achieved with all cells (0.547).

---

[2] The data set is available at `https://github.com/quhfus/table-disambiguation`

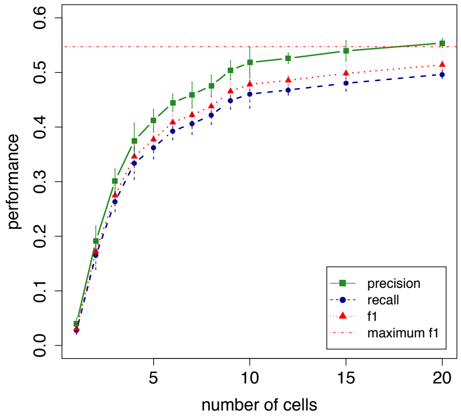


Fig. 2. Performance depending on number of cells. Showing mean and standard deviation over 10 runs, and maximum F_1 value (using all cells).

5 Conclusion and Future Work

We proposed an algorithm to annotate table headers with semantics based on the types of the column's cells. We achieved similar accuracy as previous work with more complex methods. We expect the reason for the comparable performance to be the knowledge base with more exhaustive and qualitative annotations. From our experiments it seems reasonable to use only a small number of cells for annotating the header (20 cells lead to 94% of the total achievable accuracy) if performance is an issue. We plan to exploit more relational knowledge (e.g. `same-as`) to further improve the annotations.

Acknowledgements. The presented work was developed within the CODE project funded by the EU Seventh Framework Programme, grant agreement number 296150.

References

1. Gonzalez, H., Halevy, A.Y., Jensen, C.S., Langen, A., Madhavan, J., Shapley, R., Shen, W., Goldberg-Kidon, J.: Google fusion tables: web-centered data management and collaboration. In: Proc. ACM SIGMOD, New York, ACM (2010) 1061–1066
2. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. Proc. VLDB Endow. **3**(1-2) (September 2010) 1338–1347
3. Quercini, G., Reynaud, C.: Entity discovery and annotation in tables. In: Proc. EDBT, New York, NY, USA, ACM (2013) 693–704
4. Venetis, P., Halevy, A., Madhavan, J., Paşca, M., Shen, W., Wu, F., Miao, G., Wu, C.: Recovering semantics of tables on the web. Proc. VLDB Endow. **4**(9) (6 2011) 528–538
5. Zwicklbauer, S., Seifert, C., Granitzer, M.: Do we need entity-centric knowledge bases for entity disambiguation? In: Proc. I-KNOW. (2013)

A Restful Interface for RDF Stream Processors

Marco Balduini[1] and Emanuele Della Valle[1]

DEIB – Politecnico di Milano, Italy
marco.balduini@polimi.it, emanuele.dellavalle@polimi.it

Abstract. This poster proposes a minimal, backward compatible and combinable restful interface for RDF Stream Engine.

1 Introduction

A number of RDF Stream Processors exists (e.g., CQELS [1], SPARQL$_{stream}$ [2], ETALIS/EP-SPARQL [3], Sparkwave [4], INSTANS [5] and C-SPARQL Engine [6]), but **they do not talk each other**.

This hampers comparative evaluations: existing benchmark proposals [7, 8] had to create software adapters to test the various processors. In this condition, it is difficult to assess how much the benchmark results depend on the performances of the processors and how much on those of the adapters.

Moreover, the lack of a shared protocol to transmit RDF streams hinders the combined usage of those processors. For instance, a user may want: *a*) to deploy SPARQL$_{stream}$ to natively process data streams[1]; *b*) to semantically enrich the resulting RDF streams using Sparkwave (or INSTANS); *c*) to aggregate the enriched streams in events using the C-SPARQL Engine (or CQELS); and *d*), finally, to detect complex events with ETALIS/EP-SPARQL.
This poster proposes a restful interface for RDF Stream Processors that is:

1. **minimal** – more sophisticated interface can be envisioned, but in this attempt we would like to create a broad consensus, thus we avoid proposing controversial solutions.
2. **backward compatible** – we are reusing RDF and SPARQL standards wherever we can so to guarantee that adaptation of non-streaming clients for RDF and SPARQL is straight forward.
3. **combinable** – the proposed interface enforces that the output of a processor can serve as input to a processor (including the one that generates it).

The remainder of the paper is organised as follows. Section 2 briefly presents the background required to understand the proposed interface. Section 3 proposes the interface. Section 4 shortly discusses two requirements that are not considered for this minimal proposal and how the interface can be extended to cover them. A proof of concept implementation of the proposed interfaces for the C-SPARQL engine is available for download at `http://streamreasoning.org/download`.

---

[1] SPARQL$_{stream}$ rewrites continuous SPARQL queries issued against virtual RDF streams on continuous SQL-like queries on data streams.

2 Background

From a conceptual point of view, existing RDF Stream Processors are homogenous. They define the notion of *RDF Stream* – an unbound list of tuples $< t, \tau >$ where t is an RDF triple and τ is a non-decreasing timestamp –, and *continuous SPARQL query* – a SPARQL query extended so that it can process RDF streams using continuous operators (e.g., windows to logically convert a portion of the infinite RDF stream in an RDF graph) and time-aware operators (e.g., sequence to ask that a graph pattern is detected before another one).

To the best of our knowledge, limited efforts was spent in defining a protocol for: *a*) transmitting RDF stream across RDF Stream Processor on separated machines, *b*) registering a continuous query in a processor, and *c*) observing the continuously evolving results. The only existing solution are proprietary. For instance, the C-SPARQL Engine is typically used within the Streaming Linked Data framework [9]. Similarly, CQELS is paired with the Super Stream Collider [10].

3 Services

A community effort is needed to propose a continuous SPARQL extension that can span across the existing proposals, but we believe this is the right time to propose a restful interfaces that processors can easily implement.

The following proposal specifies how to manage RDF streams, continuous SPARQL queries, and observers of continuous results (see Table 1 for details).

Complying to restful principles, users can register a new RDF stream σ in the processor using the `PUT` method on the resource `/streams/`. As a result the RDF stream `/streams/`σ becomes available in the processor. At this point, they can stream information on the RDF stream `POST`ing an RDF graph to `/streams/`σ) and they can unregister it using the `DELETE` method. The list of all registered stream is returned when `GET`ting the resource `/streams/`.

It is worth to note that, learning from flexible time management in data stream processors [11], we propose to avoid annotating the streamed RDF graphs with a timestamp. This complies to the expected input of best effort data stream processors (e.g. esper). We leave the annotation of the streamed RDF graphs with application timestamp to a future extension of this **minimal** protocol. Moreover, this design decision allows the proposal to be **backward compatible**. Any Semantic Web application can send information to an RDF Stream Processor simply posting an RDF graph.

User can register a new continuous SPARQL query γ in the processor using the `PUT` method on the resource `/queries/`. The proposed interface is agnostic w.r.t. the language used to declare the query and leaves to the processor to parse the query in the body of the request. Nonetheless, it requires the query to refer only to RDF streams already registered in the processor. If the user tries to register a query on streams that have not been registered, yet, the service must refuse to register the query. If the registration is successful, the processor starts the continuous execution of the query and the query `/queries/`γ appears in

Table 1: The herein proposed restful interfaces for RDF Stream Processors. Along with restful principles, `GET`ing a resource returns what was `PUT`ed.

RDF Streams

Method	Address	Body	Description
PUT	/streams/<id>		Register new stream
DELETE	/streams/<id>		Delete specified stream
POST	/streams/<id>	RDF model	Stream new information
GET	/streams		Get the list of streams

Continuous SPARQL queries

Method	Address	Body	Description
PUT	/queries/<id>	query	Register new query
DELETE	/queries/<id>		Delete specified query
POST	/queries/<id>	callback URL	Adds an observer
POST	/queries/<id>	Action [pause, restart]	Change query status
GET	/queries		Get the list of queries

Observers

Method	Address	Body	Description
DELETE	/queries/<id>/observers/<id>		Delete specified observer
GET	/queries/<id>/observers		Get observers list

the list of queries that can be retrieved `GET`ting the resource `/queries/`. As for the RDF streams, the query `/queries/`γ can be unregistered using the `DELETE` method. The method `POST` on the resource `/queries/`γ is used to start observing the query results, to pause the query and to restart it.

Access to query results follows an observable-observer design pattern. In order to start observing the results of a query γ, a user has to `POST` a callback URL to `/queries/`γ. The created observer ω is identified by an URL of the form `/queries/`γ/observers/ω. The user can stop observing the query by `DELET`ing this resource. Multiple observers per query are possible. Whenever γ computes new results, the processor notifies all the observers by invoking the provided callback URLs.

If the query is of the forms `SELECT` or `ASK`, results must be formatted according to SPARQL 1.1 query results[2], thus allowing for **backward compatibility** with existing SPARQL resultset parsers.

If the query is of the forms `CONSTRUCT` or `DESCRIBE`, the processor must `POST` an RDF graph containing the result. As a result our proposal is not only **backward compatibility** – it is conform to SPARQL 1.1 result formats –, but it is also **combinable** – the results of a query can be `POST`ed to another registered RDF stream. The callback URL passed as parameter in starting an observer simply has to be the URL of an existing RDF stream[3].

4 Conclusions

The proposal, being minimal, ignored important requirements w.r.t. time modelling, access control, and transmission overhead.

---

[2] Our implementation supports `http://www.w3.org/TR/2013/REC-sparql11-results-json/`

[3] In order to avoid the overhead to stream on HTTP an RDF stream that is consumed by the same processor, when a query γ of the forms `CONSTRUCT` or `DESCRIBE` is registered, an RDF stream, whose identifier is `/streams/`γ, is automatically registered. The result of the query γ is internally streamed on it.

Adding the application time to the protocol is only a matter to `POST` a timestamp together with the RDF graph. However, as explained in [11], in the case of multiple distributed sources `POST`ing to the same RDF stream, out-of-orders can appear due to lack of clock synchronisation and different network delays. In our future work, we will propose this extension and, at the same time, we will release an open-source package that includes the management out-of-orders.

The proposed interface lacks access control, but it is ready for HTTP-based access control. An HTTP server, between the user and the Restful service container, can handle access to `/streams` and `/queries`. Moreover, only the owner of a query γ can start observing the results of γ or is allowed to list all the observers (i.e., `GET`ting `/queries/`γ`/obsevers/`). However, investigating OAuth-based access-control is on our research agenda.

Last, but not least, we recognise that the transmission overhead of the proposed solution can reduce the processor throughput if the user frequently `POST`s RDF graphs containing only few triples. In our future work, we intent to explore the streaming of RDF triples in N-quads format on a Web-socket.

References

1. Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: ISWC. (2011) 370–388
2. Calbimonte, J.P., Corcho, O., Gray, A.J.G.: Enabling ontology-based access to streaming data sources. In: ISWC. (2010) 96–111
3. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: a unified language for event processing and stream reasoning. In: WWW. (2011) 635–644
4. Komazec, S., Cerri, D., Fensel, D.: Sparkwave: continuous schema-enhanced pattern matching over RDF data streams. In: DEBS. (2012) 58–68
5. Rinne, M., Nuutila, E., Törmä, S.: Instans: High-performance event processing with standard rdf and sparql. In: ISWC (Posters & Demos). (2012)
6. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Querying rdf streams with c-sparql. SIGMOD Record **39**(1) (2010) 20–26
7. Zhang, Y., Duc, P., Corcho, O., Calbimonte, J.P.: SRBench: A Streaming RDF/SPARQL Benchmark. In: ISWC. (2012) 641–657
8. Le-Phuoc, D., Dao-Tran, M., Pham, M.D., Boncz, P., Eiter, T., Fink, M.: Linked stream data processing engines: Facts and figures. In: ISWC. (2012) 300–312
9. Balduini, M., Celino, I., Dell'Aglio, D., Valle, E.D., Huang, Y., il Lee, T.K., Kim, S.H., Tresp, V.: Bottari: An augmented reality mobile application to deliver personalized and location-based recommendations by continuous analysis of social media streams. J. Web Sem. **16** (2012) 33–41
10. Quoc, H.N.M., Serrano, M., Le-Phuoc, D., Hauswirth, M.: Super stream collider–linked stream mashups for everyone. In: Proceedings of the Semantic Web Challenge co-located with ISWC2012, Boston, MA, US (2012)
11. Srivastava, U., Widom, J.: Flexible time management in data stream systems. In: PODS, New York, New York, USA (2004) 263

TripleRush

Philip Stutz, Mihaela Verman, Lorenz Fischer, and Abraham Bernstein

DDIS, Department of Informatics, University of Zurich, Zurich, Switzerland
`{stutz, verman, lfischer, bernstein}@ifi.uzh.ch`

1 Introduction

TripleRush[1] is a parallel in-memory triple store designed to address the need for efficient graph stores that quickly answer queries over large-scale graph data. To that end it leverages a novel, graph-based architecture.

Specifically, TripleRush is built on a parallel and distributed graph processing framework. The index structure is represented as a graph where each index vertex corresponds to a triple pattern. Partially matched copies of a query are routed in parallel along different paths of this index structure.

Among the existing triple stores, we only know of Trinity.RDF [3] to be implemented on top of a distributed graph processing abstraction. Trinity.RDF represents the graph with adjacency lists and combines traditional query processing with graph exploration.

In TripleRush, an RDF triple is represented as a vertex and SPARQL queries are answered with a purely exploration-based approach. In other words, TripleRush does not use any joins in the traditional sense but searches the index graph in parallel. Whilst traditional stores pipe data through query processing operators, TripleRush routes query descriptions to data. We implemented TripleRush on top of our graph processing framework SIGNAL/COLLECT [2].

TripleRush takes less than a third of the time to answer queries compared to the fastest of three state-of-the-art triple stores, when measuring time as the geometric mean of all queries for two benchmarks.

2 Foundation: Signal/Collect

SIGNAL/COLLECT [2] is a parallel and distributed large-scale graph processing system written in Scala. Akin to Pregel [1], it allows to specify graph computations in terms of vertex-centric methods.

The key features of SIGNAL/COLLECT are: a) it is suitable for expressing data-flow algorithms and transparently parallelizes them, using vertices as processing elements and edges for message propagation, b) it supports different types of vertices, c) the graph structure can be changed during computation, d) it supports bulk-messaging and combiners for message-passing efficiency, e) it supports asynchronous scheduling, and f) it is flexible with regard to edge representation, messages can be routed directly.

3 TripleRush

TripleRush is a triple store with three types of SIGNAL/COLLECT vertices:

---

[1] Source code at `https://github.com/uzh/triplerush`

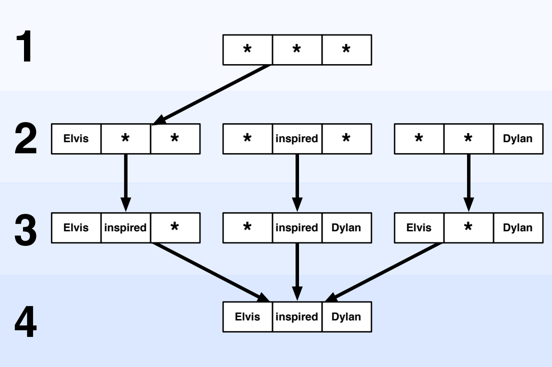


Fig. 1. TripleRush index structure that is created for the triple vertex [Elvis inspired Dylan].

Triple vertices (level 4, Fig. 1) represent triples in the database. Each contains subject, predicate, and object information.

Index vertices (levels 1-3, Fig. 1) represent triple patterns and are responsible for routing partially matched copies of queries (referred to as *query particles*) towards triple vertices that match their respective patterns. They also contain subject, predicate, and object information, but one or several of them are wildcards.

Query vertices (Fig. 2) are added to the graph for each query that is being executed. The query vertex emits the first query particle that traverses the index structure. Once all query particles—successfully matched or not—get routed back to their respective query vertex, it reports the results and removes itself from the graph.

The graph is built bottom-up, starting by creating a *triple vertex* for each RDF triple. These vertices are added to Signal/Collect, which turns them into parallel processing units. A triple vertex will add its immediate *index vertices* (if they do not exist yet) and an edge from these vertices to itself. The construction process continues recursively for the index vertices until the parent vertex has already been added or the index vertex has no parent.

To ensure the uniqueness of a path from an index vertex to all triple vertices below it, an index vertex adds an edge from at most one parent index vertex, always according to the structure that is illustrated in Fig. 1.

The index graph we just described is different from traditional index structures, because it is designed for the efficient parallel routing of messages to triples that correspond to a given triple pattern. All vertices that form the index structure are active parallel processing elements that only interact via message passing.

Consider the subgraph shown in Fig. 2 and the query processing for the query: (unmatched = [?X inspired ?Y], [?Y inspired ?Z]; bindings = {}). The query execution starts by adding the query vertex to the TripleRush graph.

❶ The query vertex emits a single query particle, which is routed (by Signal/Collect) to the index vertex that matches its first unmatched triple

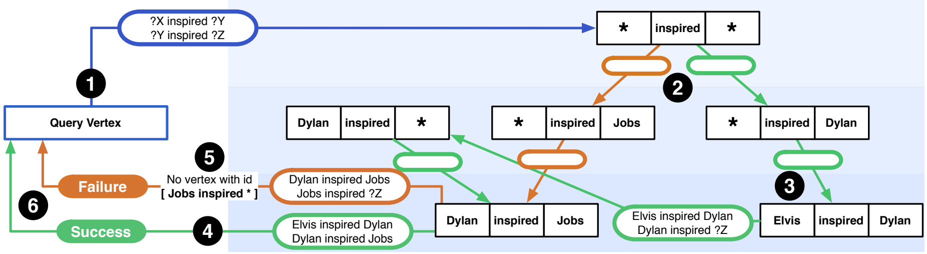


Fig. 2. Query execution on the relevant part of the index that was created for the triples [Elvis inspired Dylan] and [Dylan inspired Jobs].

pattern. To determine when a query has finished processing, the initial query particle is endowed with a large number of tickets.

② When a query particle arrives at an index vertex, a copy of it is sent along each edge. The original particle evenly splits up its tickets among its copies.

③ Once a query particle reaches a triple vertex, the vertex attempts to match the next unmatched query pattern to its triple. If this succeeds, then a variable binding is created and the remaining triple patterns are updated with the new binding. The query particle gets sent to the index or triple vertex that matches its next unmatched triple pattern.

④ If all triple patterns are matched, then the query particle gets routed back to its query vertex.

⑤ If no vertex with a matching pattern is found, then a handler for undeliverable messages routes the failed query particle back to its query vertex.

⑥ Query execution finishes when the sum of tickets of all failed and successful query particles received by the query vertex equals the initial ticket endowment of the first particle that was sent out. Then, the query vertex reports the result that consists of the variable bindings of the successful query particles, and removes itself from the graph.

We further perform some optimizations: a) we do dictionary encoding, b) we remove the triple vertices and turn the third index level into binding index vertices, which hold a compact representation of all the triples that match their pattern, c) the index vertices on levels 1 and 2, in addition to a compact edge representation, use delta-encoding and variable length integers to further reduce memory usage, d) we use a query optimizer that reorders patterns based on cardinalities, e) we only send the tickets of the failed particles back to the query vertex, and f) we use bulk-messaging and message-combiners.

4 Evaluation

In order to evaluate TripleRush, we wanted to compare it with the fastest related approaches. Trinity.RDF [3] is also based on a parallel in-memory graph store, and it is, to our knowledge, the best performing related approach. As Trinity.RDF is not available for evaluations, we made our results comparable by closely following the setup of their published parallel evaluations. The Trinity.RDF paper also includes results for other in-memory and on-disk systems that were evaluated with the same setup, which allows us to compare TripleRush with these other systems in terms of performance.

Consistent with the parallel Trinity.RDF [3] evaluation, we benchmarked the performance of TripleRush by executing the same queries on the LUBM-160 and DBPSB-10 datasets. More information about the setup is found in [3].

The execution time covers everything from the point where a query is dispatched to TripleRush until the results are returned. Consistent with the Trinity.RDF setup, the execution times *do* include the time used by the query optimizer, but *do not* include the dictionary encoding and the triple materialization.

The results in Figure 3 show the fastest execution time of 10 runs for all stores. The results for the other stores are from the Trinity.RDF paper [3] and were measured on comparable hardware. TripleRush is fastest on all but one query. These results indicate that the performance of TripleRush is competitive with, or even superior to other state-of-the-art triple stores.

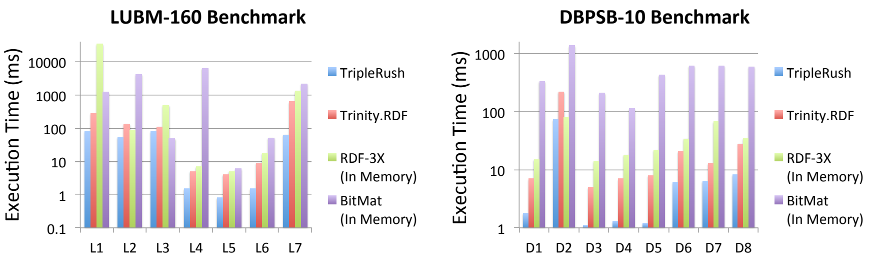


Fig. 3. Execution times for queries on the LUBM-160 and DBPSB-10 benchmarks (note the logarithmic scale). Comparison data for other stores from [3].

5 Conclusions

The need for efficient querying of large graphs lies at the heart of most Semantic Web applications. During the last decade, research in this area has made tremendous progress based on a database-inspired paradigm. Parallelizing these centralized architectures is a complex task. The advent of multi-core computers, however, calls for approaches that exploit parallelization.

With TripleRush we presented an in-memory triple store that divides the work among a large number of active processing elements that work towards a solution in parallel, and our evaluation shows that the performance of this approach is promising.

References

1. G. Malewicz, M. H. Austern, A. J. C. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In *SIGMOD Conference*, pages 135–146, 2010.
2. P. Stutz, A. Bernstein, and W. W. Cohen. Signal/Collect: Graph Algorithms for the (Semantic) Web. In P. P.-S. et al., editor, *International Semantic Web Conference (ISWC) 2010*, volume LNCS 6496, pages pp. 764–780. Springer, Heidelberg, 2010.
3. K. Zeng, J. Yang, H. Wang, B. Shao, and Z. Wang. A distributed graph engine for web scale rdf data. *Proceedings of the VLDB Endowment*, 6(4), 2013.

Adding Time to Linked Data: A Generic Memento proxy through PROV

Miel Vander Sande, Sam Coppens, Ruben Verborgh,
Erik Mannens, and Rik Van de Walle

Ghent University – iMinds – Multimedia Lab
Gaston Crommenlaan 8 bus 201, B-9050 Ledeberg-Ghent, Belgium
firstname.lastname@ugent.be

Abstract. Linked Data resources change rapidly over time, making a valid consistent state difficult. As a solution, the Memento framework offers content negotiation in the datetime dimension. However, due to a lack of formally described versioning, every server needs a costly custom implementation. In this poster paper, we exploit published provenance of Linked Data resources to implement a generic Memento service. Based on the W3C PROV standard, we propose a loosely coupled architecture that offers a Memento interface to any Linked Data service publishing provenance.

1 Introduction

Linked Data defines data to be published as resources on the web, uniquely identified by persistent URIs. However, the state of these resources changes rapidly over time, which causes inconsistencies in the links between them. This is of great concern to enterprises maintaining their data archives extensively. Requesting a consistent state of resources at a given point in time is crucial for recovery, analytics and administration purposes. A popular solution is the Memento framework [3]. It provides access to prior versions of web resources through datetime negotiation over HTTP. With a fixed datetime, clients can access a consistent state valid at that time. Unfortunately, every server needs a custom implementation, since there is no uniform way of exposing the relations between the different stored versions.

In Linked Data, *provenance* has been a hot research topic for years. It formally describes where the current resource state originates from. Recently, the W3C Provenance Working Group released the PROV [2] standard, enabling the publication of provenance in a uniform way. Based on uniform provenance, any Linked Data server could feed the Memento framework in a loosely coupled way.

In this poster paper, we extend the Memento framework with provenance discovery and exploit it to enable generic access. We redefine Memento as an independent service, compatible with any Linked Data server publishing provenance. First, we shortly introduce the Memento framework (Section 2) and propose an extended architecture (Section 3). Next, we describe our approach for generic provenance-based content negotiation (Section 4). Finally, we add conclusions and some future work (Section 5).

2 Overview of Memento

Memento is an HTTP framework for accessing prior archived versions of web resources, based on a given datetime. It defines three types of resources:

- Original Resource R_i: an existing resource on the web, of which prior versions are desired.
- Memento $M_{i,j}$: encapsulated prior states of the requested R_i.
- Timegate G_i: a resource supporting content negotiation in the datetime dimension. When requested, it decides on a best matching $M_{i,j}$, where t_j is the given datetime.

An architectural overview is given in Figure 1. The resources R_i, G_i and M_i are connected through hypermedia. When R_i is requested, the HTTP response contains a *Link* header, pointing at G_i. When G_i is requested, it responds with its *Location* header set to the selected $M_{i,j}$ resource. When M_i, j is requested, the response holds link headers to its related R_i and to the previous, the next, the first and the last M_i.

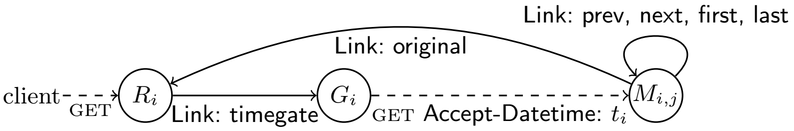


Fig. 1. Hypermedia connecting Original Resource R_i, Timegate G_i and Memento $M_{i,j}$

3 Extended architecture

Memento needs a tailored implementation on every server, due to lacking formal and uniform descriptions about the structure and locations of the different archives. In our approach, we will use provenance of web resources to create a generic Memento implementation as demonstrated in Figure 2. We define two independent types of services: a *Linked Data Service* LDS and a *Generic Memento Proxy* GMP. The former publishes a Linked Data resource L_i and their provenance in PROV P_i, which describes the archives $A_{i,j}$. The latter provides the functionality of the Memento framework, providing access to the resources R_i, $M_{i,j}$ and G_i. The decision logic in G_i now depends on P_i to select a matching $M_{i,j}$, which is explained in Section 4. This results into a loose coupling between LDS and GMP, thus making publishing provenance the only requirement for adding Memento functionality. This reduces costs and effort, while enabling other applications of provenance.

4 Generic time-based content negotiation

When requested, the timegate module decides on a $M_{i,j}$ based on the value t_i from the *Accept-DateTime* header and the descriptions in P_i. We use the semantic reasoner EYE [1] to implement our decision logic, which has three main advantages. First, PROV descriptions can be directly requested as RDF, using the PROV-O[1] ontology, which is natively supported. Second, we can describe our

---

[1] `http://www.w3.org/TR/prov-o/`

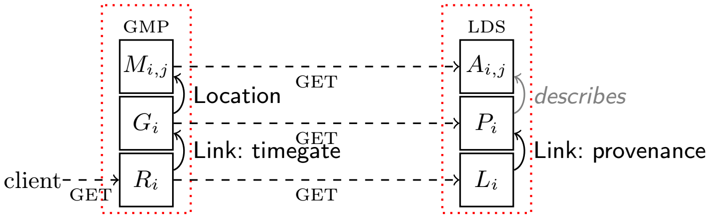


Fig. 2. The Generic Memento proxy GMP and the Linked Data service LDS are loosely coupled, creating a generic architecture

```
@prefix prov:  <http://www.w3.org/ns/prov#>.
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#>.
<Resource> prov:wasRevisionOf <Resource/1>; prov:wasGeneratedBy :rev3.
:rev3 prov:endedAtTime "2012-04-18T14:30:00Z"^^xsd:dateTime.

<Resource/1> prov:wasRevisionOf <Resource/2>; prov:wasGeneratedBy :rev2.
:rev2 prov:endedAtTime "2012-04-15T14:30:00Z"^^xsd:dateTime.

<Resource/2> prov:wasRevisionOf <Resource/3>; prov:wasGeneratedBy :rev1.
:rev1 prov:endedAtTime "2012-04-11T12:30:00Z"^^xsd:dateTime.
```

Listing 1. Provenance Record for `http://example.org/Resource` in PROV-O

logic in only a few compact N3 rules. Third, it can easily be extended with more complex logic if desired later. We can devide the approach into two main steps: discovery of provenance and selecting the memento.

Discovery of provenance Before any decisions can be made, the provenance has to be retrieved. As descibed by the PROV-AQ note[2], a link header pointing to the PROV description is added to an HTTP response. The module will lookup the resource L_i and dereference the URI in the header. As stated above, we will request the provenance in RDF. An example is given in Listing 1.

Selecting the Memento Once the provenance is retrieved, we decide on a memento $M_{i,j}$ using semantic reasoning. The resulting rules are demonstrated in Listing 2. First, we identify all mementos. Each revision is linked to its predecessor using the predicate *prov:wasRevisionOf*, forming a chain of revisions with R_i as endpoint. Relying on the transitive property defined in OWL logic, the relation between Ri and $M_{i,j}$ is derived by adding the triples on lines 5 and 6.

Next, we select a version valid at a given datetime *[line 8]*. The predicate *prov:wasGeneratedBy* links each version to an instance of *prov:Activity*, whose *prov:endedAtTime* predicate indicates the initiation of validity. The rule starts with extracting the defined datetime *[line 10]* and creating a finite list of occuring datetimes. This list is composed by finding all datetimes *[line 11]* that occur on or before the requested datetime *[line 13]*. The valid version occurs on the latest datetime in that list *[lines 15 and 16]*, and is added to the response *[line 18]*. In addition, we define analogue rules to select the first, the last, the next and the previous memento as well, since links to all of these resources are required. The complete rule file can be found here: `http://goo.gl/dz13UN`.

---

[2] `http://www.w3.org/TR/2013/NOTE-prov-aq-20130430/`

```
@prefix prov:  <http://www.w3.org/ns/prov#>.
@prefix pred:  <http://www.w3.org/2007/rif-builtin-predicate#>.
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#>.
@prefix e:  <http://eulersharp.sourceforge.net/2003/03swap/log-rules#>.
prov:wasRevisionOf rdfs:subPropertyOf :memento.
:memento a owl:TransitiveProperty.

:request :datetime "2012-04-11T12:30:00Z"^^xsd:dateTime
{
  :request :datetime ?req_datetime.
  [] e:findall (?datetime {
    ?rev prov:endedAtTime ?datetime .
    (?datetime ?req_datetime) pred:dateTime-less-than-or-equal true.
  } ?datetime_list) .
  ?datetime_list e:max ?current_datetime.
  ?current prov:endedAtTime ?current_datetime.
} => {
  :response :memento ?current.
}.
...
```

Listing 2. N3Logic selects the Memento valid at a specific Datetime

After the rule execution, the derived result can be mapped directly to the response. We add a *Location* header pointing at $M_{i,j}$ and add the necessary Memento-specific *Link* headers.

5 Conclusion and Future Work

Many enterprises publish their data archives as Linked Data. This requires access to a consistent state of all resources. The Memento framework solves this, but requires a costly custom implementation on each server. In the described approach, we avoid these costs by extending the framework using provenance descibed in PROV. We proposed a loosely-coupled architecture, where a Memento server can operate independently. We explained how semantic reasoning can implement the decision logic in a quick and scalable way.

In a related project, we have created *r&wbase* [4], a triple version control approach for triple stores. In future work, the capabilities of existing interfaces (e.g., SPARQL, Linked Data Platform and the Graph Store Protocol) to access different versions will be investigated. The approach described in this paper simplifies datetime-based version selection for these interfaces.

References

1. J. De Roo. Euler proof mechanism, 1999–2013. Available at `http://eulersharp.sourceforge.net/`.
2. P. Groth and L. Moreau. PROV-Overview: An Overview of the PROV Family of Documents. W3C Working Group Note, 2013.
3. H. Van de Sompel, M. L. Nelson, R. Sanderson, L. Balakireva, S. Ainsworth, and H. Shankar. Memento: Time travel for the Web. *CoRR*, abs/0911.1112, 2009.
4. M. Vander Sande, P. Colpaert, R. Verborgh, S. Coppens, E. Mannens, and R. Van de Walle. R&Wbase: git for triples. In *Proceedings of the 6th Workshop on Linked Data on the Web*, May 2013.

Distributed SPARQL Throughput Increase: On the effectiveness of Workload-driven RDF partitioning

Cosmin Basca and Abraham Bernstein

DDIS, Department of Informatics, University of Zurich, Zurich, Switzerland
{lastname}@ifi.uzh.ch

1 Introduction

The current size and expansion of the Web of Data or WoD, as shown by the staggering growth of the Linked Open Data (LOD) project[1], which reached to over 31 billion triples towards the end of 2011, leaves federated and distributed Semantic DBMS' or SDBMS' facing the open challenge of scalable SPARQL query processing. Traditionally, SDBMS' push the burden of efficiency at runtime on the query optimizer. This is in many cases too late (i.e., queries with many and/or non-trivial joins). Extensive research in the general field of *Databases* has identified *partitioning*, in particular horizontal partitioning, as a primary means to achieve scalability. Similarly to [2] we adopt the assumption that *minimizing the number of distributed-joins as a result of reorganizing the data over participating nodes will lead to increased throughput in distributed SDBMS'*. Consequently, the benefit of reducing the number of distributed joins in this context is twofold:

A) *Query optimization becomes simpler*. Generally regarded as a hard problem in a distributed setup, query optimization benefits, at all execution levels, from fewer distributed joins. During *source selection* the optimizer can use specialized indexes like in [5], while during *query planning* better query plans can be devised quicker, since much of the optimization burden and complexity is shifted away from the distributed optimizer to local optimizers.

B) *Query execution becomes faster*. Not having to pay for the overhead of shipping partial results around, naturally reduces the time spent waiting for usually higher latency network transfers. Furthermore, federated SDBMS' incur higher costs as they have to additionally serialize and deserialize data.

The main contributions of this poster are: ***i***) the presentation of a novel and naïve workload-based RDF partitioning method[2] and ***ii***) an evaluation and study using a large real-world query log and dataset. Specifically, we investigate the impact of various method-specific parameters and query log sizes, comparing the performance of our method with traditional partitioning approaches.

2 Method Overview

Traditional approaches like Schism construct a graph representation where vertexes are tuples that participate in workload transactions. The graph is extended

---

[1] http://linkeddata.org/

[2] This work was partially supported by the Swiss National Science Foundation under contract number 200021-118000.

to include all other tuples using a partition-trained classifier. Following this idea, triples would be considered vertexes, while edges are created when any two triples participate in the same query. This is however not feasible for RDF data. Following this graph representation in our early attempts led to very dense graphs, which proved to be too large for state of the art graph partitioning software like Metis [4].[3] Next, we detail all steps seen in Figure 1 except the simpler 1^{st} phase where queries and their results are logged.

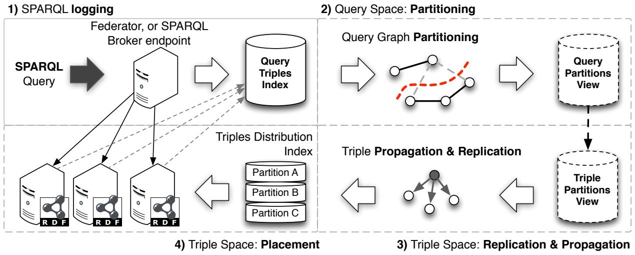

1) SPARQL **logging**
Federator, or SPARQL Broker endpoint
SPARQL Query
Query Triples Index
2) Query Space: **Partitioning**
Query Graph **Partitioning**
Query Partitions View
Triples Distribution Index
Partition A
Partition B
Partition C
Triple **Propagation & Replication**
Triple Partitions View
RDF
4) Triple Space: **Placement**
3) Triple Space: **Replication & Propagation**

Fig. 1. A simple generalized view of the partitioning process.

Data Representation & Graph Partitioning. Since mapping triples to vertexes does not scale well for RDF data, we pursued an intermediate representation: the **Queries graph**.

Each query in the workload becomes a vertex, while edges between queries are formed when some triples participate in more than one query, with the number of common triples as edge weights (Figure 2). Finally, we apply Metis on the newly formed queries graph, forcing balanced partitions as a result of the *graph-cut* operation.

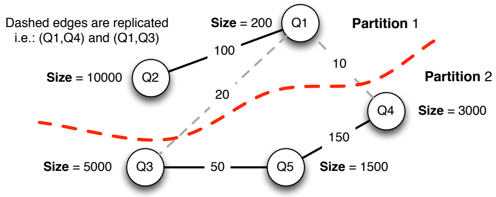


Fig. 2. Basic query log-driven data representation.

Replication. After performing the *graph-cut*, there will still be distributed joins even on the workload queries (i.e., query $Q1$ will require a distributed join while query $Q2$ not). A straight-forward solution is to replicate the triples that reside on the border between partitions. We proceed with identifying the minimum set of triples that needs to be replicated, copying the extra triples from the smaller sized query (i.e., copy extra triples from query $Q1$ over to Partition 2).

Propagation. While the process outlined so far guarantees that each query in the workload can be executed without a single distributed join there are no guarantees for future unknown queries. A method of expanding the set of all triples which participate in all workload queries is needed. For this we rely on the principle of *(Spatial) Locality of Reference* [3] adapted to the logical graph representation. In effect we *propagate* along the edges in the original RDF data graph to identify new triples "related" to existing triples which participated in all workload

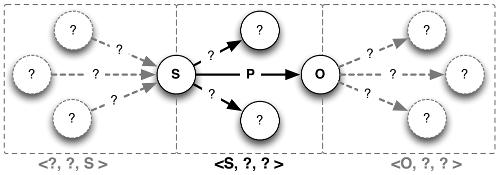


Fig. 3. Visual depiction of the propagation patterns.

---

[3] The resulting input edge file amounted to approx. 150GB on disk, crashing Metis.

queries. Hence, we perform an n-hop[4] edge propagation matching the following triple patterns given a triple <s,p,o> (also depicted Figure 3): *a*) siblings: $< s$,?,?>, *b*) outgoing edges: <o,?,?> and *c*) incoming edges: <?,?,s>. The remaining dataset triples which have not been considered so far, are randomly distributed to the K selected partitions, or by hashing by subject.

3 Results & Conclusions

We make use of the USEWOD Data challenge [1] log file to extract 400k valid and well formed SPARQL SELECT queries that produce at least 1 result, all other log entries are discarded. We use a local instance of the `Virtuoso` RDF-store to resolve them against DBpedia 3.5.1. Furthermore, we assume a *perfect* distributed query optimizer, able to find the best possible query decomposition. Measurements were conducted on a node with 72GB of RAM, 8 Cores @2.93GHz.

We compare our method against *random partitioning*, *expert (manual) partitioning*[5] and *hash partitioning*. For the latter we hash on all possible combinations of a triple: `S`, `P`, `O`, `SP`, `SO`, `PO` and `SPO`.[6] Given the small to average size of the DBpedia dataset (ca. 43.6 million triples), we fixed the number of partitions to $K = 8$, simulating a small to medium sized cluster. Furthermore, we randomly sampled the workload, with sizes consisting of 1k, 5k, 10k, 25k, 50k and 100k queries from the total of 400k logged. The number of propagation **hops** was set to 0, 1 and 2 respectively while **replication** was enabled in all cases.

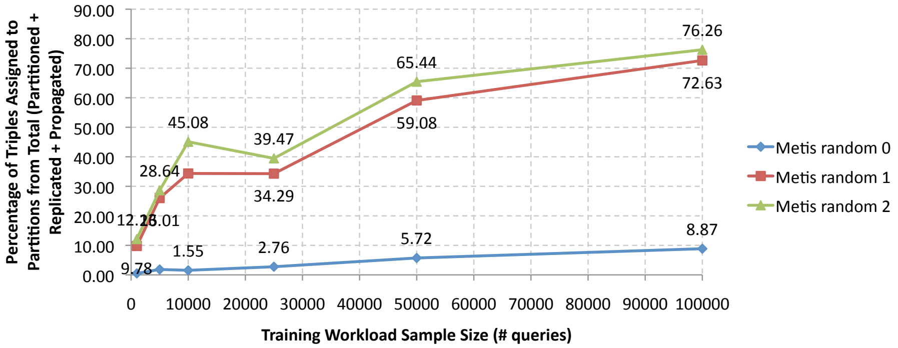


Fig. 4. The % of triples assigned to partitions from total triples, for each training query set.

As we can visually observe in Figure 4 that although the increase in number of triples reached through the partitioning process (excluding the triples not connected at all) is significant from 50k to 100k queries, there are diminishing returns as the expansion process starts to slow down. Indeed doubling the number of queries to log, yields approximatively a 10% increase at this point. Therefore, we observe that a training log size of 50k queries represents an optimal point.

Performance Impact of Graph Partitioning. Even-though the general problem of graph partitioning is known to be NP-hard, the approximating algorithm implemented in Metis performs very well, finalizing the queries graph cut in 0.17 seconds for 1k queries and 0.71 seconds for 100k queries.

---

[4] Multiple hops only enabled for in & out edges to avoid an expensive avalanche effect.

[5] Each large dump (if $> 1M$ triples) to own partition, remainder grouped together.

[6] We use of the `cityhash` family of hash functions, due to low-collision rate and speed.

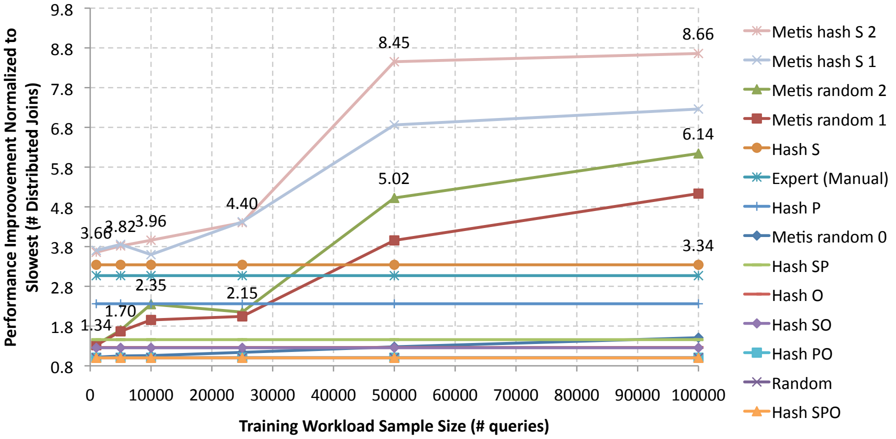


Fig. 5. The performance improvement relative to the lowest performing method (hash SPO).

Number of Hops Impact when Propagating. Figure 5 plots the relative performance improvement over the lowest performing method. Non-workload driven partitioning methods appear as horizontal lines. The worst performing ones are the *Hash SPO* method with *Random* & *Hash PO/SO* exposing similar performance levels. Hashing by subject *Hash S* is performing best, followed closely by the *Expert (manual)* distribution method. This could suggest that the majority of the workload queries are dominated by star-shaped basic graph patterns and contain few joins.[7] When using random distribution of remaining triples our method performs unsatisfactory for smaller workload sizes, but becomes substantially better by 50k queries. At 100k queries it exposes a 6.14 performance factor being 2.81 and 3.1 times better than *Hash S* and *Expert* respectively. When remaining triples are distributed based on subject hashes, the method outperforms all other methods at all workload sizes. At best (*Metis hash S 2*) our method is between 3.6 and 8.66 times better than the lowest performing and up to 5.32 times better than hashing by subject. In essence the best case partitioning would produce on average of 0.10 distributed joins per query.

References

1. B. Berendt, L. Hollink, V. Hollink, M. Luczak-Rsch, K. H. Mller, and D. Vallet. Usewod2011 - 1st international workshop on usage analysis and the web of data. in 20th international world wide web conference (www2011), hyderabad, india, 2011.
2. C. Curino, E. Jones, Y. Zhang, and S. Madden. Schism: a workload-driven approach to database replication and partitioning. *Proceedings of the VLDB Endowment*, 3:48–57, Sept. 2010.
3. P. J. Denning. The locality principle. *Communications of the ACM*, 48, July 2005.
4. G. Karypis and V. Kumar. MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0. http://www.cs.umn.edu/~metis, 2009.
5. Y. Yan, C. Wang, A. Zhou, W. Qian, L. Ma, and Y. Pan. IEEE Xplore - Efficient Indices Using Graph Partitioning in RDF Triple Stores. In *ICDE2009: IEEE 25th International Conference on Data Engineering, 2009.*, pages 1263 – 1266, 2009.

---

[7] A fact we intend to investigate in depth in the near future

Pay-as-you-go Matching of Relational Schemata to OWL Ontologies With *IncMap*[⋆]

Christoph Pinkel[1] [⋆⋆], Carsten Binnig[2], Evgeny Kharlamov[3], and Peter Haase[1]

[1] fluid Operations AG [2] University of Mannheim [3] University of Oxford

Abstract. Ontology Based Data Access (OBDA) enables access to relational data with a complex structure through ontologies as conceptual domain models. A key component of an OBDA system are mappings between the schematic elements in the ontology and their correspondences in the relational schema. Today, in existing OBDA systems these mappings typically need to be compiled by hand. In this paper we present *IncMap*, a system that supports a semi-automatic approach for matching relational schemata and ontologies. Our approach is based on a novel matching technique that represents the schematic elements of an ontology and a relational schema in a unified way. Finally, *IncMap* can extend user-verified mapping suggestions in a pay-as-you-go fashion.

1 Introduction

Today, enterprise information systems of large companies typically store petabytes of data across multiple relational databases, each with hundreds or thousands of tables (e.g., [1]). Effective understanding of complex schemata is a crucial task for enterprises to support decision making and retain competitiveness on the market. Ontology-based data access (OBDA) [2] is an approach that has recently emerged to provide semantic access to complex structured (relational) data. However, in many existing real-world systems (e.g. [2]) that follow the ODBA principle, the mappings have to be created manually, which constitutes a significant entry barrier for applying OBDA in practice.

To overcome this limitation, we propose a novel semi-automatic schema matching approach and a system called *IncMap*. We focus on finding one-to-one correspondences of ontological and relational schema elements, while we also work on extensions for finding more complex mappings.

The matching approach of *IncMap* is inspired by the Similarity Flooding (SF) algorithm [3] that works well for schemata that follow the same modeling principles. However, we show that applying the SF algorithm naively for matching relational schemata to OWL ontologies results in rather poor suggestion quality due to a conceptual mismatch between ontologies and relational schemata. The contributions of the paper are the following: In Section 2, we propose a novel graph structure called *IncGraph* to represent schema elements from ontologies and relational schemata in a unified way. In Section 3, we present our matching algorithm that supports an incremental pay-as-you-go approach that can

---

[⋆] The research was supported by the EU Commission's FP7 grant Optique (n. 318338).

[⋆⋆] E-Mail: `christoph.pinkel@fluidops.com`

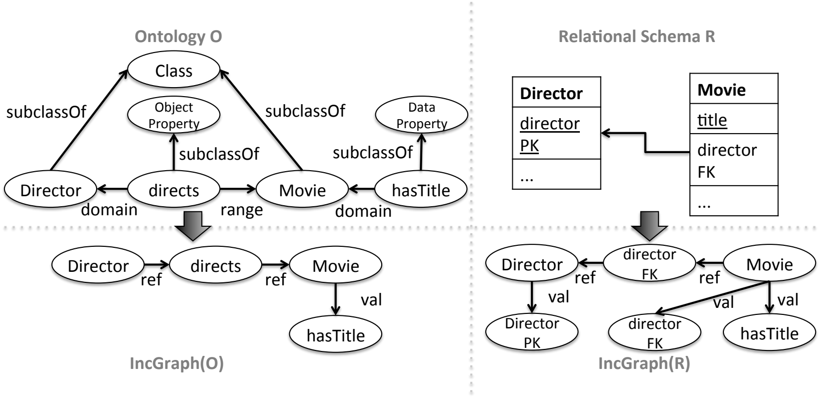


Fig. 1. *IncGraph* Construction Example

leverage existing mappings. Finally, Section 4 presents an experimental evaluation using different (real-world) relational schemata and ontologies. Experiments show that the basic version of *IncMap* reduces the effort for creating a mapping up to 20% compared to applying SF in a naive way. The incremental version of *IncMap* can reduce the total effort by another $50\% - 70\%$.

2 The *IncGraph* Model

The *IncGraph* model used by *IncMap* represents schema elements of an OWL ontology $\mathcal{O}$ and a relational schema $\mathcal{R}$ in a unified way. An *IncGraph* model is defined as directed labeled graph $(V, \texttt{Lbl}_V, E, \texttt{Lbl}_E)$. V represents a set of vertices, E a set of directed edges, $\texttt{Lbl}_V$ a set of labels for vertices and $\texttt{Lbl}_E$ a set of labels for edges. A label $l_V \in \texttt{Lbl}_V$ represents a name of a schema element whereas a label $l_E \in \texttt{Lbl}_E$ is either "ref" representing a so called `ref`-edge or "value" representing a so called `val`-edge. Figure 1 shows a cinematography related ontology $\mathcal{O}$ and relational schema $\mathcal{R}$, as well as the result of constructing graphs *IncGraph*($\mathcal{O}$) and *IncGraph*($\mathcal{R}$) according to the *IncGraph* model. While $\mathcal{O}$ and $\mathcal{R}$ describe the same entities *Directors* and *Movies* and their relationship in a different way, the *IncGraph* $\mathcal{O}$ and $\mathcal{R}$ is designed to represent both in a structurally similar fashion.

However, after constructing the *IncGraph* models, structural differences between *IncGraph*($\mathcal{O}$) and *IncGraph*($\mathcal{R}$) might still exist due to the mismatch between the high level view of the domain in ontologies and the low level view of data in relational databases. *IncMap* therefore adds *annotations* in *IncGraph* to bridge these structural gaps. Annotations are added as inactive `ref`-edges which can be activated during the schema matching process. For instance, additional `ref`-edges are added to *IncGraph* ($\mathcal{R}$) as *shortcuts* for join-paths to better match the *IncGraph* ($\mathcal{O}$). Moreover, another idea is to add *inverse* `ref`-edges to unify the structure resulting from modeling relationships in different directions (e.g., the *directs*-predicate in $\mathcal{O}$ vs. the *directorFK*-relationship in $\mathcal{R}$ in Figure 1. Finally, results from reasoning over an ontology $\mathcal{O}$ can also be integrated into *IncGraph* ($\mathcal{O}$). Analyzing these annotations in detail is a future work.

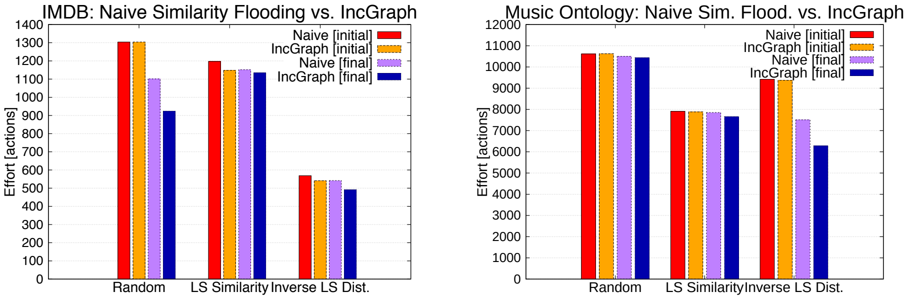


Fig. 2. Naive vs. *IncGraph*

3 The *IncMap* System

IncMap takes the *IncGraph*s produced for a relational schema $\mathcal{R}$ and for an ontology $\mathcal{O}$ as input. In its basic version, *IncMap* applies the original SF algorithm and thus creates initial mapping suggestions for the *IncGraph* of $\mathcal{O}$ and $\mathcal{R}$. Additionally, *IncMap* can activate `ref`-edges (i.e., annotations) before executing the SF algorithm to achieve better results.

One important extension is the incremental version of *IncMap*. In this version the initial suggestions are re-ranked by *IncMap* by including user feedback. The idea of user feedback is that the user confirms those mapping suggestions of the previous iteration, which are required to answer a given user query over $\mathcal{O}$.

We support three methods for incorporating user feedback into the matching process: First, the naive *Initializer* method changes the score of confirmed or rejected mappings to initialize the next run to 1.0 and 0.0, respectively. Second, *Self-Confidence Nodes* work similar but the initialization is repeated during the fix-point computation of the SF algorithm which results in a stronger influence of the user feedback. Finally, *Influence Nodes* include additional nodes in the graph structure to locally influence the score of a confirmed or rejected mappings. Please refer to [4] for a more detailed description of those methods.

IncMap is designed as a framework and provides different knobs to control which extensions and variations to use. A major avenue of future work is to apply optimization algorithms to find the best configurations automatically.

4 Experimental Evaluation

We evaluate *IncMap* using to two real-world scenarios that provided hand crafted mappings as gold standard. As a first scenario, we evaluate a mapping from movie database IMDB to the Movie Ontology (`http://www.movieontology.org`) The second scenario is a mapping from the MusicBrainz database to the Music Ontology (`www.musicontology.com`) We evaluate *IncMap* w.r.t. reducing *work time* (i.e., effort) needed to correct the correspondences suggested by *IncMap* to match the gold standard. The effort is defined as the sum of steps that users need to validate the suggested mappings for each node in the *IncGraph* ($\mathcal{O}$). For validating one mapping the user needs to reject all suggested correspondences in the decreasing order of their final ranking score until reaching the correct mapping whereas each rejection is counted as one step.

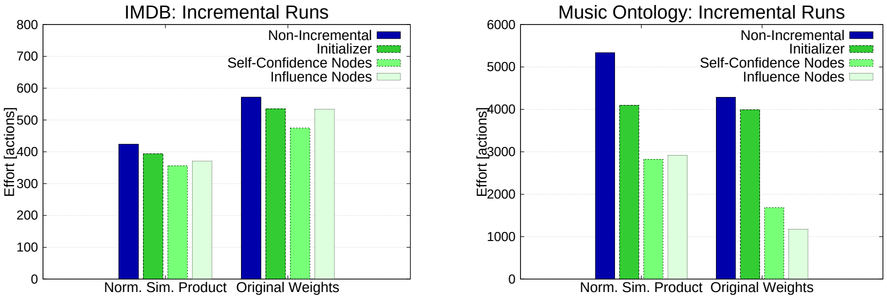


Fig. 3. Incremental Evaluation

Experiment 1 – Naive vs. IncGraph. In our first experiment we compare the work time required to correct the mapping suggestions when the schema and ontology are represented naively as schema graphs, or using *IncGraph*s. Additionally, we vary the lexical matcher using three alternatives: randomly assigned scores (base line), Levenshtein similarity and inverse Levenshtein distance. Figure 2 shows that *IncGraph* works better in all cases than the naive approach.

Experiment 2 – Incremental Mapping Generation. In the second experiment we evaluate the incremental schema matching in *IncMap*. Figure 3 show the resulting work time for the three incremental methods. Most significantly, incremental evaluation reduces the overall effort (work time) by up to $50\% - 70\%$ compared to the naive non-incremantal version. For both scenarios Self-Confidence Nodes and Influence Nodes work much better than the naive Initializer approach.

5 Conclusions and Outlook

We presented *IncMap*, a novel semi-automatic matching approach for matching relational schemata to ontologies. Our approach is based on a novel unified graph model called *IncGraph* for ontologies and relational schemata. Based on the *IncGraph* model, *IncMap* implements a novel semi-automatic matching approach inspired by the Similarity Flooding algorithm to derive mappings using both lexical and structural similarities of ontologies and relational schemata. Our experiments with *IncMap* on real-world relational schemata and ontologies showed that the effort for creating a mapping with *IncMap* is up to 30% less than using the Similarity Flooding algorithm in a naive way. The incremental version of *IncMap* reduces the total effort of mapping creation by another $50\% - 70\%$.

References

1. SAP HANA Help: http://help.sap.com/hana/html/sql_export.html (2013)
2. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The mastro system for ontology-based data access. Semantic Web Journal **2**(1) (2011) 43–53
3. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In: ICDE. (2002)
4. Pinkel, C., Binnig, C., Kharlamov, E., Haase, P.: IncMap: Pay-as-you-go Matching of Relational Schemata to OWL Ontologies. In: OM. (2013)

Interlinking Multilingual LOD Resources: A Study on Connecting Chinese, Japanese, and Korean Resources Using the Unihan Database

Saemi Jang, Satria Hutomo, Soon Gill Hong, and Mun Yong Yi

Department of Knowledge Service Engineering, KAIST, Republic of Korea
Sammy1221@kaist.ac.kr, satriahj@kaist.ac.kr, soonhong@kaist.ac.kr, munyi@kaist.ac.kr

Abstract. This study proposes a novel method with which Chinese, Japanese, and Korean (CJK) resources on the Web can be effectively matched and connected. The three countries share Chinese characters even though Japan and Korea have their own language. Utilizing the Unihan database, which covers more than 45,000 characters commonly used by the three countries, we show that the proposed method outperforms the traditional method based on string matching in finding similar characters and words used in these countries. The results represent a first step towards overcoming the multilingual barrier in semantically interlinking Asian LOD resources.

Linked Open Data (LOD) is an international endeavor to interlink structured data on the Web and create the Web of Data on a global level. Linking data can be achieved by understanding the semantic relationships between data and building explicit links for them. Hence, semantically matching and connecting resources in different languages is crucial to successfully building linked open data around the world.

Approximately 60 percent of the world population is Asians. Resolving multilingual issues for the Asian population is one of the important yet challenging tasks as Asian countries mostly use their own writing systems. Those approaches that have been developed for English alphabets and Western language systems cannot be readily adapted to Asian languages systems as their writing systems are based on different assumptions and conventions. Most of the LOD frameworks have focused on Western language resources and most of the open resources in the LOD cloud are connected to the West, significantly hampering the effort to make the LOD cloud truly a global data space.

In this study, we propose a novel method for matching and interlinking Asian LOD resources and then empirically validate the proposed method using Silk Workbench, an application developed in conjunction with the LOD2 EU-FP7 project[1]. China, Japan, and Korea, shortened as CJK, are geographically close and collectively account for the largest population in Asia. The three countries have had mutual interactions for over a thousand years influencing each other's language system. In particular, Japan and Korea have been affected by the Chinese ideographic characters (Han Chinese), which were used by the Han race a long time ago, which still has a strong impact on the Han Chinese characters used in CJK. Our work exploits the fact that these three countries share the origins and semantics of certain characters even though those characters have developed into often differently looking characters over time.

---

[1] http://lod2.eu

Our Proposed Approach.

The Unihan database is a repository for the Unicode Consortium's collective knowledge regarding the CJK Unified Ideographs. The database contains mapping data to allow conversion to and from other coded character sets and additional information about radical-stroke counts and phonetic information [1][2]. The database represents a character as a 16-bit character code and covers more than 45,000 codes. We used reading and semantic information available from the Unihan database. The reading information in Unihan database shows the pronunciations of the same unified Ideographs in China, Japan and Korea. The semantic information in Unihan includes a variety of possible alternative variants beyond the one-to-one matching of Chinese characters.

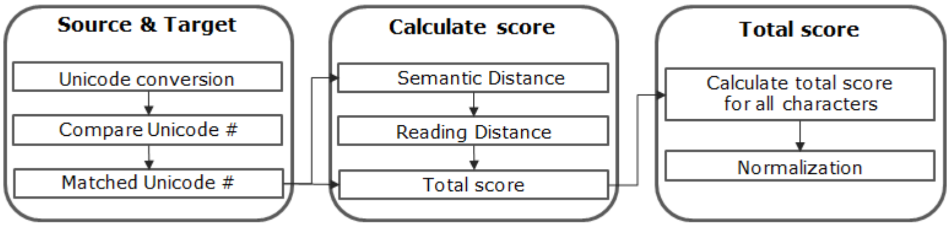


Fig. 1. Process diagram of Han edit distance

To identify matching CJK resources characters using the Unihan database, we propose a new distance measure, called Han Edit Distance (HED). Figure 1 summarizes the overall procedure for the computation of the proposed Han Edit Distance. First, the source and target Chinese words are converted into the Unicode number of each character. Then the two Unicode numbers are compared in a fixed order. It means that the Unicode number of the first character in the source word is compared with the Unicode number of the first character in the target word, the Unicode number of the second character in the source word with the Unicode number of the second character in the target word, and so on. If matching Unicode numbers are found between the characters of those two words, those matching numbers are given a score of 0. If there is not any matching, each Unicode number is converted into the Unicode number of its compatible variant properties. The next step is to check each radical stroke index properties. When the radical part of one character is the same as the other, the two characters belong to the same family. In this case, the number of different strokes is calculated. In the other case, the semantic distance (SD) is given the maximum score of 30 (which is the maximum number of strokes for common Chinese characters) and the reading distance (RD) is calculated by using the reading properties. The total distance is the minimum score between SD and RD. Finally, the Han edit distance is calculated for the two words and then it is normalized. The edit distance calculation algorithm is further detailed below.

Han Edit Distance Calculation Algorithm.

$$\mathrm{HED}\ (0, 0) = 0$$
$$\mathrm{SD}\ (i_1, i_2) = |\ \mathrm{TotalStroke}\ i_1 + \mathrm{TotalStroke}\ i_2\ |$$
$$\mathrm{RD}\ (i, j) = [\ M(i, j) + JK(i, j) + JO(i, j) + K(i, j) + H(i, j)\ ] * 6$$
$$\mathrm{HED}\ (s, t) = \min [\ \mathrm{SD}(s, t), \mathrm{RD}(s, t)\ ]$$

Han edit distance is calculated using the distance of both Semantic properties and Reading properties. When two words have the same family root, their semantic distance is calculated. Otherwise, a fixed maximum is assigned to SD, and their reading distance is calculated. Semantic distance represents the difference in the total number of different strokes from the two characters. Although the strokes for each Chinese character are different, the number of Chinese characters that have more than 30 strokes is around 0.23 percent in the Unihan database, so we defined 30 as the maximum number of the total stroke.

Reading distance mainly focuses on how characters are pronounced in each country. When each value of the reading properties is equal, 0 is given as the score; otherwise, 1 is given as the score. Then, the scores from all reading properties are added and multiplied by 6. Multiplying by 6 standardizes RD and SD because their maximum values become very close.

$$\text{Normalized Distance} = 1 - \frac{Distance}{(L_i + L_j) \times n} \quad (1)$$

Normalized distance (ND) is defined per Equation 1[3], ranging between 0 and 1 ($0 \leq ND \leq 1$). The sum of the length of elements is multiplied by 30 (n) since the maximum difference between characters is defined as 30. When the Normalized distance is 1, the two words are exactly the same. L_i denotes the length of the source word and L_j denotes the target word. Table 1 shows some examples of Han edit distance between Chinese, Japanese, and Korean words.

Table 1. Examples of Han edit distance between CJK words

	Word	Han	Meaning	Unicode#	SD	RD	HED	ND
1	國家	K	Nation	U+570B U+5BB6	0	30	0	0
	国家	C	Nation	U+56FD U+5BB6				
2	今日	J	Today	U+4ECA U+65E5	0	30	30	0.75
	今天	C	Today	U+4ECA U+5929				
3	読書	J	Reading	U+8AAD U+66F8	0	24	24	0.8
	□□	C	Reading	U+8BFB U+4E66				

Evaluation.

Korea and Japan commonly use about two thousands Chinese characters, and China commonly uses about 2500 characters. Han Chinese words can be composed of just one character or more than one characters. We evaluated the performance of the Han edit distance in two scenarios to reflect this situation. First, similarities at the character level were evaluated utilizing the most commonly used Chinese characters in CJK. 1,937 synonymous pairs were selected as a test data set for this purpose. Second, similarities at the word level were evaluated. 618 pairs of words, each pair of which has the same meaning across the three countries, were selected as a test data set for this purpose. We evaluated our approach against the Levenshtein edit distance, which is most widely used for measuring string similarities. When one character is different, the distance is 1 by the Levenshtein distance while the distance is 30 by the Han edit distance (HED). The two distance measures are compared after normalization.

Table 2. Results from the character-level comparison

Threshold =1	Chinese : Japanese			Chinese : Korean			Japanese : Korean		
	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
Levenshtein	0.5975	0.1946	0.2936	0.5919	0.1850	0.2819	0.6065	0.2525	0.3565
HED	0.5982	0.2710	0.3730	0.6477	0.2952	0.4055	0.6012	0.2739	0.3763

Table 3. Result from the word-level comparison

Threshold =1	Chinese : Japanese			Chinese : Korean			Japanese : Korean		
	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
Levenshtein	0.9726	0.3430	0.5071	0.9770	0.3414	0.5060	0.9601	0.5427	0.6934
HED	0.9705	0.4767	0.6393	0.9737	0.5958	0.7393	0.9644	0.6538	0.7793

As shown, most scores of the Han edit distance both at the character-level and at the word-level are higher than the scores of the Levensthtein distance. In particular, in all recall and F-measure comparisons, the HED approach shows superior performance consistently at the character-level and at the word-level, without exception. In the comparisons of precision, the results are mixed, but without much noticeable difference between the two approaches. On average, the f-score improvement made by the HED approach is 25% for the character-level comparison and 26% for the word-level comparison.

Concluding Remarks.

Research on LOD mainly focused on Western resources and measured the similarity of the resources at the string level. However, these approaches are not readily applicable to non-Western resources. In this study, we propose a new method to measure similarities among CJK resources, and demonstrate its effectiveness at the character-level and word-level. The results show that the proposed approach is able to identify similar resources of the three countries more effectively than the traditional Levenshtein approach. Our research represents a first step to overcoming the limitations of interlinking multilingual resources in Asia, in particular for CJK. The proposed comparator is planned to be implemented on the next version of Silk Workbench. Future research should involve expanding the approach to include other Asian countries whose characters are covered by the Unihan database such as Singapore, Taiwan, Hong Kong, and Vietnam.

Acknowledgements.

This research was conducted by the International Collaborative Research and Development Program (Creating Knowledge out of Interlinked Data) and funded by the Korean Ministry of Knowledge Economy.

References.

1. The Unicode Standard (http://www.unicode.org/versions/Unicode6.2.0/ch12.pdf).
2. Unihan database document (http://www.unicode.org/charts/unihan.html).
3. Shigeaki Kodama: String Edit Distance for Computing Phonological Similarity between Words, proceedings of International Symposium on Global Multidisciplinary Engineering, (2010).

Finite Models in RDF(S), with datatypes

Peter F. Patel-Schneider[1] and Pat Hayes[2]

[1] Nuance Communications, `pfpschneider@gmail.com`
[2] IHMC, `phayes@ihmc.us`

The details of reasoning in RDF [2] and RDFS [1] are generally well known. There is a model-theoretic semantics for RDF [3, 4] and there are sound and complete proof theories for RDF without datatypes [6]. However, the model-theoretic characteristics of RDF[3] have been less studied, particularly when datatypes are added. We show that RDF reasoning can be performed by only considering finite models or pre-models, and sometimes only very small models need be considered.

Ter Horst [6] does define Herbrand models for RDF and RDFS, providing the basis for some model-theoretic characteristics of RDF and RDFS, but he does not provide a full analysis of RDF datatypes, analyzing instead an incomplete semantics for datatypes that is easier to reason in. As well, the recent minor modifications to the semantics of RDF [4], while cleaning up some aspects of entailment in RDF, do make some technical changes that might appear to interfere with finite model-based reasoning in RDF. An analysis of finite models for RDF shows that the modified semantics does not introduce any unintended changes to reasoning in RDF. As well, it provides insights into the modeling strength of RDF, particularly when blank nodes are not present, and illustrates how finite datatypes interact with the rest of RDF.

As shown by ter Horst, sound and complete reasoning in RDF and RDFS without datatypes is decidable, even though RDF and RDFS have an infinite number of axioms. However, the decidability of RDF reasoning does not necessarily mean that RDF reasoning can be done by considering only finite models. For example, OWL [5] has decidable reasoning, but nonetheless requires infinite models, for example by encoding the number line using inverse properties and number restrictions.

In the new semantics for RDF [4], all IRIs are given denotations in all interpretations. This means that the standard construction for Herbrand models will result in an infinite model. Infinite datatypes (e.g., xsd:integer) also produce infinite models, so finite model reasoning in RDF with datatypes is technically concerned with pre-models, semantic structures that are finite and can be trivially extended to real models. Nonetheless finite model reasoning should be possible in RDF as RDF does not have inverses, counting, or even equality and inequality.

Given an RDF graph (or set of RDF graphs), we define the set of *identifiers* for the graph as the nodes and predicates of the graph plus the IRIs used in the RDF (and RDFS, if considering RDFS entailment) semantic conditions and

---

[3] In this paper, RDF by itself will generally be used to indicate both RDF and RDFS, both with and without datatypes, unless otherwise specified.

axioms, except that no container membership property not occurring in the graph is an identifier for the graph.

We then build some models for the RDF graph as follows.[4] We start with the data values for the recognized datatypes. For every identifier that is not a literal with a recognized datatype we nondeterministically either nondeterministically choose some data value as its denotation or add a new domain element as its denotation. This results in denotation functions where identifiers that do not denote data values all denote different domain elements. For IRIs and literals with unrecognized datatypes that are not identifiers of the graph we add two extra domain elements, one being the denotation of the container membership properties that are not identifiers for the graph and one being the denotation of all other identifiers. We then build up the rest of the semantic structure using the graph and the axioms and rules of inference from ter Horst augmented with axioms for datatypes and co-denoting identifiers, resulting in a structure like a datatype-aware Herbrand interpretation except that some non-literal identifiers might denote data values.

Some of these denotation functions might fail to produce an interpretation because some datatype domain or range restriction requires a domain element to be a data value for a particular datatype when it is not. However, if there is a model for the RDF graph, then this construction will produce at least one model, because there are no semantic conditions in RDF that require a particular denotation for an identifier or require or prohibit co-denotation between identifiers except those related to data values and we have not constrained data value denotations here except for non-identifiers.

So for every satisfiable RDF graph we have ended up with a set of models. We now need to show that any model of the graph is at least as strong as one of these models. For a particular set of denotations the inference rules produce the weakest possible model. Now consider the extra domain element added for unmentioned container membership properties. Replicating this domain element and splitting denotations produces a model that has the same strength as the original model because the RDF semantic conditions treat all these domain elements the same and identity cannot be detected. Similarly replicating the other additional domain element produces a model of the same strength. Because there is no inequality in RDF, identifying any two domain elements always produces model that is at least as strong, if it produces a model at all. Thus the restriction that denotations that are not data values be unique produces the weakest possible models.

In these models all the data values in a datatype that are not the denotation of some identifier have exactly the same characteristics. A pre-model can thus be constructed that collapses all these data values into one, finally resulting in finite model reasoning for RDF. (The result is, of course, not generally a model because it violates the semantic conditions on the denotation of literals.) A completely finite semantic structure can be constructed by simply ignoring these denotation mappings and the mappings for other non-identifiers.

---

[4] For purposes of space some shortcuts in notation will be taken throughout this paper.

In the absence of recognized datatypes, the above construction results in unique Herbrand models just like the ones in ter Horst. In the presence of recognized datatypes this construction is different from that in ter Horst, as it captures the full meaning of datatypes, including the requirement to consider several models. For example, consider a datatype with only two data values, say ex:two, and the RDF graph

ex:p rdf:range ex:two.
ex:a ex:p ex:u, ex:v.   ex:b ex:p ex:u, ex:w.   ex:c ex:p ex:v, ex:w.

This RDF graph entails _:x ex:p ex:u, ex:v, ex:w. To determine entailment in these situations more than one model must be considered, hence the choice of values in the graph above.

If datatypes have sufficient data values of the right kind, however, then it is possible to only consider models that are more like Herbrand models. Given a finite set of recognized datatypes D and E a subset of D, let the unconstrained portion of E be the elements of the intersection of the data spaces for each e in E that are not in any other datatype in E that is not a superset of the intersection. Consider two RDF graphs A and B and a set of recognized datatypes D. If the unconstrained portion of every E, a non-empty subset of D, is of size greater than the number of data values in it denoted by literals in A and B plus the number of identifiers in A that are not literals with recognized datatypes plus one then it is possible to always choose unconstrained elements when picking data values for identifiers that are not literals with recognized datatypes. Then all such identifiers will have different denotations, and different denotations from all literals. This in turn permits the determination of the datatypes that the denotation of an identifier must belong to by using the D* rules of ter Horst. Then when determining the denotation of an identifier, if this set is empty add a new domain element and otherwise pick an unconstrained value for this set. This results in a single, finite model that can be used for reasoning. Note, however, that the presence of even a single too-small unconstrained portion may require examining multiple models.

So we have shown that RDF reasoning can be done by considering only models of the size of the RDF graph. Is it possible to consider only very small models? (Datatypes make these considerations even more complex, so this section of the paper will ignore datatypes.) If RDF had disjunction, then it would not be possible to significantly shrink the minimum size of considered models. For example, consider RDF graphs containing n triples of the form

S_i S_1 S_i.   for $1 <= i <= n$.

In any model with less than n domain elements, there is some $1 \leq i \neq j \leq n$ such that S_i and S_j have the same denotation. In this model S_i S_1 S_j. is true and so in any such model the disjunction of all these triples is true, which is not a valid entailment.

Even with RDF lacking disjunctions, it is possible to show that very small models are not adequate. Consider RDF graphs containing triples of the form

S_i S_1 S_j .   for $1 \leq i \neq j \leq n$.

In any model with less than n domain elements, there is some $1 \leq i \neq j \leq n$ such that S_i and S_j have the same denotation, which is then related to the denotation of each of the S_i by the denotation of S_1, so the graph containing

_:x S_1 S_j.   for $1 <= i <= n$,

is true in each of these models, but this graph is not entailed. Therefore considering only models of this size or smaller is not sufficient.

If we only consider entailments with no blank nodes in the entailed graph then smaller models suffice. Consider an interpretation I (for RDF or RDFS without any recognized datatypes) containing two domain elements e_1 and e_2 that are neither properties nor classes (call these domain elements *ordinary*). Form I' from I by simply replacing e_1 and e_2 with a single domain element e throughout. Then I' is an interpretation, which can be determined by examining all the appropriate semantic conditions.

So for B_1 and B_1 identifiers whose denotation in I are neither e_1 nor e_2, I' supports any triple of the form B_1 P B_2., if and only if I supports the triple. For any particular such triple this process can be repeated until only three ordinary domain elements remain. Considering all such shrunken interpretations is adequate to rule out any invalid entailments, so we need only consider models with three ordinary domain elements, but of course we need to consider many interpretations. The ability to have such small models and to then only consider them shows how weak RDF is as a logic.

We have argued that RDF reasoning can be done by only considering finite models, even in the presence of datatypes. The exact size and number of the models that need to be considered depends on a number of factors, including which recognized datatypes are involved, but generally models of size at least the number of identifiers in the graph must be considered. If there are no datatypes or the datatypes are of sufficient size, then a single Herbrand-like model is all that need be considered. If there are no blank nodes in the entailed graph, then much smaller models suffice, although multiple models must then be considered.

References

1. Dan Brinkley and R. V. Guha. RDF vocabulary description language 1.0: RDF schema. W3C Recommendation, http://www.w3.org/TR/rdf-schema, 2004.
2. Richard Cyganiak and David Wood. RDF 1.1 concepts and abstract syntax. W3C Working Draft, http://www.w3.org/TR/rdf11-concepts, 2013.
3. Patrick Hayes. RDF Semantics. W3C Recommendation, http://www.w3.org/TR/rdf-mt/, 2004.
4. Patrick Hayes and Peter F. Patel-Schneider. RDF 1.1 Semantics. W3C Working Draft, http://www.w3.org/TR/rdf11-mt/, 2013.
5. Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 web ontology language: Structural specification and functional-style syntax. W3C Recommendation, http://www.w3.org/TR/owl2-syntax/, 2009.
6. Herman J. ter Horst. Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3(2-3):79–115, 2005.

Extending R2RML to a source-independent mapping language for RDF

Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Erik Mannens, and
Rik Van de Walle

Ghent University - iMinds - Multimedia Lab
Gaston Crommenlaan 8, bus 201, B-9050 Ledeberg-Ghent, Belgium
`firstname.lastname@ugent.be`

Abstract. Although reaching the fifth star of the Open Data deployment scheme demands the data to be represented in RDF and linked, a generic and standard mapping procedure to deploy raw data in RDF was not established so far. Only the R2RML mapping language was standardized but its applicability is limited to mappings from relational databases to RDF. We propose the extension of R2RML to also support mappings of data sources in other structured formats (indicatively CSV, TSV, XML, JSON). Broadening further its scope, the focus is put on the mappings and their optimal reuse. The language becomes source-agnostic, and resources are integrated and interlinked at a primary stage.

1 Introduction

Today, the idea of the (Linked) Open Data is widely spread and adopted. However, while reaching the fourth star of the Open Data deployment scheme[1] is easily attainable, achieving the fifth demands a well-considered approach and significantly greater effort. Current solutions are either highly customized to each case's specific needs or they follow a schematic and/or syntactic mapping approach. This fails to fully depict the semantics as it remains tied to the source file's structure. To this end, only R2RML[2] became a W3C recommendation aiming to formalize the mappings from relational databases to RDF (RDB2RDF). In practice though, one publishes data available in different source formats which, in turn, requires a more generic approach.

A generic language that maps the data independently of the source structure (*schema-agnostic*) and puts the focus on the mappings is a prominent advancement. Thereby, one deals with all different source files in a uniform way; in contrast with other languages that handle the mappings of different source formats separately. Therefore, the initial learning costs remain limited and the potential for the custom-defined mapping's reuse augments. As a result, the per-file mapping model followed so far gets surpassed, leading to contingent data integration and interlinking at a primary stage. In this paper, we propose an extension of the R2RML aiming to broaden its scope to cover also mappings from different structured data formats –CSV, TSV, XML and JSON files– to RDF.

---

[1] http://5stardata.info

[2] http://www.w3.org/TR/r2rml

2 State of the art

Beyond R2RML which has already several implementations[3], other RDB2RDF mapping languages were defined [1]. In the same context, there are corresponding languages to support CSV-to-RDF mappings (CSV2RDF), e.g., the XLWrap's mapping language [2], the Mapping Master's M2 [3] and Vertere[4]. On the other hand, in the case of mappings from XML to RDF (XML2RDF), the different tools rely mostly on existing XML solutions. To be more precise, XSLT-based approaches were explored, as the Krextor [4] and the AstroGrid-D[5] mapping tools, while other implementations deploy mappings using XPath and XQuery, e.g., the Tripliser[6] and the XSPARQL [5]. These solutions for XML sources lead to mappings on the syntactic level rather than on the semantic level or fail to provide a solution applicable to a broader domain. Beyond the standard Extract-Map-Load (EML) mappings, dynamic query translation was also explored, e,g, in the case of Tarql[7] (CSV2RDF) and XSPARQL (mapping and integration of XML, RDB and RDF resources).

In general, most tools deploy mappings from a certain source format to RDF (*source-centric approaches*). There are only a few tools that provide mappings from various source formats to RDF –DataLift [6], the DataTank [7], Karma [8], Open Refine[8] and Virtuoso Sponger[9] are the most well known– but only the DataTank uses a mapping language. For the latter's needs, Vertere was extended not only to cover CSV2RDF mappings but mappings from other structured data sources as well, namely databases, XML and JSON. Since R2RML became a W3C standard and due to its analogous nature to Vertere, the extension of R2RML is considered a prominent solution and its applicability verified.

3 Extending R2RML for a more generic use

An extension of the R2RML language is proposed, aiming to broaden its scope beyond RDB2RDF mappings, to cover every structured data format (a *Global-As-View approach*), and to address the limitations of existing languages. The R2RML's RDF graphs are used to express mappings independently of the source format. Therefore, the same custom mappings are reused whether the source files are in the same format or not, only by redetermining the references to the source values to be mapped, as the expected custom mapping definitions remain the same. The vocabulary extending the R2RML is available at `http://mmlab.be/users/andimou/rml.ttl`. The expansion is achieved as follows:

---

[3] http://www.w3.org/2001/sw/rdb2rdf/wiki/Implementations

[4] https://github.com/knudmoeller/Vertere-RDF

[5] http://www.gac-grid.de/project-products/Software/XML2RDF.html

[6] http://daverog.github.io/tripliser/

[7] https://github.com/cygri/tarql

[8] http://openrefine.org/

[9] http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtSponger

Extending RDF triples mapping. *Triples map* is extended not only to map each row in the logical table, but each resource in the logical source. To this end, the `rr:logicalTable` and `rr:tableName` become a sub-property of `rml:logicalSource` and `rml:sourceName` respectively, while `rr:elementName` for XML sources and `rr:objectName` for JSON sources are introduced. In the example, *books* is a logical table's, a JSON object's or an XML element's name.

```
<#RDB_CSV_map> rml:logicalSource [ rr:tableName "BOOKS" ];
  rr:subjectMap [ rr:template "http://data.example.com/books/{ISBN}" ];
  rr:predicateObjectMap [ rr:predicate ex:id; rr:objectMap [ rml:resource "ID" ] ].
<#XML_map> rml:logicalSource [ rml:elementName "/books" ];
  rr:subjectMap [ rr:template "http://data.example.com/books/{book/ISBN}"];
  rr:predicateObjectMap [ rr:predicate ex:id; rr:objectMap [ rml:resource "book/ISBN@id" ] ].
<#JSON_map> rml:logicalSource [ rml:objectName "books" ];
  rr:subjectMap [ rr:template "http://data.example.com/books/{book.ISBN}"];
  rr:predicateObjectMap [ rr:predicate ex:id; rr:objectMap [ rml:resource "book.id" ] ].
```

Extending resources' mapping. In the same context, *term maps* are extended to generate RDF terms from any logical resource, either this is a table row, an XML element or a JSON object. The *column-valued term map* is extended to cover every *resource term map*. Therefore, the R2RML's `rr:column` property becomes a sub-property of the `rml:resource` which is a valid column name for relational databases and CSV files, a valid XPath expression for an XML node's or attribute's absolute path and a valid path pattern in JavaScript syntax for objects in JSON source files, as in the aforementioned example.

Multiple entities per row. Most of the mapping languages (including R2RML) follow the *entity-per-row* model and consider that each row's RDF triples are mapped to the same subject. In its extended version, R2RML can map sets of columns to different subjects, which are then related among each other with a *predicate-object triples map*. For example, a row may have several columns with information about an event and a few of them refer to its location, e.g., latitude and longitude. Using this single row a *triples map* may be defined for the event while another *triples map* may be defined for the location where the event takes place (this mapping definition might be reused for other locations' mapping) and the two of them are related with a *predicate-object triples map*, as in the following example:

```
<#Event_map> rml:logicalSource [ rml:elementName "/events" ];
  rr:subjectMap [ rr:template "http://data.example.com/events/{event/id}" ];
  rr:predicateObjectMap
    [ rr:predicate ex:location; rr:objectMap [ rr:parentTriplesMap <#Location_map> ] ] ,
    [ rr:predicate ex:transport; rr:objectMap [ rr:parentTriplesMap <#Transport_map> ;
      rr:joinCondition [ rr:child "event/bus/num"; rr:parent "BUS_NUM" ] ] ].
<#Location_map> rml:logicalSource [ rml:elementName "/events" ];
  rr:subjectMap [
  rr:template "http://data.example.com/location/{event/location/lat},{event/location/long}"].
<#Transport_map> rml:logicalSource [ rr:tableName "TRANSPORTATIONS" ];
  rr:subjectMap [ rr:template "http://data.example.com/transport/{TYPE}/{BUS_NUM}"];
  rr:predicateObjectMap [ rr:predicate ex:name; rr:objectMap [ rml:resource "BUS_NAME" ] ].
```

Extended the logical sources. According to the `rr:sqlQuery`, the `rml:xmlQuery` is adapted and both are sub-properties of `rml:query` to serve a query against a source file. In the same context the `rml:queryLanguage` is defined to determine which language is used (indicatively, a W3C standard in the case of XML).

Integrated mapping. Extending the *reference object map*, one can use the subjects of another *triples map* as the objects generated by a *predicate-object map*. Since the *triples maps* may be based on different logical sources, the potential to create triples based on integrated sources emerges. At the aforementioned event example, an element node may refer to the number of the bus going to the event location, but the bus names are associated to the bus numbers at a separate table which is mapped by another *triples map*. The mappings of both of them are defined and a *predicate-object terms map* may be used to relate them.

4 Conclusions and Future Work

A generic mapping language is proposed to handle the mappings from different source formats to RDF. The uppermost goal of such an extension is to keep the focus on the mappings to be expressed rather than on the data and their original structure. With this work, we bring into discussion its feasibility, possible barriers and aspects that should be taken into consideration. In the future the arising generic mapping language will be used at the DataTank, instead of Vertere, to cover mappings from different source formats to RDF and, in the same time, to confront with the standard mapping language for the RDB2RDF mappings.

References

1. Hert, M., Reif, G., Gall, H.C.: A comparison of RDB-to-RDF mapping languages. In: Proceedings of the 7th International Conference on Semantic Systems. I-Semantics '11, New York, NY, USA, ACM (2011) 25–32
2. Langegger, A., Wöß, W.: XLWrap – Querying and Integrating Arbitrary Spreadsheets with SPARQL. In: Proceedings of the 8th International Semantic Web Conference. ISWC '09, Berlin, Heidelberg, Springer-Verlag (2009) 359–374
3. O'Connor, M.J., Halaschek-Wiener, C., Musen, M.A.: Mapping Master: a flexible approach for mapping spreadsheets to OWL. In: Proceedings of the 9th International Semantic Web Conference on The Semantic Web - Volume Part II. ISWC'10, Berlin, Heidelberg, Springer-Verlag (2010) 194–208
4. Lange, C.: Krextor - an extensible framework for contributing content math to the Web of Data. In: Proceedings of the 18th Calculemus and 10th international conference on Intelligent computer mathematics. MKM'11, Berlin, Heidelberg, Springer-Verlag (2011) 304–306
5. Bischof, S., Decker, S., Krennwallner, T., Lopes, N., Polleres, A.: Mapping between rdf and xml with xsparql. Journal on Data Semantics **1**(3) (2012) 147–185
6. Scharffe, F., Atemezing, G., Troncy, R., Gandon, F., Villata, S., Bucher, B., Hamdi, F., Bihanic, L., Képéklian, G., Cotton, F., Euzenat, J., Fan, Z., Vandenbussche, P.Y., Vatant, B.: Enabling Linked Data publication with the Datalift platform. In: Proc. AAAI workshop on semantic cities, Toronto, Canada (2012)
7. Vander Sande, M., Colpaert, P., Van Deursen, D., Mannens, E., Van de Walle, R.: The DataTank: an open data adapter with semantic output. In: 21st International Conference on World Wide Web, Proceedings. (2012)
8. Gupta, S., Szekely, P., Knoblock, C., Goel, A., Taheriyan, M., Muslea, M.: Karma: A system for mapping structured sources into the Semantic Web. In: 9th Extended Semantic Web Conference (ESWC2012). (May 2012)

PigSPARQL: A SPARQL Query Processing Baseline for Big Data

Alexander Schätzle, Martin Przyjaciel-Zablocki,
Thomas Hornung, and Georg Lausen

Department of Computer Science, University of Freiburg
Georges-Köhler-Allee 051, 79110 Freiburg, Germany
`schaetzle|zablocki|hornungt|lausen@informatik.uni-freiburg.de`

Abstract. In this paper we discuss PigSPARQL, a competitive yet easy to use SPARQL query processing system on MapReduce that allows adhoc SPARQL query processing on large RDF graphs out of the box. Instead of a direct mapping, PigSPARQL uses the query language of Pig, a data analysis platform on top of Hadoop MapReduce, as an intermediate layer between SPARQL and MapReduce. This additional level of abstraction makes our approach independent of the actual Hadoop version and thus ensures the compatibility to future changes of the Hadoop framework as they will be covered by the underlying Pig layer. We revisit PigSPARQL and demonstrate the performance improvement when simply switching the underlying version of Pig from 0.5.0 to 0.11.0 without any changes to PigSPARQL itself. Because of this sustainability, PigSPARQL is an attractive long-term baseline for comparing various MapReduce based SPARQL implementations which is also underpinned by its competitiveness with existing systems, e.g. HadoopRDF.

1 Introduction

Today, *MapReduce* has been widely adopted in manifold application fields, especially in the broad area of Big Data, with *Hadoop* being the most prominent open source implementation. Though node efficiency is known to be rather poor, its success is mainly attributed to the inherent high degree of parallelism, robustness, reliability and excellent scalability properties while running on cheap and heterogeneous commodity hardware. Furthermore, new nodes can be added to the system on demand seamlessly at runtime.

Driven by the *Semantic Web* and *Linked Open Data*, new challenges with regard to SPARQL query evaluation arise and scalability becomes an issue as RDF datasets continuously grow in size, exceeding the capabilities of state of the art non-distributed RDF triple stores [2]. The wide spread adoption of MapReduce makes it an interesting candidate for distributed SPARQL processing on large RDF graphs, especially for rather costly queries involving several joins that cannot be executed in real-time at web-scale and hence need to be processed offline. However, existing approaches in this direction are often accompanied by proof-of-concept implementations that are hard to deploy or not compatible with newer versions of Hadoop, do not run out of the box or they are even not available for download at all. Moreover, they often support only a small subset of SPARQL, usually basic graph patterns. All this hampers the comparison of different approaches as evaluation results are hard to reproduce and a comprehensive evaluation becomes very cumbersome and time consuming.

In this paper we first revisit *PigSPARQL*[1], a mapping from SPARQL to the query language of *Pig* [4], that was originally presented in [6]. PigSPARQL is easy to use without complicated deployment, installation or configuration. By using Pig Latin as an intermediate layer of abstraction between SPARQL and MapReduce, the mapping is automatically compatible to future versions of Hadoop (including major changes like the new YARN framework) while it benefits from further developments and optimizations of Pig without having to change a single line of code since the query language of Pig is kept backward compatible. This is confirmed by experiments that are presented in short in this paper (cf. Section 3). Switching the version of Pig from 0.5.0 to 0.11.0 improved the query execution times by up to one order of magnitude, while no adaptations of PigSPARQL were required. Because of this feature of sustainability, PigSPARQL is an attractive long-term baseline for comparing various MapReduce based SPARQL implementations. This is also underpinned by PigSPARQL's competitiveness with existing systems like HadoopRDF and others (cf. Section 3).

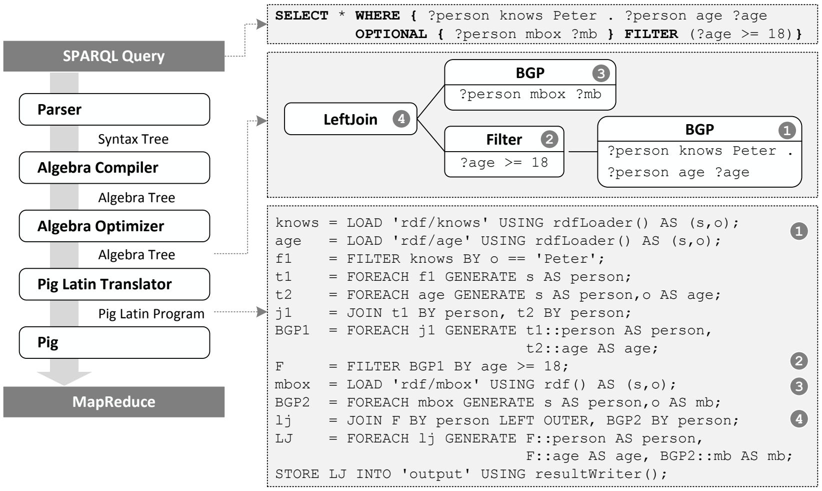
2 PigSPARQL Architecture

Pig is a data analysis platform on top of Hadoop with a fully nested data model, complemented by a comprehensive imperative query language (Pig Latin) that gives us a simple level of abstraction from the procedural model of MapReduce by providing relational style operators like filters and joins which are not available in MapReduce out of the box. We represent an RDF triple in the data model of Pig as a tuple of three atomic fields with schema (`s`, `p`, `o`). As we do not require any preprocessing and RDF triples are converted into the data model of Pig on the fly, PigSPARQL is particularly suited for ad-hoc query processing, e.g. for ETL like scenarios where we do not want to build up a costly index structure in advance. In a typical SPARQL query the predicate of a triple pattern is usually bounded. Hence, PigSPARQL also supports optional vertical partitioning of the dataset as an additional preprocessing step.

Our mapping of SPARQL to Pig Latin follows a common design principle based on an algebraic representation of SPARQL expressions (cf. Figure 1). First, a SPARQL query is parsed to generate an abstract syntax tree which is then translated into a SPARQL algebra tree. Next, we apply several optimizations on the algebra level like the early execution of filters and a rearrangement of triple patterns by selectivity. Finally, we traverse the optimized algebra tree bottom up and generate for every SPARQL algebra operator an equivalent sequence of Pig Latin expressions. At runtime, Pig automatically maps the resulting Pig Latin script into a sequence of MapReduce iterations. More details are given in [6].

---

[1] See `http://dbis.informatik.uni-freiburg.de/PigSPARQL` for download.



```
SELECT * WHERE { ?person knows Peter . ?person age ?age
         OPTIONAL { ?person mbox ?mb } FILTER (?age >= 18)}
```

```
knows = LOAD 'rdf/knows' USING rdfLoader() AS (s,o);        ①
age   = LOAD 'rdf/age' USING rdfLoader() AS (s,o);
f1    = FILTER knows BY o == 'Peter';
t1    = FOREACH f1 GENERATE s AS person;
t2    = FOREACH age GENERATE s AS person,o AS age;
j1    = JOIN t1 BY person, t2 BY person;
BGP1  = FOREACH j1 GENERATE t1::person AS person,
                            t2::age AS age;
F     = FILTER BGP1 BY age >= 18;                           ②
mbox  = LOAD 'rdf/mbox' USING rdf() AS (s,o);               ③
BGP2  = FOREACH mbox GENERATE s AS person,o AS mb;
lj    = JOIN F BY person LEFT OUTER, BGP2 BY person;        ④
LJ    = FOREACH lj GENERATE F::person AS person,
                            F::age AS age, BGP2::mb AS mb;
STORE LJ INTO 'output' USING resultWriter();
```

Fig. 1. PigSPARQL workflow from SPARQL to MapReduce

3 Experiments

Most of the published MapReduce based approaches are proof-of-concept implementations, which are neither well documented nor running out of the box, nor are available for public. Evaluation of such systems is time consuming and easily leads to inexplicable results. This hampers the comparability of different proposed solutions which is a key driver for further development. To introduce a stable basis for comparison, we suggest PigSPARQL as an easy to use baseline for SPARQL query processing with MapReduce because of the following reasons:

1. PigSPARQL is a reliable and stable system as it uses Pig as an intermediate layer which is widely-used and maintained by Yahoo! Research. Pig's processing framework is fairly competitive and continuously optimized and enhanced with new features. This is confirmed by Figure 2.a that shows exemplary the runtime improvement of PigSPARQL for SP^{2}Bench Query 2 between Pig 0.5.0 and Pig 0.11.0 where we can observe a speed up by an order of magnitude without changing a single line of code - other queries exhibit a similar behavior as can be expected because of PigSPARQL's architecture.
2. For a comprehensive evaluation of different systems, they should be installable and usable within a reasonable effort, without the need of tricky configurations. In the context of such evaluations, PigSPARQL is very attractive. The LUBM evaluation of HadoopRDF [3], for example, took us several weeks including an exhaustive troubleshooting whereas the same evaluation with PigSPARQL was done in only one day.
3. We evaluated the competitiveness of PigSPARQL with respect to three other SPARQL engines based on MapReduce by using LUBM, as some of these systems only support basic graph patterns: (1) *HadoopRDF* [3] is an advanced

SPARQL engine that utilizes a cost-based execution plan for reduce-side joins. (2) *MAPSIN* [5] is a map-side index nested loop join based on HBase. (3) *Merge Join* [1] is a MapReduce adoption of merge joins for SPARQL basic graph patterns. Figure 2.b illustrates the execution times for LUBM Query 4 distinguishing between n-way and 2-way join execution, if supported. PigSPARQL shows a competitive runtime performance while scaling smoothly when increasing the size of the dataset. MAPSIN performs a bit faster, however it uses a sophisticated storage schema based on HBase that works well for selective queries but decreases significantly in performance for less selective ones [5]. All approaches need a rather time consuming initial preprocessing of up to several hours compared to the vertical partitioning of PigSPARQL, which took less than 14 minutes for 1.6 billion triples.

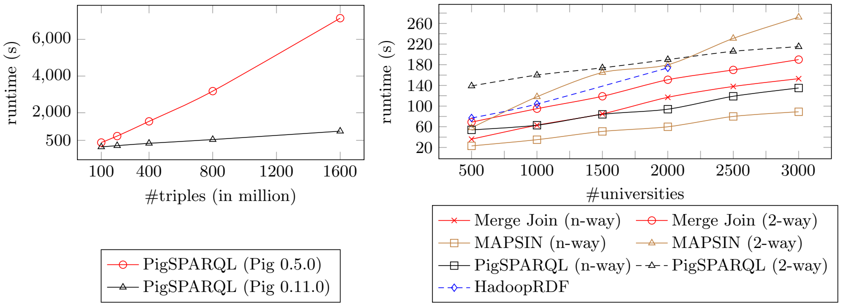


Fig. 2. (a) SP^{2}Bench Query 2. (b) LUBM Query 4.

Conclusion. PigSPARQL is an easy to use and competitive baseline for the comparison of MapReduce based SPARQL processing. With the support of SPARQL 1.0, it already exceeds the functionalities of most existing research prototypes. For future work, we plan to add support for additional SPARQL 1.1 features.

References

1. (2013), `http://dbis.informatik.uni-freiburg.de/forschung/projekte/DiPoS/`
2. Huang, J., Abadi, D.J., Ren, K.: Scalable SPARQL Querying of Large RDF Graphs. PVLDB 4(11), 1123–1134 (2011)
3. Husain, M.F., et al.: Heuristics-Based Query Processing for Large RDF Graphs Using Cloud Computing. IEEE TKDE 23(9) (2011)
4. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig Latin: A Not-So-Foreign Language for Data Processing. In: SIGMOD. pp. 1099–1110 (2008)
5. Schätzle, A., Przyjaciel-Zablocki, M., et al.: Cascading Map-Side Joins over HBase for Scalable Join Processing. In: SSWS+HPCSW. p. 59 (2012)
6. Schätzle, A., Przyjaciel-Zablocki, M., Lausen, G.: PigSPARQL: Mapping SPARQL to Pig Latin. In: Proc. SWIM. pp. 4:1–4:8 (2011)

Discoverability of SPARQL Endpoints in Linked Open Data

Heiko Paulheim[1] and Sven Hertling[2]

[1] University of Mannheim, Germany
Research Group Data and Web Science
`heiko@informatik.uni-mannheim.de`
[2] Technische Universität Darmstadt
Knowledge Engineering Group
`hertling@ke.tu-darmstadt.de`

Abstract. Accessing Linked Open Data sources with query languages such as SPARQL provides more flexible possibilities than access based on derefencerable URIs only. However, discovering a SPARQL endpoint on the fly, given a URI, is not trivial. This paper provides a quantitative analysis on the automatic discoverability of SPARQL endpoints using different mechanisms.

1 Introduction

Query languages such as SPARQL provide efficient ways of accessing Linked Open Data sources. In his *design issues* document from 2006, Tim Berners-lee states that "to make the data be effectively linked, someone who only has the URI of something must be able to find their way the SPARQL endpoint." [2].

However, automatically discovering the SPARQL endpoint for a given resource is still not a trivial problem. Approaches to solve that problem include:

- Standardized vocabularies for describing datasets, such as *VoID*, where the descriptions are provided at URLs that can be canonically derived from a URI [1], and
- Catalogs of datasets such as *datahub*[1] or the LATC data source inventory[2], which can be queried for a SPARQL endpoint with a given URI.

In this paper, we explore the success rates of those strategies using a large representative sample of URIs in Linked Open Data, and discuss the results. Furthermore, we propose a simple, multi-strategy resolution service which delivers SPARQL endpoints for URIs.

2 Strategies for Discovering SPARQL Endpoints

We examine two basic strategies for discovering SPARQL endpoints from a URI: trying to retrieve VoID descriptions, and leveraging external catalogs of datasets.

---

[1] `http://datahub.io/`

[2] `http://dsi.lod-cloud.net/`

2.1 Retrieving VoID Descriptions

The VoID specification [1] recommends to use the RFC 5785 standard [5] for publishing discoverable VoID descriptions of a dataset. This means that a URI of the form `http://hostname/.well-known/void` is to be used for publishing VoID descriptions. Although the specification states that the `/.well-known/void` path segment should be located at the root level, i.e., directly follow the host name part of the URI, our experiments have shown that it is sometimes located at deeper locations. Thus, we use the following approach for trying to discover VoID vocabularies:

> Given a URI, remove the portion after the last slash (/), and append `.well-known/void`. If no VoID description is found at that location, and there are segments left after the host name, continue from the start.

For example, given the URI `http://www.example.org/data/xyz`, we would try the following URLs for retrieving a VoID description, using the VoID [1] and Provenance [3] vocabularies:

1. `http://www.example.org/data/.well-known/void` and
2. `http://www.example.org/.well-known/void`,

assuming that the first URL does not return a VoID description.

As a second strategy to retrieving VoID descriptions, we retrieve the RDF dataset from the (dereferencable) sample URI, and look for one of the following axioms:

1. `?x void:inDataset ?d`
2. `?x prv:containedBy ?d`[3]

Although, in the literature, means other than VoID descriptions have been proposed to link data to SPARQL endpoints [4], we do not expect them to be too widely spread, since they are not backed by a standardization document.

2.2 Leveraging External Catalogs

Catalogs of datasets, such as *datahub*, list datasets as well as their metadata, including SPARQL endpoints, if applicable. For our prototype, we use the *datahub catalog*, which lists data sets as well as their SPARQL endpoints. Similar searches could be issued on any catalogs of Linked Open Data.

We have implemented all those strategies in the *SEnF* (SPARQL Endpoint Finder) service, a simple web service which can be used to retrieve SPARQL endpoints for a URI.[4]

---

[3] We do not demand that `?x` is connected to `<URI>`, e.g., by a `rdfs:definedBy` statement, in order to make this approach as versatile as possible, and since we assume that a VoID description linked from a dataset will in most cases be the description of that dataset, and not of another one.

[4] `http://tinyurl.com/sparqlsenf`

Table 1. Results on different strategies for finding SPARQL endpoints on 10,000 random URIs, reporting both the number of URIs for which *any* SPARQL endpoint was found, as well as the number of URIs for which a *valid* SPARQL endpoint was found. The numbers in parantheses denote the total number of endpoints found.

Strategy	Datahub Catalog	`/.well-known/void` (all)	`/.well-known/void` (standard)	Link to VoID
# found	7,389 (26,124)	110 (392)	94 (288)	9 (9)
# valid	1,375 (2,978)	53 (106)	53 (72)	0 (0)

3 Quantitative Analysis

We have tested the approaches discussed above on a random sample of 10,000 subjects in the 2012 billion triple challenge dataset,[5] which we deem a representative sample of Linked Open Data in the wild. Out of those 10,000 URIs, 8,893 were dereferencable.

For each endpoint retrieved by any strategy, we have checked the correctness of the result by issuing a query of the form `ASK {<URI> ?r ?x}` at the endpoint, and consider the returned endpoint as a valid result if `TRUE` is returned upon the query.

Table 1 shows the results of our evaluation. The first observation is that in many cases and by most strategies, more than one endpoint is returned, which shows that there is some redundancy in terms of SPARQL endpoints (i.e., more than one endpoint may contain information on a resource).

The main observation is that using external catalogs clearly outperforms other methods in terms of coverage, being able to locate endpoints for 74% of all URIs. However, only in 14% of the cases, at least one of the retrieved endpoints[6] was online during our experiment[7] and actually contains data about the resource in question, which also demonstrates the limitations of the approach.

The approaches using VoID and the provenance vocabulary are still not adopted on a large scale, thus, the coverage of those approaches is much lower. On the other hand, the data found by following `/.well-known/void` is much more precise than those delivered by catalogs, showing a precision of 0.48 (in contrast to 0.19 for the catalog based approach). The approach looking for direct links to VoID descriptions provided information on endpoints in some cases, however, the SPARQL statement for checking the validity of the endpoint failed in those nine cases because the original URI was redirected, and the redirect URI, which pointed to the dataset, not the resource, was not found in the endpoint.

Furthermore, we can observe that there is a deviation between the standard specification for providing VoID descriptions (i.e., providing them at the server's root directory), and the actual deployment (in some cases, they are located at deeper levels). This may hint at a practical problem with implementing the

---

[5] `http://km.aifb.kit.edu/projects/btc-2012/`

[6] In some cases, more than one endpoint is retrieved.

[7] Carried out between July 22nd and July 23rd, 2013

standard, i.e., hosting data sets on servers for which the authority providing the data set does not have root access rights.

It is further remarkable that for no URI in our sample, an endpoint could be retrieved by every strategy. This shows that there is a need to use multiple strategies in parallel, like our implementation of the *SEnF* service does.

4 Conclusion

The capability of locating SPARQL endpoints for a given URI has been stated as a desired property of Linked Open Data. In this paper, we have evaluated several strategies for performing that *URI-to-endpoint* resolution, based on a large random sample of the Billion Triple Challenge Dataset.

Approaches using proposed methods such as VoID and the provenance vocabulary are scarcely in use (and sometimes not implemented according to the specification), they lead to a valid SPARQL endpoint in less than 1% of all cases. That finding means that catalogs are essential for discovering SPARQL endpoint, at least in the short and medium term. However, although performing better than the approaches mentioned before, catalogs also do not provide information in sufficient quality at the time being.

Overall, we were not able to locate suitable SPARQL endpoints in most of the cases – for more than 85% of all URIs, no SPARQL endpoint could be found. The reasons may be two-fold: (i) it is not possible to discover the endpoints with the methods described in this paper, or (ii) no such endpoints exist. While in many cases, the latter case is likely (e.g. for single FOAF documents at websites, or blogging software that publishes RDF(a), but does not provide a SPARQL endpoint), it is beyond the scope of this paper (if not completely infeasible due to the open world assumption) to make a statement about the actual availability of SPARQL endpoints for Linked Open Data URIs.

Our evaluation has furthermore shown that no single strategy outperforms all other strategies. Thus, for practical purposes, using multi-strategy approaches such as the SEnF service is the most suitable way for discovering endpoints. Since the SEnF service follows a modular architecture, new catalogs and/or resolution strategies may be plugged in as they become available and/or standardized.

References

1. Keith Alexander, Richard Cyganiak, Michael Hausenblas, and Jun Zhao. Describing Linked Datasets with the VoID Vocabulary. `http://www.w3.org/TR/void/`.
2. Tim Berners-Lee. Linked Data. `http://www.w3.org/DesignIssues/LinkedData.html`.
3. Olaf Hartig and Jun Zhao. Provenance Vocabulary Core Ontology Specification. `http://trdf.sourceforge.net/provenance/ns.html`.
4. Kjetil Kjernsmo. The necessity of hypermedia RDF and an approach to achieve it. In *Proceedings of the First Linked APIs Workshop*, 2012.
5. Mark Nottingham and Eran Hammer-Lahav. RFC 5785 – Defining Well-Known Uniform Resource Identifiers (URIs). `http://tools.ietf.org/html/rfc5785`.

RDFChain: Chain Centric Storage for Scalable Join Processing of RDF Graphs using MapReduce and HBase

Pilsik Choi[1,2 *], Jooik Jung[1] and Kyong-Ho Lee[1]

[1]Department of Computer Science, Yonsei University, Seoul, Republic of Korea
pschoi@icl.yonsei.ac.kr, jijung@icl.yonsei.ac.kr,
khlee@cs.yonsei.ac.kr
[2]Mobile Communication Division, Samsung Electronics
pilsik.choi@samsung.com

Abstract. As a massive linked open data is available in RDF, the scalable storage and efficient retrieval using MapReduce have been actively studied. Most of previous researches focus on reducing the number of MapReduce jobs for processing join operations in SPARQL queries. However, the cost of shuffle phase still occurs due to their reduce-side joins. In this paper, we propose RDFChain which supports the scalable storage and efficient retrieval of a large volume of RDF data using a combination of MapReduce and HBase which is NoSQL storage system. Since the proposed storage schema of RDFChain reflects all the possible join patterns of queries, it provides a reduced number of storage accesses depending on the join pattern of a query. In addition, the proposed cost-based map-side join of RDFChain reduces the number of map jobs since it processes as many joins as possible in a map job using statistics.

Keywords: Map-side join, chain centric storage, HBase, NoSQL, RDF, SPARQL, MapReduce, Hadoop

1 Introduction

As an enormous amount of Linked Data is available, processing a SPARQL query into a massive RDF dataset becomes a challenging task when scalability and performance issues are taken into consideration. Progress in many researches into SPARQL query processing has been made with the use of MapReduce, a distributed parallel processing framework. In particular, Hadoop[1] is the most popular open source version of MapReduce. Particularly, the conventional MapReduce-based join processing methods are divided into two approaches: reduce-side join and map-side join [1]. Map-side join outperforms reduce-side join since the shuffle and reduce phases of reduce-side join are not required. However, map-side join requires the datasets to be equally partitioned by join keys. It is not just non-trivial but demanding for the condition to be met in the case of multi-way joins. Przyjaciel-Zablocki et al. [2] have pro-

---

[1] http://hadoop.apache.org.

posed the Map-Side Index Nested Loop Join (MAPSIN) using HBase[2], which is a distributed, scalable and column-oriented NoSQL storage. HBase is well suited for random access due to the sparse multidimensional sorted map, and is also appropriate for a data model that requires processing row keys such as table scans and lookups. MAPSIN provides an optimized algorithm for reducing the number of storage access for star pattern joins, but not for chain pattern joins. Therefore, we propose RDFChain with the following contributions:

- RDFChain reflects every possible join patterns in its storage schema. Specifically, the proposed chain centric storage, which reflects relations among the subjects and objects of RDF triples, reduces the number of storage access.
- RDFChain reduces the number of map jobs in multi-way joins. RDFChain estimates the cost of join processing using statistics to split a query. The queries separated include as many triple patterns (TPs) as possible to be processed in a map job. Thus, RDFChain processes as many joins as possible in a single map job.

2 Proposed Architecture

RDFChain consists of two components, the data loading component and the query processing one. RDF triples are converted into an N-triple format which is natively supported by Hadoop and then loaded using map jobs. At this stage, we employ the bulk load of HBase instead of directly putting every triple into a table. We also create all the statistics [3] required by our join execution.

2.1 Storage Design

The proposed method stores RDF triples as follows:

- RDF triples with the terms that co-exist in both the subject and object parts of triples are located in the T_{com} table. A RDF triple with the term as a subject is considered as a Subject-Predicate-Object (SPO) triple. A RDF triple with the term as an object correspond to Object-Predicate-Subject (OPS).
- SPO triples which are not located in T_{com} are stored in T_{spo}.
- OPS triples which are not located in T_{com} are stored in T_{ops}.

If a term is used not only as a subject but also as an object in triples, it would be a row key in T_{com}. T_{com} has two column-families to represent OPS and SPO schemas. A predicate comes to a column. The subject and object terms become the values of the corresponding columns for OPS and for SPO, respectively. A subject may have several predicates and a predicate may also have several objects. This means that triples are stored in a triple group by a subject as a row. Although T_{com} may be sparse and have a lot of empty fields, empty fields do not occupy storage in HBase. When looking into rows in T_{com}, some OPS and SPO triple groups share the same row key.

---

[2] http://hbase.apache.org.

These triple groups indicate chain pattern relationship. In other words, the target triples of a chain pattern join must exist in T_{com}. RDFChain does not index the predicates of RDF triples. Having a table with predicates as row keys has serious scalability problems because the number of predicates in an ontology is usually fixed, relatively small in RDF datasets [3].

2.2 Query Planning

A join pattern is determined by the relationship of join variables in a query. Subject-Subject (SS) and Object-Object (OO) relationships are subject to star pattern joins while Object-Subject (OS or SO) relationships are subject to chain pattern joins. A query graph is a directed graph derived from a query. A triple pattern is referred to as a node and each join pattern is referred to as an edge. A logical plan is derived from a query graph. A logical plan includes a set of triple pattern groups (TPGs) and the join order of them. RDFChain determines the logical plan with the selection rules proposed. The selection rules focus on reducing the number of bindings. A chain TPG is a priority for grouping of TPGs.

2.3 Query Execution

Where a triple pattern has rdf:type as its predicate, we analyze the class hierarchy in the corresponding ontology tree by utilizing T_{com} instead of storing the triples inferred and then extends the logical plan. The logical plan is transformed into a physical plan for actual join processing on MapReduce and HBase. Each TPG in a logical plan is transformed into a map job in a physical plan. RDFChain makes table mappings for each TPG using an HBase index and accesses tables with an HBase filter based on the structure of a TPG. The first map job uses HBase tables as the query input. The intermediate result of each map job is stored in a distributed file system and then taken as an input of a map job iteration.

For a star pattern join, RDFChain efficiently retrieves a row through a single storage access in a map job since its storage has a triple group by subject or object as a row. For a chain pattern join, RDFChain only scans T_{com} since the triple groups satisfying a chain pattern are located in T_{com}. Therefore, T_{com} is more efficient for a chain pattern join due to the reduced number of storage access. In multi-way joins, only the rows with possibility of satisfying a chain pattern join are passed on as inputs of map job iterations, thereby reducing the processing time.

Two chain TPGs with disjoint join variables are compatible [4]. RDFChain estimates the cost of processing compatible sets in a map job. The cost of a map-side join is the sum of all the cost-consuming tasks of a map job. Since we do not need to consider the shuffle and reduce phases of a reduce-side join, we only take account of the time to process a join. In a conventional reduce-side join, a map task simply writes data according to a join key for an actual join in the reduce phase. However, the proposed map task of RDFChain is relatively heavy to iterate both binding and pattern matching. So, it may exceed the execution time assigned to the task. This problem can be resolved by a static method of increasing the timeout or a computing power. However, for a stationary time duration in a map task environment, we split TPGs based on a threshold to dynamically solve this problem. We limit the number of TPGs which can be processed in a map job using statistics such as the number of objects of frequently used subject-predicate pairs and the number of predicate-object pairs for every subject. We are able to estimate an intermediate result. The number of bindings for join variables dominates the number of map jobs. All map jobs run sequentially. If the cost does not exceed time limit, a single map job is generated.

3 Experimental Results

We used a Jena SPARQL parser and the Amazon's Elastic MapReduce service. Ten clusters of large instances were run on Hadoop 0.20.2 and HBase 0.92.0. We experimented with the LUBM 10K, LUBM 20K and LUBM 30K datasets, which consist of 1.3, 2.7 and 4.1 billion triples respectively, and a subset of benchmark queries, Q1, Q2, Q3, Q4, Q7 and Q9 [5]. The queries include simple and complex structures and provide star and chain pattern joins. Q3, Q4, Q7 and Q9 require type inference. We compared RDFChain with HadoopRDF [6] in terms of reduce-side join and MAPSIN [2] for map-side join.

RDFChain showed the best performance in large non-selective queries (Q2 and Q9). In particular, Q2 and Q9 have a complex structure, a low selectivity due to unbound objects, and a relationship of a chain pattern join. RDFChain greatly reduced the size of the intermediate results by limiting RDF triples to actual candidate rows which can satisfy a chain pattern join. RDFChain also shows smaller number of storage accesses than MAPSIN. Since T_{com} is a common subset of T_{spo} and T_{ops}, it scales down the scan space. RDFChain splits TPGs with compatible mappings and process the divided TPGs in a map task. So, the number of map jobs decreases in turn.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. NRF-2013R1A2A2A01016327).

References

1. Blanas, S., Patel, J.M., Ercegovac, V., Rao, J., Shekita, E.J., Tian, Y.: A Comparison of Join Algorithms for Log Processing in Mapreduce. In: Proc. International Conference on Management of data (2010)
2. Przyjaciel-Zablocki, M., Schätzle, A., Hornung, T., Dorner, C., Lausen, G.: Cascading Map-Side Joins over Hbase for Scalable Join Processing. CoRR, abs/1206.6293 (2012)
3. Stocker, M., Seaborne, A., Bernstein, A., Kiefer, C., Reynolds, D.: SPARQL basic graph pattern optimization using selectivity estimation. In: WWW, pp. 595–604. ACM (2008)
4. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and Complexity of SPARQL. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 30–43. Springer, Heidelberg (2006)
5. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Journal of Web Semantics 3, 158–182 (2005)
6. Husain, M., McGlothlin, J., Masud, M., Khan, L., Thuraisingham, B.: Heuristics-Based Query Processing for Large RDF Graphs Using Cloud Computing. IEEE Transactions on Knowledge and Data Engineering, vol. 23, no. 9, pp.1312-1327 (2011)

Context Aware Sensor Configuration Model for Internet of Things

Charith Perera[1,2], Arkady Zaslavsky[2], Michael Compton[2], Peter Christen[1],
and Dimitrios Georgakopoulos[2]

[1] Research School of Computer Science, The Australian National University,
Canberra, ACT 0200, Australia
{charith.perera, peter.christen}@anu.edu.au,
[2] CSIRO ICT Center, Canberra, ACT 2601, Australia
{charith.perera, arkady.zaslavsky, michael.compton
dimitrios.georgakopoulos}@csiro.au

Abstract. We propose a *Context Aware Sensor Configuration Model* (CASCoM) to address the challenge of automated context-aware configuration of filtering, fusion, and reasoning mechanisms in IoT middleware according to the problems at hand. We incorporate semantic technologies in solving the above challenges.

1 Introduction

Broadly, configuration in IoT paradigm can be categorized into two: *sensor-level* configuration and *system-level* configuration. Sensor-level configuration focuses on changing a sensor's behaviour by configuring its embedded software parameters such as sensing schedule, sampling rate, data communication frequency, communication patterns and protocols. In this paper, we are focused on developing a system-level configuration model for IoT midddleware platforms. Specifically, our proposed model identifies, composes, and configures both sensors and data processing components according to the user requirements. In existing IoT middleware (e.g. GSN), many configuration files and programming codes need to be manually defined by the users (without any help from GSN). An ideal IoT middleware configuration model should address all the above mentioned challenges. The configuration model we propose in this paper is applicable towards several other emerging paradigms, such as sensing as a service [4].

2 Problem Analysis

Our research question is *'How to develop a model that allows non-IT experts to configure sensors and data processing mechanisms in an IoT middleware according to the user requirements?'*. Extended explanations are provided in [3]. Research challenges are highlighted in Figure 1. Context-Aware Sensor Configuration Model (CASCoM) simplifies the IoT middleware configuration process significantly. Figure 2 compares the execution-flow of sensor configuration in the current GSN approach and the CASCoM approach. The proposed solution CASCoM is illustrated in Figure 3.

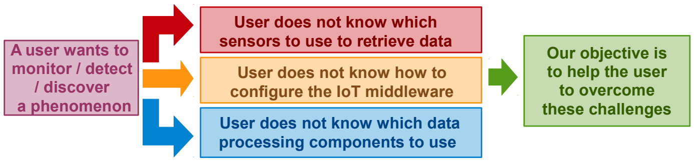

A user wants to monitor / detect / discover a phenomenon
User does not know which sensors to use to retrieve data
User does not know how to configure the IoT middleware
User does not know which data processing components to use
Our objective is to help the user to overcome these challenges

Fig. 1. Research Challenges, User Requirements, and Our Objective

3 The CASCoM Architecture

In phase 1, users are facilitated with a graphical user interface, which is based on a *question-answer (QA)* approach, that allows to express the user requirements. Users can answer as many question as possible. CASCoM searches and filters the tasks that the user may want to perform. From the filtered list, users can select the desired task. The details of the QA approach are presented later in this section. In phase 2, CASCoM searches for different programming components that allow to generate the data stream required. In phase 3, CASCoM tries to find the sensors that can be used to produce the inputs required by the selected data processing components. If CASCoM fails to produce the data streams required by the users due to insufficient resources (i.e. unavailability of the sensors), it will provide advice and recommendations on future sensor deployments in phase 4. Phase 5 allows the users to capture additional context information. The additional context information that can be derived using available resources and knowledge are listed to be selected. In phase 6, users are provided with one or more solutions[3]. CASCoM calculates the costs for each solution in using technique disucced in [2]. By default, CASCoM will select the solution with lowest cost. However, users can select the cost models as they required. Finally, CASCoM generates all the configuration files and program codes which we listed in Figure 2(a). Data starts streaming soon after.

---

[3] Solution is a combination of sensors and data processing components that can be composed together in order to satisfy the user requirements.

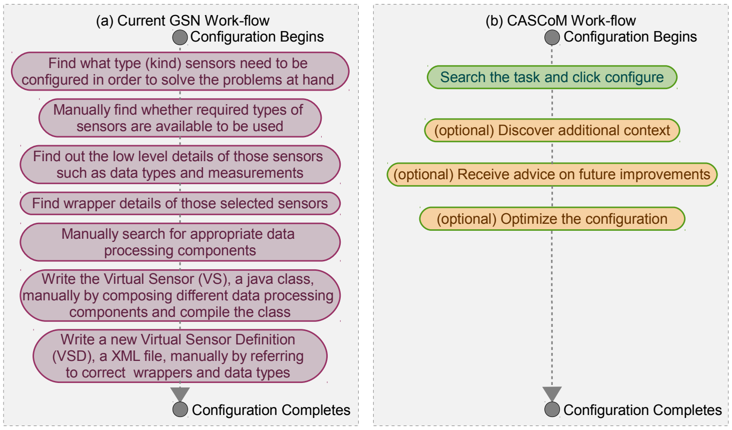

(a) Current GSN Work-flow
Configuration Begins
Find what type (kind) sensors need to be configured in order to solve the problems at hand
Manually find whether required types of sensors are available to be used
Find out the low level details of those sensors such as data types and measurements
Find wrapper details of those selected sensors
Manually search for appropriate data processing components
Write the Virtual Sensor (VS), a java class, manually by composing different data processing components and compile the class
Write a new Virtual Sensor Definition (VSD), a XML file, manually by referring to correct wrappers and data types
Configuration Completes
(b) CASCoM Work-flow
Configuration Begins
Search the task and click configure
(optional) Discover additional context
(optional) Receive advice on future improvements
(optional) Optimize the configuration
Configuration Completes

Fig. 2. Configuration Execution-flow Comparison: (a) Current GSN (b) CASCoM

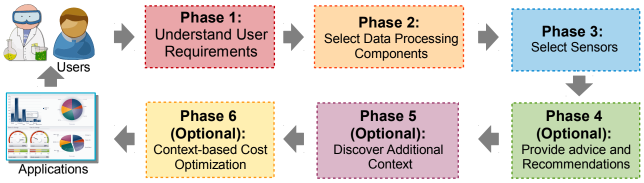

Users
Phase 1:
Understand User Requirements
Phase 2:
Select Data Processing Components
Phase 3:
Select Sensors
Phase 4 (Optional):
Provide advice and Recommendations
Phase 5 (Optional):
Discover Additional Context
Phase 6 (Optional):
Context-based Cost Optimization
Applications

Fig. 3. The Context-Aware Sensor Configuration Model (CASCoM)

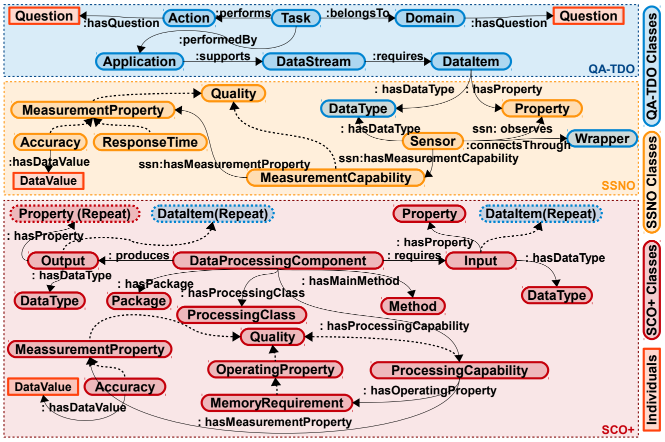


Fig. 4. Extracts of different ontological data models used in CASCoM: QA-TDO, SCO [1], and SSN ontology (w3.org/2005/Incubator/ssn/wiki/SSN). These model are used to store sensor descriptions, software component description, and domain knowledge.

4 Evaluation, Discussion and Lessons Learned

Results: Figure 5(a) shows that CASCoM allows to considerably reduce the time required for configuration of data processing mechanism in IoT middleware. Specifically, CASCoM allowed the three types of users to complete the given task 50, 80 and 250 times faster (respectively) in comparison to the existing approach. According to Figure 5(b), the Java reflection approach takes slightly more time to specially when initializing. Though the Java reflection approach can add more flexibility to our model, the additional overhead increases when the number of components and operation involved gets increased. The overheads can grow up to unacceptable level very quickly when GSN scales up (e.g. more user requests).

According to Figure 5(c), even IT experts who know GSN can save time by using CASCoM up to 88%. Specially, time taken for defining the VSD and VS class have been significantly reduced. Both files can be generated by CASCoM autonomously within a second even for complex scenarios. However, the time taken to find data processing components and sensors (and wrappers) depends on the size of the semantic data model. Figure 5(d) shows how total processing time would vary depending on the size of the semantic data model. Approximately, a semantic model with 10,000 sensor descriptions and 10,000 data processing components can be processed in order to find solutions for a given user request in less than a minute. However, most of the time is taken to read the data model.

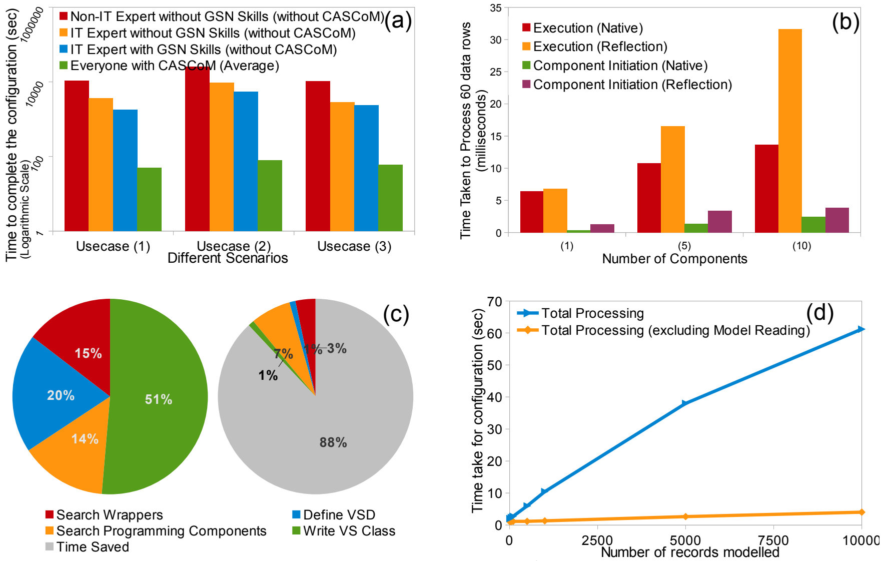


Fig. 5. Evaluation of CASCoM

The actual configuration process other than reading the data model takes only 4 seconds and it slightly increases when the model size increases.

5 Conclusion

We have shown that it is possible to offer a sophisticated configuration model to support non-IT experts. Semantic technologies are used extensively to support this model. Using our proof of concept implementation, both IT and non-IT experts were able to configure the GSN in significantly less time. In future, we plan to extend our configuration model into sensor-level. To achieve this, we will develop a model that can be used to configure sensors autonomously without human intervention in highly dynamic smart environments in the IoT paradigm.

References

1. F. E. Castillo-Barrera, R. C. M. Ramírez, and H. A. Duran-Limon. Knowledge capitalization in a component-based software factory: a semantic viewpoint. In *LA-NMR*, pages 105–114, 2011.
2. C. Perera, A. Zaslavsky, P. Christen, M. Compton, and D. Georgakopoulos. Context-aware sensor search, selection and ranking model for internet of things middleware. In *IEEE 14th International Conference on Mobile Data Management (MDM)*, Milan, Italy, June 2013.
3. C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Ca4iot: Context awareness for internet of things. In *IEEE International Conference on Conference on Internet of Things (iThing)*, pages 775–782, Besanon, France, November 2012.
4. A. Zaslavsky, C. Perera, and D. Georgakopoulos. Sensing as a service and big data. In *International Conference on Advances in Cloud Computing (ACC-2012)*, pages 21–29, Bangalore, India, July 2012.

Explaining Clusters with Inductive Logic Programming and Linked Data

Ilaria Tiddi, Mathieu d'Aquin, Enrico Motta

Knowledge Media Institute, The Open University, United Kingdom
{ilaria.tiddi,mathieu.daquin,enrico.motta}@open.ac.uk

Abstract. Knowledge Discovery consists in discovering hidden regularities in large amounts of data using data mining techniques. The obtained patterns require an interpretation that is usually achieved using some background knowledge given by experts from several domains. On the other hand, the rise of Linked Data has increased the number of connected cross-disciplinary knowledge, in the form of RDF datasets, classes and relationships. Here we show how Linked Data can be used in an Inductive Logic Programming process, where they provide background knowledge for finding hypotheses regarding the unrevealed connections between items of a cluster. By using an example with clusters of books, we show how different Linked Data sources can be used to automatically generate rules giving an underlying explanation to such clusters.

1 Introduction

Knowledge Discovery in Databases (KDD) is the process of detecting hidden patterns in large amounts of data [2]. In many real-world contexts, the explanation of such patterns is provided by experts, whose work is to analyse, visualise and interpret the results obtained out of a data mining process in order to reuse them. For instance, in Business Intelligence, the analyst uses such interpretation for decision making; in Learning Analytics, the detected patterns are used to assists people's learning; in Medical Informatics, trends can be useful for anomalies detection. This production of explanation becomes an intensive and time-consuming process, particularly when the background knowledge needs to be gathered from different domains and sources.

In a practical example, the university of Huddersfield provides books recommendations within its library catalogues[1], where records of books transactions over a decade can be used for stock management and students recommendation systems. Here, we are interested in explaining why groups of books, obtained from a clustering process, have been borrowed by the same students. Considering one such cluster, the question is: "why these books have been borrowed by those particular students?" and "where and how to find this information?".

Our hypothesis is that this answer can be given with Linked Data[2], which provide the required background knowledge (in our example, a trivial explanation for a pattern can be that authors of the books borrowed by the students enrolled in English Literature are from England). While works into data preparation and data mining using Linked Data have already been presented (see

---

[1] http://library.hud.ac.uk/data/usagedata/_readme.html

[2] http://linkeddata.org/

the ones of [4, 6, 8]), few works have considered Linked Data for results interpretation (some preliminary attempts are to be found in [1, 7]). However, the former uses Linked Data only to support the user's navigation, and the latter does not take into account the whole knowledge discovery process and focuses on the interpretation of statistical data. For this reason, we aim to exploit the interconnected knowledge from Linked Data to explain patterns resulting from a clustering process, by combining the existing semantic technologies with a Machine Learning technique, i.e. *Inductive Logic Programming* [3], to automatically produce underlying explanations for the formation of such patterns.

2 Approach

2.1 On Inductive Logic Programming

Inductive Logic Programming (ILP) is a research field at the intersection of Machine Learning and Logic Programming, investigating the inductive construction of first-order clausal theories starting from a set of examples $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$ [3]. While $\mathcal{E}^+$ represents the relation to be learnt, $\mathcal{E}^-$ are the facts where the relation does not hold. The distinguished feature of ILP is the use of some additional background knowledge $\mathcal{B}$ about the examples in $\mathcal{E}$. Believing $\mathcal{B}$, and faced with the facts in $\mathcal{E}$, the induction process derives an hypotheses space $\mathcal{H}$. The success of the induction requires that $\mathcal{H}$ covers all the positive examples ($\mathcal{H}$ is *complete*) and none of the negative ones ($\mathcal{H}$ is *consistent*), with respect to $\mathcal{B}$ (i.e., there is no contradiction with the facts written in $\mathcal{B}$).

2.2 Proposed approach

Assuming that we have retrieved some clusters, our approach is articulated as follows (see Fig. 1):

1. Linked Data Selection. We retrieve information about the data contained in each cluster from the Linked Data cloud, across several datasets.

2. Hypotheses Generation. We generate some hypotheses using ILP. A hypothesis is an explanation ("why those items are part of that particular cluster").

3. Hypotheses Evaluation. We validate the hypotheses using two rules evaluation measures: the *Weighted Relative Accuracy* (WR_{acc}, as described in [5]), providing a trade-off between coverage and relative accuracy, that we exploit to obtain explanations for small clusters, and the very well known and Information Retrieval F-measure ($\mathcal{F}$).

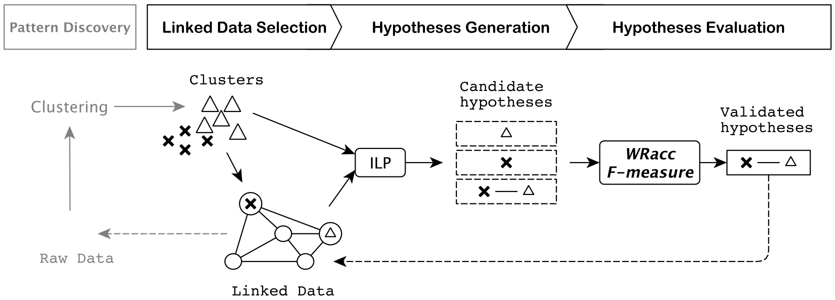


Fig. 1. Structure of the ILP approach for clusters explanation.

3 Experiments

We ran our experiments on the Huddersfield's books usage dataset introduced in the first section. Our target problem is defined as: *considering some clustered books borrowed by students from the Humanities faculty, explain what those books have in common and why they belong to a particular cluster.* The manual analysis of each cluster's centroid shows that each cluster represents the books borrowed by students from the same course, such as *Music Technologies, Politics* or *English Literature.*

For each book, we retrieve some information from the Linked Data cloud. We first use `bibo:isbn10` as an equivalence property to navigate from the Huddersfield dataset to the British National Bibliography one[3]. From there, we retrieve information about the book using the existing Linked Data vocabularies: Dublin Core[4] for topic and author, the Event Ontology[5] for the publication time, place and publisher. Finally, we exploit the `owl:sameAs` property to navigate to the Library of Congress Subject headings[6] and retrieve the broader concepts of each topic using the `skos:broader` property.

Clusters and the Linked Data extracted knowledge are encoded as Prolog clauses as follows:

$\mathcal{E}^+$	clusters	`cl`$_{MT}$`('book_1').`
$\mathcal{E}^-$		`cl`$_{MT}$`('book_4'). cl`$_{MT}$`('book_5').`
$\mathcal{B}$	RDF predicates	`subject('book_1','electronic music').`
	RDF is-a relations	`book('book_1'). topic('electronic music').`

Here we search the hypothesis space $\mathcal{H}$ specific to the *Music Technologies* cluster (`cl`$_{MT}$). $\mathcal{E}^+$ is composed by books in `cl`$_{MT}$ (as `book_1`), while books in other clusters (such as `book_4` and `book_5`) form $\mathcal{E}^-$. The process is repeated for each cluster. Both the RDF binary relations (`hud:book_1 dc:subject 'electronic music'`) and the unary ones (`hud:book_1 a bibo:book`) are also transformed into Prolog clauses and then added to $\mathcal{B}$.

We ran several experiments combining different properties (in different $\mathcal{B}$s), in order to see the properties impact on the hypotheses generation. These are shown in Table 1. Other hypotheses demonstrated the relations between different predicates, such as the relation between a publisher and a specific topic (see Table 2).

4 Conclusion and future work

We showed how ILP can be good in generating hypotheses to explain patterns, e.g. "*books borrowed by students of Music Technologies are clustered together because they talk about music*". Although it is a trivial example, the automation of such a process is not an easy task. We demonstrated how the use of Linked Data is important to generate such hypotheses, and how combining different sources

---

[3] http://bnb.data.bl.uk/

[4] http://dublincore.org/documents/dcmi-terms/

[5] http://motools.sourceforge.net/event/event.html

[6] http://id.loc.gov/authorities/subjects.html

Table 1. Expanding $\mathcal{B}_1$ with the LCSH knowledge ($\mathcal{B}_2$) improves the hypotheses. Those are read as follows: the item A belongs to the cluster `cl` because it has some properties, which appear in the body ("A's topic is mass media" or "A's broader topic is publicity").

Centroid	$\mathcal{B}$	Hypothesis	$\mathcal{F}(\%)$	WR_{acc}
Media& Journalism	$\mathcal{B}_1$	`cl(A):-subject(A,‘mass media,social aspects’)`	10.8	0.004
	$\mathcal{B}_2$	`cl(A):-broader(A,‘publicity’)`	16.4	0.007
Humanities	$\mathcal{B}_1$	`cl(A):-subject(A,‘criminology’)`	11.3	0.003
	$\mathcal{B}_2$	`cl(A):-broader(A,‘social sciences’)∧broader(A,‘auxiliary sciences’)`	15.5	0.003
Music Technologies	$\mathcal{B}_1$	`cl(A):-subject(A,‘sound, recording and reproducing’)`	10.6	0.003
	$\mathcal{B}_2$	`cl(A):-broader(A,‘digital electronics’)`	18.8	0.006
	$\mathcal{B}_1$	`cl(A):-subject(A,‘popular music, history and criticism’)`	14.5	0.005
	$\mathcal{B}_2$	`cl(A):-broader(A,‘music’)`	21.2	0.008
English& Media	$\mathcal{B}_1$	`cl(A):-subject(A,‘language acquisition’)`	11.6	0.005
	$\mathcal{B}_2$	`cl(A):-broader(A,‘child development’)∧broader(A,‘philology’)`	13.7	0.006

Table 2. Hypotheses revealing hidden connections between properties.

Centroid	Hypothesis	$\mathcal{F}(\%)$	WR_{acc}
Media	`cl(A):-broader(A,‘psychology’)∧pubPlace(A,‘oxford’)`	10.3	0.004

English Literature	`cl(A):-publisher(A,‘routledge’)∧broader(A,‘literature’)`
	`∧broader(A,‘philology’)`
		11.1	0.003

Politics	`cl(A):-publisher(A,‘macmillan’)∧broader(A,‘political science’)`
	`∧broader(A,‘social sciences’)`
		4.3	0.001

of background knowledge (i.e., different datasets) produces better explanations of patterns of data. The future work concerns the automatic selection of the datasets from Linked Data, the use of a more appropriate evaluation measure and the generalisation of the approach to other data mining techniques.

References

1. d'Aquin, M., & Jay, N. (2013). Interpreting Data Mining Results with Linked Data for Learning Analytics: Motivation, Case Study and Directions. In *Third Conference in Learning Analytics and Knowledge (LAK)*, Leuven, Belgium.
2. Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, 17(3), 37.
3. Muggleton, S., & De Raedt, L. (1994). Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19, 629-679.
4. Narasimha, V., Kappara, P., Ichise, R., & Vyas, O. P. (2011). LiDDM: A Data Mining System for Linked Data.
5. Lavrač N., Flach P., and Zupan B. (1999). Rule Evaluation Measures: A Unifying View. In Proceedings of the 9th International Workshop on Inductive Logic Programming (ILP '99). Springer-Verlag, London, UK, 174-185. .
6. Paulheim, H., & Fümkranz, J. (2012, June). Unsupervised generation of data mining features from linked open data. In Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics (p. 31). ACM.
7. Paulheim, H. (2012). Generating possible interpretations for statistics from Linked Open Data. In The Semantic Web: Research and Applications. Springer, pp. 560574.
8. Verborgh, R., Van Deursen, D., Mannens, E., & Van de Walle, R. (2010). Enabling advanced context-based multimedia interpretation using linked data.

D-SPARQ: Distributed, Scalable and Efficient RDF Query Engine

Raghava Mutharaju[1], Sherif Sakr[2], Alessandra Sala[3], and Pascal Hitzler[1]

[1] Kno.e.sis Center, Wright State University, Dayton, OH, USA.
{mutharaju.2, pascal.hitzler}@wright.edu
[2] College of Computer Science and Information Technology,
University of Dammam, Saudi Arabia.
University of New South Wales, High Street, Kensington, NSW, Australia 2052.
ssakr@cse.unsw.edu.au
[3] Alcatel-Lucent Bell Labs, Blanchardstown Industrial Park, Dublin, Ireland.
alessandra.sala@alcatel-lucent.com

Abstract. We present D-SPARQ, a distributed RDF query engine that combines the MapReduce processing framework with a NoSQL distributed data store, MongoDB. The performance of processing SPARQL queries mainly depends on the efficiency of handling the join operations between the RDF triple patterns. Our system features two unique characteristics that enable efficiently tackling this challenge: 1) Identifying specific patterns of the input queries that enable improving the performance by running different parts of the query in a parallel mode. 2) Using the triple selectivity information for reordering the individual triples of the input query within the identified query patterns. The preliminary results demonstrate the scalability and efficiency of our distributed RDF query engine.

1 Introduction

With the recent surge in the amount of RDF data, there is an increasing need for scalable RDF query engines. In SPARQL, even a simple query may translate to multiple triple patterns which have to be joined. In practice, centralized RDF engines lack scalability and query performance is abridged as they are highly dependent on main memory constraints in order to efficiently process these join operations. MapReduce-based processing platforms are becoming the *de facto* standard for distributed processing of large scale datasets.

We present D-SPARQ, a distributed and scalable RDF query engine that combines the MapReduce processing framework with a NoSQL distributed data store, MongoDB. In particular, we make the following contributions:

- We describe how RDF data can be partitioned, stored and indexed in a horizontally scalable NoSQL store, MongoDB.
- We describe a number of distributed query optimization techniques which consider the patterns of the input query together with selectivity information to minimize the processing time by efficiently parallelizing the query execution.
- A comparative performance evaluation of our approach with the state-of-the-art in distributed RDF query processing.

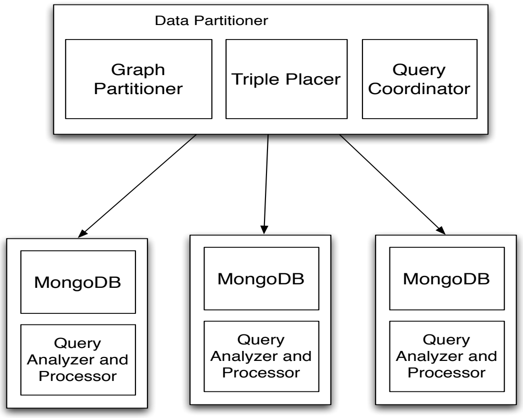

Data Partitioner
Graph Partitioner
Triple Placer
Query Coordinator
MongoDB
Query Analyzer and Processor
MongoDB
Query Analyzer and Processor
MongoDB
Query Analyzer and Processor

Fig. 1. System architecture

2 Approach

Figure 1 illustrates an overview of D-SPARQ's architecture. The system receives RDF triple datasets that are imported into MongoDB using a single MapReduce job which also captures all the required statistical information needed by our join reordering module in the query optimization process. A graph is constructed from the given RDF triples and using a graph partitioner [2], triples are spread across the machines in the cluster where the number of partitions is equal to the number of machines in the cluster. 'rdf:type' triples are removed from the data before partitioning the graph in order to make the graph more connected and reduce the quality of partitions. After partitioning, all the triples whose subject matches a vertex, are placed in the same partition as the vertex. A partial data replication is then applied where some of the triples are replicated across different partitions to enable the parallelization of query execution. In particular, in each partition, for all the vertices that are already assigned to that partition, vertices along a path of length n (in either direction) are added to that partition [1].

Triples assigned to each partition (machine) are stored in MongoDB[4], a NoSQL document database. In general, each RDF triple has three parts, *subject*, *predicate* and *object*. In a general key-value store, even if two parts of the triple are compressed into one, it would be less efficient to index them. Therefore, we used a document store, MongoDB, that has a good read and write speed along with good indexing, querying and sharding mechanisms. For example, in MongoDB, all triples with the same subject can be grouped in to one document. While many SPARQL queries in their basic graph patterns have a "star" pattern [3], these the patterns can share (joined on) either the same subject or object. By grouping the triples with the same subject, we would be able to retrieve

---

[4] `http://www.mongodb.org`

triples which satisfy subject-based star patterns in one read call. In addition, MongoDB supports indexing on any attribute of a document. It also supports single and compound indexes. We create compound indexes involving both of *subject-predicate* and *predicate-object* pairs. In MongoDB, a compound index handles queries on any prefixes of the index. For example, queries on predicate alone can be handled by the compound index *predicate-object*.

We tackle query processing by identifying the following patterns in the input query:

1. Triple patterns which are independent of each other and can be run in parallel.
2. Star patterns, i.e., triple patterns which need to be joined on the same subject or object.
3. Pipeline patterns, i.e., dependency among the triple patterns such as object of one triple pattern is same as the subject of another triple pattern (object-subject, subject-object, object-object joins).

Identifying these patterns enable us to run different parts of the query in parallel. In order to identify these patterns, an undirected labelled graph is constructed. In this graph, we find articulation points and biconnected components. In particular, articulation points provide the triples involved in pipeline pattern. A star pattern is treated as a block or a component here i.e., a star pattern cannot be split further. In general, a star pattern would have at least one articulation point and would be split up into smaller pieces if a regular biconnected component algorithm is run on it. With this tweak (keeping the star pattern as an indivisible block), all the independent star patterns can be obtained from the query graph.

In a star pattern, selectivity of each triple pattern plays an important role in reducing the query runtime. Therefore, for each predicate, we keep a count of the number of triples involving that particular predicate. For a star pattern, this information is used to reorder the individual triple patterns within a star pattern. After identifying the patterns from the query graph, processing of queries becomes a straightforward task of using querying capabilities provided by MongoDB, which automatically makes use of the appropriate indices while retrieving the records from the database. If pipeline patterns are involved in the query, care is taken to share the output of the dependent variable among all the triple patterns involved in the pipeline.

3 Evaluation

For our experimental evaluation, we used RDF datasets which are generated using SP^{2}Bench benchmark [5]. The benchmark generates DBLP data in the form of RDF triples. Our cluster consists of 3 nodes where each node has a quad-core AMD Opteron Processor with 16GB RAM and 2300MHz processor speed. MongoDB version 2.2.0 is used as a backend for our query engine. We compared our approach with the approach of [1], a distributed RDF query engine that uses RDF-3X [4] query processor as its backend. RDF-3X[5] Version 0.3.7

---

[5] `http://www.mpi-inf.mpg.de/~neumann/rdf3x`

#Triples	Query2		Query3		Query4	
	RDF-3X	D-SPARQ	RDF-3X	D-SPARQ	RDF-3X	D-SPARQ
77 million	217s	192.5s	80s	69.43s	OutOfMemory	319.87s
163 million	1537s	398s	434s	166s	OutOfMemory	671s

Table 1. Query runtimes (in seconds) for RDF-3X and D-SPARQ

has been used in our experiments. In particular, RDF-3X have been running on each node of the cluster and the same number of triples which are handled by our implementation are also loaded into RDF-3X on each node.

We picked three queries of the benchmark for our experiments (*Query2*, *Query3* and *Query4*). In particular, we have not considered the queries which uses the *OPTIONAL*, *FILTER*, *ORDER* features of the SPARQL query language as they are out of the scope of this paper where we are mainly focusing on the efficient execution of the join operations between the RDF triple patterns. The numbers of triples of our experimental datasets which are illustrated in Table 1 are the average number of triples loaded into RDF-3X, MongoDB of each node. So the total number of triples across all three nodes in the first case (with average of 77 million) is around 230 million and for the second case (163 million) is around 490 million triples. Each query has been executed five times and average of the runtime across all these runs has been collected. The results of Table 1 show that the query runtimes of our implementation are significantly better than that of RDF-3X, especially for larger number of triples. We observed that the performance of RDF-3X decreases with increase in the number of triples. This is a clear advantage for our query optimization techniques and the scalability of our data storage backend that relies on a NoSQL store, MongoDB.

4 Conclusion

We presented a distributed RDF query engine that combines a scalable data processing framework, MapReduce, with a NoSQL distributed data store, MongoDB. A comparative performance evaluation show that our approach can outperform the state-of-the-art in distributed RDF query processing. We are planning to continue evaluating our approach using different and bigger datasets and extend our approach to support other features of the SPARQL query language.

References

1. Huang, J., Abadi, D.J., Ren, K.: Scalable SPARQL Querying of Large RDF Graphs. PVLDB 4(11), 1123–1134 (2011)
2. Karypis, G., Kumar, V.: A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. SIAM Journal on Scientific Computing 20(1), 359–392 (Dec 1998)
3. Kim, H., Ravindra, P., Anyanwu, K.: From SPARQL to MapReduce: The Journey Using a Nested TripleGroup Algebra. PVLDB 4(12), 1426–1429 (2011)
4. Neumann, T., Weikum, G.: The RDF-3X engine for scalable management of RDF data. VLDB J. 19(1), 91–113 (2010)
5. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP^{2}Bench: A SPARQL Performance Benchmark. In: ICDE. pp. 222–233 (2009)

Efficient Computation of Relationship-Centrality in Large Entity-Relationship Graphs

Stephan Seufert[1], Srikanta J. Bedathur[2], Johannes Hoffart[1], Andrey Gubichev[3], and
Klaus Berberich[1]

[1] Max Planck Institute for Informatics, Germany
{sseufert,jhoffart,kberberi}@mpi-inf.mpg.de
[2] IIIT Delhi, India
bedathur@iiitd.ac.in
[3] Technische Universität München, Germany
andrey.gubichev@in.tum.de

Abstract. Given two sets of entities – potentially the results of two queries on a knowledge-graph like YAGO or DBpedia– characterizing the relationship between these sets in the form of important people, events and organizations is an analytics task useful in many domains. In this paper, we present an intuitive and efficiently computable vertex centrality measure that captures the importance of a node with respect to the explanation of the relationship between the pair of query sets. Using a weighted link graph of entities contained in the English Wikipedia, we demonstrate the usefulness of the proposed measure.

1 Introduction

Consider a journalist researching the political relations between *France* and *Germany*. In order to gain insight into the underlying relationship, it is an important task to identify entities (e. g. events, organizations, etc.) that play an important role in the interactions between these countries. This task can be greatly simplified if the journalist could simply input two sets of entities – corresponding to the classes "French Politicians" and "German Politicians" – and the system automatically generates a ranking of entities in the knowledge base, reflecting their potential for characterizing the relationship between the two entity-sets. Variants of this *relationship characterization* problem can be found in settings ranging from political studies to analysis of relationships in computational biology and economics. With the availability of massive entity-relationship networks such as Wikipedia, DBLP, and BioCyc networks as well as large Semantic Web ontologies like YAGO2 [5], solutions not only need to be effective, but also scalable. State-of-the-art approaches for identifying important nodes in networks include various centrality measures (e.g., closeness- and betweenness-centrality) which operate on the entire network, without any specific input entity sets.

In this paper, we develop a novel centrality measure called *relationship centrality*, that assesses the 'strength' of a node in the relationship path between the given two sets of nodes. These scores can be computed exactly, or can be well-approximated to scale to networks as large as the entire Wikipedia graph, comprising tens of millions of edges. The resulting rankings can further be restricted to entities of certain types (e.g, `Organization` or `Location` etc.), leveraging semantic knowledge-bases such as YAGO2 [5] or DBPedia [1]. In the following section, we formally introduce our novel centrality measure.

2 Relationship Centrality

Graph centrality measures, which assign to every node a score reflecting its importance in the graph structure, are a valuable tool for analyzing different kinds of graphs. Although they have been studied extensively in the scope of social networks, the use of centrality measures in the context of Semantic Web is gaining importance only recently. The classical measures proposed in past include *closeness* [3] and *betweenness centrality* [2]. However, these measures become computationally expensive when we consider large networks. Also, their utility in ranking nodes with respect to an input set of entities is rather limited.

In contrast, the measure we introduce in this work, called *relationship centrality*, is easier to compute since it is designed to assign scores that reflect the centrality only with respect to the two input entity sets, rather than on a global scale. Formally, given two query entity sets S and T from the network, we define the relationship centrality of a node v as follows:

$$c_R(v) = \sum_{s \in S} \sum_{t \in T} \frac{1}{\rho(s,v,t)},$$

where $\rho(s,v,t)$ is a penalty function for a path connecting a node $s \in S$ and $t \in T$ passing through v, given by: $\rho(s,v,t) = (1 + d(s,v)) \cdot (1 + d(v,t))$. The distance $d(\cdot,\cdot)$ between two nodes connected by an edge can be customized based on the underlying network to measure the semantic distance between the corresponding entities. The corresponding edge-weighting schemes we envision can be based on the graph structure (Milne-Witten inlink overlap measure [6]) or textual representations of the entities (keyphrase overlap measure [4]), among others.

Relationship centrality takes into account different paths than betweenness centrality (which regards the shortest paths between all pairs of vertices). For every vertex in the graph and every pair $(s,t) \in S \times T$, the shortest path from s to t passing through v contributes to the centrality score of v.

The corresponding paths are computed as follows: For every vertex $s \in S$ and $t \in T$, the shortest distances to each vertex $v \in V$ are computed using Dijkstra's algorithm. Then, the centrality scores for every vertex can be computed from the resulting distance vectors. While the $O(m + n\log(n))$ time complexity induced by each of the $|S| + |T|$ required shortest path computations is rather lightweight, for very large graphs the corresponding computation time can be too demanding, especially for interactive applications. For this purpose, we have experimented with an alternative scoring scheme which only considers shortest path distances up to a value $\Delta \in \mathbb{R}$. Then, the distance from a query node q to a vertex v is approximated in the following way:

$$\tilde{d}(q,v) = \begin{cases} d(q,v) & \text{for } d(q,v) \leq \Delta, \\ \text{diam}(G) & \text{else,} \end{cases}$$

where $\text{diam}(G)$ denotes the diameter of the (weighted) graph. In our experimental evaluation in Section 4, we evaluate the quality of computed scores based on different choices of the *cutoff parameter*, Δ.

Rank	Entity
1	2009 G-20 Pittsburgh sum.
2	2010 G-20 Toronto summit
3	37th G8 summit
4	Iraq War
5	35th G8 summit
6	2009 G-20 London Summit
7	36th G8 summit
8	2010 G-20 Seoul summit
9	Presidency of G. W. Bush
10	2009 Nobel Peace Prize

(a) Q1

Rank	Entity
1	The Expendables (2010 film)
2	Crouching Tiger, Hidden Dragon
3	The Forbidden Kingdom
4	Rush Hour 2
5	Police Story (1985 film)
6	Once Upon a Time in China II
7	Fist of Fury
8	Romeo Must Die
9	Kung Fu Hustle
10	Fearless (2006 film)

(b) Q2

Rank	Entity
1	Iraq War
2	War in Afghanistan
3	Gulf War
4	Op. Enduring Freedom
5	Yom Kippur War
6	War on Terror
7	Battle of Karameh
8	Palestinian diaspora
9	Operation Opera
10	Suez Crisis

(c) Q3

Table 1. Top ranked entities for example queries

3 Application

In this section, we briefly present the application scenarios we envision. We target analytical tasks at the downstream of Semantic Web applications. In particular, we consider a large knowledge-base (such as YAGO or DBPedia), over which S and T sets are derived as results of SPARQL queries.

Example: As a concrete scenario, the following two queries retrieve all organizations conducting research on (variants of) lung cancer, and all tobacco companies respectively:

```
SELECT ?p WHERE { ?p <rdf:type> <Organization> . ?p <worksOn> <Lung_Cancer> }
```

```
SELECT ?c WHERE { ?c <rdf:type> <American_Tobacco_Company> }
```

An analytics task could be to identify legal cases that played an important role in the relationship between the entity sets corresponding to the query results. We can utilize the relationship centrality measure developed above, and then use a type hierarchy, e. g. Wikipedia categories or the WordNet lexical database, to retain only the relevant entity types in the generated ranking.

4 Experimental Evaluation

In this section, we provide an overview over our experimental evaluation of the relationship centrality measure. In order to empirically validate the assigned scores, we have compiled several example queries over an edge-weighted entity-relationship graph obtained from Wikipedia: The vertices of the graph correspond to Wikipedia pages that represent an entity contained in the YAGO knowledge base. Two vertices u, v are connected via a weighted, undirected edge if there exists an internal Wikipedia link in either direction between the corresponding articles, $A(u), A(v)$. The weight we assign to the edge (u, v) should capture the semantic relatedness of the respective concepts. For this purpose, we employ the inlink overlap measure originally proposed by Milne and Witten [6]. For nodes u, v the weight (inverse semantic relatedness) is given by

$$d(u,v) = \frac{\log(\max\{|I_u|, |I_v|\}) - \log(|I_u \cap I_v|)}{\log(n) - \log(\min\{|I_u|, |I_v|\})},$$

where I_u and I_v denote the set of pages linking to $A(u)$ and $A(v)$, respectively and n corresponds to the overall number of pages. The resulting weights lie in the interval $[0, \infty]$. Using this measure, vertices u and v exhibit a high semantic relatedness if the weight of the edge (u, v) is close to zero. Finally, we discard all edges with $d(u, v) \geq 1$.

Query	$\Delta = \infty$	$\Delta = 1$		$\Delta = 0.5$	
	Time	Time	τ	Time	τ
Q1	41,960.40 ms	18,629.80 ms	1.0	4,616.05 ms	0.55
Q2	48,174.80 ms	15,002.70 ms	1.0	5,117.02 ms	0.60
Q3	71,162.50 ms	32,028.50 ms	1.0	7,858.39 ms	0.87

Table 2. Computation time and ranking quality

The resulting graph structure contains around 2.5 million vertices (entities) and roughly 37 million edges.

In order to empirically evaluate our ranking, we use three example queries where the sets of entities correspond to a collection of

Q1: *Events* between European politicians (S) and US American politicians (T)[4]
Q2: *Movies* between US action movie stars (S) and Asian action movie stars (T)[5]
Q3: *Events* between countries from the Middle East/Central Asia (S) and Western countries (T)[6]

In Table 1 we present the top-10 ranked results by relationship centrality for each of the queries. The resulting rankings suggest that our measure is useful for the explanation of the relationship between the sets of query entities. Regarding the computation time, we give an overview over the effect of pruning the shortest path computation using different cutoff parameters Δ, as well as the resulting rank correlation (measured by Kendall's τ) for the top 10 entities in Table 2.

5 Conclusions & Outlook

In this work we have presented the *relationship centrality* measure, a vertex centrality score that reflects the potential of an individual vertex for the explanation of the relationship between two sets of query nodes. Our preliminary experimental results over the edge-weighted Wikipedia entity-relationship graph indicate that our measure can provide valuable insights into the relationship between sets of real-world entities. In future work, we plan to conduct a large-scale evaluation of our result ranking in a user study. In addition, we plan to use our centrality measure as a building block for extracting interesting subgraphs between the query entities.

References

1. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *J. Web Sem.*, 7(3):154–165, 2009.
2. L. C. Freeman. A set of measures of centrality based upon betweenness. *Sociometry*, 40:35–41, 1977.
3. L. C. Freeman. Centrality in social networks: Conceptual clarification. *Social Networks*, 1(3):215–239, 1979.
4. J. Hoffart, S. Seufert, D. B. Nguyen, M. Theobald, and G. Weikum. KORE: Keyphrase Overlap Relatedness for Entity Disambiguation. In *CIKM'12: Proceedings of the 21th ACM International Conference on Information and Knowledge Management*, pages 545–555. ACM, 2012.
5. J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. *Artificial Intelligence*, 2013.
6. D. Milne and I. H. Witten. An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links. In *WIKIAI'08: Proceedings of the 2008 AAAI Workshop on Wikipedia and Artificial Intelligence*. AAAI, 2008.

---

[4] $S = \{$Angela Merkel, Nicolas Sarkozy, David Cameron, Silvio Berlusconi$\}$,$T = \{$Barack Obama, Hillary Clinton$\}$

[5] $S = \{$Chuck Norris, A. Schwarzenegger, Sylvester Stallone,Bruce Willis$\}$,$T = \{$Jet Li, Jackie Chan, Chow Yun-Fat$\}$

[6] $S = \{$Iraq, Iran, Israel, Palestine, Afghanistan, Pakistan$\}$,$T = \{$Germany, France, Spain, Italy, Netherlands, Portugal$\}$

A Hybrid Natural Language Approach to Manage Semantic Interoperability for Public Health Analytics

Maxime Lavigne, Arash Shaban-Nejad, Anya Okhmatovskaia, Luke Mondor,
David L. Buckeridge

McGill Clinical & Health Informatics, Department of Epidemiology and Biostatistics,
McGill University, Montreal, Quebec, H3A 1A3, Canada
`maxime.lavigne@mail.mcgill.ca`
`(arash.shaban-nejad|anya.okhmatovskaia|luke.mondor|david.buckeridge)@mcgill.ca`

Abstract. This paper discusses the integration of an ontology with a natural language query engine to calculate and interpret epidemiological indicators for population health assessment. In this paper, we discuss the application of this approach to one type of possible query, which retrieves health determinants, causally associated with diabetes mellitus.

Keywords: Ontology, Natural language interface, Causal inference, Epidemiology

1 Introduction

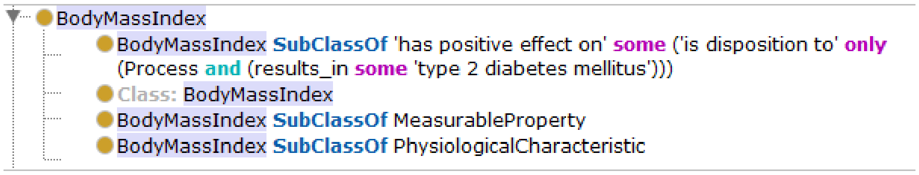
The Population Health Record (PopHR) platform [1][2] aims to improve population health decision-making. It calculates and presents measures or indicators of health determinants and health outcomes in a manner that, unlike most current web portals, is intuitive to access and provides up-to-date indicators that are contextualized by public health knowledge. In this paper, we describe our approach to querying the PopHR knowledge base using an natural language interface (NLI).

Early in its development, it became apparent that even though we were restraining the language of recognized queries, the breadth of pre- and post-conditions made implementation difficult. We therefore partitioned the space of possible queries and called these, query types. By partitioning intents of user inputs into collectively exhaustive and mutually exclusive query types, we were able to overcome the difficulty of designing a single data processing pathway for all queries. Linking a query's concepts with our domain ontology is simplified and it allows us, for example, to disambiguate concepts, which could have different interpretation in different query types.Partitioning restricts the software contract of our system when processing a query. Finally, this approach allows us to make assumptions about the domains of concepts, such as statistics and geography, which are relevant in our context.

In this paper, we use a representative query as an example: *What determinants increase the risk of diabetes?* The following sections introduce the relevant parts of our domain ontology and then describe the strategy used to answer the query.

2 Ontological Representation

PopHR uses its own domain ontology representing knowledge relevant to population health, including a taxonomy of human diseases, various groups of health determinants and public health interventions, measures of disease occurrence and other epidemiological concepts. In addition to the hierarchy of concepts, the ontology encodes associative relations to allow for meaningful inference. One specific type of associative relation represents a causal link between two entities (i.e., cause and effect). For example, body mass index (BMI) has a positive effect on an individual's disposition towards developing type 2 diabetes mellitus (see Figure 1). More generally, this relationship is an example of a probabilistic causal link from a health determinant to a process of developing a disease. We can also describe a causal relation between a health determinant and a process that modifies another health determinant.



```
BodyMassIndex
    BodyMassIndex SubClassOf 'has positive effect on' some ('is disposition to' only
    (Process and (results_in some 'type 2 diabetes mellitus')))
    Class: BodyMassIndex
    BodyMassIndex SubClassOf MeasurableProperty
    BodyMassIndex SubClassOf PhysiologicalCharacteristic
```

Fig. 1. Example of encoding BMI in the ontology, as seen with the Protégé editor.

3 Processing Pipeline

In PopHR, the natural language interface is the preferred method for querying information. All queries must respect a proper subset of the English language that is formally defined to be context free. The subset is built around question answering and was conceived with the intent to provide all the expressivity needed. This design decision implied that we needed an intuitive, consistent user experience. For the system to succeed at providing proper guidance, it needed to suit both the needs of the inexperienced users and experts.

3.1 Lexical and Syntactic Analysis

We used our formally defined grammar in conjunction with the ANTLR framework[3] for langage processing. The first step of the process is to break the input down into lexemes. The token stream produced by this step from the example input is:

`QUESTION, ID, VERB, ARTICLE, ID, QUALIFIER_START, ID, QUESTION_MARK`

Once the individual components of the question are separated, an LL(*) parser uses production rules to generate a syntactic tree. If the creation of such a tree is impossible, then we know that the input text was not part of our language and proper guidance will be given on how to correct the issue. This syntactic tree (Figure 2) is the artifact that will be used by the rest of the system.

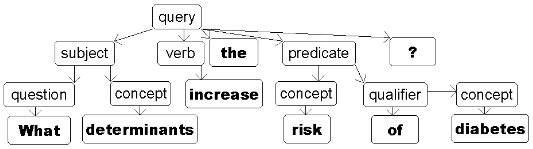


Fig. 2. Syntactic Tree Produced by our Example

A formal representation of the question is a necessary but not a sufficient step to understand the intent of the user. Although it is trivial for a human, performing this step programmatically requires the ability to match the query to some known patterns. This role is played by the oracle: all known patterns are manually entered in the system and take the general form: *What (To Be) ID? is a description query*. With this mapping, we are making the assumption that a question that starts with the question word *What* and uses a derivate of the verb *To Be* that has a final concept *ID* is asking the system for a description of this concept. Applying the Oracle to our example would classify it as a *CausalityEnumerationQueryWithConcept*. We can intuitively concur that we did want an enumeration of all determinants that have some causal relationship to diabetes mellitus.

3.2 Semantics

At this point in the process, we have gathered information regarding the domain and general intent of the query. Nevertheless, we still have no information on which concepts are used and what they mean. It is at this point that we query the ontology for concept such as determinants, increase, risk, and diabetes. Fetching these concepts by their textual representation, searching labels, synonyms and other annotations, we obtain the following:

Table 1. Association Between Query Terms and Ontology

Input	Concept or Relation in Ontology
determinants	health determinants
increase	'has positive effect on' some _
risk	'is disposition of' only (Process and 'results in' only _)
diabetes	diabetes mellitus

It is noteworthy that misspelings, difference in case and such are handled outside of the ontology. We then check for special markings that define processing triggers to activate. In our example, *'has positive effect on'* requires transitively walking upstream to identify additional causal factors. At this point, all of the information needed to understand what the user requested has been gathered. We would then reformulate the question into a format that can be answered by a description logic (DL) reasoner, such as Fact++[4]. From our example, we need *SubClassOf+* of *'Health Determinants'* that are described by: *" 'has positive effect on' some ('is disposition to' only (results_in some 'diabetes mellitus') "*

The results, will be a list of health determinants that directly influence the risk of the event in a positive way. From our processing trigger associated with *'has positive effect on'*, we know that the answer should also include any health determinants that positively affects health determinant having a direct influence on the risk event. We know, for example, that BMI is one of those direct factors and that it is a measurable property. Therefore, we look for other health determinants that have a positive effect on a disposition to increase the level of BMI. If the result is a measurable property we would repeat the same step.

4 Discussion

Developing a system that is accessible via a natural language interface is challenging. To address this challenge, we make use of all the contextual information we can learn about the intent of the query, we restrict ourselves to a proper context-free subset of the English language, and we use a domain ontology. The resulting system gives useful and correct answers to practical questions. We are looking forward validating our solution in user testing. It will enable us to broaden our scope from a prototype state to that of day to day use.

Acknowledgments

The Canadian Foundation for Innovation (CFI) and the Canadian Institutes of Health Research (CIHR) provide funding for this research.

References

1. Buckeridge DL, Izadi MT, Shaban-Nejad A, Mondor L, Jauvin C, Dubé L, Jang Y, and Tamblyn R. An infrastructure for real-time population health assessment and monitoring. IBM Journal of Research and Development, 2012, 56(5): 2
2. Izadi M, Shaban-Nejad A, Okhmatovskaia A, Mondor L, Buckeridge DL (2013). Population Health Record: An Informatics Infrastructure for Management, Integration, and Analysis of Large Scale Population Health Data. In Proc. of AAAI (HIAI 2013), Belleveue, Washington, USA July 14-18.
3. Parr T., Fisher KS, LL(*): The Foundation of the ANTLR Parser Generator, PLDI 2011
4. Tsarkov, D, and Horrocks, I. FaCT++ description logic reasoner: system description. In proc. of IJCAR 2006, Springer, pp. 292-297.

Towards the Natural Ontology of Wikipedia

Andrea Giovanni Nuzzolese[1,2], Aldo Gangemi[1,3],
Valentina Presutti[1], and Paolo Ciancarini[1,2]

[1] STLab-ISTC, National Research Council, Rome, Italy.
[2] Dept. of Computer Science and Engineering, University of Bologna, Italy.
[3] LIPN, University Paris 13, Sorbone Cité, UMR CNRS, France

Abstract. In this paper we present preliminary results on the extraction of ORA: the Natural Ontology of Wikipedia. ORA[4] is obtained through an automatic process that analyses the natural language definitions of DBpedia entities provided by their Wikipedia pages. Hence, this ontology reflects the richness of terms used and agreed by the crowds, and can be updated periodically according to the evolution of Wikipedia.

1 An ontology for Wikipedia

The DBpedia Ontology [5] (DBPO) and Yago [7] are the two reference ontologies for DBpedia. Both of them provide only partial extensional and intensional coverage of DBpedia entities because they rely on Wikipedia categories (Yago) and infoboxes (DBpedia), which induce an intrinsic limit of domain coverage [3]. Tìpalo [3] is a tool that automatically produces a RDF taxonomy of types (aligned with WordNet and Dolce) for a DBpedia entity by analysing its natural language definition in Wikipedia. This approach is aimed at identifying the most natural types for an entity as they are expressed by the crowds. By running Tìpalo on the whole DBpedia we aimed at deriving a natural ontology for Wikipedia and approximating as much as possible a complete domain coverage. In this paper, we show the results obtained so far from this process: the first version of the Natural Ontology of Wikipedia (ORA)[6], and we discuss emerging issues and possible solutions for its refinement. This article is organized as follows: (i) in section 2, we introduce the main related work; (ii) in section 3 we describe the material and the method used for generating the ontology; (iii) finally in section 4 we describe the results obtained so far, and discuss ongoing work and future research directions.

2 Related work

The DBpedia project [4] and YAGO [7] are the most relevant approaches at generating an ontology from semi-structured information in Wikipedia. DBpe-

---

[4] ORA is the italian translation of NOW

[5] http://dbpedia.org/ontology

[6] http://isotta.cs.unibo.it:8080/sparql - select the graph *now*

dia provides an ontology extracted from Wikipedia infoboxes based on hand-generated mappings of infoboxes to the DBpedia ontology (DBPO). DBPO counts 359 concepts (version 3.8) but only 2.3M entities over more than 4M are classified with respect to this ontology. YAGO types are extracted from Wikipedia categories and aligned to a subset of WordNet. The YAGO ontology is larger that DBPO and counts $\sim$290K concepts. YAGO has a larger (although still incomplete, 2.7M typed entities) coverage of DBpedia entities. ORA introduces a third *dimension*: the terminology of the crowds; furthermore, it provides a larger coverage (currently 3.0M typed entities). Recently, the Schema.org [7] initiative has provided alignments to the DBPO. However, such effort does not add value from the perspective of the intensional and extensional coverage issues. Other relevant work related to our method includes Ontology Learning and Population (OL&P) techniques [1]. Typically OL&P is implemented on top of machine learning methods, hence it requires large corpora, sometimes manually annotated, in order to induce a set of probabilistic rules. Such rules are defined through a training phase that can take a long time. The method used for ORA and implemented by Tìpalo [3] differs from existing approaches as it is mainly rule-based, hence it does not require a training phase and it is faster than the other approaches.

3 Automatic extraction of an ontology for Wikipedia: materials and methods

Tìpalo is implemented as a pipeline of components and data sources. Each component in the pipeline implements a step of the computation: (i) extraction of an entity's natural language definition from its Wikipedia abstract; (ii) natural language deep parsing (provided by FRED [6]) whose output is a RDF/OWL representation of the entity definition; (iii) selection of candidate types (based on graph-pattern-based heuristics applied to FRED output); (iv) word-sense disambiguation of candidate types; and (v) type alignment to OntoWordNet [2], WordNet supersenses and to a subset of and DUL+DnS Ultralite. We refer to [3] for details about the design and the implementation of Tìpalo. In [3] we evaluated Tìpalo by extracting the types for a sample of 627 resources, while in this work we want to extract the ontology of Wikipedia by running Tìpalo on 3,769,926 DBpedia entities taken from the `dbpedia_long_abstracts_en` dataset of DBpedia, which include only entities having a Wikipedia abstract: this is a main constrain for applying our method.

4 The Natural Ontology of Wikipedia (ORA): results and discussion

The process described above has been run on a Mac Pro Quad Core Intel Xeon 2.8Ghz with 10Gb RAM and took 15 days (which can be easily reduced by parallelizing the activity on a cluster of machines with similar or more powerful

---

[7] http:schema.org

characteristics). The process resulted in 3,023,890 typed entities and associated taxonomies of types. Most of the missing results are due to the lack of matching Tìpalo heuristics, which means that by improving Tìpalo we will improve coverage (this is part of our current work). The resulting ontology includes 585,474 distinct classes organized in a taxonomy with 396,375 `rdfs:subClassOf` axioms; 25,480 if these classes are aligned through `owl:equivalentClass` axioms to 20,662 OntoWordNet synsets by means of a word-sense disambiguation process. The difference between the number of disambiguated classes (25,480) and the number of identified synsets (20,662) means that there are at least 4,818 synonym classes in the ontology. We expect the number of actual synonyms to be greater. Hence, we are planning to investigate some sense-similarity-based metric in order to reduce the number of distinct classes in the ontology by merging synonyms or at least providing explicit similarity relations with confidence scores between classes.

In order to prevent polysemy deriving from merging classes with same names but aligned to different synsets, it has been adopted a criterion of uniqueness for the generation of the URIs of these classes. For example, let us consider the entity `dbpedia:The_Marriage_of_Heaven_and_Hell`[8]. For this entity Tìpalo generates the following RDF:

```
dbpedia:The_Marriage_of_Heaven_and_Hell
    a  fred:Book .
fred:Book
    owl:equivalentClass  wn30-instance:synset-book-noun-2 .
```

Similarly, for the entity `dbpedia:Book_of_Revelation`[9] Tìpalo generates the following RDF:

```
dbpedia:Book_of_Revelation
    a  fred:CanonicalBook .
fred:CanonicalBook
    rdfs:subClassOf  fred:Book .
fred:Book
    owl:equivalentClass  wn30-instance:synset-book-noun-10 .
```

The two `fred:Book` classes refers to two distinct concepts. Hence, they cannot be merged during the generation of the ontology. We solve this by appending the ID of the closest synset in the taxonomy to the URI of the new generated classes: this approach guarantees to prevent polysemy and to identify synonymity at the same time. Finally, all the classes aligned to OntoWordNet have been also aligned to WordNet supersenses and a subset of DOLCE+DnS Ultra Lite classes by means of `rdfs:subClassOf` axioms. The following example shows a sample of the ontology which has been derived by typing the two entities used as examples previously:

---

[8] The definition of `dbpedia:The_Marriage_of_Heaven_and_Hell` is: *"The Marriage of Heaven and Hell is one of William Blake's books."*

[9] The definition of `dbpedia:Book_of_Revelation` is: textit"The Book of Revelation is the last canonical book of the New Testament in the Christian Bible."

```
dbpedia:The_Marriage_of_Heaven_and_Hell
    a  fred:Book_102870092 .
dbpedia:Book_of_Revelation
    a  fred:CanonicalBook_106394865 .
fred:CanonicalBook_106394865
    rdfs:subClassOf  fred:Book_106394865 ;
    rdfs:label  "Canonical Book"@en-US .
fred:Book_102870092
    owl:equivalentClass  wn30-instance:synset-book-noun-2 ;
    rdfs:label  "Book"@en-US .
fred:Book_106394865
    owl:equivalentClass  wn30-instance:synset-book-noun-10 ;
    rdfs:subClassOf  wn30-instance:supersense-noun_communication ,
                     d0:InformationEntity ;
    rdfs:label  "Book"@en-US .
```

Conclusion. The main result of this work is the Natural Ontology of Wikipedia (ORA): an ontology that reflects the richness of terms used and agreed by the crowds for defining entities in Wikipedia. All produced datasets are available for download[10]. We claim that this ontology provides an important resource that can be used as alternative or complement for YAGO and DBPO, and that it can enable more accurate usage of DBpedia in Semantic Web based applications such as: mash-up tools, recommendation systems, and exploratory search tools (see for example Aemoo [5]), etc. Currently, we are working at refining ORA and to align it to DBPO and YAGO.

References

1. P. Cimiano. *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer, 2006.
2. A. Gangemi, R. Navigli, and P. Velardi. The OntoWordNet Project: extension and axiomatization of conceptual relations in WordNet. In *in WordNet, Meersman*, pages 3–7. Springer, 2003.
3. A. Gangemi, A. G. Nuzzolese, V. Presutti, F. Draicchio, A. Musetti, and P. Ciancarini. Automatic Typing of DBpedia Entities. In *International Semantic Web Conference (1)*, volume 7649 of *Lecture Notes in Computer Science*, pages 65–81. Springer, 2012.
4. J. Lehmann, C. Bizer, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - A Crystallization Point for the Web of Data. *Journal of Web Semantics*, 7(3):154–165, 2009.
5. A. G. Nuzzolese, V. Presutti, A. Gangemi, A. Musetti, and P. Ciancarini. Aemoo: Exploring knowledge on the web. In *Proceedings of the 5th Annual ACM Web Science Conference*, pages 272–275. ACM, 2013.
6. V. Presutti, F. Draicchio, and A. Gangemi. Knowledge extraction based on discourse representation theory and linguistic frames. In *Knowledge Engineering and Knowledge Management (EKAW 2012)*, pages 114–129. Springer, 2012.
7. F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, pages 697–706, New York, NY, USA, 2007. ACM Press.

---

[10] `http://stlab.istc.cnr.it/stlab/ORA`

The Empirical Robustness of Description Logic Classification

Rafael S. Gonçalves, Nicolas Matentzoglu, Bijan Parsia, and Uli Sattler

School of Computer Science, University of Manchester, Manchester, United Kingdom

Abstract. In spite of the recent renaissance in lightweight description logics (DLs), many prominent DLs, such as that underlying the Web Ontology Language (OWL), have high worst case complexity for their key inference services. Modern reasoners have a large array of optimization, tuned calculi, and implementation tricks that allow them to perform very well in a variety of application scenarios, even though the complexity results ensure that they will perform poorly for some inputs. For users, the key question is how often they will encounter those pathological inputs in practice, that is, how robust are reasoners. We attempt to determine this question for classification of existing ontologies as they are found on the Web. It is a fairly common user task to examine ontologies published on the Web as part of their development process. Thus, the robustness of reasoners in this scenario is both directly interesting and provides some hints toward answering the broader question. From our experiments, we show that the current crop of OWL reasoners, in collaboration, is very robust against the Web.

1 Motivation

A serious concern about both versions 1 [4] and 2 [3] of the Web Ontology Language (OWL) is that the underlying description logics ($\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$) exhibit extremely bad worst case complexity (NEXPTIME and 2NEXPTIME) for their key inference services. While since the mid-1990s, highly optimized description logic reasoners have been exhibiting rather good performance in real cases, even in those more constrained cases there are ontologies (such as Galen) which have proved impossible to process for over a decade. Indeed, concern with such pathology stimulated a renaissance of research into tractable description logics with the $\mathcal{EL}$ family [1] and the DL Lite [2] family being incorporated as special "profiles" of OWL 2. However, even though the number of ontologies available on the Web has grown enormously since the standardization of OWL, it is still unclear how robust modern, highly optimized reasoners are to such input. Anecdotal evidence suggests that pathological cases are common enough to cause problems, however, systematic evidence has been scarce.

In this paper we investigate the question of whether modern, highly-optimized description logic reasoners are *robust* over Web input. The general intuition of a robust system is that it is *resistant to failure in the face of a range of input*. For any particular robustness determination, one must decide: *1)* the range of input, *2)* the functional or non-functional properties of interest, and *3)* what counts as failure. The instantiation of these parameters strongly influences robustness judgements, with the very same reasoner being highly robust under one scenario and very non-robust under another. For our current purposes, the key scenario is that an ontology engineer, using a tool like Protégé [6], is inspecting ontologies published on the Web with an eye to possible reuse, and, as is common, they wish to classify the ontology using a standard OWL 2 DL reasoner as part of their evaluation. This scenario yields the following constraints: *1)* for input, we examine Web-based corpora, *2)* functional: acceptance (will the reasoner load and process the ontology); non-functional: performance (i.e., will the reasoner complete classification before the ontology engineer gives up), *3)* w.r.t. acceptance, failure means either rejecting the input or crashing while processing, and we might reasonably expect an engineer to wait up to 2 hours if the ontology seems "worth it". If a reasoner (or a set of reasoners) is successful for 90% of a corpus, we count that reasoner as robust over that corpus, with 95% and 99% indicating "strong" and "extreme" robustness. While these levels are clearly arbitrary (as is the timeout), they provide a framework to set expectations. Robustness under these assumptions does not ensure robustness under other assumptions (e.g., over subsets of these ontologies as experienced during development or over a more stringent time constraint), yet they are challenging enough that it was unclear to us *ex ante* whether any reasoner would be robust for any corpus.

In fact, we find that the reasoners are robust or near robust for most of the cases we examine, including for lower timeouts. More significantly, if we take the best result for each ontology (which represents a kind of "meta-reasoner", where our test reasoners are run in parallel), then the *set* of reasoners is extremely robust over all corpora. Thus, in a fairly precise, if limited, sense, we demonstrate that classification over OWL ontologies (even those based on highly expressive description logics, such as $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$) is practical, even despite the worst case being intractable in some cases.

2 Results

Overall we have processed a total of 1,071 ontologies, the largest such reasoner benchmark (similar benchmarks typically use at most a few hundred ontologies, e.g., the recent study in [5]), having found that amongst the 4 tested reasoners Pellet is the most robust of all (see Table 1). Surprisingly, Pellet is followed by JFact on our robustness test, due to having far less errors than FaCT++. HermiT and FaCT++ have the same overall robustness, but FaCT++ has less errors and higher impatient robustness.

While Pellet is the most robust reasoner, we urge some caution in that reading. In particular, this does not mean that Pellet will always do best or even perform reasonably. In fact, it may timeout where other reasoners finish reasonably fast. The set of reasoners (taken together and considering the best results) is extremely robust across the board (for each reasoner's contribution to the best case reasoner, see Figure 1). Thus, we have strong empirical evidence that the *ontologies* on the Web do not supply any *in principle* intractable cases, but only cases which are difficult for particular reasoners.

Note that FaCT++ and JFact fail to process several ontologies due to poor support for OWL 2 datatypes. Both of these reasoners, as well as HermiT, seem to have little support for OWL 1 datatypes. By removing the non OWL 2 datatype errors, we would end up with FaCT++ being the most robust w.r.t. OWL 2, followed by HermiT and Pellet. That is, if we restrict the test corpus to those ontologies that use only datatypes from the OWL 2 datatype map, then FaCT++ would be the most robust reasoner.

	Pellet	HermiT	JFact	FaCT++	Best Combo	Worst Combo
Very Easy	787 (73%)	706 (66%)	741 (69%)	784 (73%)	878 (82%)	645 (60.2%)
Easy	116 (11%)	112 (10%)	103 (10%)	55 (5%)	73 (7%)	102 (9.5%)
Medium	83 (8%)	65 (6%)	43 (4%)	94 (9%)	101 (9%)	45 (4.2%)
Hard	24 (2%)	53 (5%)	76 (7%)	7 (1%)	6 (1%)	75 (7.0%)
Very Hard	6 (1%)	4 (0%)	1 (0%)	4 (0%)	4 (0%)	3 (0.3%)
Timeout	29 (3%)	14 (1%)	16 (1%)	15 (1%)	9 (1%)	25 (2.3%)
Errors	26 (2%)	117 (11%)	91 (8%)	112 (10%)	0 (0%)	176 (16.4%)
Total (excl. Errors)	1016	940	964	944	1062	870
Total (incl. Errors)	1071	1071	1071	1071	1071	1071
Impatient Robustness	92%	82% [90%]	83%	87% [96%]	98%	74% [87%]
Overall Robustness	95%	88% [96%]	90%	88% [97%]	99%	81% [96%]

Table 1: Binning of all three corpora: BioPortal, NCIt (2013), and Web crawl. Under robustness rows, values in square brackets indicate robustness w.r.t. OWL 2 alone.

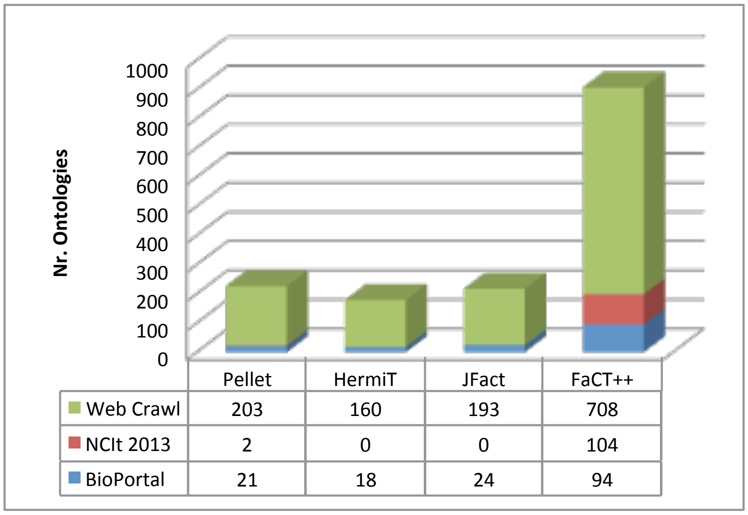
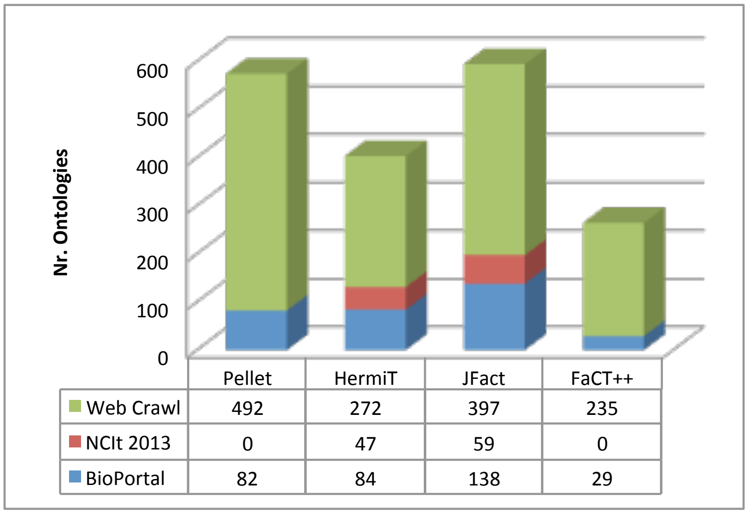


Fig. 1: Number of times each reasoner outperforms all other reasoners in each corpus.

From Figure 1 we see that FaCT++ outperforms other reasoners on many occasions, but, due to the high number of errors thrown, its robustness w.r.t. our input data is not as good as this figure might indicate. In Figure 2 we show the frequency with which reasoners are the worst case in each corpus: Notice that FaCT++ is, overall, less often the worst reasoner, followed by HermiT. However, HermiT and JFact both dominate the worst cases in the NCIt corpus. Pellet, while being most often the worst case reasoner in the Web Crawl corpus, is so (in many cases) by a mere fraction of a second; as pointed out it is the most robust for that corpus.

It is clear that deriving a sensible ranking even simply using average or total time is not straightforward. Our results have rather strong implications for reasoner experi-



	Pellet	HermiT	JFact	FaCT++
🟩 Web Crawl	492	272	397	235
🟥 NCIt 2013	0	47	59	0
🟦 BioPortal	82	84	138	29

Fig. 2: Number of times that each reasoner equals the worst case, for each corpus.

ments, especially those purporting to show the advantages of an optimisation or a technique or an implementation: The space is very complex and it is very easy to simultaneously generate a biased sample for one system and against another. Even simple, seemingly innocuous things like timeouts and classification failures require tremendous care in handling. If results are going to be meaningful across papers we need to converge on experimental inputs, methods, and reporting forms.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: Proc. of IJCAI-05 (2005)
2. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning 39(3), 385–429 (2007)
3. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. J. of Web Semantics (2008)
4. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. J. of Web Semantics 1(1), 7–26 (2003)
5. Kang, Y.B., Li, Y.F., Krishnaswamy, S.: Predicting reasoning performance using ontology metrics. In: Proc. of ISWC-12 (2012)
6. Knublauch, H., Fergerson, R.W., Noy, N.F., Musen, M.A.: The Protégé OWL plugin: An open development environment for semantic web applications. In: Proc. of ISWC-04 (2004)

Network-Aware Workload Scheduling for Scalable Linked Data Stream Processing

Lorenz Fischer, Thomas Scharrenbach, Abraham Bernstein

University of Zurich, Switzerland*
{lfischer,scharrenbach,bernstein}@ifi.uzh.ch

1 Introduction

In order to cope with the ever-increasing data volume, distributed stream processing systems have been proposed. To ensure scalability most distributed systems partition the data and distribute the workload among multiple machines. This approach does, however, raise the question how the data and the workload should be partitioned and distributed. A uniform scheduling strategy—a uniform distribution of computation load among available machines—typically used by stream processing systems, disregards network-load as one of the major bottlenecks for throughput resulting in an immense load in terms of inter-machine communication.

We propose a graph-partitioning based approach for workload scheduling within stream processing systems. We implemented a distributed triple-stream processing engine on top of the Storm realtime computation framework and evaluate its communication behavior using two real-world datasets. We show that the application of graph partitioning algorithms can decrease inter-machine communication substantially (by 40% to 99%) whilst maintaining an even workload distribution, even using very limited data statistics. We also find that processing RDF data as single triples at a time rather than graph fragments (containing multiple triples), may decrease throughput indicating the usefulness of semantics.

2 Problem Statement and Definitions

A linked data stream processing system essentially continuously ingests large volumes of temporally annotated RDF triples and emits the results again as data stream. Such systems usually implement a version of the SPARQL algebra that has been modified for processing dynamic data. The processing model considered is a directed graph, where the nodes are algebra operators and data is sent along the edges. Hence, each query can be transformed to a query tree of algebra expressions – the topology of the processing graph.

---

* The research leading to these results has received funding from the Europ. Union 7th Framework Programme FP7/2007-2011 under grant agreement no 296126 and from the Dept. of the Navy under Grant NICOP N62909-11-1-7065 issued by Office of Naval Research Global.

In order to scale the system horizontally (i.e., executing its parts on multiple processing units concurrently) we may replicate parts (or the whole) of the query's topology and execute clones of the operators, i.e., *tasks* in parallel. The workload of the system, can then be distributed across several machines in a compute cluster. We refer to the assignment of tasks to machines as *scheduling.*

The goal of our approach is to find a schedule (i.e., assignment of tasks to machines) for a given topology that minimizes the total number of data messages transferred over the network, whilst maintaining an even workload distribution across machines in terms of CPU cycles.

Many stream processing platforms attempt to uniformly distribute compute loads possibly incurring high network traffic. Approaches like Borealis [1] schedule the processors according to the structure of the query, where every operator is is assigned to one machine. This approach has an upper limit in parallelization equal to the number of operators and may incur high network traffic between two machines containing active operators. As Aniello et al. in their recent work [2], we propose to partition the data, to parallelize the operators, and then to minimize network traffic allowing for more flexibility for distributing the workload. In contrast to previous work, we don't try to implement a new scheduling algorithm *but much rather make use of existing graph partitioning algorithms to optimize the amount of data sent between machines.*

Hypothesis: *Combining data partitioning between tasks with a scheduler that employs graph partitioning to assign the resulting task instances outperforms a uniform distribution of data and tasks to machines.*

Our hypothesis assumes that different distribution strategies significantly influence the number of messages sent between the machines.

3 Evaluation and Results

We evaluated our system using two example queries that are built around a real world streaming use case: *SRBench* [5], (streams of weather measurements), and an open government dataset (self-acquired from public sources), which combines data on public spending in the US with stock ticker data.[1] We devised a query that would highlight (publicly traded) companies, that double their stock price within 20 days and are/were awarded a government contract in the same timeframe. The data for both use cases has been converted to and consumed as time annoated n-triple records.

Procedure First, we partitioned each dataset and compiled the queries into execution topologies. We then recorded the number of messages that were sent between tasks at runtime. Second, to test our hypothesis, we needed to partition the resulting communication graph based on the network load of each channel. Since the channel loads are not known before running the query we chose two experimental scenarios. In the first scenario we assume an *oracle* optimizer that would know the number of messages that would flow along every channel. This scenario allows to establish a hypothetical upper bound of quality that our

---

[1] http://www.usaspending.gov, https://wrds-web.wharton.upenn.edu/wrds

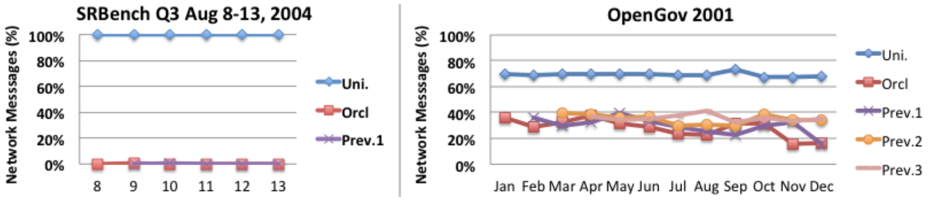


Fig. 1. Percentage of messages sent over the network for the uniform distribution and the graph partitioned setup, using either the test data itself (oracle) or data from the previous one to three time-slices as input for the graph partitioning algorithm.

method could attain, if it were to have an oracle. In a second scenario we assumed a *learning* optimizer that first observes channel statistics for a period of time and then partitions the graph accordingly. To that end we sliced the *SRBench* data into daily and the *OpenGov* data into monthly slices. We then measured the the performance of our approach based on learning during the preceding one to three time-slice essentially providing a adaptively learning system.
Third, to partition the graph we employed *METIS* [3]. We used the *gpmetis* in its standard configuration, which creates partitions of equal size, and only changed the *-objtype* parameter to instruct *METIS* to optimize for total communication volume when partitioning, rather than minimizing on total edgecut.

The Suitability of Graph Partitioning for Scheduling *The critical element for optimizing the scheduling using graph partitioning is that the operators can be parallelized with an adequate data partitioning.* The results show that using a graph partitioning algorithm to schedule task instances on machines does indeed reduce the number of messages sent over the network (Fig 1). We graph the number of network messages divided by the number of total messages as a measure for the optimality of the distribution. The *SRBench* data can be optimally partitioned by the id of the reporting weather station even when using only the data of the immediately preceeding time slice (left side, Prev.1). Once this task has been achieved, which we got due to the pre-partitioned datasets, all computation can be managed on a local machine, as no further joins are necessary. This clearly indicates that some queries can be trivially distributed when a good data partition is either known or can be learned.

For the *OpenGov* dataset (right side) the tested join operation requires a significant redistribution of messages. First, we find that our approach clearly outperforms the uniform distribution strategy by a factor of two to three. Second, even longer learning periods, using two (Prev.2) and even three previous time slices (Prev.3), do not necessarily improve the overall performance - maybe due to over-fitting or concept drift [4]. We also found that this only leads to a slightly less even load distribution.

For the *SRBench* query we observed a reduction in network usage by over 99%. For the *OpenGov* query, workload distribution using a graph partitioning approach yields savings in terms of network bandwidth of over 40%.

Balancing Computation Load In order to make good use of the available resources, a distributed system should assign equal workloads to all machines.

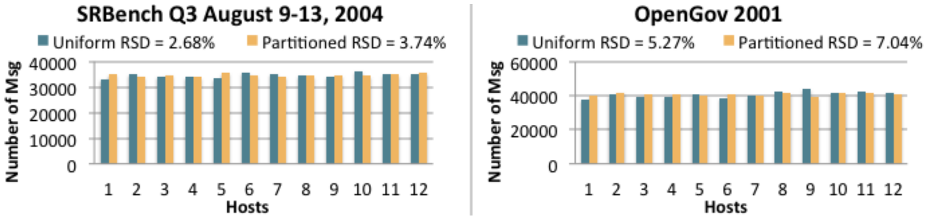


Fig. 2. Average computation load distribution for all time-slices of each dataset. RSD = Relative Standard Deviation

For this reason we analyzed how many messages were processed by all tasks of each partition for the two queries (Figure 2). The load distribution resulting from the graph partitioned task assignment only differs slightly from the one found by uniform task distribution (average relative standard deviation (RSD) *OpenGov*: 7.04% for partitioning vs. 5.27% for uniform baseline; SRBench: 3.74% for partitioning vs. 2.68% for uniform baseline).

The most important shortcomings of our study are its limitation to two datasets and queries and the fixed setup of the distributed system. For the first we intend to systematically extend our evaluation in the future in terms of number of datasets and queries. For the latter, is it the interactions between number of machines and cores available and the degree of parallelism that require further research. Especially the impact of such interactions on throughput in terms of messages ingested per second is of interest here. Future work will also investigate whether the principle of finding the smallest possible data partition given the desired degree of parallelism is as important as our experiments indicate.

We are confident that our findings help making DSFP systems more scalable and ultimately enable reactive systems that are capable of processing billions of triples or graph fragments per second with a negligible delay. It is our firm belief that the key to addressing these challenges needs to and will have to be revealed from the data itself.

Acknowledgements We would like to thank Thomas Hunziker, who wrote the first prototype of the *KATTS* system during his master's thesis in our group.

References

1. Abadi, D.J., Ahmad, Y., Balazinska, M., Hwang, J.h., Lindner, W., Maskey, A.S., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y., Zdonik, S.: The Design of the Borealis Stream Processing Engine. In: Proc. CIDR2005. pp. 277–289 (2005),
2. Aniello, L., Baldoni, R., Querzoni, L.: Adaptive online scheduling in storm. In: DEBS2013 (2013),
3. Karypis, G., Kumar, V.: A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. SIAM J. on Scientific Comp. 20(1), 359–392 (Jan 1998),
4. Vorburger, P., Bernstein, A.: Entropy-based Concept Shift Detection. In: Proc. ICDM2006. pp. 1113–1118 (2006)
5. Zhang, Y., Duc, P.M., Corcho, O., Calbimonte, J.p.: SRBench : A Streaming RDF / SPARQL Benchmark. In: Proc. ISWC 2012 (2012)

Semantic Enrichment of Mobile Phone Data Records Using Linked Open Data

Zolzaya Dashdorj, Luciano Serafini

SKILL, Telecom Italia and DKM, Fondazione Bruno Kessler
University of Trento, ICT International Doctoral School, Trento, Italy
{dashdorj@disi.unitn.it, serafini@fbk.eu}

1 Introduction

Users of mobile (smart) phones, generate an enormous amount of data every day. Most of them are not accessible due to privacy reason, but anonimized metadata, such as for instance, the location, the time and the duration of the interaction with the smartphone are nowadays available for analysis. We address these data as "Call Data Records" (CDR). CDR metadata constitute an important source of information for investigating on general human behavior, such as mobility [5, 3, 6], and communication patterns [9, 2, 4, 1, 8]. Currently, most of the analyses provide a quantitative description of human behavior, which is presented via visual analytics techniques. The outcome of these analysis are usually quantitative models estimating for instance, the number of people present in a certain area at a certain time, the number of people who moves from point a to point b within a certain period, and so on. Less interest has been dedicated to the creation of model that describe human activity at qualitative/semantic level, i.e., in terms of semantically rich concepts in order to estimate/predict for instance, the actions performed by a group of people in a certain situation or the type of event is happening around in a certain location, on the basis of the CDR.

The analyses presented in [11, 10, 1, 5, 3] need to be extended making use of a relevant knowledge of the context of the user. User contextual information includes all those information describing the objects present and the events happening in the place and at the time that the user is interacting with his/her smartphone. Contextual information includes information about the territory (e.g., points of interests - POIs), weather conditions, public and private events (e.g., concerts, sport events, public spontaneous meeting etc) and emergency events (accidents, strikes, etc.), transportation schedule, energy or water consumption, etc.

In this paper, we propose a first step of investigation developing an ontological and statistical model (HRBModel) capable to predict the possible human activities in a certain place and on a certain time on the basis of the contextual information describing the POI's of the place and information about the time of the day at which certain actions are usually performed. POI's are taken from Openstreetmap[1]. The model enables early identification of standard type of hetergenous human activities in various geographical area profiles and at different times in which CDR occur.

2 Experimental Setup

We are interested in predicting the most probable human activity of a user when he/she is in a specific geographical area at a given time. We develop our experiment consider-

---

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

ing the area of Trento - Italy, but the process and the software is a general enough to be applicable to any geographical area.

Details about the experiment are described in the following.

We divide the geographical area of the city into sub-divisions in a way that POIs are uniformly distributed over the subareas (see Figure 1(a)). In each sub area, POIs are extracted using geographical tools like OSM2PGSQL. We do not consider the POIs which don't derive human activities, such as benches, towers, emergency phones, etc. In total for this city, we extracted 333,809 POIs from the OSM map which cleaned to 159,314.

To annotate the POIs with the human activities, we propose a Human Behavioral Ontology (HBOnto) that with the help of the OSMOnto ontology[7], associates POIs with all the human activities that can be performed or hosted there or nearby. The human activities are hierarchically organized from specific activities to 10 high level categories, that around 220 human activities. For this association, we take day and time into account as all the activitiies are connected to a day-time range of validity. A stochastic behavior model (SBM) we propose that estimates the probability of human activities given the location and time of an event, on the basis of the ontological model as follow:

$$\mathrm{P}(a|t,l) = P(a|t) * P(a|l) \text{ (t, l are independent)} \tag{1}$$

As an initial step, we manually created a fuzzy model for $P(a|t)$ that performs reasoning on the importance of activities given a time that is estimated as follow:

$$\mathrm{P}(a|t) = \frac{FuzzyActivity(a,t)}{FuzzyTime(t)} \tag{2}$$

$P(a|l)$ is estimated based the importance of the activities given a location using TF/IDF for the POIs importance that derives a weight to the activities as follow:

$$\mathrm{tf}-\mathrm{idf}(f,l) = \frac{N(f,l)}{\underset{w}{\mathrm{argmax}}\{N(w,l) : w \in l\}} * \log\frac{|L|}{|\{l \in L : f \in l\}|} \tag{3}$$

To avoid the spatial gap, $P(a|l)$ can be extended if we consider the nearby activities in a given radius r of a circular area around the location.

$$P(a|l,r) = \frac{\underset{a}{\mathrm{argmax}}\{W(a,l_i) * \lambda_i\}}{\sum_{a\in r}\underset{a}{\mathrm{argmax}}\{W(a,l_i) * \lambda_i\}} : r\bigcap_{i=1}^{n} l_i \tag{4}$$

We collected the user-data[2] through the experiment application described above (see Figure 1(b)) for one week with 32 partipants involved (see Table 1). It emerged that most of the user feedback comes from the areas of (Trento-Povo and Trento-Downtown).

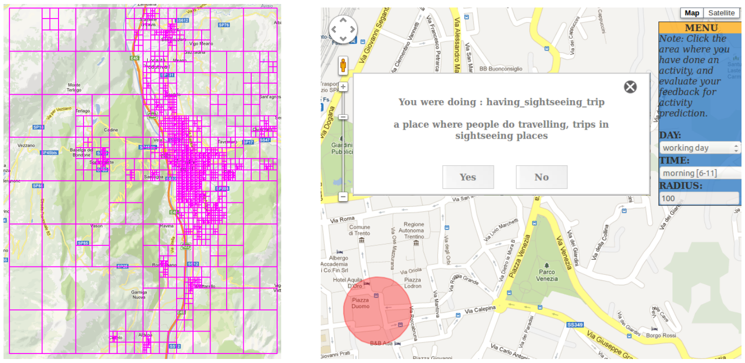
Every user's feedback is collected in a record containing: the latitude/longitude of the location selected on the map, the radius of the area, the selected activity (among top-5 or one freely chosen), the semantic day and time (e.g., weekday, saturday.., early morning, mid morning.. etc), and the current time of the system.

---

[2] http://dkmlab.fbk.eu:8080/BHRModelTest/data/semantic_data_BHRModel2013.csv

Number of feedback	Participants	Duration	Feedback clusters	Feedback in each cluster
481	32	1 week	5	Trento.Nord - 21, Downtown - 180, Povo - 151, Santa Chiara - 60 Trento.Sud - 67
Feedback on Weekday		morning 158, mid day 29, afternoon 59, evening 61, night 28		
Feedback on Saturday		morning 19, mid day 5, afternoon 33, evening 19, night 8		
Feedback on Sunday		morning 22, mid day 5, afternoon 25, evening 6, night 4		

Table 1: Data collection by user feedback over different locations and times



You were doing : having_sightseeing_trip

a place where people do travelling, trips in sightseeing places

Note: Click the area where you have done an activity, and evaluate your feedback for activity prediction.

DAY: working day

TIME: morning [6-11]

RADIUS: 100

(a) Geographical area divisions by the density of POIs

(b) Visualization of demonstration app, at Piazza Duomo on weekday mornings

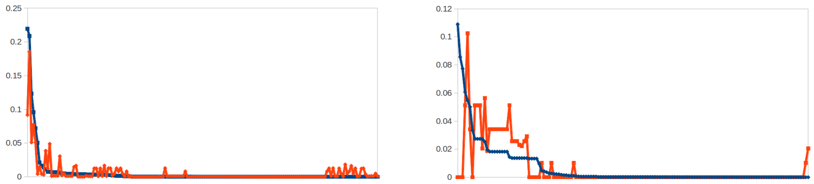
Fig. 1: Application UI for High level Behavioral Representation Model

3 Preliminary results

Given the collected data, we measured the accuracy of the HRBModel considering the correctly predicted activities w/regardless of the ranking position. We also analyzed the divergence of the probability activities comparing to the probability from the feedback in the areas with the highest number of feedback: Trento-Downtown and Trento-Povo. The example of the results, Figure 2(a) and 2(b) describe the divergence of the probability activities that occur in those areas on weekday mornings and afternoons, in which we propogated the probability activity to the child activities. In these figures, the activity indexing (x-axis) is different in each area, y-axis is the probability associated to the activities. The figures show that the probability activities from the user feedback can still follow the trend of the probability from our model.

4 Conclusion and Future works

Within the 481 users' feedback collected, 341 activities have been correctly predicted, corresponding to an overall accuracy of 70.89%. The overall accuracy of a correct activity prediction (among the top-5) corresponds to 61.95%. We have done the evaluation considering the high level (parent) activities, the overall accuracy has been increased to 80.23%. We showed the divergence between the probability activities in our model compared to the probability from the feedback by various locations and times that can be further studied in order to understand the correlation between human activities and contexts. We will extend the evaluation of the model making use of mobile phone survey,



(a) Trento-Downtown area on weekday mornings

(b) Trento-Povo area on weekday afternoons

Fig. 2: Divergence of the activities probability from our model (blue) and from user feedback (red).

crowd sourcing, and social networks (e.g., twitter, foursquare). The proposed model will be a baseline to characterize geographical areas by activity of interests in the areas where CDRs are occurred. This allows identification and prediction of human behaviors by various area profiles (e.g., business, shopping, or leisure areas etc) in certain contextual conditions and detection of anomalous behavioral events.

References

1. B.Furletti, L.Gabrielli, C.Renso, and S.Rinzivillo. Identifying users profiles from mobile calls habits. In *the Proc. of the ACM SIGKDD Int.Workshop on Urban Computing*, UrbComp '12, pages 17–24. ACM, 2012.
2. C.Ratti, S.Sobolevsky, F.Calabrese, C.Andris, J.Reades, M.Martino, R.Claxton, and S.H.Strogatz. Redrawing the map of great britain from a network of human interactions. *PLoS ONE*, 5(12):e14248, 12 2010.
3. F.Calabrese, F.C.Pereira, G.Di Lorenzo, L.Liu, and C.Ratti. The geography of taste: analyzing cell-phone mobility and social events. In *the Proc. of the 8th Intl.Conf. on Pervasive Computing*, Pervasive'10, pages 22–37, 2010.
4. J.Candia, M.C.González, P.Wang, T.Schoenharl, G.Madey, and A.Barabási. Uncovering individual and collective human dynamics from mobile phone records. *Journal of Physics A: Mathematical and Theoretical*, 41(22):224015, June 2008.
5. J.P.Bagrow, D.Wang, and A.Barabási. Collective response of human populations to large-scale emergencies. *CoRR*, abs/1106.0560, 2011.
6. M.C.Gonzalez, C.A.Hidalgo, and A.Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, June 2008.
7. M.Codescu, G.Horsinka, O.Kutz, T.Mossakowski, and R.Rau. Osmonto - an ontology of openstreetmap tags. In *State of the map Europe (SOTM-EU)*, 2011.
8. N.Eagle and (Sandy) A.Pentland. Reality mining: sensing complex social systems. *Personal Ubiquitous Comput.*, 10(4):255–268, March 2006.
9. J. P. Onnela, J. Saramäki, J. Hyvönen, G. Szabó, D. Lazer, K. Kaski, J. Kertész, and A. L. Barabási. Structure and tie strengths in mobile communication networks. *Proceedings of the National Academy of Sciences*, 104(18):7332–7336, May 2007.
10. S.Phithakkitnukoon, T.Horanont, G.Di Lorenzo, R.Shibasaki, and C.Ratti. Activity-aware map: identifying human daily activity pattern using mobile phone data. In *the Proc. of the 1st Intl. Conf. Human Behavior Understanding*, pages 14–25, 2010.
11. Z.Dashdorj and L.Serafini. Semantic interpretation of mobile phone records exploiting background knowledge. In *Intl.Conf. Semantic Web Conference (ISWC 2013), Doctoral Consortium*, 2013.