Valentina Ivanova Patrick Lambrix
Steffen Lohmann Catia Pesquita (Eds.)

VOILA! 2016

Proceedings of the 2nd International Workshop on

Visualization and Interaction for Ontologies and Linked Data

Co-located with ISWC 2016, Kobe, Japan, October 17, 2016

Title: Visualization and Interaction for Ontologies and Linked Data (VOILA! 2016)

Editors: Valentina Ivanova, Patrick Lambrix, Steffen Lohmann, Catia Pesquita

ISSN: 1613-0073

CEUR Workshop Proceedings
(CEUR-WS.org)

Copyright © 2016 for the individual papers by the papers' authors. Copying permitted for private and academic purposes. This volume is published and copyrighted by its editors.

Organizing Committee

Valentina Ivanova, Linköping University, Sweden
Patrick Lambrix, Linköping University, Sweden
Steffen Lohmann, Fraunhofer IAIS, Germany
Catia Pesquita, University of Lisbon, Portugal

Program Committee

Benjamin Bach, Monash University, Melbourne, Australia
Isabel F. Cruz, University of Illinois at Chicago, USA
Aba-Sah Dadzie, Knowledge Media Institute, The Open University, UK
Roberto García, Universitat de Lleida, Spain
Anika Gross, University of Leipzig, Germany
Willem Robert van Hage, Netherlands eScience Center, The Netherlands
Ali Hasnain, The Insight Centre for Data Analytics, Ireland
Eero Hyvönen, Aalto University & University of Helsinki, Finland
Hanmin Jung, Korea Institute of Science and Technology Information, Korea
Tomi Kauppinen, Aalto University School of Science, Finland
Ali Khalili, Vrije Universiteit Amsterdam, The Netherlands
Suvodeep Mazumdar, University of Sheffield, UK
Paul Mulholland, Knowledge Media Institute, The Open University, UK
Stefan Negru, MSD IT Global Innovation Center, Czech Republic
Francesco Osborne, Knowledge Media Institute, The Open University, UK
Heiko Paulheim, University of Mannheim, Germany
Silvio Peroni, University of Bologna & CNR-ISTC, Italy
Emmanuel Pietriga, INRIA Saclay, France
Harald Sack, Hasso Plattner Institute, University of Potsdam, Germany
Daniel Schwabe, Pontifical Catholic University of Rio de Janeiro, Brazil
Gem Stapleton, University of Brighton, UK
Vojtěch Svátek, University of Economics, Prague, Czech Republic

Additional Reviewers

Victor Christen
Aidan Delaney
Magnus Knuth
Peter Chapman
Tabea Tietz

Preface

A picture is worth a thousand words, we often say, yet many areas are in demand of sophisticated visualization techniques, and the Semantic Web is not an exception. The size and complexity of ontologies and Linked Data in the Semantic Web constantly grows and the diverse backgrounds of the users and application areas multiply at the same time. Providing users with visual representations and intuitive interaction techniques can significantly aid the exploration and understanding of the domains and knowledge represented by ontologies and Linked Data.

Ontology visualization is not a new topic and a number of approaches have become available in recent years, with some being already well-established, particularly in the field of ontology modeling. In other areas of ontology engineering, such as ontology alignment and debugging, although several tools have recently been developed, few provide a graphical user interface, not to mention navigational aids or comprehensive visualization and interaction techniques.

In the presence of a huge network of interconnected resources, one of the challenges faced by the Linked Data community is the visualization of multidimensional datasets to provide for efficient overview, exploration and querying tasks, to mention just a few. With the focus shifting from a Web of Documents to a Web of Data, changes in the interaction paradigms are in demand as well. Novel approaches also need to take into consideration the technological challenges and opportunities given by new interaction contexts, ranging from mobile, touch, and gesture interaction to visualizations on large displays, and encompassing highly responsive web applications.

There is no one-size-fits-all solution but different use cases demand different visualization and interaction techniques. Ultimately, providing better user interfaces, visual representations and interaction techniques will foster user engagement and likely lead to higher quality results in different applications employing ontologies and proliferate the consumption of Linked Data.

These and related issues are addressed by the VOILA! workshop series concerned with *Visualization and Interaction for Ontologies and Linked Data*. The second edition of VOILA! is co-located with the 15th International Semantic Web Conference (ISWC 2016) and will take place as a full day event on October 17, 2016 in Kobe, Japan. It will be organized around scientific paper presentations and discussions, and will be accompanied by interactive software demonstrations, giving developers a chance to gather feedback from the community.

The call for papers for VOILA! 2016 attracted 22 submission in different paper categories. Three reviewers were assigned to each submission. Based on the reviews, we selected 15 contributions for presentation at the workshop in the following categories: full papers (7), system papers (2), short papers (2), demo papers (4).

We thank all authors for their submissions and all members of the VOILA! program committee for their useful reviews and comments. We are also grateful to Chiara Ghidini and Heiner Stuckenschmidt, the workshop chairs of ISWC 2016, for their continuous support during the workshop organization.

October 2016

Valentina Ivanova,
Patrick Lambrix,
Steffen Lohmann,
Catia Pesquita

Contents

Workshop Summary

While reading through the papers this year, one topic appeared more often than others—*navigation*. Navigation in the physical world is a complex cognitive process; finding one's way in a digital environment can be even more challenging, and has its own peculiarities. Navigating an information space is indispensable for understanding its content and structure, it is an activity accompanying higher-level tasks. Acquiring information about the environment—obtaining an overview, identifying objects and anchors or landmarks as well as relationships between them—is often the first step in an unfamiliar setting. This leads to devising an internal structure, storing and integrating our prior and new knowledge in order to plan and execute a task at hand by interacting with the environment. To facilitate this pursuit, we employ general-purpose visual representations and interaction techniques or tailor them to a specific information space or application. Visualizations are targeted to varying user groups, since individuals possess diverse backgrounds and differ in navigational strategies and abilities; they seek to fulfill different information needs and tasks. Visualizing semantic data introduces an extra tint to the picture—it provides advanced means to explore an information space but may increase visualization and interaction complexity.

The workshop program includes a mixture of general and application-tailored works, most of them implicitly touching the issue of navigation in an information space. They target user groups with different expertise—lay users, technically-fluent users and users possessing domain expertise. The first session—*Visualization of Domain Ontologies and Data*—takes us through several examples of how semantic techniques facilitate the exploration of domain-specific datasets. In the ontology realm, extensive navigation is often in demand to support high-level tasks—developing, modifying, consistency checking, etc. These topics together with ontology visualization are part of the second session—*Visualization in Ontology Engineering*. The next session—*Visualization of SPARQL Queries and Endpoints*—focuses on extracting and visualizing schema information using SPARQL, and visually navigating and exploring Linked Data endpoints, while the last session—*Visualization in Semantic Annotation*—is composed of two talks presenting approaches for semantic annotation in order to improve navigation and exploration.

Session I: Visualization of Domain Ontologies and Data

The first session opens the workshop by presenting several application-tailored visualizations supporting the navigation in domain-specific datasets and ontologies. The first two talks—*VIZ–VIVO: Towards Visualizations-driven Linked Data Navigation* and *PrEVIEw: Clustering and Visualizing PubmEd using Visual Interface*—take us through a topic familiar to the research community—exploration of scholarly data—, and both employ concept maps as a representation of choice. While the latter work centers around the biomedical domain and addresses PubMed data, the former makes use of

a community-driven ontology and it is not limited to a particular domain. Visually exploring geographical data from multiple heterogeneous sources is the focus of the third talk—*A Linked Data Driven Visual Interface for the Multi-Perspective Exploration of Data Across Repositories*. Staying in the realm of geo data and making a preamble to the second session, the last talk—*Exploring Visualization of Geospatial Ontologies using Cesium*—uses the globe to visualize instances from geospatial ontologies.

Session II: Visualization in Ontology Engineering

The second session is dedicated to the different stages of an ontology life cycle. The first talk—*Starting Ontology Development by Visually Modeling an Example Situation - A User Study*—compares a common approach for ontology development using the Protégé editor to an alternative workflow—visually developing an ontological background model of an example situation. The second work—*Advanced UML Style Visualization of OWL Ontologies*—brings us back to the familiar topic of ontology visualization, while the third—*Extending Ontology Visualization with Interactive Contextual Verbalization*—extends visual notations with on-demand verbal descriptions to facilitate domain experts' understanding of selected graphical elements. Developing (large) ontologies increasingly involves a number of people with different roles and development styles. The fourth talk—*Visualizing User Editing Behavior in Collaborative Ontology-Engineering Projects*—provides means to facilitate the analysis of users' editing behavior in a collaborative setting. Consequently, ontologies are changing, and tools to explore the modifications are in demand; the last talk—*Visualization for Ontology Evolution*—surveys existing ontology evolution tools and catalogues their functionalities.

Session III: Visualization of SPARQL Queries and Endpoints

A common step when navigating an information space is acquiring an overview of its content. By knowing the schema of an endpoint, the users can pose queries to fulfil varying information needs. This topic is tackled during the third session. The first talk—*A Visual Aide for Understanding Endpoint Data*—establishes three requirements necessary for providing a visual overview of an endpoint's schema. The tool that has been implemented is suitable for SPARQL-fluent users. The next work—*LD-VOWL: Extracting and Visualizing Schema Information for Linked Data Endpoints*—adapts the VOWL notation, suitable for casual users, to present an endpoint's schema extracted through a dynamic stepwise approach. While the aforementioned two works employ node-link diagrams as a representation of the graph metaphor, the third talk—*ViziQuer: Notation and Tool for Data Analysis SPARQL Queries*—adapts UML class diagrams for defining data aggregation queries. These are general approaches which facilitate the understanding of an endpoint's structure. Unlike them, the last talk—*SQuaRE: a Visual Support for OBDA Approach*—focuses on a particular scenario and provides visual support for developing mappings between relational databases and ontologies.

Session IV: Visualization in Semantic Annotation

The last session comprises two works which depict the benefits from semantic annotations to the navigation task. The first work—*Semantic Annotation and Information Visualization for Blogposts with refer*—helps content authors (lay-users) to annotate textual content in order to facilitate content consumers in discovering background information and exploring relationships between entities. The second work—*Visual Development & Analysis of Coreference Resolution Systems with CORVIDAE*—employs radial diagrams to support NLP developers in discovering errors in coreference annotation during information extraction.

Conclusion

The workshop program is a mixture of works navigating different domains, targeting diverse user groups and combining various visual metaphors. Some of the authors have conducted user evaluations to uncover the strengths and weaknesses of their approaches. More user evaluations are, however, needed to get a better understanding of which (set of) visual metaphors and representations meet best the requirements of particular user groups, domains and goals and are thus essential in this area.

October 2016

Valentina Ivanova,
Patrick Lambrix,
Steffen Lohmann,
Catia Pesquita

Workshop Program

Session I: Visualization of Domain Ontologies and Data

- VIZ-VIVO: Towards visualizations-driven linked data navigation, *Muhammad Javed, Sandy Payette, Jim Blake and Tim Worrall*
- PrEVIEw: Clustering and Visualizing PubmEd using Visual Interface, *Syeda Sana E Zainab, Qaiser Mehmood, Durre Zehra, Dietrich Rebholz-Schuhmann and Ali Hasnain*
- A Linked Data Driven Visual Interface for the Multi-Perspective Exploration of Data Across Repositories, *Gengchen Mai, Krzysztof Janowicz, Yingjie Hu and Grant McKenzie*
- Exploring Visualization of Geospatial Ontologies using Cesium, *Abhishek Potnis and Surya Durbha*

Session II: Visualization in Ontology Engineering

- Starting Ontology Development by Visually Modeling an Example Situation - A User Study, *Marek Dudáš, Vojtěch Svátek, Miroslav Vacura and Ondřej Zamazal*
- Advanced UML Style Visualization of OWL Ontologies, *Jūlija Ovčiņņikova and Kārlis Čerāns*
- Extending Ontology Visualization with Interactive Contextual Verbalization, *Uldis Bojārs, Renārs Liepiņš, Normunds Grūzītis, Kārlis Čerāns and Edgars Celms*
- Visualizing User Editing Behavior in Collaborative Ontology-Engineering Projects, *Simon Walk, Tania Tudorache and Mark Musen*
- Visualization for Ontology Evolution, *Patrick Lambrix, Zlatan Dragisic, Valentina Ivanova and Craig Anslow*

Session III: Visualization of SPARQL Queries and Endpoints

- A Visual Aide for Understanding Endpoint Data, *Fernando Florenzano, Denis Parra, Juan L. Reutter and Freddie Venegas*
- LD-VOWL: Extracting and Visualizing Schema Information for Linked Data Endpoints, *Marc Weise, Steffen Lohmann and Florian Haag*
- ViziQuer: Notation and Tool for Data Analysis SPARQL Queries, *Kārlis Čerāns and Jūlija Ovčiņņikova*
- SQuaRE: a Visual Support for OBDA Approach, *Michał Blinkiewicz and Jarosław Bąk*

Session IV: Visualization in Semantic Annotation

- Semantic Annotation and Information Visualization for Blogposts with refer, *Tabea Tietz, Joscha Jäger, Jörg Waitelonis and Harald Sack*
- Visual Development & Analysis of Coreference Resolution Systems with CORVIDAE, *Nico Möller and Gunther Heidemann*

Extending OWL Ontology Visualizations with Interactive Contextual Verbalization

Uldis Bojārs[1,2], Renārs Liepiņš[1], Normunds Grūzītis[1,2], Kārlis Čerāns[1,2], and Edgars Celms[1,2]

[1] Institute of Mathematics and Computer Science, University of Latvia
Raina bulvaris 29, Riga, LV-1459, Latvia
[2] Faculty of Computing, University of Latvia
Raina bulvaris 19, Riga, LV-1459, Latvia
uldis.bojars@lumii.lv, renars.liepins@lumii.lv, normunds.gruzitis@lumii.lv, karlis.cerans@lumii.lv, edgars.celms@lumii.lv

Abstract. To participate in Semantic Web projects, domain experts need to be able to understand the ontologies involved. Visual notations can provide an overview of the ontology and help users to understand the connections among entities. However, users first need to learn the visual notation before they can interpret it correctly. Controlled natural language representation would be readable right away and might be preferred in case of complex axioms, however, the structure of the ontology would remain less apparent. We propose to combine ontology visualizations with contextual ontology verbalizations of selected ontology (diagram) elements, interactively displaying controlled natural language (CNL) explanations of OWL axioms corresponding to the selected visual notation elements. Thus, the domain experts will benefit from both the high-level overview provided by the graphical notation and the detailed textual explanations of particular elements in the diagram.

Keywords: OWL · Ontology visualization · Contextual verbalization · Controlled natural language

1 Introduction

Semantic Web technologies have been successfully applied in pilot projects and are transitioning toward mainstream adoption in the industry. In order for this transition to go successfully, there are still hurdles that have to be overcome. One of them are the difficulties that domain experts have in understanding mathematical formalisms and notations that are used in ontology engineering.

Visual notations have been proposed as a way to help domain experts to work with ontologies. Indeed, when domain experts collaborate with ontology experts in designing an ontology "they very quickly move to sketching 2D images to communicate their thoughts" [8]. The use of diagrams has also been supported by an empirical study done by Warren et al. where they reported that "one-third [of participants] commented on the value of drawing a diagram" to understand what is going on in the ontology [21].

Despite the apparent success of the graphical approaches, there is still a fundamental problem with them. When a novice user wants to understand a particular ontology, he or she cannot just look at the diagram and know what it means. The user first needs to learn the syntax and semantics of the notation – its mapping to the underlying formalism. This limitation has long been noticed in software engineering [18] and, for this reason, formal models in software engineering are often translated into informal textual documentation by systems analysts, so that they can be validated by domain experts [4].

A similar idea of automatic conversion of ontologies into seemingly informal controlled natural language (CNL) texts and presenting the texts to domain experts has been investigated by multiple groups [14,20,9]. CNL is more understandable to domain experts and end-users than the alternative representations because the notation itself does not have to be learned, or the learning time is very short. Hoverer, the comparative studies of textual and graphical notations have shown that while domain experts that are new to graphical notations better understand the natural language text, they still prefer the graphical notations in the long run [13,17]. It leads to a dilemma of how to introduce domain experts to ontologies. The CNL representation shall be readable right away and might be preferred in case of complex axioms (restrictions) while the graphical notation makes the overall structure and the connections more comprehensible.

We present an approach that combines the benefits of both graphical notations and CNL verbalizations. The solution is to extend the graphical notation with interactive contextual verbalizations of the axioms that are represented by the selected graphical element. The graphical representation gives the users an overview of the ontology while the contextual verbalizations can explain what the particular graphical element means. Thus, domain experts that are novices in ontology engineering shall be able to learn and use the graphical notation rapidly and independently without special training.

Throughout this paper we refer to the OWLGrEd visual notation and ontology editing and visualization tools that are using it. The OWLGrEd notation [1] is a compact and complete UML-style notation for OWL 2 ontologies. It relies on Manchester OWL Syntax [7] for certain class expressions. This notation is implemented in the OWLGrEd ontology editor[3] and its online ontology visualization tool[4] [11]. The approach proposed in this article is demonstrated on an experimental instance of the OWLGrEd visualization tool.

In Section 2, we present the general principles of extending graphical ontology notations with contextual natural language verbalizations. In Section 3, we demonstrate how the proposed approach may be used in practice by extending ontology visualizations in the OWLGrEd notation with interactive contextual verbalizations in controlled English. In Section 4, we discuss the benefits and limitations of our approach. Section 5 discusses related work while Section 6 summarizes the article.

[3] `http://owlgred.lumii.lv`
[4] `http://owlgred.lumii.lv/online_visualization/`

2 Extending Graphical Notations with Contextual Verbalizations

This section describes the proposed approach for contextual verbalization of graphical elements in ontology diagrams, starting with a motivating example. We are focusing particularly on OWL ontologies, assuming that they have already been created and that the ontology symbols (names) are lexically motivated and consistent, i.e., in this paper we do not consider the authoring of ontologies, although the contextual verbalizations might be helpful in the authoring process as well and it would provide a motivation to follow a lexical and consistent naming convention.

2.1 Motivating Example

In most diagrammatic OWL ontology notations, object property declarations are shown either as boxes (e.g. in VOWL [12]) or as labeled links connecting the property domain and range classes (e.g. in OWLGrEd [1] or Graffoo [3]). Figure 1 illustrates a simplified ontology fragment that includes classes *Person* and *Thing*, an object property *likes* and a data property *hasAge*. This fragment is represented by using three alternative formal notations: Manchester OWL Syntax [7], VOWL and OWLGrEd. As can be seen, the visualizations are tiny and may already seem self-explanatory. Nevertheless, even in this simple case, the notation for domain experts may be far from obvious. For example, the Manchester OWL Syntax uses the terms *domain* and *range* when defining a property, and these terms may not be familiar to a domain expert. In the graphical notations, the situation is even worse because the user may not even suspect that the edges represent more than one assertion and that the assertions are far-reaching. In the case of *likes* in Figure 1, it means that everyone that likes something is *necessarily* a person.

We have encountered such problems in practice when introducing ontologies visualized using the OWLGrEd notation to users familiar with the UML notation. Initially, the users were misunderstanding the meaning of the association edges. For example, they would interpret that the edge *likes* in Figure 1 means "persons *may* like persons", which is true, however, they would also assume that other disjoint classes could have this property, which is false in OWL because multiple domain/range axioms of the same property are combined to form an intersection. Thus, even having a very simple ontology, there is a potential for misunderstanding the meaning of both the formal textual notation (e.g., Manchester OWL Syntax) and the graphical notations.

2.2 Proposed Approach

We propose to extend graphical ontology diagrams with contextual on-demand verbalizations of OWL axioms related to the selected diagram elements, with the goal to help users to better understand their ontologies and to learn the graphical notations based on their own and/or real-world examples.

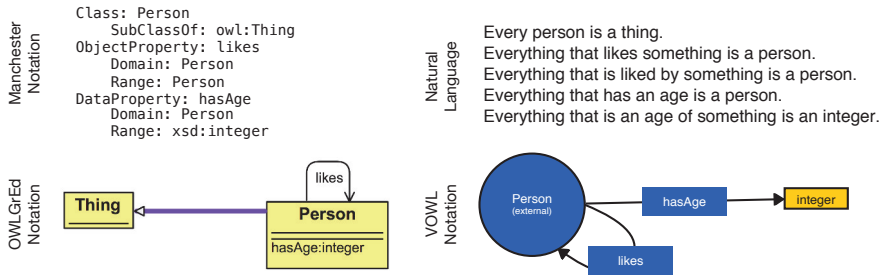


Fig. 1. A simplified ontology fragment alternatively represented by using Manchester OWL Syntax, VOWL and OWLGrEd, and an explanation in a controlled natural language

The contextual verbalization of ontology diagrams relies on the assumption that every diagram element represents a set of ontology axioms, i.e., the ontology axioms are generally presented locally in the diagram, although possibly a single ontology axiom can be related to several elements of the diagram.

The same verbalization can be applied to all the different OWL visual notations, i.e., we do not have to design a new verbalization (explanation) grammar for each new visual notation, because they all are mapped to the same underlying OWL axioms. Thus, the OWL visualizers can reuse the same OWL verbalizers to provide contextual explanations of any graphical OWL notation.

By reusing ontology verbalizers, existing ontology visualization systems can be easily extended with a verbalization service. Figure 2 illustrates how the proposed approach might work in practice:

1. *Visualizer* is the existing visualization component that transforms an OWL ontology into its graphical representation.
2. The system is extended by a *User Selection* mechanism that allows users to select the graphical element that they want to verbalize.
3. *Collector* gathers a subset of the ontology axioms that correspond to the selected graphical element.
4. The relevant axioms are passed to *Verbalizer* that produces CNL statements – a textual explanation that is shown to the user.

The actual implementation would depend on the components used and on how the output of the verbalization component can be integrated into the resulting visualization.

With the proposed approach, when domain experts encounter the example ontology in Figure 1, they would not have to guess what the elements of this graphical notation mean. Instead, they can just ask the system to explain the notation using the ontology that they are exploring. When the user clicks on the edge *likes* in Figure 1 (in either visual notation), the system would show

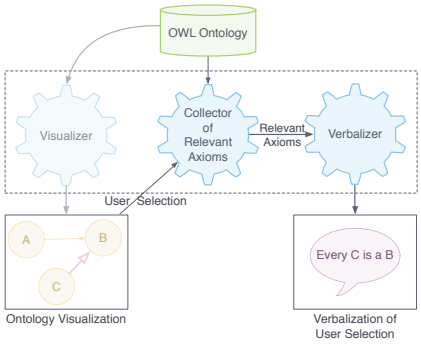


Fig. 2. Architecture of a contextual ontology verbalizer

the verbalization that unambiguously explains the complete meaning of this graphical element:

> *Everything that likes something is a person. Everything that is liked by something is a person.*

By applying the proposed approach and by using natural language to interactively explain what the graphical notation means, developers of graphical OWL editors and viewers can enable users (domain experts in particular) to avoid misinterpretations of ontology elements and their underlying axioms, resulting in a better understanding of both the ontology and the notation.

The verbalization can help users even in relatively simple cases, such as object property declarations where user's intuitive understanding of the domain and range of the property might not match what is asserted in the ontology. The verbalization of OWL axioms makes this information explicit while not requiring users to be ontology experts. The value of contextual ontology verbalization is even more apparent for elements whose semantics might be somewhat tricky even for more experienced users (e.g., *some*, *only* and cardinality constraints on properties, or OWLGrEd generalization forks with *disjoint* and *complete* constraints).

The verbalization of ontology axioms has been shown to be helpful in teaching OWL to newcomers both in practical experience reports [16] as well as in statistical evaluations [9] and thus it would be a valuable addition to ontology visualizations.

3 Proof of Concept: Interactive Verbalizations in OWLGrEd Visualizations

We illustrate the approach by extending OWLGrEd ontology visualizations (i.e. visualizations in the OWLGrEd notation used by the OWLGrEd ontology editor and its online ontology visualization tool) with on-demand contex-

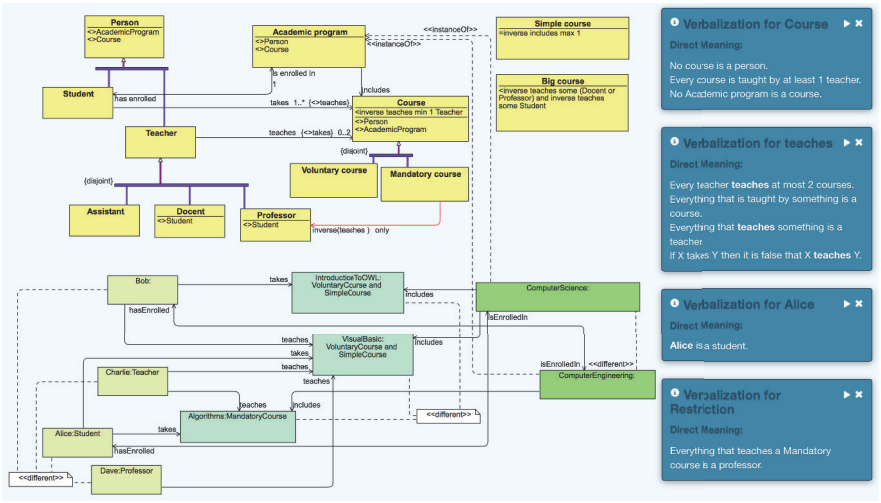


Fig. 3. The example ontology in the OWLGrEd notation with CNL verbalizations (explanations) of selected diagram elements.

tual verbalizations of the underlying OWL axioms using Attempto Controlled English (ACE) [5].

The interactive ontology verbalization layer allows users to inspect a particular element of the presented ontology diagram and to receive a verbal explanation of the ontology axioms that are related to this ontology element. By clicking a mouse pointer on an element, a pop-up widget is displayed, containing a CNL verbalization of the corresponding axioms in Attempto Controlled English. By default, the OWLGrEd visualizer minimizes the number of verbalization widgets shown simultaneously by hiding them after a certain timeout. For users to simultaneously see the verbalizations for multiple graphical elements, there is an option to "freeze" the widgets and prevent them from disappearing.

3.1 Ontology Verbalization Example

A demonstration of our approach, based on an example mini-university ontology, is available online[5]. Figure 3 shows a screenshot of the OWLGrEd visualization of this ontology containing a number of verbalizations.

These verbalizations describe the ontology elements that represent the class *Course*, the object property *teaches*, the individual *Alice* and the restriction on the class *MandatoryCourse*. Verbalizations are implicitly linked to the corresponding elements using the element labels. While it might be less convenient to identify the implicit links in a static image, the interactive nature of the combined ontology visualization and verbalization tool makes it easier for users to keep track of relations between diagram elements and their verbalizations.

[5] `http://owlgred.lumii.lv/cnl-demo`

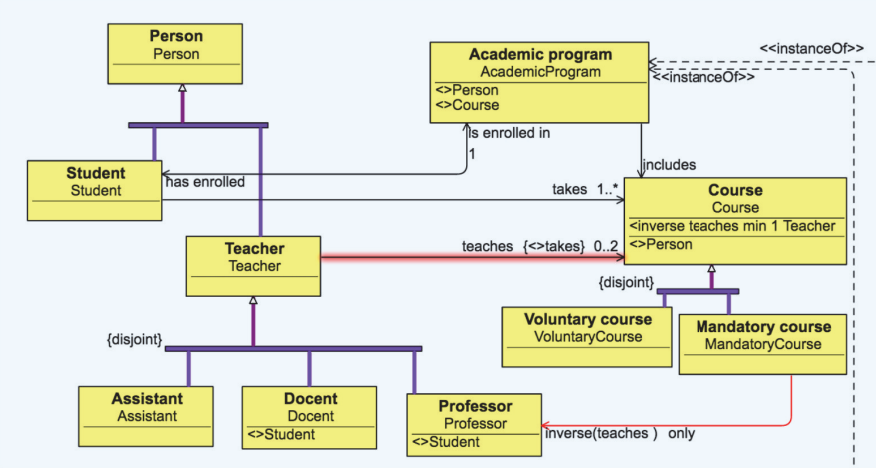


Fig. 4. A fragment of the ontology visualization example showing the *teaches* object property (highlighted).

To illustrate the verbalization functionality, let us look at the object property *teaches*, represented in the diagram by an edge connecting the class *Teacher* to the class *Course* (Figure 4). It leads to the following ACE verbalization:

(V1) *Every teacher teaches at most 2 courses.*

(V2) *Everything that is taught by something is a course.*

(V3) *Everything that teaches something is a teacher.*

(V4) *If X takes Y then it is false that X teaches Y.*

Note that the specific OWL terms, like *disjoint*, *subclass* and *inverse*, are not used in the ACE statements. The same meaning is expressed implicitly – via paraphrasing – using more general common sense constructions and terms.

In this case, the edge represents not only the domain and range axioms of the property (verbalized stentences V2, V3) but also the cardinality of its range (V1) and the restriction that *teaches* is disjoint with *takes* (expressed by the if-then statement in V4).

Further information about combining interactive contextual CNL verbalization with OWLGrEd ontology visualizations is available at [10].

4 Discussion

This section discusses the use of contextual ontology verbalization, focusing on its applicability to various graphical notations, multilinguality and the potential limitations of the approach.

4.1 Applicability to Different Visual Notations

The proposed approach is applicable to any ontology visualization where graphical elements represent one or more OWL axioms. The value of using verbalization functionality is higher for more complex notations (e.g., OWLGrEd) where graphical elements may represent multiple axioms but even in simple graphical notations, where each graphical element corresponds to one axiom, users will need to know how to read the notation. Contextual ontology verbalization addresses this need by providing textual explanations of diagram elements and the underlying OWL axioms.

A more challenging case is notations where some OWL axioms are represented as spatial relations between the elements and are not directly represented by any graphical elements (e.g., Concept Diagrams represent *subclass-of* relations as shapes that are included in one another [19]). In order to represent these axioms in ontology verbalization they need to be "attached" to one or more graphical elements that these axioms refer to. As a result, they will be included in verbalizations of relevant graphical elements. In the case of Concept Diagrams, the *subclass-of* relation, which is represented by shape inclusion, would be verbalized as part of the subclass shape.

4.2 Lexical Information and Multilinguality

In order to generate lexically and grammatically well-formed sentences, additional lexical information may need to be provided, e.g., that the property *teaches* is verbalized using the past participle form "taught" in the passive voice (*inverse-of*) constructions or that the class *MandatoryCourse* is verbalized as a multi-word unit "mandatory course".

In terms of multilinguality, contextual verbalizations can be generated for multiple languages, assuming that the translation equivalents are available for the lexical labels of ontology symbols. This would allow domain experts to explore ontologies in their native language and avoid information getting lost in translation.

An appropriate and convenient means for implementing a multilingual OWL verbalization grammar is Grammatical Framework (GF) [15] which provides a reusable resource grammar library for about 30 languages, facilitating rapid implementation of multilingual CNLs[6]. Moreover, an ACE grammar library based on the GF resource grammar library is already available for about 10 languages [2]. This allows for using English-based entity names and the OWL subset of ACE as an interlingua, following the two-level OWL-to-CNL approach suggested in [6].

We have used this GF-based approach in order to provide an experimental support for lexicalization and verbalization in OWLGrEd tools for both English and Latvian, a highly inflected Baltic language.

[6] `http://www.grammaticalframework.org/`

4.3 Limitations

Verbalization techniques that are a part of the proposed approach have the same limitations as ontology verbalization in general. In particular, verbalization may require additional lexical information to generate grammatically well-formed sentences. To some degree, by employing good ontology design practices and naming conventions as well as by annotating ontology entities with lexical labels, this limitation can be overcome. Another issue is specific kinds of axioms that are difficult or verbose to express in natural language without using terms of the underlying formalism.

5 Related Work

To the best of our knowledge, there are no publications proposing combining OWL ontology visualizations with contextual CNL verbalizations but there has been a movement towards cooperation between both fields. In ontology visualizations, notations have been adding explicit labels to each graphical element that describes what kind of axiom it represents. For example, in VOWL a line representing a *subclass-of* relation is explicitly labeled with the text "Subclass of". This practice makes the notation more understandable to users as reported in the VOWL user study where a user stated that "there was no need to use the printed [notation reference] table as the VOWL visualization was very self-explanatory" [12]. However, such labeling of graphical elements is only useful in notations where each graphical element represents one axiom. In more complex visualizations where one graphical element represents multiple axioms there would be no place for all the labels corresponding to these axioms. For example, in the OWLGrEd notation, class boxes can represent not just class definitions but also *subclass-of* and *disjoint classes* assertions. In such cases, verbalizations provide understandable explanations. Moreover, in some notations (e.g., Concept Diagrams [19]) there might be no graphical elements at all for certain kinds of axioms, as it was mentioned in Section 4.1.

In the field of textual ontology verbalizations there has been some exploration of how to make verbalizations more convenient for users. One approach that has been tried is grouping verbalizations by entities. It produces a kind of a dictionary, where records are entities (class, property, individual), and every record contains verbalizations of axioms that refer to this entity. The resulting document is significantly larger than a plain, non-grouped verbalization because many axioms may refer to multiple entities and thus will be repeated in each entity. Nevertheless, the grouped presentation was preferred by users [22]. Our approach can be considered a generalization of this approach, where a dictionary is replaced by an ontology visualization that serves as a map of the ontology.

An ad-hoc combination of verbalization and visualization approaches could be achieved using existing ontology tools such as Protégé by using separate visualization and verbalization plugins (e.g., ProtégéVOWL[7] for visualization and

[7] `http://vowl.visualdataweb.org/protegevowl.html`

ACEView[8] for verbalization). However, this would not help in understanding the graphical notation because the two views are independent, and thus a user cannot know which verbalizations correspond to which graphical elements. Our approach employs closer integration of the two ontology representations and provides contextual verbalization of axioms that directly correspond to the selected graphical element, helping users in understanding the ontology and learning the graphical notation used. The usefulness of combining ontology visualization and verbalization is also demonstrated by a course for ontology engineering that uses a CNL-based Fluent Editor and employs OWLGrEd visualizations for illustrating the meaning of CNL assertions[9].

6 Conclusions

Mathematical formalisms used in ontology engineering are hard to understand for domain experts. Usually, graphical notations are suggested as a solution to this problem. However, the graphical notations, while preferred by domain experts, still have to be learned to be genuinely helpful in understanding. Until now the only way to learn these notations was by reading the documentation.

In this article, we propose to combine ontology visualizations and CNL verbalizations in order to solve the learning problem. Using this approach the domain expert can interactively select a graphical element and receive the explanation of what the element means. The explanation is generated by passing the corresponding axioms of the element through one of the existing verbalization services. The service returns natural language sentences explaining the OWL axioms that correspond to the selected element and thus explaining what it means.

CNL explanations can help domain experts to rapidly and independently learn and use the graphical notation from the beginning, without extensive training, thus making it easier for domain experts to participate in ontology engineering thus solving one of the problems that hinder the adoption of Semantic Web technologies in the mainstream industry.

Acknowledgments. This work has been supported by the Latvian State Research program NexIT project No.1 "Technologies of ontologies, semantic web and security", the ESF project 2013/0005/1DP/1.1.1.2.0/13/APIA/VIAA/049, and the University of Latvia Faculty of Computing project "Innovative information technologies".

[8] `http://attempto.ifi.uzh.ch/aceview/`

[9] `http://www.slideshare.net/Cognitum/introduction-to-ontology-engineering-with-fluent-editor-2014/19`

References

1. Bārzdiņš, J., Bārzdiņš, G., Čerāns, K., Liepiņš, R., Sproģis, A.: UML-style graphical notation and editor for OWL 2. In: Perspectives in Business Informatics Research, pp. 102–114. Springer (2010)
2. Camilleri, J., Fuchs, N., Kaljurand, K.: ACE grammar library. Tech. Rep. Project Deliverable D11.1, MOLTO project (2012)
3. Falco, R., Gangemi, A., Peroni, S., Shotton, D., Vitali, F.: Modelling OWL ontologies with Graffoo. In: The Semantic Web: ESWC 2014 Satellite Events: ESWC 2014 Satellite Events, Anissaras, Crete, Greece, May 25-29, 2014, Revised Selected Papers. pp. 320–325. Springer (2014)
4. Frederiks, P.J., Van der Weide, T.P.: Information modeling: The process and the required competencies of its participants. Data & Knowledge Engineering 58(1), 4–20 (2006)
5. Fuchs, N.E., Kaljurand, K., Kuhn, T.: Attempto Controlled English for knowledge representation. In: Reasoning Web, pp. 104–124. Springer (2008)
6. Gruzitis, N., Barzdins, G.: Towards a more natural multilingual controlled language interface to OWL. In: Proceedings of the 9th International Conference on Computational Semantics. pp. 1006–1013 (2011)
7. Horridge, M., Patel-Schneider, P.F.: OWL 2 Web Ontology Language Manchester Syntax. W3C Working Group Note (2009)
8. Howse, J., Stapleton, G., Taylor, K., Chapman, P.: Visualizing ontologies: A case study. In: The Semantic Web – ISWC 2011, pp. 257–272. Springer (2011)
9. Kuhn, T.: The understandability of OWL statements in controlled English. Semantic Web 4(1), 101–115 (2013)
10. Liepiņš, R., Bojārs, U., Grūzītis, N., Čerāns, K., Celms, E.: Towards self-explanatory ontology visualization with contextual verbalization. In: Databases and Information Systems: 12th International Baltic Conference, DB&IS 2016, Riga, Latvia, July 4-6, 2016, Proceedings. pp. 3–17. Springer (2016)
11. Liepins, R., Grasmanis, M., Bojars, U.: OWLGrEd ontology visualizer. In: ISWC Developers Workshop 2014. CEUR (2015)
12. Lohmann, S., Negru, S., Haag, F., Ertl, T.: VOWL 2: User-oriented visualization of ontologies. In: Knowledge Engineering and Knowledge Management, pp. 266–281. Springer (2014)
13. Ottensooser, A., Fekete, A., Reijers, H.A., Mendling, J., Menictas, C.: Making sense of business process descriptions: An experimental comparison of graphical and textual notations. Journal of Systems and Software 85(3), 596–606 (2012)
14. Power, R., Third, A.: Expressing OWL axioms by English sentences: dubious in theory, feasible in practice. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters. pp. 1006–1013. Association for Computational Linguistics (2010)
15. Ranta, A.: Grammatical Framework, a type-theoretical grammar formalism. Journal of Functional Programming 14(2), 145–189 (2004)
16. Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., Wroe, C.: OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. In: Engineering Knowledge in the Age of the Semantic Web, pp. 63–81. Springer (2004)
17. Sharafi, Z., Marchetto, A., Susi, A., Antoniol, G., Guéhéneuc, Y.G.: An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension. In: 21st International Conference on Program Comprehension. pp. 33–42. IEEE (2013)

18. Siau, K.: Informational and computational equivalence in comparing information modeling methods. Journal of Database Management 15(1), 73–86 (2004)
19. Stapleton, G., Howse, J., Bonnington, A., Burton, J.: A vision for diagrammatic ontology engineering. In: Visualizations and User Interfaces for Knowledge Engineering and Linked Data Analytics. CEUR (2014)
20. Stevens, R., Malone, J., Williams, S., Power, R., Third, A.: Automating generation of textual class definitions from OWL to English. J. Biomedical Semantics 2(S-2), S5 (2011)
21. Warren, P., Mulholland, P., Collins, T., Motta, E.: The usability of Description Logics. In: The Semantic Web: Trends and Challenges, pp. 550–564. Springer (2014)
22. Williams, S., Third, A., Power, R.: Levels of organisation in ontology verbalisation. In: Proceedings of the 13th European Workshop on Natural Language Generation. pp. 158–163. Association for Computational Linguistics (2011)

PrEVIEw: Clustering and Visualising PubMed using Visual Interface

Syeda Sana e Zainab[1], Qaiser Mehmood[1], Durre Zehra[1], Dietrich Rebholz-Schuhmann[1], and Ali Hasnain[1]

Insight Centre for Data Analytics, National University of Ireland, Galway
firstname.lastname@insight-centre.org

Abstract. The life sciences domain has been one of the early adopters of Open Data Initiative and a considerable portion of the Linked Open Data cloud is comprised of datasets from Life Sciences Linked Open Data (LSLOD). This deluge of biomedical data and active research over the past decade resulted in the flux of scientific publications in this domain. PubMed resource provides access to MEDLINE, NLM's database of citations and abstracts in the biomedical domain. PubMed Central provides links to full-text articles along with publisher web sites, and other related resources. In this paper we present PubMed Visual Interface (PrEVIEw)- a web based application to access information related to *publication, research topic, author* and *institute* through a visual interface. PrEVIEw not only provides useful information e.g. research topic of interest, research collaboration at personal or institute level etc, for the biomedical research community but also helpful for the working Data Scientist. We also evaluate the usability of our system by using the standard system usability scale as well as a custom questionnaire, particularly designed to test the usability of the interface. Our overall usability score of **83.69** suggests that web based interface is easy to learn, consistent, and adequate for frequent use.

Keywords: PubMed, Publication, Visual Interface

1 Introduction

The deluge of biomedical data in the last few years, partially caused by the advent of high-throughput gene sequencing technologies [3,5,2], has been a primary motivation for efforts in this area. The significant contributors includes the Bio2RDF project[1], Linked Life Data[2], Neurocommons[3], Health care and Life Sciences knowledge base[4] (HCLS Kb) and the W3C HCLSIG Linking Open Drug Data (LODD) effort[5]. These efforts have been partially derived and are

[1] http://bio2rdf.org/ (l.a.: 2016-03-31)
[2] http://linkedlifedata.com/ (l.a.: 2016-03-01)
[3] http://neurocommons.org/page/Main_Page (l.a.: 2016-03-16)
[4] http://www.w3.org/TR/hcls-kb/ (l.a.: 2016-03-16)
[5] http://www.w3.org/wiki/HCLSIG/LODD (l.a.: 2016-07-16)

still motivated by the deluge of data in biomedical facilities in the past few years, partially caused by the decrease in price for acquiring large datasets such as genomic sequences (e.g. the cost of sequencing the genome has dropped faster than Moore's law) and the trend towards personalised medicine, pharmacogenomics and integrative bioinformatics. This increase in biomedical research has resulted in the drastic increase in scientific publications in this area. This includes both Conference as well as Journal publications. In order to investigate the ongoing research trends, similar or related research contribution and possible research collaboration one has to keep an eye on the research papers from peer scientist. There is a need for an integrated system where one can find the answers for the questions e.g. *List of all the authors who publish their research on Lungs Cancer in any conference*. With the emergence of PubMed it become possible to access the MEDLINE, NLM's database of citations and abstracts in the biomedical domain including medicine, nursing, veterinary medicine, dentistry, health care systems, and preclinical sciences. PubMed Central provides links to full-text articles along with publisher web sites, and other related resources. Using PubMed, one has to access their web interface in order to search for any publication or through RESTful Web Services. Using these RESTful services poses limitations for Biomedical researchers searching for relevant articles as they are domain user with limited or no knowledge of using such services[15,4]. In this paper we introduce PrEVIEw a web based application that uses RESTful Web Services provided by PubMed and user can search for any topic e.g. *Cancer*, and all the relevant information regarding the publications about the searched concepts along with the publication type, author and institutes involved in the *Cancer* research are retrieved through graphical and intuitive interface. This interface make it easier to cluster Authors working on similar topics or institutes involved in any particular research area.

The remaining part of this paper is organised as follows: we highlight the related work in section 2. We introduce our methodology and PrEVIEw salient features in section 3. Later we present the usage scenario in section 4. Subsequently, we present an evaluation of our approach in section 5. We finally conclude the paper.

2 Related Work

AMiner by Tang J et. al creates semantic-based profile for each researcher in ored to built researcher social network.[12] Osborne F Et. al introduce tool Rexplore which mainly focus on relating authors semantically in order to understand the dynamics of research area. [11] Monaghan F et. al introduce application Saffron which extract information from unstructured documents using Natural Language Processing techniques.[10] Wu et. al [13] visualised author collaboration network for schizophrenia disease, between year 2003 to 2012 using CiteSpace III visualisation. Xuan et. al [14], developed Medline exploration approach for interactive visualisation. They made use of PubViz, for grouping, summary, sorting and active external content retrieval functions. Similarly Joseph T et. al, introduced

TPX [7] which is a web-based PubMed search enhancement tool that enables faster article searching using analysis and exploration features in tabular form. In the book "Analysis and Visualisation of Citation Networks" Zhao et. al [16], discussed mapping research fields through citation analysis. H Chen et. al presented NLP-based text-mining approach, *"Chilibot"*, [1] which constructs content-rich relationship graph networks among biological concepts, genes, proteins, or drugs from pubmed (abstract). Table 1 shows a brief comparison of PrEVIEw with similar systems.

Table 1: Comparison of Applications based on Pubmed Data (Time Limit **TL**, Topic Limit **TLs**, Visualisation **Vs**, Explorative Search **ES**, Topic Structuring **TS**, Time Based Analysis **TbA**, Community Structure **CS**, Social Network Focused **SNF**, Genes Proteins Drugs Relations **GPDR**, Schizophrenia Disease Only for year (2003-2012), **SDO**, Tabular Output Only **TOO**, Most Cited Only **MCO**, Mediline Only **MO**)

Publications	Tool	TL	TLs	Vs	ES	TS	TbA	CS	Limitations
Tang J et. al [12]	AMiner	X	X	X	✓	✓	✓	✓	SNF
Osborne F Et. al[11]	Rexplore	X	X	X	✓	✓	✓	✓	SNF
Monaghan F et. al[10]	Saffron	X	X	X	✓	✓	✓	✓	SNF
H Chen et. al [1]	Chilibot	X	✓	✓	X	X	X	X	GPDR
Ying Wu et. al [13]	CiteSpace III	✓	✓	✓	X	✓	X	✓	SDO
Joseph T et. al[7]	TPX	X	X	X	✓	✓	X	X	TOO
Zhao et. al [16]	Citation Networks	X	X	X	X	X	X	X	MCO
Weijian Xuan et. al [14]	PubViz	X	X	✓	X	✓	X	✓	MO

The table 1 shows that PrEVIEw is not limited for searching particular topic, not confined for searching articles for specific time frame, or specific database of pubmed neither for the social networking purposes. Furthermore tabular display, can be difficult to pull off linked information while concept map approach [6] used by PrEVIEw show better interactions. PrEVIEw can also retrieved results based on most recent publications instead of most cited ones [16]. PrEVIEw also supports searching and navigation through visualisation that makes it more intuitive to use.

3 Methodology

Our methodology consists of two steps namely: 1) result retrieval using Rest API's and 2) building visual interface .

Result Retrieval Using Rest API's REST (Representational State Transfer) is a communication approach that is often used in the development of Web services[6] and involves reading any Web page that contains an XML file. PrEVIEw

[6] `http://searchsoa.techtarget.com/definition/REST`

make use of REST APIs[7] provided by PubMed and overall result retrieval works as follow: (1) User search a *topic, institute, title of the publication or author* and send the request, (2) Rest API's fetch data in the form of XML, (3) XML parser with data manipulator creates JSON file and transfer to presentation layer.

Building Visual Interface The analysis of large numbers of biomedical publications, in the form of graphical representation made it possible for researchers to extract the relevant publications for future collaboration. We chose the concept map approach [6] for building the visual interface, which is a graphical method representing the relationship between nodes and links, and has been used in various domains for organising knowledge. Using this approach we represent concepts of title, topic, authors and institutes and relevant publications. Greater the occurrence of any searched concept bigger the size of the circular node representing that concept. Due to large amount of data we only present top 100 cited occurrence for any searched concept. Subsequent results are displayed in a tabular form using *"View All Records"* option. Interface also provides the graphical representation of topic, authors or institute and respective occurrence (the number of times it occur in the result).

Technologies PrEVIEw is browser-based client application that provides a flexible front-end for searching. To build this application variety of web technologies are used including HTML5, CSS, JavaScript, JQuery[8], Rest API's [9], Java Servlet, SVG[10], AJAX[11] and JSON[12].

Availability The PrEVIEw application can be accessed at `http://srvgal78.deri.ie/PrEVIEw/`.

3.1 System Overview

Figure 1, presents the architecture of PrEVIEw that comprises of three layers namely *"Data Layer", "Business Logic Layer" and "Presentation Layer"*. Mining PubMed data brings two challenges. The first challenge is to retrieve data from PubMed, based on their terminological, author or institute content. The second challenge is to classify and visualise the results on the basis of selected field. For addressing the first challenge, at the Data Layer, we use Rest APIs, provided by PubMed to retrieve user searched term directly from PubMed resource in the form of XML. The business layer parses XML data and create respective JSON file. For example in case of topic search "cancer", XML of

[7] `http://europepmc.org/restfulwebservice`
[8] `https://jquery.com/`
[9] `http://searchsoa.techtarget.com/definition/REST`
[10] `www.w3schools.com/svg/`
[11] `http://api.jquery.com/jquery.ajax/`
[12] `http://json.org/`

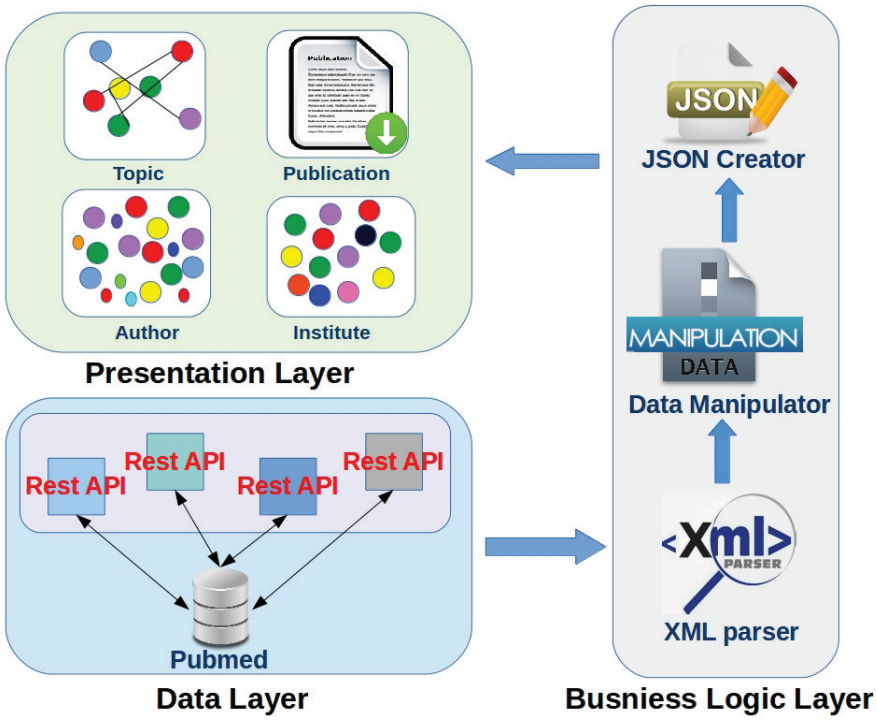


Fig. 1: PrEVIEw Architecture: Data, Business and Presentation layer.

all the relevant publications is retrieved at the *Data Layer* and corresponding JSON is generated at the *Business Layer*. At the *Presentation Layer*, graphical and visual representation of PId's (PubMed Id's), authors and institute are displayed as clusters. Complete metadata description of publications can be accessible by clicking on PId's . For the second challenge, the classification, we adopt the technique that relies on publication mapping, a web-based client application PrEVIEw is developed. Publications are mapped in the form of clusters classifying author, topic, title and institute separately. Authors and institutes with remarkable contribution in regard to particular topic search, represented as larger node. Additionally, the publications can be downloaded. We applied two mapping techniques to this application: force directed placement [8] for clusters and chart summary for highly ranked search results.

4 Usage Scenarios

Recognising the value of research networking, two example scenarios are discussed to demonstrate the use of PrEVIEw. First use-case visualises PubMed ID's and authors as the outcome from topic/title query. For second use-case,

Fig. 2: Highly cited publications with the author list on "cancer" research.

the institutional outcomes have been used. The combination of these two use-case enables establishing research networks and collaborations at authors and institution level. Details of these two outcomes have been discussed below.

Highly cited publications with the author list having paper on "cancer" research The step-by-step approach is discussed as follow (Fig. 2):

1. The first step is to specify the selection of the corresponding text. User search *"cancer"* while selecting Topic from the drop-down menu. (window A).
2. The visualisation of top 100 highly cited papers on *cancer* can be seen in (window B) where user can explore them using PubMed Id's and and description about the full paper can be explored (window D).

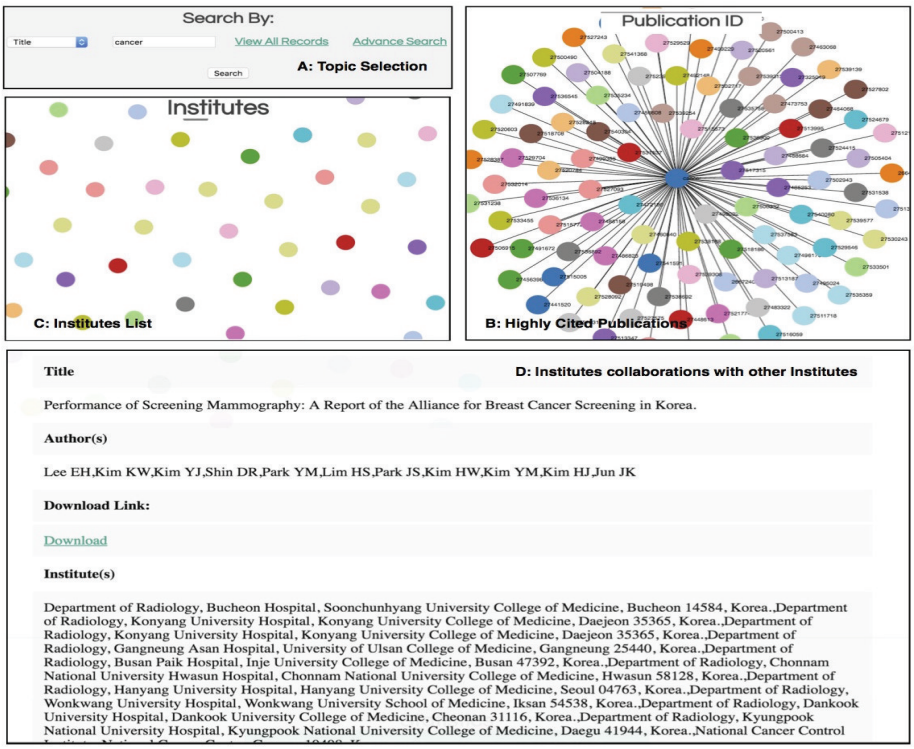


Fig. 3: research Institutes working on "cancer", and their collaborations with other institutes in the same domain.

3. List of all authors (from top 100 cited papers) working on cancer domain is shown in (window C) where a mouse-hover further display the *"author's name"*.
4. Selected author contribution in cancer research (window E) provide user the ability to download all the relevant papers in cancer domain.

Authors collaborations at international level increase citations of research manuscripts. One of the novel advancement in the mode represented in Figure 2 is establishing authors research networks and collaborations on a focused domain.

Research Institutes working on *"cancer research"*, and their collaborations with other institutes The step-by-step approach is discussed as follows(Fig. 3):

1. The first step is to specify the selection with the corresponding text. In this case *"cancer"* while selecting *"Topic"* from "drop-down menu" (window A).
2. The visualisation of highly cited cancer papers can be seen (window B) where user can explore them using PubMed Id's.

3. List of all institutes working on *"cancer research"* are shown in (window C) where user is able to explore any of them by selection.
4. Selected institute collaboration with other institutes in cancer research is shown in publication metadata, (window D).

The exponential growth in international collaboration on focused scientific research questions shows the novelty in the mode represented in Figure 3. Institutional collaboration enables shared learning, new research opportunities, establishing new research projects, joint applications for funds, and technology transfer. Findings from these different kinds of networks can be used in many ways. Collaboration networks in terms of highly cited authors , institutions, and countries are highlighted in the publication set. Semantic networks can identify various research directions the focused concepts. Further more publication citation networks can be used to quantify the citation impact in research directions and disciplines.

5 Evaluation

The goal of our evaluation is to quantify the usability and usefulness of PrEVIEw graphical interface. We evaluate the usability of the interface by using the standard *System Usability Scale* (SUS) [9] as well as a customised questionnaire designed for the users of our system. In the following, we explain the survey outcomes.

5.1 System Usability Scale Survey

In this section, we explain the SUS questionnaire[13] results. This survey is more general and applicable to any system to measure the usability. The SUS is a simple, low-cost, reliable 10 item scale that can be used for global assessments of systems usability[9].

As of 3rd May 2016, 22 users[14] including researchers and engineers in Semantic Web participated in survey. According to SUS, we achieved a mean usability score of **83.69** indicating a high level of usability according to the SUS score. The average scores (out of 5) for each survey question along with standard deviation is shown in Figure 4.

The responses to question 1 (average score to question 4.2 = ± 0.72) suggests that PrEVIEw is adequate for frequent use. The responses to question 3 indicates that PrEVIEw is easy to use (average score 4.5 ± 0.51) and the responses to question 7 (average score 4.4 ± 0.59) suggests that most people would learn to use this system very quickly. However, the slightly higher standard deviation to question 9 (standard deviation = ± 0.74) and question 10 (standard deviation

[13] SUS survey can found at: `https://goo.gl/hkuMDM`

[14] Users from AKSW, University of Leipzig and INSIGHT Centre, National University of Ireland, Galway. Summary of the responses can be found at: `https://goo.gl/iKaZjQ`

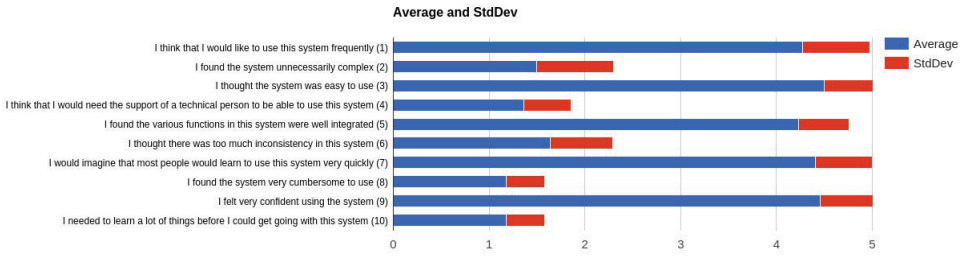


Fig. 4: SUS Evaluation

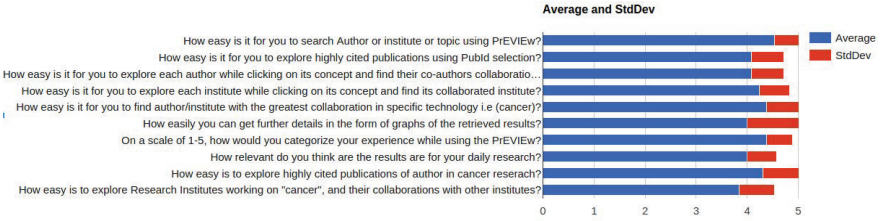


Fig. 5: Usability Evaluation

$= \pm$ 0.39) suggest that we may need a user manual to explain the different functionalists provided by the PrEVIEw interface.

5.2 Custom Survey

This survey[15] was particularly designed to measure the usability and usefulness of the different functionalists provided by PrEVIEw. In particular, we asked users to use our system and share their experience through question 9 and question 10. As of 3rd May 2016, 13 researchers including Computer Scientist[16] and Bioinformaticians have participated in survey. The average scores (out of 5 with 1 meaning strongly disagree and 5 meaning strongly agree) for each survey question along with standard deviation is shown in Figure 5. The average scores to question 10 (i.e., 4.45 $\pm$ 0.73), show that most of the users feel confidence to use the system and need not to learn much about the PrEVIEw before using. The responses to question 2 (average score = 4.07 $\pm$ 0.64) suggest that exploring highly cited publications using PubId selection is easy in PrEVIEw. A slightly lower scores to question 7 (average score = 4.38 $\pm$ 0.51) suggests that we need to further improve the user experience with visualisation components of the PrEVIEw.

[15] Custom survey can be found at: `https://goo.gl/syZzAM`
[16] Summary of the responses can be found at: `https://goo.gl/JKogDQ`

6 Conclusion and Future Work

In this paper we introduce PrEVIEw as an online visual and graphical interface for searching PubMed resource. We evaluate our approach and usability of our system using the standard system usability scale as well as through domain experts. Our preliminary analysis and evaluation revels the overall usability score of *83.69*, concluding PrEVIEw an interface, easy to learn and help users accessing PubMed resource intuitively. As a future work we aim to extend PrEVIEw with Faceted browsing and also provide visualisation at entity level e.g, Genes and Molecules where the search criteria retrieve these entities. Current work visualise top 100 cited results and in future we aim to visualise all retrieved results.

Acknowledgement

This research has been supported in part by Science Foundation Ireland under Grant Number SFI/12/RC/2289.

References

1. Chen, H., Sharp, B.M.: Content-rich biological network constructed by mining pubmed abstracts. BMC bioinformatics 5(1), 1 (2004)
2. Hasnain, A., Fox, R., Decker, S., Deus, H.F.: Cataloguing and linking life sciences LOD Cloud. In: 1st International Workshop on Ontology Engineering in a Data-driven World collocated with EKAW12 (2012)
3. Hasnain, A., Kamdar, M.R., Hasapis, P., Zeginis, D., Warren Jr, C.N., et al.: Linked Biomedical Dataspace: Lessons Learned integrating Data for Drug Discovery. In: International Semantic Web Conference (In-Use Track), October 2014 (2014)
4. Hasnain, A., Mehmood, Q., e Zainab, S.S., Decker, S.: A provenance assisted roadmap for life sciences linked open data cloud. In: Knowledge Engineering and Semantic Web, pp. 72–86. Springer (2015)
5. Hasnain, A., e Zainab, S.S., Kamdar, M.R., Mehmood, Q., Warren Jr, C.N., Fatimah, Q.A., Deus, H.F., Mehdi, M., Decker, S.: A roadmap for navigating the life sciences linked open data cloud. In: Semantic Technology, pp. 97–112. Springer (2014)
6. Jonassen, D.H., Beissner, K., Yacci, M.: Structural knowledge: Techniques for representing, conveying, and acquiring structural knowledge. Psychology Press (1993)
7. Joseph, T., Saipradeep, V.G., Raghavan, G.S.V., Srinivasan, R., Rao, A., Kotte, S., Sivadasan, N.: Tpx: Biomedical literature search made easy. Bioinformation 8(12), 578 (2012)
8. Kobourov, S.G.: Force-directed drawing algorithms (2004)
9. Lewis, J.R., Sauro, J.: The factor structure of the system usability scale. In: HCD (2009)
10. Monaghan, F., Bordea, G., Samp, K., Buitelaar, P.: Exploring your research: Sprinkling some saffron on semantic web dog food. In: Semantic Web Challenge at the International Semantic Web Conference. vol. 117, pp. 420–435. Citeseer (2010)
11. Osborne, F., Motta, E., Mulholland, P.: Exploring scholarly data with rexplore. In: International semantic web conference. pp. 460–477. Springer (2013)

12. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: extraction and mining of academic social networks. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 990–998. ACM (2008)
13. Wu, Y., Duan, Z.: Visualization analysis of author collaborations in schizophrenia research. BMC psychiatry 15(1), 27 (2015)
14. Xuan, W., Dai, M., Mirel, B., Wilson, J., Athey, B., Watson, S.J., Meng, F.: An active visual search interface for medline. In: Comput Syst Bioinform Conf. vol. 6, pp. 359–69. World Scientific (2007)
15. e Zainab, S.S., Hasnain, A., Saleem, M., Mehmood, Q., Zehra, D., Decker, S.: Fedviz: A visual interface for sparql queries formulation and execution. In: Visualizations and User Interfaces for Ontologies and Linked Data (VOILA 2015), Bethlehem, Pennsylvania, USA. (2015)
16. Zhao, D., Strotmann, A.: Analysis and Visualization of Citation Networks. Morgan & Claypool Publishers (2015)

Semantic Annotation and Information Visualization for Blogposts with *refer*

Tabea Tietz[1], Joscha Jäger[2], Jörg Waitelonis[1], and Harald Sack[1]

[1] Hasso-Plattner-Institute, Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany
firstname.lastname@hpi.de
[2] yovisto GmbH, August-Bebel-Str. 26-53, 14482 Potsdam, Germany
firstname@yovisto.com

Abstract. The growing amount of documents in archives and blogs results in an increasing challenge for curators and authors to tag, present, and recommend their content to the user. *refer* comprises a set of powerful tools focusing on Named Entity Linking (NEL) which help authors and curators to semi-automatically analyze a platform's textual content and semantically annotate it based on Linked Open Data. In *refer* automated NEL is complemented by manual semantic annotation supported by sophisticated autosuggestion of candidate entities, implemented as publicly available Wordpress plugin. In addition, *refer* visualizes the semantically enriched documents in a novel navigation interface for improved exploration of the entire content across the platform. The efficiency of the presented approach is supported by a qualitative evaluation of the user interfaces.

Keywords: visualization, annotation, named entity linking, DBpedia

1 Introduction

When searching for an arbitrary subject in weblogs or archives, users often don't get the information they are really looking for. Often they are overwhelmed with an overflow of information while sometimes the presented information is too scarce to make any use of it. Without further knowledge about the context or background of the intended subject users are easily frustrated because they either cannot handle the amount of information or they might give up because they cannot make sense of the topic at all. Furthermore, authors of online-platforms often deal with the issue to provide useful recommendations of other articles and to motivate the readers to stay on the platform to explore more of the available but most times hidden content of their blog or archive. With *refer* users are encouraged to take an active part in discovering a platform's information content interactively and intuitively, rather than just to have to read the entire textual information provided by the author. They can discover background information as well as relationships among persons, places, events, and anything related to the subject in current focus and are inspired to navigate the previously hidden information on a platform.

To enable content discovery in blogs and archives, *semantic annotations* are used to enrich texts with additional information to explain relations among entities as well as to provide meaningful content-based recommendations. However, common issues with (semantic) annotations are that their provision and maintenance is an extra effort to handle and lay-users find it rather difficult to deal with Linked Data [1]. With *refer*, content creators are enabled to (semi-)automatically annotate their text-based content with DBpedia resources as part of the original writing process and visualize them automatically. Thereby, authors can engage the readers to further explore the available content and to provide background information from DBpedia and Wikipedia without having to leave the platform.

In this paper the newly developed user interfaces of *refer* for semantic annotation and visualization together with a qualitative evaluation are presented. The goal of the evaluation is to better understand how to display entities in semantic annotation interfaces in order to support lay-users to annotate text as completely, accurately, and conveniently as possible. A preliminary user study on the proposed visualization interfaces to explore the annotated content was performed with the intention to receive insights on how to display the information to actually provide valuable additional content without overwhelming the user.

The original concept of the presented user interface and a first prototype have already been presented in [8]. This paper focuses on the achieved improvements based on two years of user experience with the implementation of the working system. The contributions of this paper include the implementation of new annotation and visualization interfaces as Wordpress-plugin, a proof-of-concept by integrating the system into a daily weblog[3], the Wordpress-plugin of *refer* publicly available for download[4], supported by a detailed user study on the proposed Linked Data annotation techniques as well as a preliminary user study on the three proposed Linked Data visualizations. All data gathered during the evaluation process is publicly available for further use[5].

The paper is structured as follows. In Section 2, related annotation and visualization systems, as well as alternative scientific approaches are discussed, followed by a detailed description of all *refer* components in Sect. 3. Sect. 4 presents the evaluation of the proposed user interfaces and the achieved results are discussed in Sect. 5. A conclusion and outlook on future work is provided in Sect. 6.

2 Related Work

Dadzie and Rowe [1] provided an in depth survey on various Linked Data visualization and exploration techniques. In this section, more recent approaches and techniques similar to *refer* are discussed and compared against our approach.

[3] `http://blog.yovisto.com`
[4] `http://refer.cx/`
[5] `http://s16a.org/refer`

Trinh et al. have proposed an autocomplete input box for manual semantic annotations, developed for content creators to provide readers with additional meta-information about the content [9]. *refer* combines automated and manual annotation to improve the annotation quality. Furthermore, in this paper also the visualization of enriched information is evaluated.

One major goal of *refer* is to enable users to explore the content of a platform actively. However, content consumers are not required to provide annotations as e.g. with Pundit [6]. *refer* leaves this task to the content authors for the following reasons: (1) Annotation quality: being an authoritative source the author knows best about information context and pragmatics. Thus, author annotations are considered as being more accurate and being provided faster and with less effort. (2) Abuse: website hosts do not have to take care of the potential abuse of the annotation interface by malicious users. (3) Linked Data complexity: lay-users often find it difficult to work with Linked Data and might easily give up when semantically annotating content written by another author. Even though most authors will also be lay-users (regarding Linked Data), they have profound knowledge about their provided content and will be able to learn how to correctly annotate their texts much faster. Pundit allows users to choose and define their own properties and knowledge bases for the annotation. However, *refer's* target users are not only journalists, professionals, and researchers, but everyone who creates content on the web, which may include travel blogs as well as websites about cooking or fashion platforms. *refer* offers less complexity and only requires the authors to annotate content with DBpedia entities.

The Poolparty thesaurus[6] plugin for Wordpress imports a SKOS thesaurus or a thesaurus from a public SPARQL endpoint and automatically links terms in blog posts to the thesaurus. On mouse-over, definitions of the linked entities are provided. The KEA named entity linking (NEL) tool [11] as part of *refer* annotates texts with DBpedia entities automatically and is supported by manual override to keep control of the content completely with the authors. In addition to the showing definitions on mouse-over as in the Poolparty thesaurus, *refer* also visualizes links to related entities in DBpedia and recommendations.

WYSIWYM by [3] is an approach for integrated visualization, exploration and authoring of unstructured and semantic content. As part of the approach, the authoring tool RDFaCE [4] and the conTEXT [5] interface were implemented, which are similar to *refer*. However, *refer* was optimized especially for lay-users and thus has to cope with different challenges compared to [3], such as, e.g. how to display semantic information to be useful for non-experts in the authoring and exploration environment. Furthermore, our evaluation differs from [3] because it includes more participants outside the academic and computer science domain.

3 *refer* Components and Infrastructure

The *refer* system consists of the following tools and visualizations, which are integrated into Wordpress. The *Annotator* is an extension of the text editing

[6] `https://wordpress.org/plugins/poolparty-thesaurus/`(accessed: June 29, 2016)

interface to create semantic text annotations based on DBpedia. *Infoboxes* are used to visualize annotations in the article view. The *Relation Browser and Recommender* visualize relationships between annotations as well as suggestions for further reading.

3.1 Annotation Tools

To create annotations, the author selects a text fraction ranging from a single word to the entire article text in the Wordpress editing environment. The author can further choose between manual and automated annotation by means of automated NEL, which is the task of computationally determining the identity of entities mentioned in the text by linking the textual mention to a knowledge-base entity. Thereby ambiguous textual mentions are algorithmically disambiguated by analysing the context of the mentions. For automated annotation, *refer* deploys the KEA-NEL [12], which implements entity linking with DBpedia entities [10]. Nevertheless, there are still errors that have to be revised via manual annotation. The user has to be supported in selecting the correct entity from the large knowledge-base, which is especially difficult for highly ambiguous textual mentions. For example, for the term 'Michigan', e.g.`dbp:Lake_Michigan`, `dbp:Michigan_wine`, `dbp:Michigan`, and many more can be considered as potential candidates. Some entity mentions yield to lists of thousands of candidates

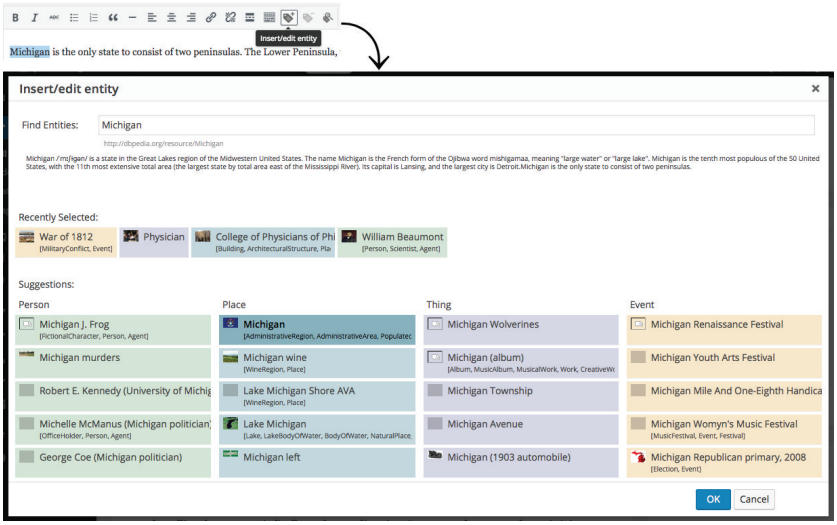


Fig. 1. Modal Annotator

which a human cannot survey quickly to find the correct one. Therefore, *auto-suggestion* utilities are applied to rank and organize the candidate lists according

to e. g. string similarity with the entity mention, or popularity of the entity [7]. But also the visual presentation of suggestions determines the users' annotation performance. To support this claim, we introduce and evaluate two different visual presentations of autosuggestion for text annotation.

The *refer* Annotator provides two configurable user interface modes: *modal* and *inline*. The **Modal Annotator** (see Fig. 1) builds upon the native TinyMCE editor controls (part of the Wordpress installation) to trigger the display of suggested entities in a modal dialog window. Upon text selection, the user can choose to open the suggestion dialog or automatically scan the selected text for entities via new buttons in the TinyMCE control panel. Entities added to the text either via manual or automated annotation can always be edited or removed by the user via a context-menu located right beside each entity in the text. The suggestion dialog starts with a text input field, which initially contains the selected text fragment and can be used to refine the search term. Suggested entities are shown below in a table-based layout, divided into the four categories Person (green), Place (blue), Event (yellow) and Thing (purple), including a list of recently selected entities for faster selection of already annotated entities in the same text. A suggested entity is displayed by its label, thumbnail, and main categories. The text abstract and entity IRI are displayed on mouseover. The selected annotation is encoded in RDFa markup, which is added to the according text fragment.

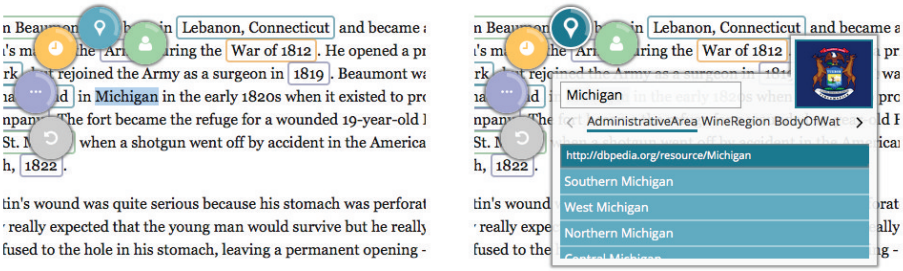


Fig. 2. Inline Annotator

The **Inline Annotator** (see Fig. 2) enables to choose entities directly in the context of a selected text. The basis of the inline annotation solution is a circular category menu attached to a text fragment upon selection and allows the user to instantly show suggestions from the respective category (Person, Place, Event, Thing). Additionally, a list of recently selected entities from all categories can be displayed. By selecting a category, the suggested entities are displayed. In order to provide more context within the relatively small space, these entities are divided into dynamically retrieved sub-categories, which are rendered horizontally as navigable tabs and are based on the list of categories

per entity provided by the DBpedia type system. The rationale of the Inline Annotator is to provide fast and simple means of semantic text annotation by minimizing the steps required to open the interface, visually scan the suggestions in several categories and to choose the most appropriate entity. Compared to the modal annotation interface, the Inline Annotator integrates directly into the text area, requires less space and preserves the context of the annotated text fragment. By combining the interactions required to open the suggestion menu and choose a category, the user is able to choose an entity more quickly. On the other hand, the modal interface leaves more space for annotations and additional information, and provides a parallel view of all categories.

3.2 Infobox Visualization

Annotated entities are indicated directly in the article text. To avoid disrupting the reading flow and visual design of the surrounding webpage, entities in the text are visualized by thin, semi-transparent, colored lines below the respective fragments. The color code indicates the same four categories (Person, Place, Event, Thing) as in the annotation interfaces. On mouseover, an infobox as in Fig. 3 is shown right below the annotated text fragment, which contains basic information about the entity, e.g. a label and thumbnail as well as additional data in a table-layout. The visual design and content of infoboxes varies per category and allows the user to gather basic facts about an entity as well as relations to other entities. While some basic information can be derived just from the webpage's RDFa microdata and is displayed instantly, additional content is asynchronously loaded from the web service once the infobox is shown for the first time. When the text fragment or any of the infobox entities are clicked, the *Relation Browser* slides down from the top of the page with the selected entity in focus.

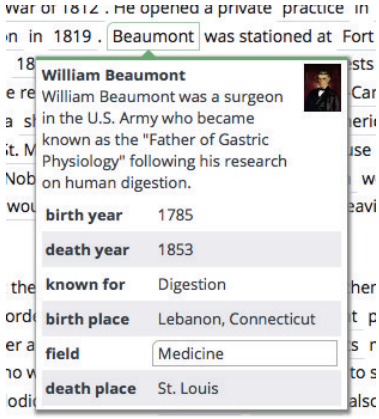


Fig. 3. Infobox

3.3 Relation Browser and Recommender

The *Relation Browser* (cf. Fig. 4) allows users to navigate and explore relations among entities. It can be opened at any time by the user either via click on the *refer* icon bar on top of the page or by selecting an entity in the article text. The rationale here is that if a user is interested in an entity annotated in the text, (e.g.

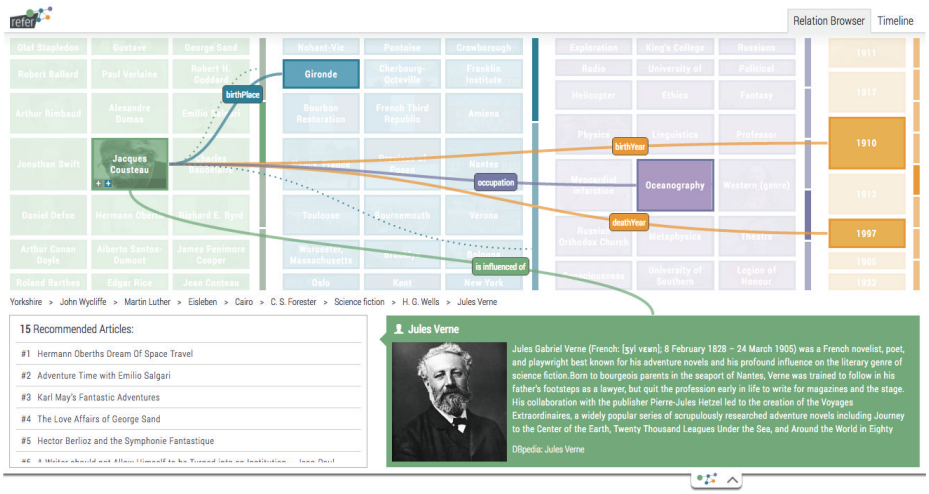


Fig. 4. Relation Browser with entity Jules Verne in focus and the Recommender on the bottom left

Jules Verne), she clicks on the entity in the text and thereby opens the Relation Browser. The entity *Jules Verne* then becomes the focus-entity. Its DBpedia abstract and image are displayed on the bottom of the Relation Browser and its background color depends on its category (Person, Place, Event, Thing). Based on the focus-entity, related entities (derived from DBpedia and all annotations available on the platform) are displayed in a four column grid.

On the right hand side of each column, pagination bars indicate the amount of further entities available within each category. Clicking the bars allows the users to browse through all entities in a category. When hovering one of the displayed entities (e.g. *Jacques Cousteau* in Fig. 4), relations to the focus-entity (e.g. *Jules Verne*) and to further entities in the grid-view (e.g. *Oceanography*) are visualized by line connectors. A label (property) indicates the direction and type of connection. If there are more entity-relations than displayed in the first overview, connections to hidden entities are indicated by dotted lines, which can be activated via hovering a small 'plus' icon inside the entity box.

A click on an entity in the grid-view replaces the focus-entity (*Jules Verne*) with the selected item and refreshes the related entities in all categories. A ranked list of recommended blogposts for the entity in focus is displayed on the bottom-left. The recommendations comprise blogposts that cover the focus-entity as well as entities related to the focus entity. The more entities are related with the entity in focus, the higher is the rank of the recommended article in the list. A recent survey on sophisticated Linked Data based recommender systems is given in [2].

4 Evaluation

A qualitative user study was performed to evaluate both annotation interfaces (Modal and Inline) as well as the visualization interfaces (Infobox, Relation Browser, Recommendation). In total, 20 participants took part in the study, aged between 21 and 45. Half of the users have a background in computer science, the others in various domains, such as teaching, biology, engineering, sports, marketing, beauty, and design including participants from the non-academic field as well.

Only 5 participants considered themselves experts with Linked Data technologies while 11 test-users had either no prior knowledge about Linked Data or had only heard about it before. All participants use the Web several times a day. Since all test-users are German native-speakers, the experiment has been performed in German language, while the user interface and annotated texts have been presented in English. Therefore, the test users had to be fluent in the English language. For each participant the experiment lasted 40 to 50 minutes and took place in a controlled environment with one interviewer present, who took notes on the participants' comments as well as their annotation and navigation behavior. The evaluation covered two parts. First, the participants were asked to annotate two consecutive text snippets with one annotation interface each. Second, the users had to solve specific tasks given in the navigation and exploration environment.

All survey sheets and evaluation results are available for download.[7]

4.1 Semantic Annotation

To find out which features are most helpful to annotate text with DBpedia entities, both annotation interfaces were tested for usability and accuracy. After a short introduction, each participant received a text paragraph containing a variety of entity-types, including persons, dates, events, places, and common nouns. Moreover, the text includes terms for which the users had to highly focus on the context of the sentence in order to disambiguate correctly. The paragraphs and interfaces alternated for each user, who annotated one text with each interface. After reading the presented paragraph, the participants were told to annotate the text as accurate, as complete, and as specific as possible. Specific in this context means that e.g. in the case of the compound *John F. Kennedy Airport*, the entire term should be annotated with one DBpedia entity `dbp:John_F._Kennedy_International_Airport` [8] instead of `dbp:John_F._Kennedy` and `dbp:Airport` separately. For each annotation task, the interviewer measured the required time. Next, the participants completed a short survey and an open interview was performed after both annotation tasks were finished. All questions concerned the understandability, readability, ease and fit of use, the ease of learning, and subjective speed and accuracy of both interfaces. A ground truth

[7] `http://s16a.org/refer`

[8] The prefix `dbp:` stands for `http://dbpedia.org/resource/`

containing correct annotations for both texts has been published previously [11] and was used to measure the annotation accuracy of all participants. The evaluation further helped to categorize common mistakes made by the users to optimize the interface in future work.

4.2 Navigation and Exploration

The second part of the user study covered all visualization and exploration interfaces. The goal was to find out how semantic information should be displayed in the context of a blogpost to make sure the enriched information is actually useful and does not overwhelm or distract the participants. As starting point of this study served an already annotated article[9]. Each user was asked 11 questions to be answered orally, including:

- What is Michael Polanyi best known for?
- How is Eugene Wigner connected with Technical University Berlin?
- Which blogpost can be recommended for the year 1902?[10]

Most answers cannot be found in the available article text, but via navigating the interface. While the participants were searching for the correct answers, the interviewer took notes on how the participants attempted to achieve the information of interest. After the task was finished the participants again completed a survey.

5 Results and Discussion

In this section, all achieved results are presented in detail followed by an in depth discussion.

5.1 Annotation Interfaces

Table 1 depicts the relative scores calculated from the Likert-type survey each user completed after using each annotation interface along with the average annotation duration per paragraph. While the participants found that the modal annotation interface was slightly easier to learn and both interfaces received the same score in terms of understandability, the inline annotator is valued slightly better in all the remaining categories.

As Table 1 shows the inline annotator just slightly achieved better results, therefore we also took into account the comments on both interfaces provided by the participants on the survey sheets and orally. Thereby, it became clearer that the inline annotator was favored by most participants in terms of usability. The participants felt that annotations can be made faster, due to its size the context of the paragraph was still available, and the interface was triggered automatically

[9] `http://blog.yovisto.com/?p=9`
[10] The complete questionnaire is available at `http://s16a.org/refer`

	Inline Annotator	Modal Annotator
Understandability	0.86	0.86
Readability	0.91	0.86
Learnability	0.97	0.98
Usability	0.86	0.87
Utility	0.79	0.77
Subjective Accuracy	0.86	0.84
Subjective Speed	0.94	0.9
Average Duration (mm:ss)	06:04	07:12

Table 1. Relative usability scores retrieved from the Likert-type questions

instead of having to click on a button to initiate entity suggestion. On the other hand, some still favored the modal interface because it provided a more complete overview of all entity categories as well as short entity descriptions. In order to measure whether one of the interfaces enabled more accurate annotations, the results from all participants are compared to the ground truth and to the automated annotation via KEA-NEL. Table 2 shows that the modal annotation interface enabled the users to annotate more accurately by 3% F-measure. Both interfaces have almost the same recall at ca. 68-69%, meaning that about 31% of annotations are missing. Consequently, the modal annotation interface exhibits a better precision (+5%). The KEA-NEL found every annotation with a recall of 100%, but the precision drops below 60%.

	Precision	Recall	F-measure
Inline	0.826	0.676	0.752
Modal	0.882	0.693	0.788
KEA-NEL	0.582	1	0.791

Table 2. Comparison of annotation accuracy between both interfaces and KEA-NEL

	Inline	Modal	Total	KEA-NEL
Missing	0.64	0.66	0.65	0
Compound Split	0.13	0.13	0.13	0.10
General/Specific	0.13	0.12	0.12	0.10
Wrong Entity	0.11	0.10	0.10	0.81

Table 3. Relative occurrence of all error-categories regarding both annotation-interfaces, overall manual annotations, and automated annotations by KEA-NEL.

All errors resulting from the manual and automated annotation have been manually classified into predefined error categories (cf. Table 3). The goal was to identify the most and least common mistakes in both interfaces which might be resolved by improving information arrangement in future versions of the interfaces. Four different error-categories have been identified: (1) Missing: terms which have not been annotated, but should have according to the ground truth. (2) Compound Split: entities such as e.g. `dbp:Nobel_Prize_in_Physics` which have been split into two separate entities, as e.g. `dbp:Nobel_Prize` and `dbp:Physics`. (3) General vs. Specific: terms for which a more general entity has

been chosen instead of a more specific one as required by the ground truth, e.g. `dbp:Army` instead of `dbp:United_States_Army`. (4) Wrong Entity: wrongly annotated entities not classified in category 1-3, such as e.g., `dbp:Michael_Polanyi_Center` as a location instead of `dbp:Michael_Polanyi` as a person. Table 3 shows that the most common mistakes for manual annotations belong to category (1), which was the least common mistake for the automated KEA-NEL and also reflects the recall-result in Table 2. Vice versa, category (4) was calculated as the least common mistake for the human annotators while it was the most frequent KEA-NEL error. In both interfaces, about 13% of all errors have been classified as a compound split error and 12% of all errors have been made because the users have chosen too general entities. Table 3 shows that humans tend to make category (2) and (3) errors, which shows that users might have difficulties to recognize compounds and specific words as annotation subject. Interestingly, the KEA-NEL detects this language phenomenon more accurately, which might be caused by the optimization of text analysis in a top-down fashion, preferring larger text fragments over single words [12]. Contrarywise, in the actual disambiguation process KEA-NEL produces much more erroneous annotations than the users. In conclusion, it seems that the most complete and accurate results might probably be achieved by a combination of automated and manual annotation. First, the automated process could 'suggest' annotations, which later can be revised by the users. Furthermore, the general vs. specific problem could be improved by recognizing the candidate lists in the autosuggestion by means of grouping specific items below general items. To improve the wrong entity rate the differences between entities should be made more clear, e.g. with sophisticated entity summaries.

5.2 Navigation and Exploration Interfaces

The preliminary user study on the navigation and exploration interfaces resulted in further insights on how Linked Data based visualizations should be presented to the users. The *Infobox* visualization was preferred the most by the participants. During the navigation task, users had no problems to find relevant information and commented positively on the way the additional information is presented. All but one participant could imagine to use the *Infobox* visualization on the Web. About 60% of the participants could imagine to use the *Relation Browser* as well. However, some users had difficulties understanding the interface. Especially the direction of line connectors was not intuitive for most lay-users. Also many users had problems finding the hidden entities in the grid-view via the pagination bars or the +icon on the bottom of an entity tile. Further comments on the *Relation Browser* revealed that by activating the *Relation Browser*, the context of the blog-post goes missing. The *Recommendation* visualization was used quite intuitively by the participants. Most users could imagine using the visualization, even though not all did understand that the recommendations are based on the specific focus-entity instead of the current blog-post as a whole.

6 Conclusion

In this paper, we have proposed two semantic annotation interfaces to enable authors to link their content with DBpedia entities as convenient, fast, and accurate as possible and Linked Data based visualizations to enable users to actively explore and navigate the entire content of a platform. The qualitative user study on both annotation interfaces resulted in the insight that the *Inline Annotator* delivered better usability results due to its compactness, faster interactions, and the possibility to better retain the context of the text. The *Modal Annotator* however enabled the users to annotate more accurately. In future research it will be investigated how to combine the best features of the modal and inline interface to enable better manual annotation and how automated and manual annotation tools can be combined to achieve the best results. The preliminary qualitative user-study on the *Infobox*, the *Relation Browser*, and the *Recommender* revealed that the *Infobox* was highly preferred by the participants and well understood while many improvements have to be made on the *Relation Browser* and *Recommender* to enable better exploration and navigation. Further research regarding the visualization tools will focus on how to display the additional information without hiding the context of the blog-post and how to enable the user to better navigate through entities which are not displayed in the first overview.

Acknowledgement: This work has been funded by the German Government, Federal Ministry of Education an Research under 03WKCJ4D.

References

1. A.-S. Dadzie and M. Rowe. Approaches to visualising linked data: A survey. *Semantic Web Journal*, 2(2):89–124, 2011.
2. T. Di Noia and V. C. Ostuni. Recommender systems and linked open data. In *Reasoning Web. Web Logic Rules. 11th International Summer School 2015*, Lecture Notes in Computer Science, pages 88–113. Springer, 2015.
3. A. Khalili and S. Auer. WYSIWYM – integrated visualization, exploration and authoring of semantically enriched unstructured content. *Semantic Web Journal*, 6(2):259–275, 2014.
4. A. Khalili, S. Auer, and D. Hladky. The RDFa content editor - from WYSIWYG to WYSIWYM. In *Proceedings of of COMPSAC 2012 - Trustworthy Software Systems for the Digital Society*, 2012.
5. A. Khalili, S. Auer, and A.-C. N. Ngomo. context – lightweight text analytics using linked data. In *11th Extended Semantic Web Conference (ESWC)*, pages 628–643. Springer, 2014.
6. C. Morbidoni and A. Piccioli. *The Semantic Web: ESWC 2015 Satellite Events*, chapter Curating a Document Collection via Crowdsourcing with Pundit 2.0, pages 102–106. Springer, 2015.
7. J. Osterhoff, J. Waitelonis, and H. Sack. Widen the Peepholes! Entity-Based Auto-Suggestion as a rich and yet immediate Starting Point for Exploratory Search. In *Proceedings of 2nd Workshop Interaction and Visualization in the Web of Data (IVDW)*. GI, 2012.

8. T. Tietz, J. Waitelonis, J. Jäger, and H. Sack. Smart media navigator: Visualizing recommendations based on linked data. In *Proceedings of the Industry Track at the International Semantic Web Conference (ISWC)*, pages 48–51, 2014.
9. T. Trinh, P. Wetz, B. Do, P. R. Aryan, E. Kiesling, and A. M. Tjoa. An autocomplete input box for semantic annotation on the web. In *Proceedings of the International Workshop on Visualizations and User Interfaces for Ontologies and Linked Data co-located with 14th International Semantic Web Conference (ISWC)*, pages 97–103. CEUR-WS, Vol. 1456, 2015.
10. R. Usbeck et al. Gerbil: General entity annotator benchmarking framework. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*, WWW '15, pages 1133–1143. ACM, 2015.
11. J. Waitelonis, C. Exeler, and H. Sack. Linked Data Enabled Generalized Vector Space Model to Improve Document Retrieval. In *Proceedings of NLP & DBpedia 2015 workshop in conjunction with 14th International Semantic Web Conference (ISWC)*. CEUR-WS, Vol. 1486, 2015.
12. J. Waitelonis and H. Sack. Named entity linking in #tweets with kea. In *Proceedings of 6th workshop on 'Making Sense of Microposts', Named Entity Recognition and Linking (NEEL) Challenge in conjunction with 25th International World Wide Web Conference (WWW)*, 2016.

SQuaRE: A Visual Support for OBDA Approach

Michał Blinkiewicz and Jarosław Bąk

Institute of Control and Information Engineering,
Poznan University of Technology,
Piotrowo 3a, 60-965 Poznan, Poland
firstname.lastname@put.poznan.pl

Abstract. We present SQuaRE, the SPARQL Queries and R2RML mappings Environment which provides a visual approach for creating R2RML mappings which can be immediately tested by executing SPARQL queries. SQuaRE is a web-based tool with easy to use visual interface that can be applied in the ontology-based data access applications. We describe SQuaRE's main features, its architecture as well as implementation details. We present an example use case to indicate how SQuaRE can be useful in an OBDA-based scenario.

1 Introduction

Ontologies, as a way of expressing knowledge, are becoming more and more popular in various research and practical fields. They allow definition of a knowledge base using abstract concepts, properties and relations between them. A properly created ontology can be then automatically processed to obtain new inferences and, as a result, a newer version of the knowledge base. Ontologies can be expressed in the Web Ontology Language 2 (OWL 2) [11]. This is a well-known format of ontologies and widely used. Ontologies require data to be in a format of RDF[1] triples. Then, using an appropriate reasoner we can obtain new data in the same format. Moreover, we can query such RDF data using SPARQL[2] queries. In such a way we are adding semantics to data and as a consequence, may obtain semantic results by executing SPARQL queries.

Nevertheless, ontologies and data need to follow the RDF-based representation. Since most of data are stored in different formats, any application of an OWL/OWL2 ontology rises the integration problem between an ontology and stored data. In this case we can transfer our current data format into RDF-based representation and change our software and architecture environment or we can create mappings between the ontology and our data and use an appropriate tool that handles such a solution. The first option is very cost-expensive and needs a lot of changes in the current software architecture. The second approach is easier and cheaper since minor changes in the software architecture are required. We need to create mappings and then query non-RDF data with SPARQL using

[1] https://www.w3.org/RDF/
[2] https://www.w3.org/TR/sparql11-overview/

ontology, mappings and a tool that enables on-the-fly transfer from non-RDF into RDF data. In this method the most important part is to create appropriate mappings. Currently, a very popular standard for expressing mappings from relational databases to RDF data is W3C's R2RML[3]. The standard allows to use existing relational data in the RDF data model, and then use SPARQL to query such data.

In this paper we provide a detailed description of SQuaRE, the SPARQL Queries and R2RML mappings Environment, which provides a visual editor for creating and managing R2RML mappings as well as for creating and executing SPARQL queries. SQuaRE is a web-based application that simplifies the creation of mappings between a relational database and an ontology. It also enables to test created mappings by defining and executing SPARQL queries in a textual manner.

The remainder of this paper is organized as follows. Firstly, we provide preliminary information, then we describe main features of SQuaRE. Next, we present its architecture as well as implementation details. Then, we provide a simple instruction on how to use SQuaRE by presenting an example use case. Finally, we provide conclusions along with future development plans.

2 Preliminaries

SQuaRE is an OBDA-oriented tool which helps an inexperienced user to create mappings between a relational database and an ontology, and then to test those mappings by creating SPARQL queries. Moreover, the tool can be used to write and execute SPARQL queries in a text-based form whereas results are presented in a graphical way. In this section we present the main overview of OBDA and R2RML.

Ontology-based Data Access (OBDA) is an approach [12] to separate a user from data sources by means of an ontology which can be perceived as a conceptual view of data. Moreover, by using concepts and relations from the ontology one can define a query in a convenient way. In this case the user operates on a different abstract level than data source. As a result the user defines queries using concepts and relations from the domain of interest and creates complex semantic conditions instead of expressing queries in terms of a relational data model. Nevertheless, in order to use OBDA approach with relational data one needs to develop mappings between a relational database and an ontology.

The R2RML recommendation provides a language for expressing mappings from a relational database to RDF datasets. Those mappings allow to view the relational database as a virtual RDF graph. Then, the relational database can be queried using the SPARQL language. Each R2RML mapping is a triples map (an RDF graph) that contains: a logic table (which can be a base table, a view or a valid SQL query); a subject map which defines the subject of all RDF triples that will be generated for a particular logical table row; and a set

[3] `https://www.w3.org/TR/r2rml/`

of predicate-object maps that define the predicates and objects of the generated RDF triples. In order to create R2RML mappings manually, one needs to know about ontologies (OWL/OWL2), RDF, R2RML and SQL at the same time.

SQuaRE overcomes the aforementioned issues. The main goal of the tool is to support creation of R2RML mappings and SPARQL queries in a graphical manner. However, at the current state of development SQuaRE supports a graphical editor for R2RML mappings and a text-based interface for creating and executing SPARQL queries. Nevertheless, results returned by a SPARQL query may be presented as a graph to a user.

3 SQuaRE, the SPARQL Queries and R2RML Mappings Environment

3.1 Features

The SQuaRE environment is aimed at providing easy-to-use functions that will support creation and execution of SPARQL queries as well as creation of R2RML mappings. Moreover, SQuaRE allows for management of queries and mappings. A user can save both: mappings and queries for future reference, execution and management. Currently, the tool supports a graphical interface for the creation of mappings and the text-based creation of SPARQL queries. Moreover, results returned by a SPARQL query can be presented in a graphical way (according to the query type). Currently, SPARQL results of queries DESCRIBE and CONSTRUCT are presented on a graph, ASK queries return 'yes' or 'no' while results of SELECT queries are presented in a table.

Nevertheless, SQuaRE provides the following useful features (appropriate figures are shown in Section 4):

1. Browsing a relational database – a user can choose a data source and browse its schema. In this view the user sees table names, column names as well as data types stored in each column. An example view of a relational database is shown in Figure 2. The figure contains two tables named `emp` and `dept` with four and three columns, respectively.
2. Browsing an OWL ontology – a user sees hierarchies of classes, object properties and datatype properties. The user can browse an ontology (Figure 3) and search for its elements.
3. Browsing mappings – a list of all created mappings is shown to a user. The user can choose a mapping and then edit it in the mapping creation view (shown in Figure 5).
4. Graphical creation of R2RML mappings – in a mapping creation view a user can create R2RML mappings. The user needs to choose tables that are going to be mapped. Then, she/he needs to search for an appropriate classes and properties to create mappings using a graphical interface. An example mapping is shown in Figure 5. Classes are represented with an orange background, datatype properties with a green background and object properties with a blue background.

5. Management of R2RML mappings – each created mapping can be saved for future reference. A user can delete mappings, correct them or generate an R2RML file that contains all or selected mappings.
6. Textual creation of SPARQL queries – current version of SQuaRE provides an option to create SPARQL queries using a text-based interface. A user can write and execute a query. The view of a user interface for creating queries is shown in Figures 6, 7, 8, 9 and 10. Graphical editor for creating queries is one of our future development plans.
7. Management of SPARQL queries – each constructed SPARQL query can be saved and used in future. A user can execute, delete or export a SPARQL query. Moreover, the user can select few queries (or all of them) and generate a separate .txt file that contains their definitions.
8. Execution of SPARQL queries – created SPARQL queries can be executed and results are shown in a form of a table, an RDF graph or answer 'yes' or 'no'.

The aforementioned main list of features provides an intuitive ontology-based access to relational data. Moreover, by exporting functionality (importing features are still in development) a user can use SQuaRE to create mappings and test them by creating SPARQL queries, and then save everything into external files. This allows to import queries and mappings into another tool that supports both SPARQL and R2RML.

3.2 Architecture and Applied Tools

SQuaRE is developed in Java as a web application. The architecture of SQuaRE is presented in Figure 1. The tool consists of modules that provide different functionality.

The SQuaRE architecture is divided into two separate sides – client and server. The client side consists of modules working on a client's machine in a user's web browser and provides a presentation layer.

The main module, from the user's point of view, is Visual R2RML Mapper which provides tools for visual (graph based) mappings of relational database metadata, such as table columns, and user provided ontology entities.

Moreover, there are modules responsible for data source configuration and ontology management. The former allows the user to configure a data source by providing DBMS, host location and port, username and password. The latter provides an interface to import an ontology and browse hierarchies of classes and properties.

The server-side modules consist of Data Source and DBMS Manager which manages the user defined data sources and provides JDBC-based access, Ontology Handler for OWL ontology processing, and SPARQL Query Executor which utilizes already defined mappings and allows to execute SPARQL queries in the context of a relational database.

SQuaRE applies well-known tools to handle OWL ontologies, relational data and SPARQL queries. The main tools that SQuaRE uses are the following:

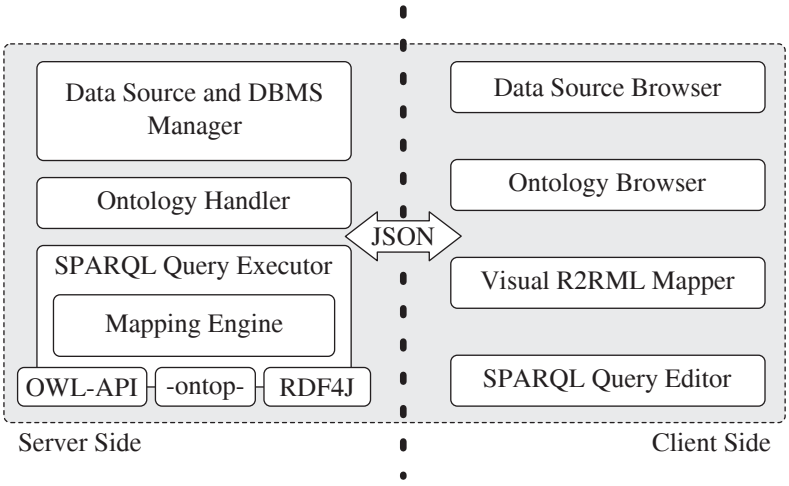


Fig. 1: The architecture of SQuaRE.

- OWL-API [7] – this is a very often used Java library to handle OWL/OWL2 ontologies. It contains a lot of features that are useful when manipulating ontology elements, using reasoner or serialising ontologies. The tool is open source[4].
- The Spring Framework[5] – it is an application framework for the Java platform. Among others, it allows for easy creation of RESTful web services and building backend API. The above mentioned features are heavily used by SQuaRE server side modules interface. In order to simplify application deployment and development Spring Boot module is used. It provides convention-over-configuration solution for stand-alone, production-ready applications. Moreover, Spring Boot embeds Tomcat or Jetty web application server which enables creation of self-contained application.
- -ontop-[6] – it is a platform to query relational databases as virtual RDF graphs using SPARQL. The tool accepts mappings in R2RML and its own OBDA mapping language. SQuaRE uses -ontop- to query relational database using mappings and SPARQL.
- RDF4J[7] – it is a framework for processing RDF data. It enables to parse, store, infer and query of/over such data. The tool supports SPARQL in version 1.1 and is used in many third party storage applications. SQuaRE uses it to save all data connected with created mappings and queries.

[4] `http://owlapi.sourceforge.net/`
[5] `http://projects.spring.io/spring-framework/`
[6] `http://ontop.inf.unibz.it/`
[7] `http://rdf4j.org/`

– Javascript libraries – we use a set of popular Javascript tools such as: AngularJS[8], jQuery[9], Cytoscape.js[10] with CoSE Bilkent layout [6], jsPlumb[11] and jsTree[12] in order to provide visual and interactive functions.

4 Example Use Case

In order to present the usability of SQuaRE we provide an example use case and a simple instruction how to use the tool. We used the W3C's R2RML [4] examples which are easy to understand and show fundamental capabilities of the presented tool.

After creating a new project (here named "W3C (1) Project") user needs to configure a data source providing DBMS, a database location and name, username and password. When all settings are correct the data source view may look like in Figure 2. It consists of two tables named `emp` and `dept` with

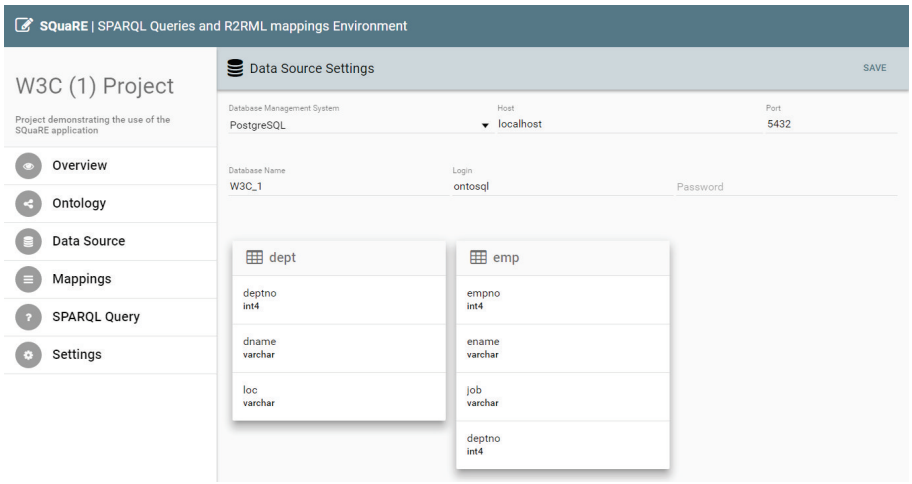


Fig. 2: Data source view of the W3C (1) Project.

information about employees and departments. Each table contains one row which content may be seen in Section 2.1 of [4].

After connecting to a database, the second task is to import OWL domain ontology which then is processed and presented to the user in a form of hierarchies of classes, object type properties and datatype properties (Figure 3).

[8] `https://angularjs.org/`
[9] `https://jquery.com/`
[10] `http://js.cytoscape.org/`
[11] `https://jsplumbtoolkit.com/`
[12] `https://www.jstree.com/`

When data source and ontology are selected the user is able to start creating new mappings. The mapping process is preceded by choosing tables (Figure 4) which will be used during the creation of a particular mapping.

Fig. 3: Ontology view of the W3C (1) Project.

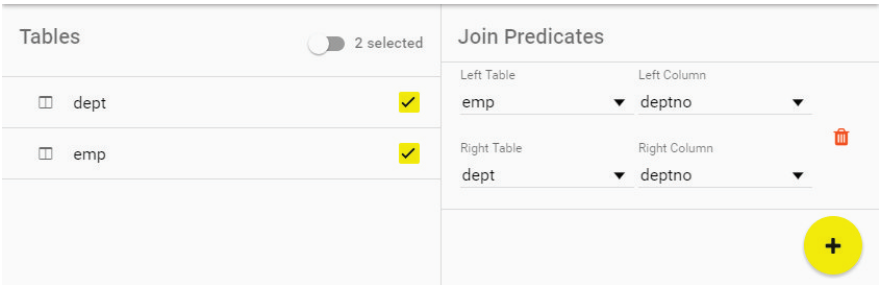


Fig. 4: Tables selection and specification of join predicates.

If the user selects more than one table she/he should define join predicates (Figure 4) which will be used for proper joining records from the selected tables.

The main mapping part consists of selecting classes, object type properties and/or datatype properties from the right-hand side ontology view shown in Figure 5). The user may drag and drop them into the center located canvas. Then she/he may link selected ontology entities with each other as well as with the left-hand side located columns descriptions of particular data source tables.

A complete mapping of W3C's R2RML example (sections 2.1 to 2.5 inclusive of [4]) is presented in Figure 5. Then the last step before the execution of SPARQL queries is to save the mapping.

Based on the aforementioned data source, ontology and mapping the user may begin executing SPARQL queries. SQuaRE is capable of executing all four SPARQL query forms. The `SELECT` form returns variables and their bindings which are presented in a form of a table (shown in Figure 6).

The SPARQL `DESCRIBE` and `CONSTRUCT` forms return RDF graphs which are presented in a form of graphs where orange color indicates objects, grey

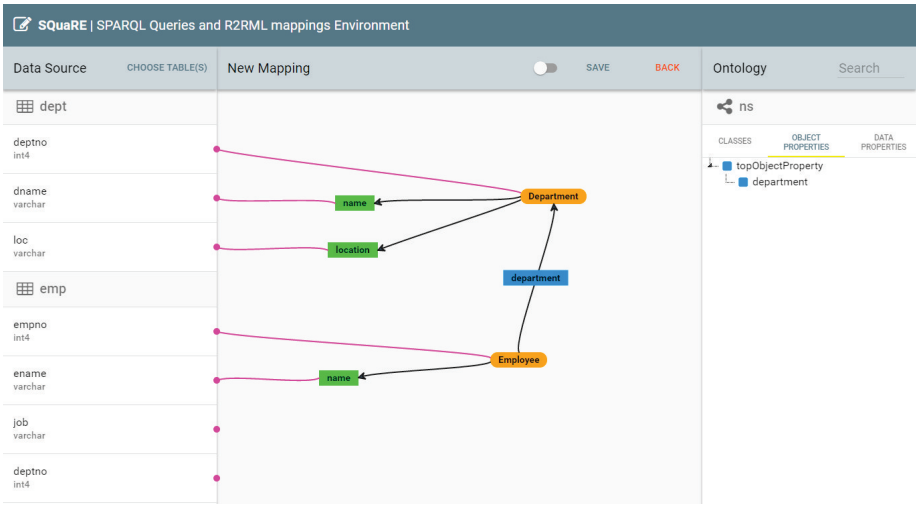


Fig. 5: Mapping creation view of W3C (1) Project.

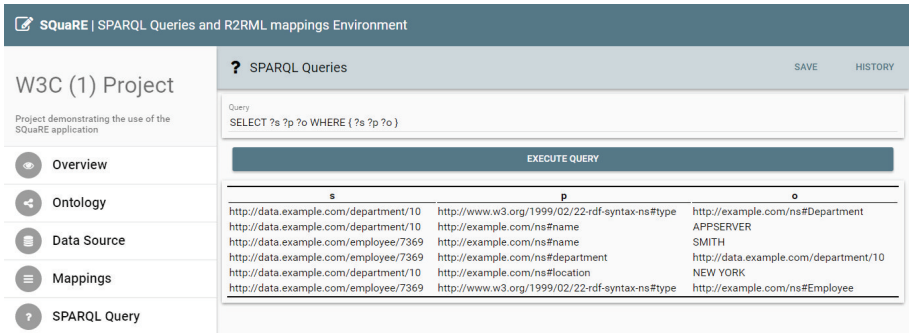


Fig. 6: The SPARQL SELECT query result.

color indicates literals, blue and green colors indicates object type properties and datatype properties, respectively (shown in Figures 7 and 8).

The SPARQL `ASK` query form returns boolean answer indicating whether a query pattern matches or not. Boolean query results are presented in Figure 9 (positive) and in Figure 10 (negative).

The W3C R2RML Recommendation's section 2.6 example [4] shows mapping of many-to-many relationship. The description includes a sample tables with exemplary contents. It also includes R2RML mappings and expected output triples. The above-mentioned mappings mapped with the use of SQuaRE application are shown in Figure 11. The example SPARQL `CONSTRUCT` query result is shown in Figure 12.

As a result we obtained an environment that is ready to be used by an inexperienced user.

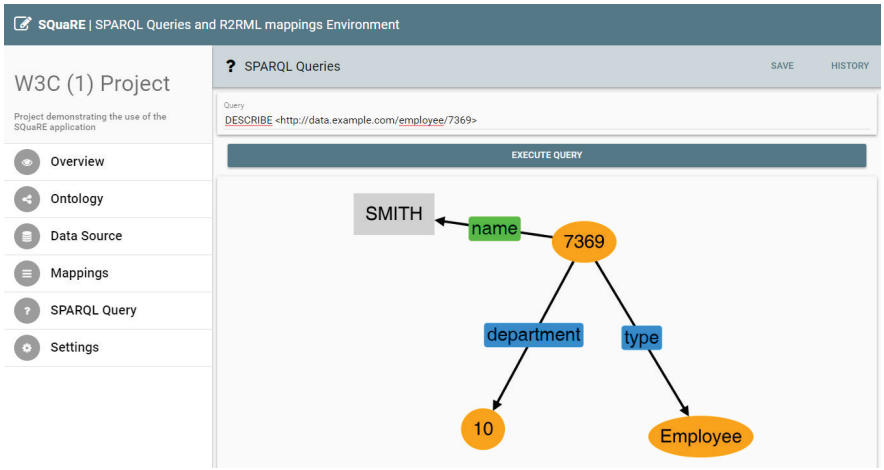


Fig. 7: The SPARQL DESCRIBE query result.

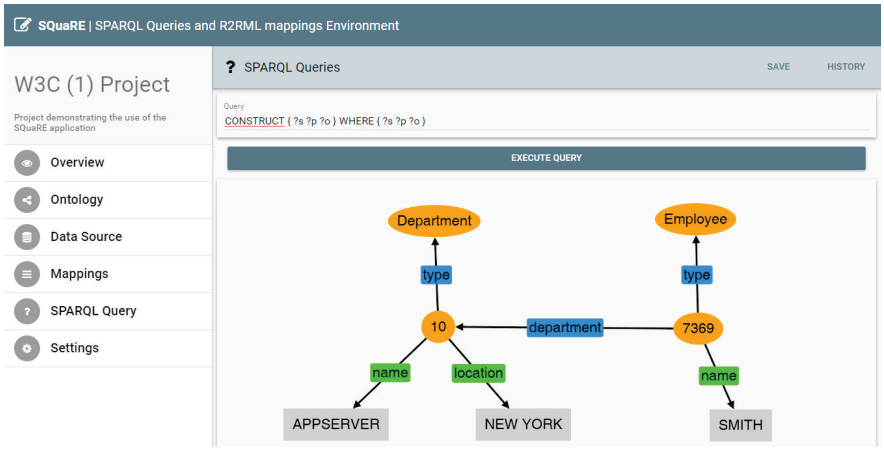


Fig. 8: The SPARQL CONSTRUCT query result.

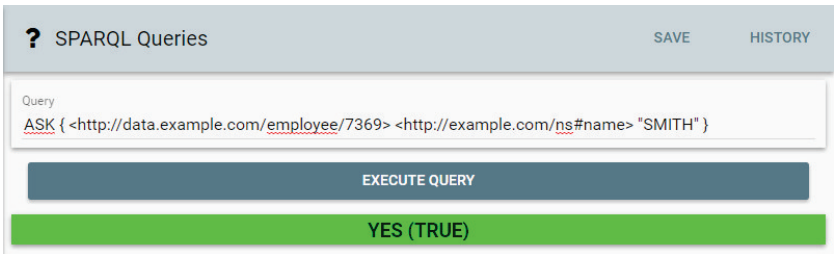


Fig. 9: The SPARQL ASK query result with positive answer.

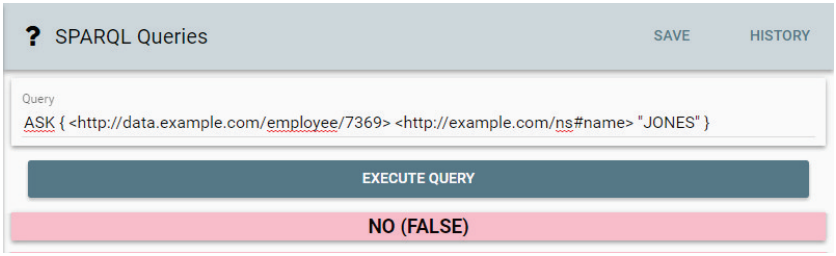


Fig. 10: The SPARQL ASK query result with negative answer.

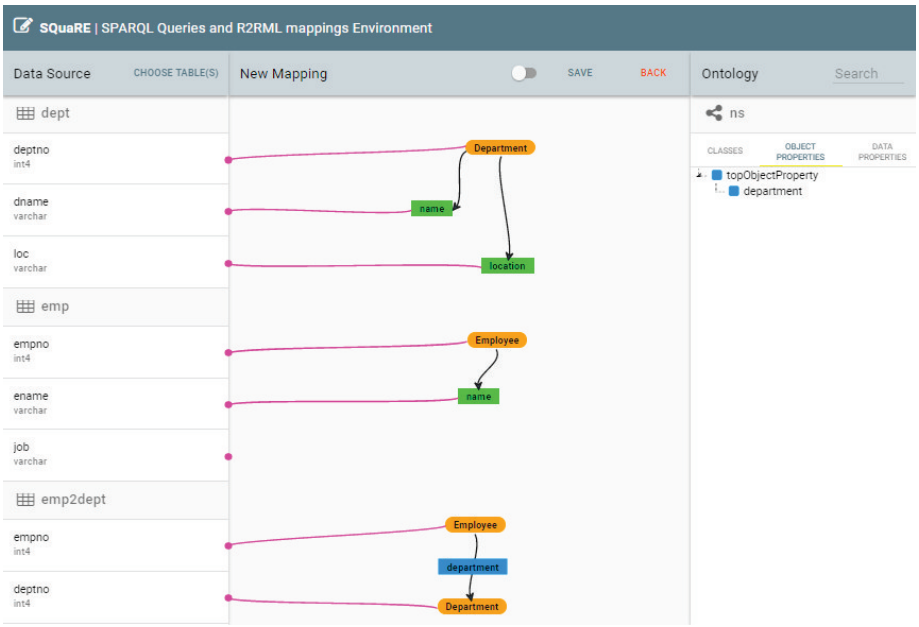


Fig. 11: The mappings of many-to-many relationship.

5 Related Tools

Several tools have been implemented to support a user in defining mappings between data sources and ontologies. We provide the list of the most similar tools to SQuaRE:

– OntopPro [3] – it provides a mapping editor inside Protégé. It allows to create mappings and to execute SPARQL queries. We can also generate RDF data according to an ontology, mappings and a data source. Users need to fill special templates in order to create mappings. In this case when creating mappings the user needs to understand how they work and how to define them. There is no graphical layout of mappings.

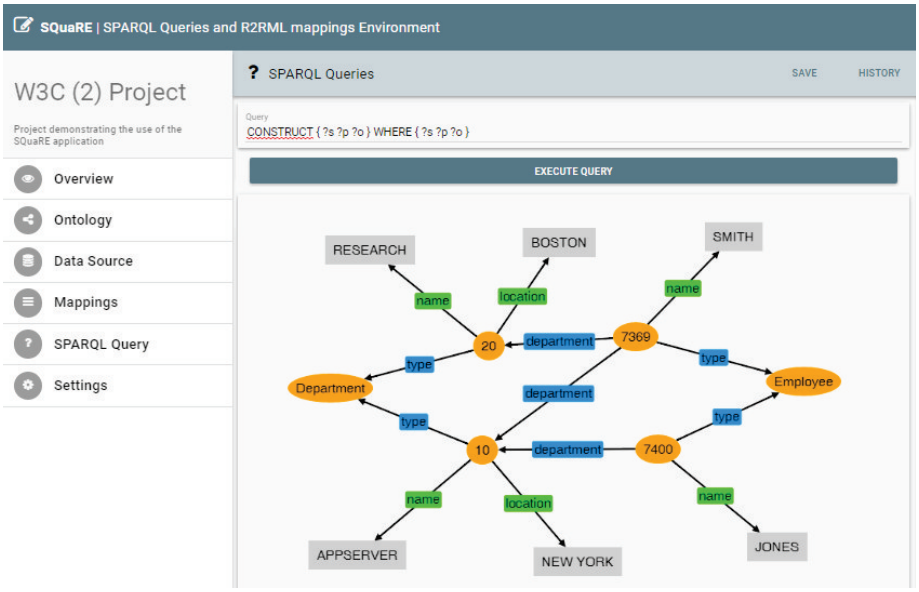


Fig. 12: The mappings of many-to-many relationship.

- Map-On [14] – it provides a graph layout for creating mappings as well as viewing ontologies and databases. This kind of representation is very convenient but when manipulating a number of mappings it is quite easy to be lost which element comes from an ontology and which one from a database. Nevertheless, the tool is very handy and easy to use.
- ODEMapster [13] – it provides a method of creating mappings in a graphical way. Mappings are expressed in the R2O language[1]. It supports OWL and RDF(S) ontologies (not OWL2) and MySQL/Oracle databases. The tool supports a tree graphical layout for database schema and ontology.
- Karma [9] – it is a web application that provides a graphical interface for creating and managing mappings between ontology and different data sources (relational databases, JSON, CSV etc.). It supports R2RML recommendation and tree layout of an ontology. However, data sources are represented with a table-like layout. Mappings are represented as a graph.
- RBA (R2RML By Assertion) [10] – it supports a tree layout for displaying databases and ontologies. R2RML mappings can be created by defining SQL queries (aka views) and then generating appropriate mappings. However, we are not able to see a graphical form of mappings.

The aforementioned tools provide different features that overlap in some cases. However, none of them provide the comprehensive functionality for OBDA-based scenario. SQuaRE provides features for creating and managing of both: R2RML mappings and SPARQL queries. Moreover, it supports users in the execution of queries and presents results in a graphical way. Moreover, we are going

to implement support for a graphical creation of SWRL rules [8], which will be another difference to the aforementioned tools.

SQuaRE is aimed at providing a simple user interface and easy to use methodology. Nevertheless, it should be perceived as a tool that tries to acquire the best features of other applications and provide them in a graphical way with an easy-to-use interface. The most similar tool at this stage of development is Karma, but without handling SPARQL queries and results in a graphical manner (but Karma provides more mapping methods than SQuaRE and more features regarding data integration). It is worth to notice that SQuaRE is still at the early stage of development whereas most of the tools from the list are being developed in the last few years. Some of them are even discontinued, like ODEMapster or RBA.

6 Summary and Future Work

In this paper we presented the SQuaRE tool which is a web-based environment that provides: (i) creation of R2RML mappings between relational databases and OWL ontologies, and (ii) creation and execution of SPARQL queries. The tool provides a lot of useful features that can be applied in an OBDA-based scenario.

Currently, we are developing a graph-based method for creating SPARQL queries. In this case we will fully support a graphical environment for handling R2RML and SPARQL. We also plan to include RuQAR [2] to extend reasoning capabilities and provide support for SWRL rules. Moreover, the long term plans are to support other mapping languages, like D2RQ[13] and RML [5]. As a result we will be able to map different data sources like CSV, JSON and others. We plan to make SQuaRE open source as soon as we finish the graphical method of creating SPARQL queries.

Acknowledgments. The work presented in this paper was supported by 04/45/DSMK/0158 project.

References

1. Jesús Barrasa, Óscar Corcho, and Asunción Gómez-pérez. R2o, an extensible and semantically based database-to-ontology mapping language. In *in In Proceedings of the 2nd Workshop on Semantic Web and Databases(SWDB2004*, pages 1069–1070. Springer, 2004.
2. Jarosław Bąk. RuQAR : Reasoning with OWL 2 RL using forward chaining engines. In *Informal Proceedings of the 4th International Workshop on OWL Reasoner Evaluation (ORE-2015) co-located with the 28th International Workshop on Description Logics (DL 2015), Athens, Greece, June 6, 2015.*, pages 31–37, 2015.
3. Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. Ontop: Answering sparql queries over relational databases. *Semantic Web*, (Preprint):1–17.

[13] `http://d2rq.org/`

4. Souripriya Das, Richard Cyganiak, and Seema Sundara. R2RML: RDB to RDF mapping language. W3C recommendation, W3C, September 2012. http://www.w3.org/TR/2012/REC-r2rml-20120927/.
5. Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. RML: a generic language for integrated RDF mappings of heterogeneous data. In *Proceedings of the 7th Workshop on Linked Data on the Web*, April 2014.
6. Ugur Dogrusoz, Erhan Giral, Ahmet Cetintas, Ali Civril, and Emek Demir. A layout algorithm for undirected compound graphs. *Information Sciences*, 179(7):980 – 994, 2009.
7. Matthew Horridge and Sean Bechhofer. The OWL API: A Java API for OWL Ontologies. *Semant. web*, 2(1):11–21, January 2011.
8. Ian Horrocks, Peter F. Patel-schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A semantic web rule language combining OWL and RuleML. 2004. Accessed: 04/04/2013.
9. Craig A. Knoblock, Pedro Szekely, José Luis Ambite, Aman Goel, Shubham Gupta, Kristina Lerman, Maria Muslea, Mohsen Taheriyan, and Parag Mallick. *Semi-automatically Mapping Structured Sources into the Semantic Web*, pages 375–390. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
10. Luís Eufrasio T. Neto, Vânia Maria P. Vidal, Marco A. Casanova, and José Maria Monteiro. *R2RML by Assertion: A Semi-automatic Tool for Generating Customised R2RML Mappings*, pages 248–252. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
11. W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 11 December 2012. Available at `http://www.w3.org/TR/owl2-overview/`.
12. Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. *Linking Data to Ontologies*, pages 133–173. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
13. Jesús Barrasa Rodriguez and Asunción Gómez-Pérez. Upgrading relational legacy data to the semantic web. In *Proceedings of the 15th international conference on World Wide Web*, pages 1069–1070. ACM, 2006.
14. Álvaro Siciliaa, German Nemirovskib, and Andreas Nolleb. Map-on: A web-based editor for visual ontology mapping. *Semantic Web Journal*, (Preprint):1–12.

Visualization for Ontology Evolution

Patrick Lambrix[1], Zlatan Dragisic[1], Valentina Ivanova[1], and Craig Anslow[2]

[1]Department of Computer and Information Science
and the Swedish e-Science Research Centre, Linköping University, Sweden
[2]Department of Computer Science, Middlesex University, London, UK

Abstract. One of the challenges for the ontology engineering community is the user involvement in the engineering process. Ontologies are not static entities and there is a demand for tools to support the user during the ontology evolution process. This paper aims to provide a set of functionality requirements for ontology evolution systems, with a particular focus on the visualization of the ontologies, their versions and information needed for ontology evolution tasks. Further, we review the current state of the art in ontology evolution systems with respect to the requirements and the visualization. We also view ontologies as software and discuss approaches from the software visualization area that could be used for ontology evolution visualization.

Keywords: ontology evolution, ontology visualization, software visualization

1 Introduction

Ontologies are a key technology for the semantic web and are used in semantically-enabled applications. Ontologies are not static entities but evolve over time. In [36] ontologies are classified into initial (with a large number of changes), expanding (many additions, with deletions and modifications), refining (mainly refining existing concepts), mature (some additions and modifications, few deletions) and dormant (little activity). There are different reasons for changes in ontologies. In [34], a study on the evolution of Gene Ontology, the authors identified the following reasons for change: (1) dealing with anomalies (essentially modeling defects), (2) extending the scope to take into account new fields, (3) dealing with diverging terminology across communities (e.g., use of the same name for similar but not exactly the same processes in plants and animals), (4) mirroring scientific advance and (5) adding relations between ontology terms. Several other papers report the fourth reason that includes new discoveries and changes in the domain, e.g., [11, 44, 54]. The first reason can be extended in scope to also include semantic defects. These defects could be found by inspection by domain experts or as a result of a debugging and completion step in the ontology development process [15] using tools as RepOSE [24, 30] or OOPS! [45]. Another reason for change discussed by [44, 54] is the need to match the changing activities of the users.

In practice ontologies are used in a number of semantically-enabled applications in which the evolution plays an important part (Case 1). One of the ways to use ontologies for *semantic search* is query expansion, where, based on ontologies, an original query

is expanded to take into account synonyms of the terms in the query, more specific terms, or other related terms. When the ontologies change, the results of the queries may change as well [15]. Ontologies are also often used for *data integration* where the ontologies take the role of content explicitation, e.g., [49]. For the integrated data sources the ontologies can also take the role of query model and during the integration the ontologies can be used for verification. When the ontologies change, the data sources may not be correctly integrated anymore. Further, many data sources, e.g., in the biomedical domain, annotate their entries with ontology terms [28]. This allows for browsing a data source using an ontology, helps with semantic search as well as with data integration. Annotations can also be used in specialized data analysis such as functional enrichment analysis. When the ontologies change, the semantics of the annotations may also have changed and therefore not be complete or appropriate anymore. For instance, [19] reports on the impact of the evolution of Gene Ontology on functional analyses.

Ontology evolution is also used for obtaining knowledge about the evolving ontology (Case 2). Information about the evolution can be used for *quality assessment* for the ontology. For instance, in Evolutionary Terminology Auditing [8] adding terms to a new version of a terminology reflects dealing with unjustified absences while deleting terms reflects dealing with unjustified presences in previous versions of the ontology. In [36] the evolution of ontologies is used to classify the ontologies into five profiles of *activity*: initial, expanding, refining, mature and dormant. Further, information about how the ontology evolves allows us to find *trends*, what changes are dominating and what has been changed and what not, the latter which is particularly important in collaborative ontology development [21].

Although ontology evolution is considered important in ontology engineering (and should be part of mature ontology development tools), there are few tools. Further, one of the current challenges for the ontology engineering community is the user involvement in the engineering process. Although ontology development tools such as Protégé have large support for user interaction, it is not always the case for systems focusing on other ontology engineering tasks. As an example, in the ontology alignment task, that produces mappings between ontologies, it has been recognized that user involvement is necessary in the validation phase, but the performance and quality of the final set of mappings could also be significantly improved with user involvement in the algorithm selection and mapping generation [46]. Therefore, requirements for such systems and their user interfaces have been proposed [17, 25] and some systems are focusing on user involvement [29].

In this paper we discuss how user involvement can and should be introduced in tools supporting an ontology evolution process. After introducing an ontology evolution methodology and discussing the types of changes in ontologies in Section 2, we introduce desired functionality for ontology evolution systems in Section 3. In Section 4 we provide a literature review of how current systems support the desired functionality with a focus on visualization. As ontologies are also software we also look into the software evolution area and review the used visualization techniques in Section 5. We conclude the paper with a short discussion in Section 6.

2 Ontology Evolution

2.1 Ontology Evolution Process

Several methodologies for dealing with ontology evolution have been proposed. We briefly describe the process recently proposed in [54], that extends or overlaps with several previously proposed methodologies, e.g., [13, 18, 47]. The proposed process for ontology evolution contains five steps. The first step deals with detecting the need for evolution. This need could be based on any of the reasons for change described in Section 1 and can be initiated by domain experts or through analysis of data directly related to the ontology (e.g., structure, instances) or indirectly related external sources (e.g., text documents, databases, queries to data sources annotated with the ontology's terms). In the second step changes are suggested. The suggested changes can be generated by extracting entities from text documents, by using ontology learning techniques, using lexical databases as WordNet, using other ontologies as background knowledge or using debugging and completion systems. The changes suggested in the second step are validated in the third step. The suggested changes are validated regarding their correctness in the domain as well as regarding logical properties. In the latter case it is checked that the changes do not introduce inconsistency or incoherence. In the fourth step the impact of the changes to external artifacts is assessed. This includes invalidation of data instances, dependent ontologies and applications. Finally, in the fifth step the changes are managed which includes keeping track of the changes and the different versions of the ontology. In the case that direct recording of changes is possible, a change language is needed. If changes cannot be recorded, approaches for change detection should be used. We refer to [54] for an overview of tools that can be used for the different steps.

We note that the process does not explicitly take into account case 2 where the evolution is used to gain information about the ontology itself (quality, trends), although one could argue that the information gained from case 2 may be used to decide on the need of changes (step 1) or for suggesting changes (step 2).

2.2 Types of Changes

Different authors classify the changes in different ways. In [44] different terms are used for the ontology evolution based on the kind of change. Ontology extension is used when new single elements are added. Ontology refinement is used when there is an addition of new concepts where an is-a relation is established between an existing concept and a new concept. Finally, ontology enrichment encompasses the addition of non-taxonomic relations or other axioms. In [27] a distinction is made between non-logical changes (e.g., change in the natural language description of a concept) and logical definition changes which affect the formal semantics.

Most papers identify the addition and deletion of concepts, relations, instances and axioms as single or elementary changes, e.g., [23, 27, 47, 48]. Some approaches consider modifications as elementary changes while others consider modification as a combination of deleting and adding. When introducing changes, the user may be interested in a higher-level change, e.g., substituting a concept by another concept, rather than in the actual sequence of elementary changes that implement the higher-level change.

Therefore, some approaches also introduce composite or complex changes. For instance, [21] introduces mapping concepts in different ontologies, substituting concepts, moving a concept and its descendants to another place in the is-a hierarchy, changing attribute values, merging concepts, splitting concepts, adding, deleting, merging and splitting for leaf concepts, adding and deleting subgraphs, and making a concept obsolete or revoke this decision. In [47] variants of merging, splitting, and moving as well as copying are introduced. In [48] complex changes are user-defined.

A study on how users edit ontologies [50] proposes that the hierarchical structure of the ontologies has the strongest influence on the editing behavior. Other influencers are the entity similarity and the semantic distance of concepts. Further, users edit the ontology in a combined top-down and breadth-first fashion.

3 Functionality for Ontology Evolution Systems

There are a number of tasks that need to be performed in cases 1 and 2 which lead to desired functionality for ontology evolution systems, and to be able to provide this functionality different kinds of information needs to be gathered and analyzed.

The approaches that we reviewed start with different versions of an ontology or support creating new versions. However, in [36] it was observed that discovering previous versions of ontologies is not always straightforward. For many ontologies there are issues with provenance and we may not find all versions.

Based on a study of projects that use Protégé (Perot Systems, NCI, OBO), it was stated in [39] that the following features are often requested.[1] Users requested the change history of a concept as well as the representation of changes between ontology versions. This should include information about the actual change as well as other provenance information such as who edited, when and why. (These features can be used in steps 1-4.) To be able to provide provenance information change annotations should be supported (step 5). In the case changes are not recorded, functionality to compare different versions should be provided (steps 1-4). A printed as well as electronic summary of changes between versions should be available (steps 1-4). Also a specialized view of changes (e.g., the changes by a specific author) was requested (steps 1-4). Users also wanted to be able to query old versions of the ontology using terminology of the new version (steps 1-4). Mechanisms for the identification of conflicts as well as to accept and reject changes should be provided (step 3). Additional features include a roll-back mechanism, the ability to save the current state in the middle of the reviewing of an ontology, and the ability to post and describe new versions. As the study included collaborative ontology development, the users also requested access privileges such that work of different authors would not clash. Further, there was a need for a negotiation mechanism to resolve conflicts that occurred as a result of work by different authors.

The functionality to compare different versions is also requested by other authors, not only when the changes cannot be recorded, but also as a way to discover trends and gaining insights regarding change and stability for the ontology, e.g., [5, 9]. Additional

[1] We annotate in parentheses the features with the number of the step in the process in Section 2.1. We also note that many of the features can be found in other articles.

Table 1. Functionality for ontology evolution systems.

Step	Category	Functionality
step 1	I	1 show an ontology version
	I, E	2 show different ontology versions in evolution graph
	I, E	3 show changes/diff between ontology versions (with change types)
	I	4 show summary of changes
	I	5 show specialized view of changes
	I	6 show change history of a concept/relation
	I, E	7 show provenance information
	I	8 show information about/context of concept/relation
	I, E	9 compare different versions
	I	10 search and query ontology
	I	11 query old versions using terminology of new version
	I, E	12 discover trends
	I, E	13 discover volatile and stable regions
step 2:	-	1 identify and suggest changes
step 3:	-	1 identify conflicts
	I	2 show conflicts
	M	3 resolve conflicts
	M	4 accept and reject suggested changes
step 4:	-	1 evaluate influence on dependent artifacts
	I	2 show influence on dependent artifacts
	M	3 update of dependent artifacts
step 5:	M	1 execute changes
	E	2 identify and show implications of change in ontology
	M	3 add/edit change annotations
	M	4 roll-back mechanism
	-	5 save current state

functionality that can be found in the literature includes detecting the changes, evaluating the influence of the evolution on dependent artifacts and semi-automatic updates of these artifacts to reflect evolution, e.g. [11].

We show a summary and, based on our experience in ontology engineering, a slight extension, of the functionality in Table 1. We have added the functionality for case 2 as part of step 1. For the functionality that presents information or requires interaction we also categorize the functionality in terms of inspection (I, e.g., exploring, searching), manipulation (M, adding/transforming information) and explanation (E) as in [25].

Different kinds of information are needed for the requested functionality (Table 2). For most kinds of functionality we require information about the actual ontology versions and their concepts, relations, axioms and instances. Some functionality such as discovering trends or volatile and stable regions [12, 23] needs statistics about the ontology versions. Similarly, most of the functionality requires information about the changes [8, 27], while functionality such as discovering volatile and stable regions also requires statistics about the changes [23, 36]. Provenance information [47, 8] is used in decision making related to several kinds of functionality, e.g., resolving conflicts and

Table 2. Information for ontology evolution functionality.

Category	**Components**
Ontology and ontology versions representations	concepts, relations, instances, axioms [41, 43, 47]
Ontology version statistics	number of concepts/relations/instances [22, 20], number of structural relations (is-a, part-of) [22, 20], concept types (obsolete vs non-obsolete; leaf vs inner node) [22], proportion of leaves [12], nodes' average height and average depth [12], in- and out-degrees of nodes [12, 23], number of paths, path lengths [23]
Change representations	logs of elementary and complex changes [41, 43, 47, 20]
Change statistics	number of concepts and relations added/deleted [22, 20, 9, 48], number of axiomatic changes [22, 20], changes wrt time interval [22, 9], add-delete ratio, growth rates [22, 20, 9]
Provenance	change author [41, 43, 47], change time [41, 43, 47, 22, 9], cost of change [47, 9], cause of change [47], change description [41, 43, 47, 22, 9]
Connections	ontology version level connections [27], conceptual relations between concepts/relations in different versions [20, 27]

executing changes, as well as for computing specialized views of changes. To be able to show the ontology evolution we need information about the connections between different versions of the ontology as well as the conceptual relations, e.g., equivalence and is-a, between the concepts and relations in different versions [27].

4 Ontology Evolution Visualization

We conducted a literature review of current ontology evolution systems and discuss how they support the desired functionality as in Table 1 with a focus on visualization[2].

CODEX [20] can be used in parts of cases 1 (for semantically-enabled applications) and 2 (obtaining knowledge about the evolving ontology). It provides support for determining complex changes between two versions of ontologies (1.3). It uses COnto-Diff [21], a rule-based algorithm for computing the complex changes. The user interface is composed of multiple views. A high-level view provides statistics about the number of relations and concepts in two ontology versions as well as the number of changes between the versions, both simple and complex changes (1.4). The distribution of different change types is presented in the form of a piechart. The change explorer view and the change navigator view provide support for navigating through changes from

[2] We denote the functionality as x.y where x is the step number and y the functionality number within the step.

the complex ones to simple ones. The change explorer view utilizes tag clouds to show the frequency of different change types or number of times a concept has been changed (1.12). After selecting a change in both explorer view and change navigator view the changes can be explored in a tree-like manner. Finally, the change impact view gives a possibility of exploring which of the provided list of items were influenced by a change (5.2). Again, changes are explored in a tree-like manner.

REX [9] is tool for case 2, focusing on exploring evolution of ontology regions. A region is defined as an ontology concept with its associated is-a subgraph. The extent of changes in a certain region of an ontology is defined via a cost model which assigns costs to ontological changes. The model is customizable. The tool consists of three components: structural analysis, quantitative change analysis (1.4) and trend analysis (1.3, 1.6, 1.9, 1.12-13). The structural analysis component provides two views, a table view and a browser view. The table view is in the form of a spreadsheet containing information about the average cost of a region as well as concepts contained in the particular region. Average cost is defined as total cost of changes in a region divided by the size of the region. The browser view is in the form of a fisheye view of an ontology version. The ontology version is represented as a graph where nodes represent concepts and edges represent is-a relations. The fisheye view implies that the selected concept is in the center while its subconcepts are organized around it. The color of a node describes the concept's change intensity, red implies high change intensity (volatile region), while green marks stable concepts (stable regions). The quantitative change analysis component provides information on how many changes of certain type occurred in a specific time interval. The information is shown in the form of a line chart with change count on one axis and ontology versions (time) on the other axis. The trend analysis component provides a means for studying and comparing evolution of regions. Users can select the time interval as well as regions of interest. A line chart shows the average cost for selected regions at different time points. Partial provenance information (time of changes) is provided by the tool (1.7).

OnEX [22] can be used in parts of case 1 (by migrating annotations) and case 2. It is a tool for exploring ontology changes (1.3, 1.9) and implements two ways for following an ontology's evolution: quantitative evolution analysis and concept-based analysis. The quantitative evolution analysis gives an overview of an ontology's evolution (1.4) in the form of spreadsheets with different levels of granularity. For example, the high-level view provides the number of concepts and relations in the first and current version of all ontologies in the repository. A chart gives an overview of the trends in the number of relations and concepts over time for a selected ontology (1.12). After selecting an ontology the user is presented with numbers for different change types between versions of the ontology. Selecting a change type for a version gives a list of changes of this particular type that were done in this version. In the concept-based analysis the users can follow a concept's evolution (1.6). A concept's evolution is presented in a spreadsheet with information such as the date of change, old and new values, type of change (1.7).

PromptDiff [41] is a Protégé plugin for ontology evolution. It covers both cases. The framework for ontology evolution consists of two components: a change management plugin and a plugin for comparing ontology versions. The change management plugin provides a list of changes made to an ontology with associated annotations (time, au-

thor, comment) (1.7, 5.3). It also provides two views for changes. The detailed view provides information on individual simple changes, while the summary view groups together simple changes. The plugin for comparing ontology versions (1.3, 1.9) visualizes indented trees of two ontology versions as a single tree. It allows visualization of concept level changes as well as tree-level changes. Color coding and symbols are used to visualize changes. For example, names of new concepts are underlined, names of deleted concepts are crossed out, names of moved concepts are greyed out in the old position and bold in the new position. Visualization of ontology versions (1.1) and editing/searching support is provided directly by Protégé (1.10, 5.1, 5.3-5).

OntoView [27] covers parts of cases 1 and 2. It is a system which supports users in specifying relations between ontology versions (1.3, 1.9). The tool also provides some limited support for analyzing effects of changes (5.2). It highlights the places in the ontology where changed concepts and relations are used. The visualization for comparing two ontologies is provided in the form of a unix-style diff, i.e., ontologies are given side by side (in XML format) and changed parts are highlighted. The user can characterize the conceptual implications of a change using a menu as identical (no change in meaning, only explanation is changed) or conceptual (in this case a relation between the two versions of a term can be given as e.g., a subsumption relation).

The NeOn toolkit [43] focuses on case 1 and provides support for the ontology evolution process via different plugins. The RaDON plugin is used for diagnosis and repair. The Evolva plugin is used for discovery of changes from external data sources and checking the relevance of a change. The toolkit provides limited decision support for accepting/rejecting changes by presenting a list of side-effects for a change (5.2). A change capture plugin allows for logging of changes with provenance information such as author, time, type of change, etc. (1.7, 5.3). The logs are presented in the form of spreadsheets. Ontology versions are visualized as trees (1.1), however there exist other plugins for visualizing ontology versions.

The KAON Framework [47] is used in case 1. The user has to define an ontology evolution strategy which defines how to deal with consequences of a change (e.g., what to do with instances of a deleted concept). After applying a change, the system computes consequences of a change given the defined strategy and presents it to the user in a spreadsheet. In the KAON framework ontology versions are visualized as graphs (1.1). The framework provides support for executing changes (5.1), querying and searching (1.10), undo/redo (5.4) as well as saving the current state (5.5). Identifying and suggesting changes is implemented via a companion tool Text2Onto.

The Ontology Lookup Service provides possibilities to visualize the evolution of ontologies [48]. It performs change detection and visualizes the number of different types of simple changes (1.3-4). By clicking on the numbers the user can retrieve information about the actual changes. Ontologies can be shown as indented trees (1.1) and queried (1.10).

Although not directly related to any system, in [5] visualizing ontology evolution with dynamic graphs [4] is discussed. They distinguish time-to-time mappings which are animations, and time-to-space mappings which are static displays. As these are graph visualizations they need to take into account requirements regarding the representation of nodes and edges, as well as topology, structure and hierarchy. Animated

diagrams lead to cognitive overload when trying to detect trends, but are good at tracing particular nodes in a graph over time.[3] One way to use static displays is to use a small multiples approach where different versions are displayed next to each other.

5 Software Evolution Visualization

While our focus is on visualization for ontology evolution, a broader area of research is software visualization [14]. Software evolution visualization is a specific area of focus within software visualization and is defined as the process of visualizing the evolution of software by representing how different aspects have changed over time [14]. A comprehensive systematic mapping study of software evolution visualization [38] found that most papers focused on the following areas: change comprehension, contribution analysis, reverse engineering, identification of anomalies, and development communication. Change comprehension, contribution analysis and reverse engineering focused most on visualization. The most common data source used in the visualizations comes from source configuration management tools and the code itself. We briefly review some techniques and connect them to the ontology evolution functionality in Table 1.

The most identified software evolution visualization technique was *change comprehension* which aims to understand how software has been changed in a given period of time, identify evolutionary patterns, or identify stable parts of the software (similar to 1.1-6, 1.9, 1.12-13). Many tools have explored different aspects of change comprehension. SeeSoft [16] was a seminal tool that explored how lines of code have changed over different versions (1.6). This tool displays all lines of a system for all different versions as minimized. Each line is a specific color and each time the line changes per version it is color coded differently. In [53] changes are represented in a time line (1.2). Semantic zooming allows users to analyze changes on different levels (raw, statement, method, type) (1.6). For complex changes rectangles in the time line represent the relative number of edits with respect to the entire file (height) as well as the time spent on the edits (width) (1.5). Chronicler [52] displays pieces of code as abstract syntax trees (AST) and builds a history graph that represents changes (insert, delete, split, merge) on individual nodes in the AST. Nodes can represent structures on different levels and history paths in the history graph represent changes to specific individual nodes (1.6).

Contribution analysis shows the activities of developers by visualizing who worked on the software (1.7) based on source code repositories like Git, SVN, and CVS. One of the most common ways to visualize contributors' changes is with graphs such as the Gevol tool [10]. Some tools have extended the graph-based approach by creating novel visualizations that include animation. Code_swarm [42] visualizes the contributions made by developers in version control systems represented as swarms and shows a histogram timeline of additions and deletions of files. Gource [7] visualizes contributions made by developers and uses a force directed layout to display the source files that are in the system and then have avatars of people flying through the visualizations to show what files they made changes to. A more recent example is Developer Rivers [6] which uses a timeline-based visualization technique to show developer contributions.

[3] This was also found in [3].

A common technique for *reverse engineering* software is to visualize source code with Polymetric Views [32]. The suite of Polymetric Views uses software metrics to represent the size of classes such as number of lines of code, number of classes, number of packages, number of methods, number of dependencies, and inheritance hierarchies (1.1). The Evolution Matrix [31] is one technique part of the Polymetric Views suite which visualizes the size of classes represented as a matrix. SourceVis [1] used Polymetric Views to show how the structure of systems, packages, and classes have changed over different versions and displayed on a large multi-touch tabletop (1.3). Some empirical studies have identified that the Polymetric View techniques are an effective technique for software evolution visualization using a 3D city metaphor and large visualization wall [2, 51].

6 Discussion and Conclusion

Visualization for ontology evolution can help engineers better understand how an ontology has evolved. Despite the existence of ontology evolution tools and many software evolution visualization tools and techniques, there is a lack of research on *ontology evolution visualization*. Our work aims to fill this void by developing ontology evolution visualization tools to help ontology engineers. As a first step in realizing this goal we identified a set of functional requirements for ontology evolution.

In this paper we did not focus on the visualization of ontology versions, as this bears similarity to ontology visualization (for overviews we refer to [26, 33, 35]). One of the challenges in front of ontology visualization techniques is representing the richness of the ontologies in a comprehensive and scalable way. Ontology evolution tools additionally need to represent the differences between two versions of the ontology in a comprehensive way on different levels of granularity. To provide an overview and an initial exploration point in the presence of many entities some tools provide statistics for the changes and further allow drilling down to individual change. Multiple connected views can be utilized to account for the complexity of the ontologies and the need of provenance information. Although relevant to ontology development as well, the demand for provenance information is even higher in the case of ontology evolution to support decision making and auditing tasks.

The reviewed systems usually do not cover the complete ontology evolution process. Steps 2-4 are often not addressed although there are other systems that can be used for these steps [54]. Regarding visualization of information, spreadsheets are often used for ontology version and change statistics, change logs as well as provenance information about changes. Also line charts, pie charts and tag clouds are used for change statistics. Changes are often visualized on indented tree visualizations of ontology versions or in a unix-style diff. Highlighting and color coding are used to visualize the changes. Indented trees and graphs are most often used to visualize ontologies and the systems we reviewed represent the ontology versions as such.

While there are many techniques for software evolution visualization, these techniques are yet to be applied effectively for ontology evolution. In particular we see adapting software visualization techniques in the areas of change comprehension, contribution analysis and reverse engineering as possible techniques to explore visualiza-

tion for ontology evolution. For instance, approaches for showing version graphs and for showing changes such as matrices and polymetric views could be used and the contribution analysis approaches are largely missing in ontology evolution systems.

In the future we will investigate also the field of schema versioning and evolution. Although ontology evolution is not the same as schema evolution [40], some of the visualization techniques in the latter (e.g., [37]) may be useful for ontology evolution. As a next step we will implement different visualization techniques for the identified functionality and conduct user studies to evaluate their applicability.

Acknowledgments. We acknowledge the EU FP7 project VALCRI (FP7-IP-608142), the National Graduate School in Computer Science (CUGS), and the Swedish e-Science Research Centre (SeRC) for financial support.

References

1. C Anslow, S Marshall, J Noble, and R Biddle. SourceVis: Collaborative software visualization for co-located environments. In *International Working Conference on Software Visualization*, VISSOFT, pages 1–10. IEEE, 2013.
2. C Anslow, S Marshall, J Noble, E Tempero, and R Biddle. User evaluation of polymetric views using a large visualization wall. In *International Symposium on Software Visualization*, SOFTVIS, pages 25–34. ACM, 2010.
3. B Bach, E Pietriga, and J-D Fekete. Graphdiaries: Animated transitions andtemporal navigation for dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 20:740–754, 2014.
4. F Beck, M Burch, S Diehl, and D Weiskopf. A taxonomy and survey of dynamic graph visualization. *Computer Graphics Forum*, 2016.
5. M Burch and S Lohmann. Visualizing the evolution of ontologies: a dynamic graph perspective. In *International Workshop on Visualizations and User Interfaces for Ontologies and Linked Data*, volume 1456 of *CEUR Workshop Proceedings*, pages 69–76, 2015.
6. M Burch, T Munz, F Beck, and D Weiskopf. Visualizing work processes in software engineering with developer rivers. In *International Working Conference on Software Visualization*, VISSOFT, pages 116–124. IEEE, 2015.
7. A. Caudwell. Gource: visualizing software version control history. In *Proceedings of the OOPSLA Companion*, Splash, pages 73–74. ACM, 2010.
8. W Ceusters. Applying evolutionary terminology auditing to the gene ontology. *Journal of Biomedical Informatics*, 42:518529, 2009.
9. V Christen, A Gross, and M Hartung. Region Evolution eXplorer a tool for discovering evolution trends in ontology regions. *Journal of Biomedical Semantics*, 6:Article 26, 2015.
10. C Collberg, S Kobourov, J Nagra, J Pitts, and K Wampler. A system for graph-based visualization of the evolution of software. In *Symposium on Software Visualization*, SoftVis, pages 77–86. ACM, 2003.
11. M Da Silveira, JC Dos Reis, and C Pruski. Management of dynamic biomedical ontologies: Current status and future challenges. In *IMIA Yearbook of Medical Informatics*, pages 125–133. 2015.
12. O Dameron, C Bettembourg, and N Le Meur. Measuring the evolution of ontology complexity: The gene ontology case study. *PLOS ONE*, 8(10):Article e75993, 2013.
13. P De Leenheer and T Mens. Ontology evolution. In M Hepp, P De Leenheer, A De Moor, and Y Sure, editors, *Ontology Management*, pages 131–176. 2008.

14. S Diehl. *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer, 2007.
15. Z Dragisic, P Lambrix, and E Blomqvist. Integrating ontology debugging and matching into the extreme design methodology. In *6th Workshop on Ontology and Semantic Web Patterns*, 2015.
16. S Eick, J Steffen, and E Sumner. Seesoft-a tool for visualizing line oriented software statistics. *IEEE Transactions on Software Engineering*, 18(11):957–968, November 1992.
17. S M Falconer and M A Storey. A Cognitive Support Framework for Ontology Mapping. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC/ASWC*, volume 4825 of *LNCS*, pages 114–127, 2007.
18. G Flouris, D Manakanatas, H Kondylakis, D Plexousakis, and G Antoniou. Ontology change: Classification and survey. *Knowledge Engineering Review*, 23(2):117–152, 2008.
19. A Gross, M Hartung, K Prufer, J Kelso, and E Rahm. Impact of ontology evolution on functional analyses. *Bioinformatics*, 28(20):2671–2677, 2012.
20. M Hartung, A Gross, and E Rahm. CODEX: Exploration of semantic changes between ontology versions. *Bioinformatics*, 26(6):895–896, 2012.
21. M Hartung, A Gross, and E Rahm. COnto-Diff: Generation of complex evolution mappings for life science ontologies. *Journal of Biomedical Informatics*, 46(1):15–32, 2013.
22. M Hartung, T Kirsten, A Gross, and E Rahm. OnEX: Exploring changes in life science ontologies. *BMC Bioinformatics*, 10:Article 250, 2009.
23. M Hartung, T Kirsten, and E Rahm. Analyzing the evolution of life science ontologies and mappings. In *Data Integration in the Life Sciences, DILS*, volume 5109 of *LNCS*, pages 11–27, 2008.
24. V Ivanova and P Lambrix. A unified approach for aligning taxonomies and debugging taxonomies and their alignments. In *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC*, volume 7882 of *LNCS*, pages 1–15, 2013.
25. V Ivanova, P Lambrix, and J Åberg. Requirements for and evaluation of user support for large-scale ontology alignment. In *The Semantic Web. Latest Advances and New Domains, 12th European Semantic Web Conference, ESWC*, volume 9088 of *LNCS*, pages 3–20, 2015.
26. A Katifori, C Halatsis, G Lepouras, C Vassilakis, and E G. Giannopoulou. Ontology visualization methods - a survey. *ACM Computing Surveys*, 39(4), 2007.
27. M Klein, D Fensel, A Kiryakov, and D Ognyanov. Ontology versioning and change detection on the web. In *13th International Conference on Knowledge Engineering and Knowledge Management*, volume 2473 of *LNCS*, pages 197–212, 2002.
28. P Lambrix. Towards a semantic web for bioinformatics using ontology-based annotation. In *14th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises - WETI-ICE*, pages 3–7, 2005.
29. P Lambrix and R Kaliyaperumal. A session-based ontology alignment approach enabling user involvement. *Semantic Web Journal*, 2016.
30. P Lambrix, F Wei-Kleiner, and Z Dragisic. Completing the is-a structure in light-weight ontologies. *Journal of Biomedical Semantics*, 6:Article 12, 2015.
31. M Lanza. The evolution matrix: Recovering software evolution using software visualization techniques. In *4th International Workshop on Principles of Software Evolution*, IWPSE, pages 37–42. ACM, 2001.
32. M Lanza and S Ducasse. Polymetric views - a lightweight visual approach to reverse engineering. *Transactions on Software Engineering*, 29(9):782–795, 2003.
33. M Lanzenberger, J Sampson, and M Rester. Ontology visualization: Tools and techniques for visual representation of semi-structured meta-data. *Journal of Universal Computer Science*, 16(7):1036–1054, 2010.
34. S Leonelli, A Diehl, K Christie, M Harris, and J Lomax. How the gene ontology evolves. *BMC Bioinformatics*, 12:Article 325, 2011.

35. S Lohmann, S Negru, F Haag, and T Ertl. Visualizing ontologies with VOWL. *Semantic Web Journal*, 7:399–419, 2016.
36. J Malone and R Stevens. Measuring the level of activity in community built bio-ontologies. *Journal of Biomedical Informatics*, 46:5–14, 2013.
37. L Meurice and A Cleve. DAHLIA: A visual analyzer of database schema evolution. In *IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering, CSMR-WCRE*, pages 464–468, 2014.
38. R Lima Novais, A Torres, T Souto Mendes, M Mendonça, and N Zazworka. Software evolution visualization: A systematic mapping study. *Information and Software Technology*, 55(11):1860–1883, November 2013.
39. NF Noy, A Chugh, W Liu, and MA Musen. A framework for ontology evolution in collaborative environments. In *The Semantic Web, 5th International Semantic Web Conference, ISWC*, volume 4273 of *LNCS*, pages 544–558, 2006.
40. NF Noy and M Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, 6:428440, 2004.
41. NF Noy and MA Musen. Promptdiff: A fixed-point algorithm for comparing ontology versions. In *18th National Conference on Artificial Intelligence, AAAI*, page 744750, 2002.
42. M Ogawa and K-L Ma. Code_swarm: A design study in organic software visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1097–1104, November 2009.
43. R Palma, F Zablith, P Haase, and O Corcho. Ontology evolution. In MC Suarez-Figuero, A Gomez-Prez, E Motta, and A Gangemi, editors, *Ontology Engineering in a Networked World*, pages 235–255. 2012.
44. C Pesquita and F Couto. Predicting the extension of biomedical ontologies. *PLOS Computational Biology*, 8:e1002630, 2012.
45. M Poveda-Villalon, A Gomez-Perez, and MC Suarez-Figueroa. Oops! (ontology pitfall scanner!): An on-line tool for ontology evaluation. *International Journal on Semantic Web & Information Systems*, 10(2):7–34, 2014.
46. P Shvaiko and J Euzenat. Ontology Matching: State of the Art and Future Challenges. *Knowledge and Data Engineering*, 25(1):158–176, 2013.
47. L Stojanovic, A Maedche, B Motik, and N Stojanovic. User-driven ontology evolution management. In *13th International Conference on Knowledge Engineering and Knowledge Management*, volume 2473 of *LNCS*, pages 285–300, 2002.
48. O Vrousgrou, T Burdett, H Parkinson, and s Jupp. Biomedical ontology evolution in the EMBL-EBI ontology lookup service. In *Workshop proceedings of the EDBT/ICDT 2016 Joint Conference*, volume 1558 of *CEUR Workshop Proceedings*, 2016.
49. H Wache, T Vögele, U Visser, H Stuckenschmidt, G Schuster, H Neumann, and S Hübner. Ontology-based integration of information - a survey of existing approaches. In *IJCAI Workshop on Ontologies and Information Sharing*, pages 108–117, 2001.
50. S Walk, Ph Singer, L Espin Noboa, T Tudorache, MA Musen, and M Strohmaier. Understanding how users edit ontologies: Comparing hypotheses about four real-world projects. In *The Semantic Web - 14th International Semantic Web Conference, ISWC*, volume 9366 of *LNCS*, pages 551–568, 2015.
51. R Wettel, M Lanza, and R Robbes. Software systems as cities: A controlled experiment. In *International Conference on Software Engineering*, ICSE, pages 551–560. ACM, 2011.
52. M Wittenhagen, C Cherek, and J Borchers. Chronicler: Interactive exploration of source code history. In *CHI Conference on Human Factors in Computing Systems*, pages 3522–3532, 2016.
53. YS Yoon and BA Myers. Semantic zooming of code change history. In *IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 95 – 99, 2015.

54. F Zablith, G Antoniou, M d'Aquin, G Flouris, H Kondylakis, E Motta, D Plexousakis, and M Sabou. Ontology evolution: a process-centric survey. *The knowledge engineering review*, 30:45–75, 2013.

Visualizing User Editing Behavior in Collaborative Ontology-Engineering Projects

Simon Walk, Tania Tudorache, and Mark A. Musen

Stanford Center for Biomedical Informatics Research
Stanford University
Stanford, CA 94305, USA
{lastname}@stanford.edu

Abstract. Over the last decade, ontologies have become the mainstay in the biomedical domain. Their size and complexity, as well as, the required expert domain knowledge to create these ontologies have increased significantly. In addition, many projects resort to collaborative approaches for building these ontologies, using the Internet as a cooperation platform. While online collaborative projects have become common, the processes that drive these collaborations are still not well understood. In this paper, we are investigating novel approaches and visualizations using Markov chains to improve our understanding of how users build large, real-world ontologies. Using our novel methods, we analyze, visualize and compare the editing behavior of users in two collaborative ontology-engineering projects from the biomedical domain. The contributions of our work are two-fold. First, we visually explore the editing behavior and dynamics of users in collaborative ontology-engineering projects; and second, we quantify the differences between the editing behaviors in these two projects. We also discuss the implications and potential applications of our findings, which, we believe, may be used to create adaptive user interfaces that better support the editing behaviors of the users.

Keywords: editing behavior, visualization, collaborative ontology engineering, Markov chain, transition matrix

1 Understanding Editing Behaviors in Ontology-Development Projects

In recent years, we have seen an increased adoption of ontologies, especially in the biomedical domain. These ontologies play a critical role in acquiring, representing and processing information about human health. With a steadily increasing relevance, ontologies had to cover new findings and application domains, which also triggered an increase in size and complexity. For example, the World Health Organization has embraced OWL as a representation language for the 11^{th} revision of the International Classification of Diseases (ICD-11), which now consists of roughly $50,000$ classes (diseases and causes of death), and it is edited in an online collaborative environment on the Web [18].

As no small group of domain experts, let alone single individuals, have the required expertise and resources to develop such highly-specialized and large-scale ontologies,

new requirements for the ontology-engineering process are emerging. In order to better support users in editing these large real-world ontologies collaboratively, and to support the project managers to get an overview of the entire process, we need to better understand the intricacies of the collaborative process for building ontologies.

Uncovering such new insights—which can potentially be used to adapt and improve existing ontology-engineering tools, or devise new and expand existing development and evaluation strategies—represents a very important first step towards overall improved and more easily maintainable structured knowledge representations.

In this paper, we focus on visually analyzing, exploring and comparing user editing behavior across two different collaborative ontology-engineering projects from the biomedical domain. To that end, we expand on our previous analyses [24–26], fit first-order Markov chain models on sequences of change-type actions, and visualize the resulting transition matrices. In contrast to most existing visualization techniques, the figures presented in this paper include information about the sequential nature of the change-logs. Further, we extend the utility of the presented approach by visually highlighting the differences and commonalities between the two investigated projects.

The remainder of the paper is structured as follows: We discuss the related work (Section 2), then we describe the datasets and methods to extract user-editing behaviors from the change history of collaborative ontology-engineering projects (Section 3). We present the results of our analyses in Section 4, and then discuss the implications and potential applications of the presented visualizations in Section 5.

2 Related Work

In general, research in the field of ontology engineering addresses all the tasks, actions, tools and processes required for developing and evaluating ontologies [5]. As a result, many researchers and practitioners have developed guidelines and methodologies [3, 6, 7, 14, 17] or compiled best practices [11] for engineering ontologies. Simperl and Luczak-Rösch [15] provide an exhaustive overview of different collaborative ontology-engineering methodologies and tools.

To learn more about the impact and implications of collaboration and consensus finding in collaborative ontology-engineering projects, researchers have analyzed several aspects of such projects. For example, Falconer et al. [4] investigated if contributors of collaborative ontology-engineering projects exhibit specific roles, and if these roles can be used to group and classify these users when contributing to the ontology. Strohmaier et al. [16] investigated the hidden social dynamics that take place in collaborative ontology-engineering projects from the biomedical domain and provided new metrics to quantify various aspects of the collaborative engineering processes. Pesquita et al. [12] showed that the location and specific structural features can be used to determine if and where the next change is going to take place in the Gene Ontology[1]. To analyze user editing patterns, Wang et al. [27] used association-rule mining on the change-logs of collaborative ontology-engineering projects, and showcased the utility of the identified editing patterns in a prediction experiment.

[1] `http://www.geneontology.org`

In 2014, Van Laere et al. [21] used k-means and the GOSPL methodology to classify users by analyzing and clustering the different interactions that users engage in, while collaboratively working on engineering an ontology.

To support the collaborative development of ontologies, the Semantic Web community has already developed a number of tools and visualizations. For example, Protégé, and its versions for collaborative ontology development, such as WebProtégé [20], iCAT [18] and Collaborative Protégé [19], are prominent stand-alone tools, which are used by a large community worldwide to develop ontologies for a variety of different projects.

Many of the existing ontology-engineering tools already provide built-in functionality or plug-ins to visualize and browse ontologies [1, 8, 10]. However, most of these visualizations lack the ability to visualize the dynamic nature of collaborative ontology-engineering projects. Falconer et al. [4] developed the Change Analysis plugin for Protégé that presented user-activity graphs and author dependency graphs from the change history of a project. In 2013, Pöschko et al. [13], and Walk et al. [22] developed new visualizations, which take the historical evolution of an ontology into account. The tool, called *PragmatiX*, allows users to browse aspects of the history of collaboratively-engineered ontologies. PragmatiX also provides quantitative insights, which allow for an easier monitoring of the progress of collaborative ontology-engineering projects. In 2015, Burch and Lohmann [2, 9] presented visualizations of dynamic and time-varying graphs, using the VOWL notation.

In our previous work, we have used Markov chains to visualize different aspects of the editing behavior of users in collaborative ontology-engineering projects [24–26]. In this paper, we expand on our previous work and present a new analysis and a novel visualization that extends the arsenal of available analysis tools for investigating social interactions in collaborative ontology-engineering projects.

3 Materials & Methods

3.1 Datasets

We used the change logs of two real-world ontology-engineering projects to conduct the analyses presented in this paper. Both projects were developed with iCAT [18], a customized version of WebProtégé [20], which records a log of all changes performed

Table 1. Characteristics of the ICD-11 and ICTM datasets used in our analyses.

	ICD-11	**ICTM**
Classes #	48,771	1,506
Changes #	439,229	67,522
Users #	109	27
First change date	2009/11/18	2011/02/02
Last change date	2013/08/29	2013/07/17
Editing period (ca.)	4 years	2.5 years

by each user. Each entry in the record stores additional metadata about the change, such as the user who performed the change, a brief description of the change, the timestamp, and all the classes and properties involved in the change. Additionally, we have removed all automatically executed changes.

Using these additional metadata, we can create sequences of change actions for each user, which can then be used to fit a (first-order) Markov chain model. A brief characterization of the datasets can be found below and in Table 1. Note that we have removed users with less than 2 changes from our analysis as no transitions between changes could be observed.

The International Classification of Diseases (ICD),[2] developed by the World Health Organization (WHO), is the international standard for diagnostic classification used to encode information relevant to epidemiology, health management, and clinical use in nearly all of the United Nations' 193 member states. The 11th revision of the classification, **ICD-11**,[3] is currently in progress, with a planned finalization in 2018. In contrast to previous revisions, ICD-11 is developed as a rich OWL ontology [18], with over 48,000 classes and authored by over 100 domain experts.

The International Classification of Traditional Medicine (ICTM)[4] is a project that aimed to produce an international standard terminology and classification for diagnoses and interventions in Traditional Medicine and was led by WHO. Analogously to ICD-11, ICTM was developed collaboratively as an OWL ontology with the goal of unifying the traditional medicine practices from China, Japan and Korea. The content of the ontology is authored in English, Chinese, Japanese and Korean. More than 20 domain experts from the three countries developed ICTM, using a customized version of WebProtégé, until the development of ICTM ended in 2012. Currently, there is ongoing work to make ICTM a chapter in ICD-11.

Table 2. Listing of all change-type actions in the change-logs.

Change Type	Description
Add Condition	A restriction is added to a class.
Add Direct Type	A direct type is added to an entity.
Add Property Value	A new value is added to a property.
Create Class	A new class is created.
Create Reference	A new reference is created.
Delete Class	A class is deleted.
Delete Condition	A restriction is deleted from a class.
Delete Property Value	A property value is deleted.
Edit Property Value	A property value is edited.
Import Property	A property value is imported from an external ontology.
Move Class(es)	One or more classes are moved in the class hierarchy.
Remove Superclass	A superclass of a class is removed.
Replace Reference	A reference is replaced.
Retire Class	A class is retired.
BREAK	30 minutes of inactivity between two actions.

[2] `http://who.int/classifications/icd/en/`

[3] `http://who.int/classifications/icd/ICDRevision/`

[4] `http://who.int/mediacentre/news/notes/2010/trad_medicine_20101207/en/`

3.2 Extracting Editing Behaviors

In order to analyze the editing behavior of users in two collaborative ontology-engineering projects, we have extracted and aggregated several different types of changes (see Table 2). Each change-type represents one action that a user performed in the user interface. For example, *Edit Property Value* describes a change, where an existing value of a property of an entity is edited by a user. In contrast, *Add Property Value* describes the action of adding a value to a property of an entity, which was previously empty. We extracted these change types from the change history recorded by iCAT. Note that even though ICD-11 and ICTM were both created using iCAT, not all types of changes were performed in both projects during our observation periods.

By extracting and analyzing the sequences of change-actions of all users in ICD-11 and ICTM, we can calculate the "average" editing behavior of all users involved in the corresponding project—in the form of a first-order Markov chain—and identify commonly followed workflows. Further, whenever more than 30 minutes have passed between two consecutive changes of one user, we have added a *BREAK* state. This allows us, not only to learn more about common workflows among users, but also which actions are conducted before users take a break, and which actions are conducted, once they resume work.

It is important to understand that the types of changes listed in Table 2 are manually aggregated change-types of all the different types of changes created by iCAT. The visualizations and analyses are not limited to these, and could be easily applied to a different set of change-logs created by a different ontology-development tool that provides similarly granular log-information. Aggregating similar change types into one action helps make the visualizations easier to read.

3.3 Fitting Markov Chains

A Markov chain consists of a finite state-space S, where each state $s_1, s_2, ..., s_n \in S$ with $n = |S|$ and a transition matrix P, which lists all probabilities p_{ij} to traverse from the state s_i to the state s_j, and for each i, $\sum_j p_{ij} = 1$.

For example, if the state space of one specific Markov chain consists of all the types of changes that can be conducted on an ontology, the values listed in the transition matrix P reflect the probabilities of a user to conduct a specific type of change immediately after that user conducted the same or a different type of change.

Similarly, Markov chains can be used to model the transition probabilities between multiple transitions or higher orders. This means that the next state does not only depend on the current state, but on a sequence of k previous states, as well.

In our previous work, we have already demonstrated that higher-order Markov chain models can be extracted from the logs of changes of different collaborative ontology-engineering projects [24–26]. However, for the purpose of creating visualizations to manually inspect these transitions, higher-order Markov chains quickly become cluttered, as all permutations of possible states up to length k have to be included in the state space as well. Hence, in this paper, we focus on the visualization of first-order Markov chains.

To be able to calculate the transition matrix P, we first have to define W with each element w_{ij} representing the number of transitions between states s_i and s_j. Hence, $\sum_j w_{ij}$ describes the absolute number of occurrences of state s_i in the corresponding dataset. Finally, we calculate P by normalizing each row of W by its corresponding ℓ_1-norm, and use it for our visualizations.

To be able to compare two projects, we calculate the transition matrix Q_{abs} by subtracting W_{ICTM} from W_{ICD-11}, and again normalizing each row by its ℓ_1-norm, as described in Equation 1.

$$Q_{abs} = ||W_{ICD-11} - W_{ICTM}||_1 \tag{1}$$

Note that the transition probabilities in Q_{abs} reflect the differences between the absolute numbers of occurrences of change-type transitions of the two datasets. After normalization, frequent transitions, which are mostly appearing in only one of the two dataset, will either be close to 1.0 for the minuend (W_{ICD-11}) or -1.0 for the subtrahend (W_{ICTM}).

In contrast, $Q_{rel} = P_{ICD} - P_{ICTM}$ represents the difference between the relative transition probabilities of the change-type transitions. Analogously, after normalization, transition probabilities will range between 1.0 and -1.0. Hence, we can either decide to compare the absolute number of transitions between the different change-type actions (Q_{abs}), or the relative transition probabilities (Q_{rel}).

A detailed description of all the steps necessary to apply Markov chains on collaborative ontology-engineering projects, as well as additional results that complement the analysis presented in this paper can be found in Walk et al. [25]. Please note that the analyses presented in this paper extend the previously presented analyses [23, 26], and include a novel approach to visualize the differences in the editing behaviors of users in ICD-11 and ICTM.

4 Results

Figures 1(a) and 1(b) depict the results of our Markov chain analyses for ICD-11 and ICTM, respectively. According to the histograms, depicted on top of the transition matrices, the top changes that are conducted in ICD-11 and ICTM are related to adding or editing values of property values (*Edit Property Value* and *Add Property Value*). Additionally, users in ICD-11 have also focused on creating and moving classes (*Create Class* and *Move Class(es)*) in the ontology, while users in ICTM focused on conducting *Import Property* and *Create Reference* changes.

For both projects, we can observe a strong tendency for all users to concentrate on a particular type of change, evident in the higher transition probabilities in the diagonal (between the same change actions). Further, we are also able to identify specific "workflow" patterns in the visualizations of both projects. For example, users have a tendency to create a class, then edit an existing property value, followed by adding a new property value to an empty property.

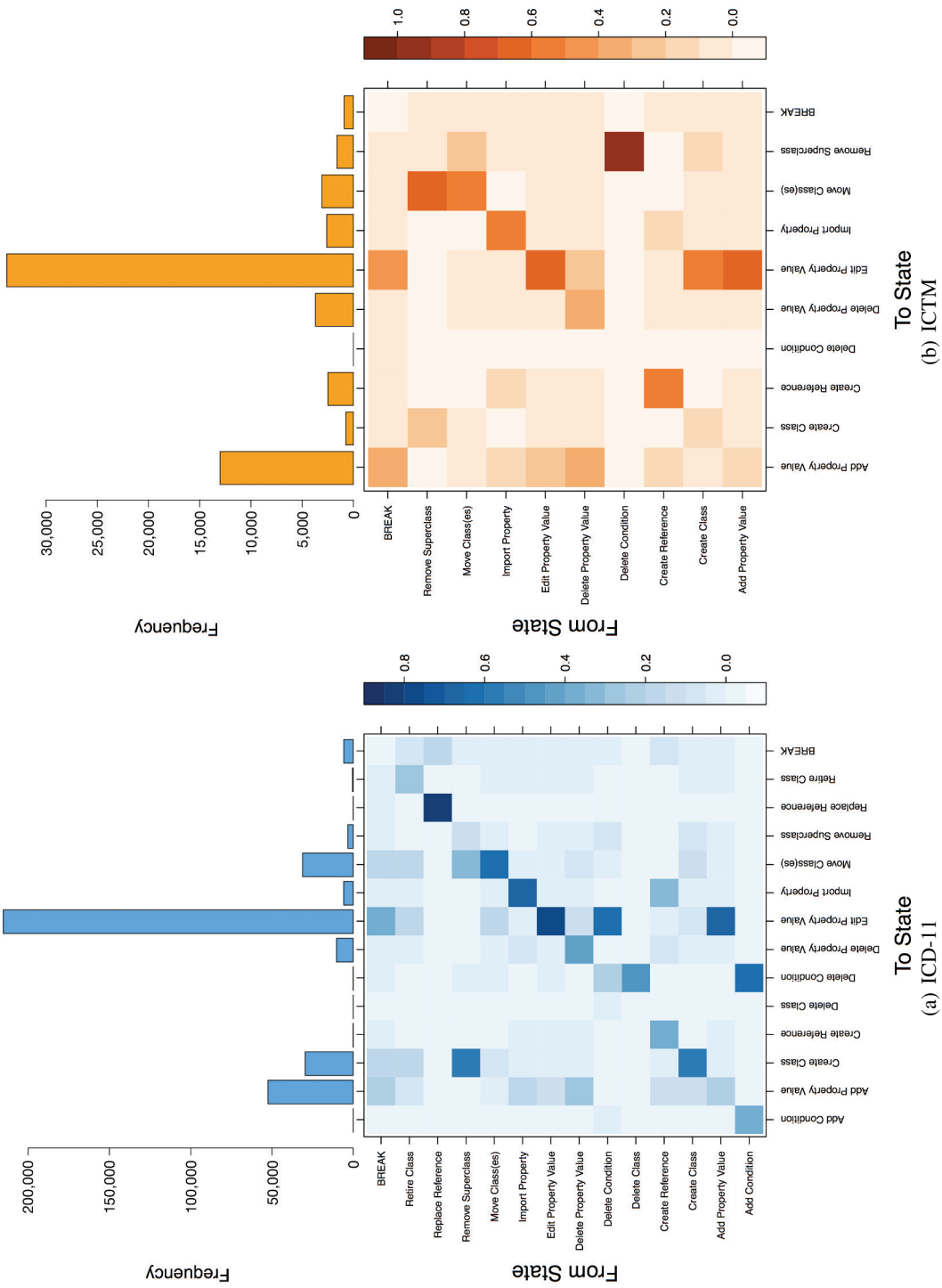


Fig. 1. Visualization of *User Editing Behaviors*: The **top** of the figures depict the histograms of the absolute occurrences of the corresponding change type actions. The transition maps are depicted on the **bottom** and visualize the probabilities (darker means higher probability) to transition *From State* (left) to *To State* (bottom) for the different change type actions in ICD-11 (a) and ICTM (b).

Additionally, we can observe that users who delete a property value are equally likely to delete another property value or add a new property value to a class. For both projects, we can further see that users, who conducted a *Remove Superclass* change, are very likely going to create a new class next or move classes in the hierarchy (*Create Class* and *Move Class(es)*). However, after moving a class, users in ICD-11 exhibit a higher tendency to edit a property value, while users in ICTM are likelier to remove another superclass.

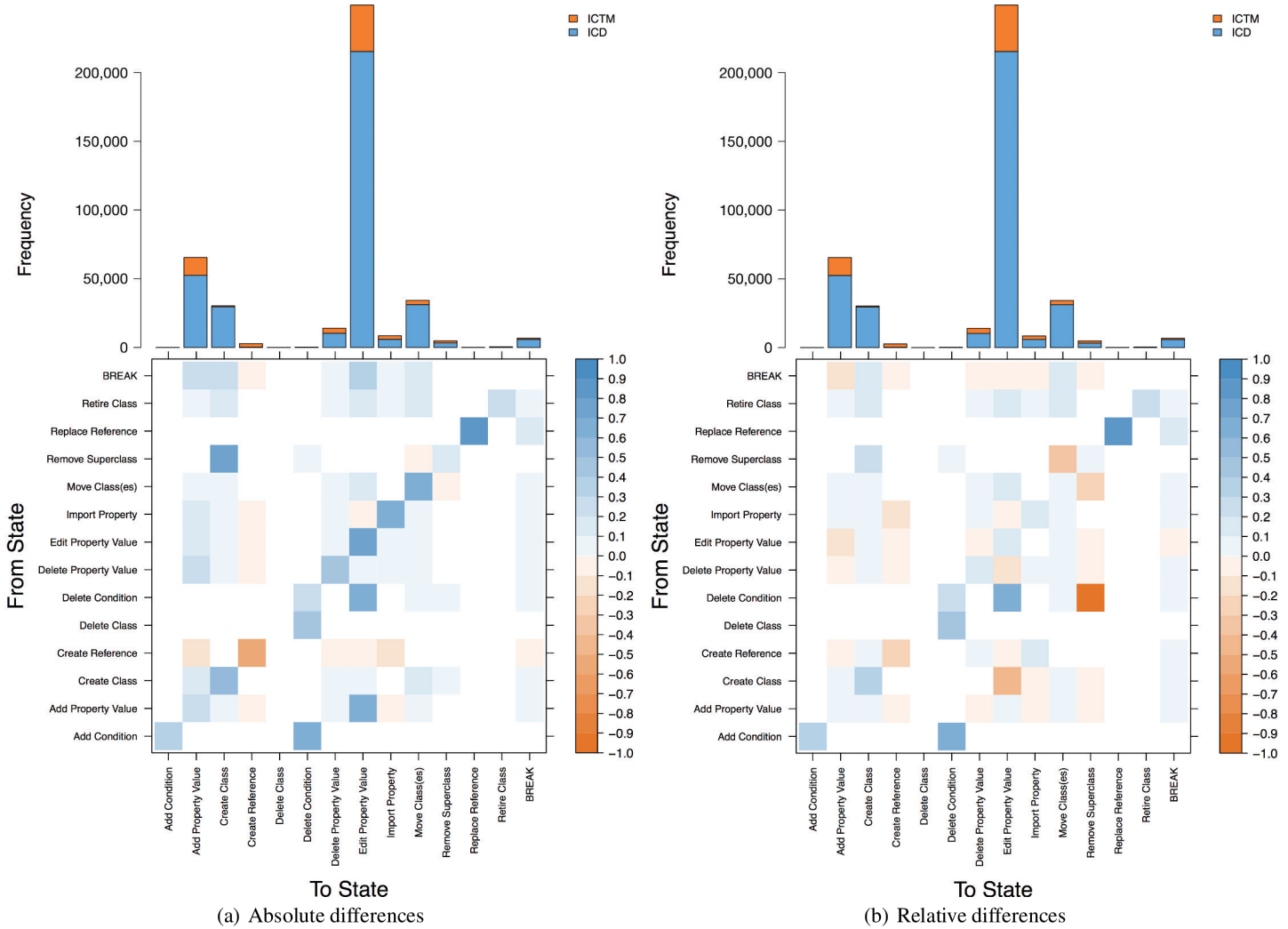
For both projects, there does not appear to be a specific change action that users conduct before taking a break from work (see columns of *BREAK* in Figure 1), and the probabilities to conduct specific types of changes when returning from a break are according to the frequencies of the different types of changes depicted in the histograms (see rows of *BREAK* in Figure 1).

To further highlight the differences between the editing behaviors of the users of the two projects, we visualized Q_{abs} (see Figure 2(a)) and Q_{rel} (see Figure 2(b)). As ICD-11 exhibits roughly six times the amount of changes compared to ICTM (see Table 1), transition probabilities in Q_{abs} are mostly positive. The only exceptions are *Create Reference* and *Import Property*, which were performed more often by the users of ICTM. As described before, we can now see that users in ICTM have a higher tendency to first conduct a *Remove Superclass* change, immediately followed by a *Move Class(es)* change than users who work on ICD-11. In contrast, users in ICD-11 are likelier to conduct a *Create Class* change after removing a superclass (*Remove Superclass*).

When looking at Q_{rel}, this difference becomes even stronger (Figure 2(b)). We can also see that specific workflows, such as creating a class, editing a property value and then adding a property value are more dominant for ICTM, which is likely caused by the overall lower number of changes and thus, their increased relative importance. In contrast, users in ICD-11 exhibit a higher tendency than users of ICTM to consecutively perform the same change-type actions, as depicted by the diagonal in Figures 2(a) and 2(b).

5 Discussion

The analyses of the presented visualizations provide important insights into the editing behavior of users in collaborative ontology-engineering projects, which can not be inferred directly from static data. We have demonstrated that common workflows can be identified when exploring the editing behavior visualizations presented in Section 4. We believe that this newly obtained information could be used by ontology-engineering tool developers to identify opportunities for improving the ontology-development tool. For example, the user interface of the tool can be better aligned to the observed workflows, and tool developers could use the information about the obtained sequences of changes to adapt the interface with the goal of reducing the number of clicks that are required to perform certain workflows.



(a) Absolute differences

(b) Relative differences

Fig. 2. Visualization of *User Editing Behavior Differences*: The **top** of the figures depict the stacked histograms of the absolute occurrences of the corresponding change type actions of both projects. The transition maps are depicted on the **bottom** and visualize the probabilities to transition *From State* (left) to *To State* (bottom) for the different change type actions in ICD-11 (blue; 1.0) and/or ICTM (orange; −1.0). The closer the transition probability to zero the lighter the color in the transition map, meaning that the transition is equally important in both datasets.

Further, the presented visualizations could act as source of information for automatically adapting interfaces, which guide users through the process of developing an ontology by considering their historical editing behavior. We have also observed that users tend to edit "vertically" in the ontology (e.g., if they edit the property value of a class, they will do so for all the subclasses). Such a workflow suggests that a tabular editing interface, similar to spreadsheets, presenting one class per row, would speed up the content entry by the users.

The visualization of the differences in editing behaviors between the two projects may be used by project administrators to identify and analyze differences in the workflows between projects, and especially between the ones that use different tools or guidelines for creating the ontology. In contrast to existing visualizations, our presented method specifically concentrates on the sequential nature of the change-log data, and provides novel insights into the dynamic nature of collaborative ontology-engineering projects.

For future work, we plan on further refining the presented analyses to compare the editing behavior of different groups of users. For example, biologists might exhibit different workflows and patterns, when developing an ontology than chemists or computer scientists do. To extend the utility of the Markov chain visualization, we plan on making the visualizations interactive. This extension would allow us to dynamically aggregate similar types of changes into abstract classes of changes, which could be expanded and collapsed by users while exploring the visualization. Additionally, this extension would make it possible to visualize higher-order Markov chains, and avoid visual clutter due to the increased number of states. Finally, we are also interested in comparing the editing behavior of users across different ontology-development tools to assert the extent to which the tool influences the editing behavior.

6 Conclusions

In this paper we have extended the analyses from our previous work, and demonstrated how to visualize and compare the editing behavior of users of two different collaborative ontology-engineering projects from the biomedical domain. In that process, we have uncovered and discussed several editing workflows, and we have presented a novel visualization, which highlights the differences in editing behaviors between two projects. Finally, we have discussed the implications of our findings for ontology tool developers and project administrators, and we have outlined potential applications of the visualizations for future work.

Acknowledgments This work was supported by Grant GM086587 from the U.S. National Institute of General Medical Sciences (NIGMS) of the National Institutes of Health. The Protégé project is support by NIGMS grant GM103316.

References

1. Alani, H.: Tgviztab: an ontology visualisation extension for protégé (2003)

2. Burch, M., Lohmann, S.: Visualizing the evolution of ontologies: a dynamic graph perspective. In: Proceedings of the International Workshop on Visualizations and User Interfaces for Ontologies and Linked Data (VOILA 2015). CEUR-WS. vol. 1456, pp. 69–76 (2015)
3. Debruyne, C., Reul, Q., Meersman, R.: Gospl: Grounding ontologies with social processes and natural language. In: 2010 Seventh International Conference on Information Technology. pp. 1255–1256. IEEE (2010)
4. Falconer, S.M., Tudorache, T., Noy, N.F.: An analysis of collaborative patterns in large-scale ontology development projects. In: Musen, M.A., Corcho, O. (eds.) K-CAP. pp. 25–32. ACM (2011)
5. Gomez, A., Fernandez, M., Corcho, O.: Ontological engineering. 2 nd Edition, Springer-Verlag (2004)
6. Gómez-Pérez, A., Suárez-Figueroa, M.C.: Scenarios for building ontology networks within the neon methodology. In: Proceedings of the fifth international conference on Knowledge capture. pp. 183–184. ACM (2009)
7. Grüninger, M., Fox, M.S.: Methodology for the design and evaluation of ontologies (1995)
8. Horridge, M.: Owlviz-a visualisation plugin for the protégé owl plugin. the university of manchester, 2004 (2008)
9. Lohmann, S., Link, V., Marbach, E., Negru, S.: WebVOWL: Web-based visualization of ontologies. In: Proceedings of EKAW 2014 Satellite Events. LNAI, vol. 8982, pp. 154–158. Springer (2015)
10. Lohmann, S., Negru, S., Bold, D.: The protégévowl plugin: ontology visualization for everyone. In: European Semantic Web Conference. pp. 395–400. Springer (2014)
11. Noy, N.F., McGuinness, D.L., et al.: Ontology development 101: A guide to creating your first ontology (2001)
12. Pesquita, C., Couto, F.M.: Predicting the extension of biomedical ontologies. PLoS Comput Biol 8(9), e1002630 (09 2012), http://dx.doi.org/10.1371%2Fjournal.pcbi.1002630
13. Pöschko, J., Strohmaier, M., Tudorache, T., Noy, N.F., Musen, M.A.: Pragmatic analysis of crowd-based knowledge production systems with icat analytics: Visualizing changes to the icd-11 ontology. In: Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Spring Symposium: Wisdom of the Crowd. Stanford, CA, USA (2012)
14. Schreiber, G., Wielinga, B., de Hoog, R., Akkermans, H., Van de Velde, W.: Commonkads: A comprehensive methodology for kbs development. IEEE expert 9(6), 28–37 (1994)
15. Simperl, E., Luczak-Rösch, M.: Collaborative ontology engineering: a survey. The Knowledge Engineering Review 29(01), 101–131 (2014)
16. Strohmaier, M., Walk, S., Pöschko, J., Lamprecht, D., Tudorache, T., Nyulas, C., Musen, M.A., Noy, N.F.: How ontologies are made: Studying the hidden social dynamics behind collaborative ontology engineering projects. Web Semantics: Science, Services and Agents on the World Wide Web 20(0) (2013), http://www.websemanticsjournal.org/index.php/ps/article/view/333
17. Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A.: Ontology engineering in a networked world. Springer Science & Business Media (2012)
18. Tudorache, T., Falconer, S.M., Nyulas, C.I., Noy, N.F., Musen, M.A.: Will Semantic Web technologies work for the development of ICD-11? In: Proceedings of the 9th International Semantic Web Conference (ISWC 2010). ISWC (In-Use), Springer, Shanghai, China (2010)
19. Tudorache, T., Noy, N.F., Tu, S., Musen, M.A.: Supporting collaborative ontology development in Protégé. Springer (2008)
20. Tudorache, T., Nyulas, C., Noy, N.F., Musen, M.A.: WebProtégé: A Distributed Ontology Editor and Knowledge Acquisition Tool for the Web. Semantic Web Journal 4(1/2013), 89–99 (2013)

21. Van Laere, S., Buyl, R., Nyssen, M.: A Method for Detecting Behavior-Based User Profiles in Collaborative Ontology Engineering. In: On the Move to Meaningful Internet Systems: OTM 2014 Conferences. pp. 657–673. Springer (2014)
22. Walk, S., Pöschko, J., Strohmaier, M., Andrews, K., Tudorache, T., Noy, N.F., Nyulas, C., Musen, M.A.: Pragmatix: An interactive tool for visualizing the creation process behind collaboratively engineered ontologies. International journal on Semantic Web and information systems 9(1), 45 (2013)
23. Walk, S., Singer, P., Noboa, L.E., Tudorache, T., Musen, M.A., Strohmaier, M.: Understanding how users edit ontologies: Comparing hypotheses about four real-world projects. In: The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I. pp. 551–568 (2015), http://dx.doi.org/10.1007/978-3-319-25007-6_32
24. Walk, S., Singer, P., Strohmaier, M.: Sequential action patterns in collaborative ontology-engineering projects: A case-study in the biomedical domain. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014. pp. 1349–1358 (2014), http://doi.acm.org/10.1145/2661829.2662049
25. Walk, S., Singer, P., Strohmaier, M., Helic, D., Noy, N.F., Musen, M.A.: How to apply markov chains for modeling sequential edit patterns in collaborative ontology-engineering projects. International Journal of Human-Computer Studies 84, 51 – 66 (2015), http://www.sciencedirect.com/science/article/pii/S107158191500124X
26. Walk, S., Singer, P., Strohmaier, M., Tudorache, T., Musen, M.A., Noy, N.F.: Discovering beaten paths in collaborative ontology-engineering projects using markov chains. Journal of Biomedical Informatics 51, 254–271 (2014), http://dx.doi.org/10.1016/j.jbi.2014.06.004
27. Wang, H., Tudorache, T., Dou, D., Noy, N.F., Musen, M.A.: Analysis of user editing patterns in ontology development projects. In: On the Move to Meaningful Internet Systems: OTM 2013 Conferences. pp. 470–487. Springer (2013)

VIZ–VIVO: Towards Visualizations-driven Linked Data Navigation

Muhammad Javed[1], Sandy Payette[2], Jim Blake[3], Tim Worrall[4]

Cornell University Library (CUL)
Cornell University
Ithaca, NY 14850, United States
{mj495[1], sdp6[2], jeb228[3], tlw72[4]}@cornell.edu

Abstract. Scholars@Cornell is a new project of Cornell University Library (CUL) that provides linked data and novel visualizations of the scholarly record. Our goal is to enable easy discovery of explicit and latent patterns that can reveal high-impact research areas, the dynamics of scholarly collaboration, and expertise of faculty and researchers. We describe VIZ-VIVO, an extension for the VIVO framework that enables end-user exploration of a scholarly knowledge-base through a configurable set of data-driven visualizations. Unlike systems that provide web pages of researcher profiles using lists and directory-style metaphors, our work explores the power of visual metaphors for navigating a rich semantic network of scholarly data modeled with the VIVO-ISF ontology. We produce dynamic web pages using D3 visualizations and bridge the user experience layer with the underlying semantic triplestore layer. Our selection of visual metaphors enables end users to start with the big picture of scholarship and navigate to individuals faculty and researchers within a macro visual context. The D3-enabled interactive environment can guide the user through a sea of scholarly data depending on the questions the user wishes to answer. In this paper, we discuss our process for selection, design, and development of an initial set of visualizations as well as our approach to the underlying technical architecture. By engaging an initial set of pilot partners we are evaluating the use of these data-driven visualizations by multiple stakeholders, including faculty, students, librarians, administrators, and the public.
Keywords: Data Driven Documents (D3), Visualizations, Linked data, RDF, VIVO, User Interface, Web of Data (WoD)

1 Introduction

As stewards of the scholarly record, Cornell University Library (CUL) has consistently leveraged new and emerging technologies to improve access and discovery to scholarly resources and to preserve and archive them for future generations. A new project of CUL is *Scholars@Cornell*[1], a data and visualization service built upon a semantic, linked data knowledge-base that represents the record of scholarship produced by Cornell faculty and researchers. One key question we set out

[1] http://scholars.cornell.edu/

to answer was how can visual mediation help users navigate the rich semantic data that represent the scholarly record? We have developed *VIZ-VIVO*, as an extension to the VIVO[2] framework to enable exploration of the scholarly record through a configurable set of data-driven visualizations that leverage relationships between people (e.g., faculty/researchers), their affiliations (e.g., academic departments or colleges), and their research outputs (e.g., journal articles, books, datasets). Our goal is to enable easy discovery of both explicit and latent patterns that reveal high-impact research areas, patterns of scholarly collaboration, and expertise of faculty and researchers.

In this paper, we discuss how we bridged the chasm between our backend of semantically rich data in VIVO triplestore, with a fresh frontend user experience that takes advantage of new developments in the dynamic world of web visualizations. We present our ongoing work on integration of D3 visualizations into the VIVO frontend pages. The visualizations-driven approach provides efficient overviews of the huge network of interconnected resources. D3 visualizations are intuitive for the users to interact and offer the ability to visualize, filter and navigate. The word cloud (presenting the *keywords* from the scholarly works) is used not only to visualize the domain expertise of a researcher, but also to provide access to the actual VIVO page of such scholarly work. The concept network map and sunburst visualizations are used to present the person-to-subject-areas map and inter-unit/cross-unit collaborations, respectively. Additionally, both traditional metrics (i.e. citation count, journal ranking.) and alternate metrics [8, 9] are used to illustrate the impact of a particular scholarly work. By engaging our initial of pilot partners at Cornell University, we are currently evaluating the use of these data-driven visualizations by multiple stakeholders, including faculty, students, librarians, administrators, and the public.

This paper is structured as follows: In Section 2 we discuss the user viewpoints that we intended to accommodate with the VIZ-VIVO visualizations. In Section 3, we discuss how we conceived of particular visualizations that could be driven by linked data navigation. We highlight specific examples showing sub-graphs from the underlying VIVO data can result in particular visualizations in the user interface layer. In Section 4, we discuss our methodology for integrating visualizations into the VIVO framework. Our approach involved developing a new component and API to be used with the core VIVO software and working with users to evaluate usability and functional suitability of visualizations. A short evaluation is given in Section 5. Related work is presented in Section 6 and we end with some discussion.

2 VIZ–VIVO: Motivation and Viewpoints

In our work at Cornell University Library, the primary entity of interest is *scholarship*, of which people and organizations are, by definition, both the creators and consumers. From this perspective, attention is focused on aggregate views

[2] http://www.vivoweb.org

of scholarship and our visualizations become the entry points into a rich graph of knowledge that can be explored interactively. Using VIZ-VIVO capabilities, a user can begin with a Cornell-wide perspective of subject areas, move to views of particular research topics, and discover faculty members in context, along with traversing lateral pathways to collaborators across the university and beyond. While other faculty profiling systems exist, including other VIVO-based applications and as well as services such as ResearchGate[3], these tend to use limited list-oriented metaphors that draw the user's attention to a primary entity – an individual, which is presented as a profile web page. Our selection of visual metaphors is motivated by enabling end users to start with the big picture, and encounter individuals (e.g., faculty members, collaborators) in context, within an interactive environment that can visually guide the user through the sea of scholarly data depending on the questions the user wishes to answer. While users can still view profiles or download data about individual entities, they are found within the process of exploration when using active graph visualizations. This is a paradigm shift in faculty profiling systems that is consistent with the mission of the library in supporting discovery of knowledge, by starting with macro views of scholarship upfront.

Among the user stories that inspired our Scholars@Cornell pilots, are how the scholarly data can answer questions from different user viewpoints. For example, how academic deans and department chairs identify where collaborations are happening and the impact of their unit's scholarly output? How can librarians use novel views of publication patterns to inform decisions about library holdings and preservation strategies. How can faculty and researchers have new ways to expose their work or locate other researchers with whom they might collaborate. Finally, how can university executives explore macro-level overviews of the university and its research to highlight high impact research areas and assist them in decision making related to invest in emerging research areas. Based on the above given viewpoints, we opt for a *bottom up approach* where we started from a set of questions. Following are some specific examples of such questions that different users would like Scholars@Cornell to answer.

- List of contributions/research output of a unit.
- List of contributions/research output of a researcher.
- List of potential collaborators (people/organizations).
- List of inter-unit/cross-unit collaborations.
- List of domain experts (for a specific subject area).
- Impact of a scholarly work.

3 Visualizations-driven Linked Data Navigation

Once user stories and the set of questions were outlined, next step was to realize what data we need to answer such questions and how such data can be presented. In regards to the presentation, we opt for Data Driven Documents

[3] https://www.researchgate.net

(D3) visualization [7]. They can bridge between the best of the two worlds (i.e. frontend and backend).

At the backend, we record data in a RDF triplestore that is modelled based on a standard, community-driven ontology – VIVO-ISF[4]. Having data in RDF not only allowed us to create links from local entities to the external resources but also to use the entity URIs as the local authorities, specifically for the persons, publications and the organizations. Creation of such URI-driven authority records is in alignment with the CUL's current goals for digital preservation and open access. Further, use of ontology-driven data modelling approach allowed us to use inference mechanisms to realize the knowledge that is not explicitly given in the triplestore.

A picture is worth a thousand words. At the frontend, we use D3 visualizations. Visualizations permit a user to navigate through the large linked data network in an intuitive way. These are semantic-data driven visualizations that allowed us to move from list-driven data views to visualization-driven data views and navigation. Filters are used to narrow down the lists that consist of hundreds of entries. In addition to all other benefits, visualizations also give a fresh look and feel to the user interface.

In the following section, we discuss few of the visualizations and the questions they address.

3.1 Person-to-Subject-Area Map (*Department level*)

At the department level, the *concept map network* (see Fig. 2) presents a mapping between a person and his/her research interests. Different citation indexing applications (e.g. Web of Science, PubMed, arXiv) classify the publication venues (e.g. journals, conferences, workshops) in different subject categories. Therefore, all the scholarly works, published in a selected venue, receive the same category as applied to the venue. For example, if an article is published in a journal that is classified under the category of *polymer science*, the article is considered to be related to the subject *polymer science*. We made use of the same transitivity approach in order to represent the research interests of an author (see Fig. 1).

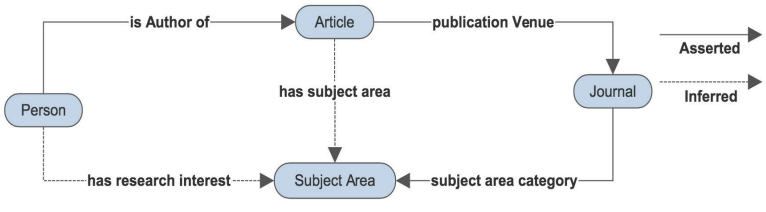


Fig. 1. Person-to-subject-area map inference

[4] https://wiki.duraspace.org/display/VIVO/VIVO-ISF+Ontology

Let, journal j be the publication venue for an article t and the person p is one of the author of the article t. If journal j is categorized under the subject area category c, we can infer that the article t can be classified under the subject area c and the person p has the research interest in the subject area c.

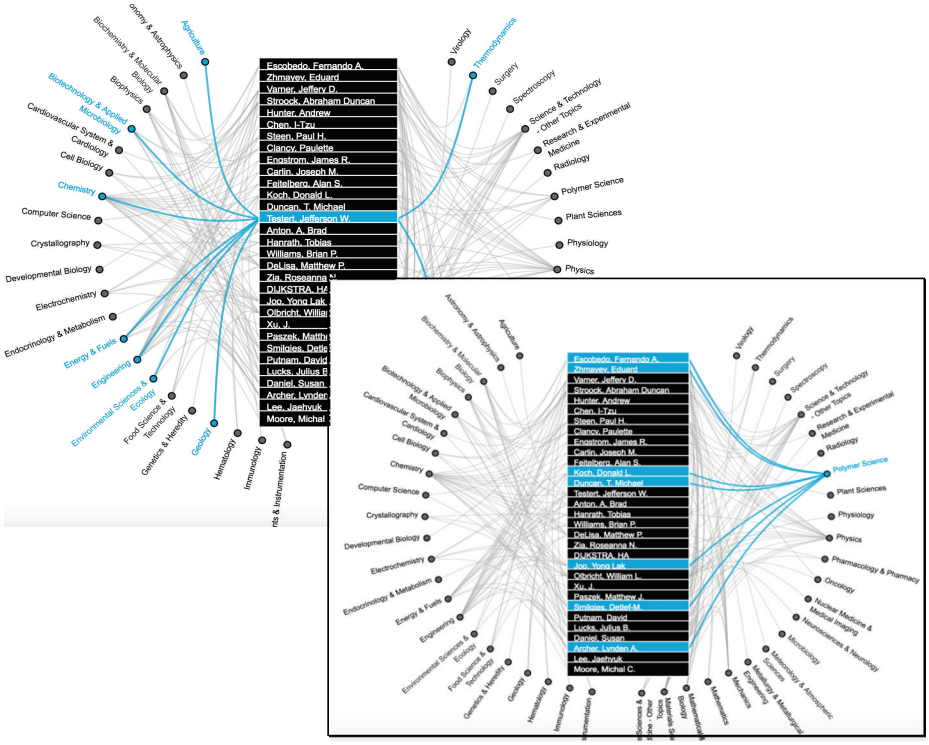


Fig. 2. Concept map network - presenting Person-to-Subject-area (P2S) map

The person-to-subject-area map is helpful for the identification of i) list of research interests of a person and ii) list of potential collaborators. The map not only demonstrates the overlap of research interests among different persons but also allows a user to navigate through the underlying linked data. A profile page link is given on every personal subject area map view. The link navigates the user to the persons's profile page. On the other hand, subject area views direct a user to the subject area's profile page, where one can learn what other entities (e.g. book chapters, dissertations, organizations, collections) are associated to the same subject area.

3.2 Domain Expertise of a Scholar (*Person level*)

The subject area terms given above in the P2S map are fairly generalized. These terms are useful to categorize the persons (and their research publications) and

narrowing down the lists. However, in order to realize the core domain expertise of a person, we need to take a step further.

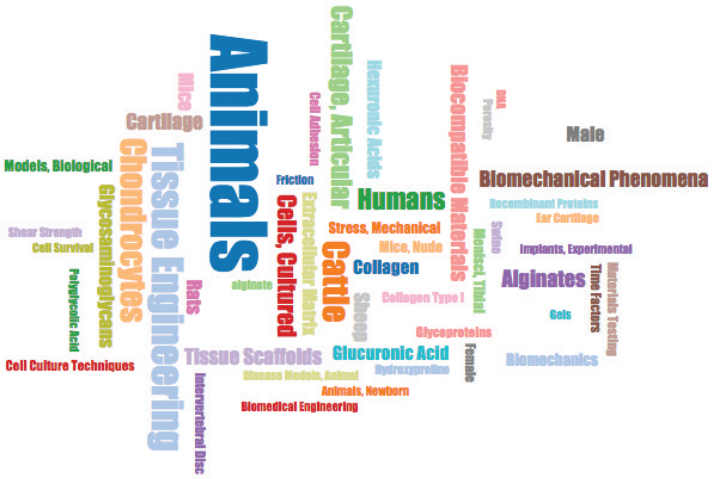


Fig. 3. Keyword cloud - presenting domain expertise of a person

Keyword cloud (see Fig. 3) presents the domain expertise of a person. Most of the scholarly works (such as articles, conference papers, posters, dissertations etc.) contains a *keyword* section. At the person level, the cloud consists of the keywords that are accumulated from the scholarly works being published by a researcher. The size of each keyword (in the cloud) is directly proportional to the number of scholarly works in which the keyword is been mentioned. Clicking on a keyword allows a user to view the list of such scholarly works. The user can further click on one of the work titles in order to navigate to the specific profile page of the work and learn more details about it.

3.3 Inter-unit/Cross-unit Collaborations (*College level*)

The inter-unit/cross-unit collaborations are presented at the college level. At Cornell, each college consists of a number of departments, institutes, centers and laboratories. Though, we receive up to date persons and their positions (affiliations) data from the human resource (HR), in order to identify the collaborative works, we cannot rely on this data. This is due to the following two reasons.

- People change their affiliations with time.
- Contribution/Output of an organization do not change with joining/leaving or change of affiliation (within organization) of a person.

For example, if a faculty member p_1 (who works at an organization org_1) co-authored an article a with a faculty member p_2 (who works at organization

org_2), article a will be considered as a collaborative work between organizations org_1 and org_2. Later, if faculty member p_1 changes his/her affiliation from org_1 to org_3 and faculty member p_2 changes his/her affiliation from org_2 to org_4 (where org_3 and org_4 can be local or external organizations), article a will still be considered as a collaborative work between org_1 and org_2.

The inter departmental and cross-unit collaborations were identified based on the *affiliations* extracted from the citation data of a scholarly work. For example, if citation data of an article describes multiple distinct affiliations for the co-authors, then it is a collaborative work. The reconciliation process for the *affiliation data strings* and the *person name strings* (coming from different citation data sources) to VIVO's person and organization URIs, is not discussed here and is out of the scope of this paper.

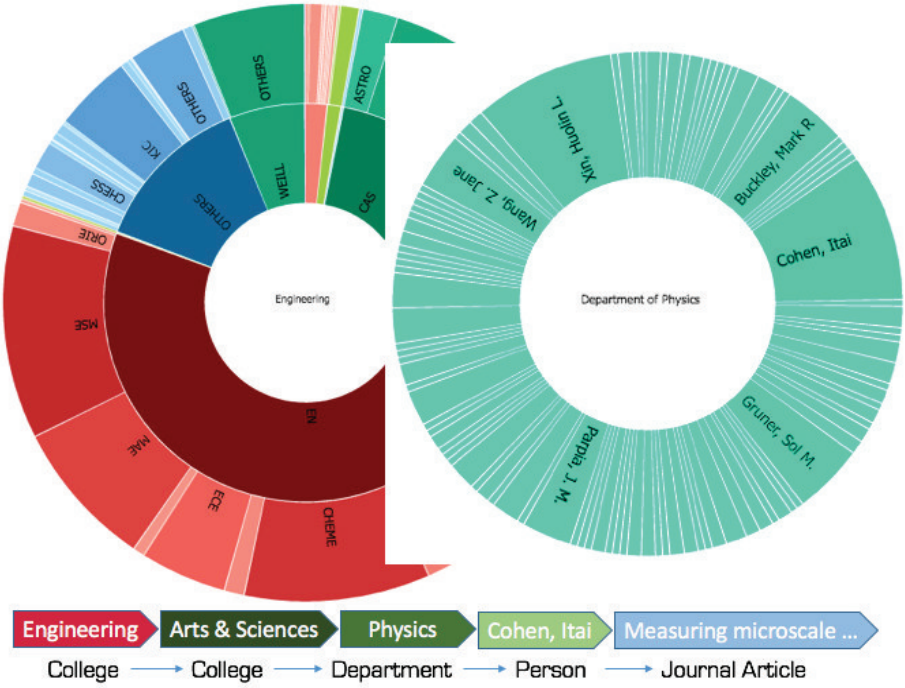


Fig. 4. Sunburst visualization - presenting inter-unit/cross-unit collaborations

Sunburst visualization (see Fig. 4) is used with the tooltips in order to present the inter-unit/cross-unit collaborations. Figure 4 presents an inter-unit/cross-unit collaboration view for the *College of Engineering (EN)*[5]. The collaboration view consists of three layers i.e., college-level layer (inner most), departmental layer (middle) and the person-level layer (outer most). To view the person-level layer, a user is first required to select the college of interest in the inner most

[5] https://www.engineering.cornell.edu

layer. To browse the collaborations within a unit, in this example EN-to-EN, one can select the *Engineering (EN)* in the college-level layer. If a user is interested in cross-unit collaborations, s/he can select a college other than *Engineering*. A user can start navigation from the college level layer and while traversing through the department (of his/her interest), can reach to a specific person who has co-authored some of the collaborative work. Clicking on a person's name section displays a tooltip that consists of the list of the (collaborative) work titles. User can further click on one of the work titles from the list in order to redirect to the specific profile page of such work.

4 Integration of D3 Visualizations in VIVO

Visualizations can be created in the browser using powerful JavaScript libraries like D3 or InfoViz. Such libraries create visualizations that are animated, responsive, and captivating. Server is only required to provide the data – a difficult task in itself. We envision a range of data sets that would be required for these visualizations. This will certainly include queries against the VIVO's triplestore and search index. It will also include data sets that require extensive processing and will likely be created by batch processes and cached for serving on demand. Existing APIs in VIVO framework are not flexible enough to handle these requirements. Further, the APIs do not provide the granularity of authorization that we require. If the JavaScript on the browser can obtain data, then the same data may be obtained by any client using HTTP, and used for any purpose.

4.1 Approach and the API Design

A data distribution API is created and is already been submitted to the VIVO community for adoption into the core. The API permits a site administrator to create a configuration file that enables requests for data from different sources. The JavaScript code on the browser requests a dataset by name, supplying qualifying parameters as appropriate. As each request type must be configured by a site administrator, a user cannot make unexpected or unauthorized requests. Requests may be restricted to logged-in users, or even to users who are logged in with a specific authorization level. Users without sufficient authorization will receive no data. It is important that the actual source of the data is hidden from the JavaScript request. For example, if a request takes too long to fulfill, the site admins can change the configuration, satisfying the request from a file of cached data, instead of a query in real-time.

On startup, the server loads a configuration file that associates data requests with the means of satisfying them. The configuration file includes the name of a Java class that will supply the data, along with any parameters that the class requires. The Java class implements a `Data Distributor` interface. This means that the API controller can ask what action the class satisfies and what MIME type should be assigned to the resulting data (see Fig. 5). The API controller

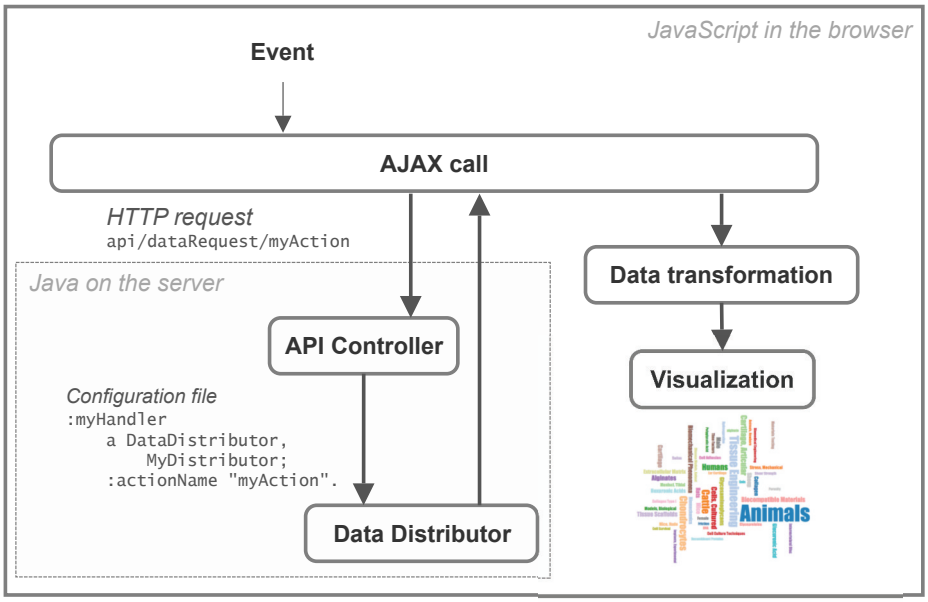


Fig. 5. The API design

asks the distributor class to produce the data and the controller routes the data to the user.

One of the powerful concepts in this design is that a Java class can be written as special-purpose code for a single use, or as general-purpose code which can be configured for each use case. We expect to see more general-purpose `DataDistributor` classes, such as:

- `SolrDataDistributor`, for queries against the search index.
- `HttpRelayDataDistributor`, for queries against services at other sites.

4.2 Data Transformation

We divide the workload into two tasks i.e., i) designing and implementing the visualization and ii) providing the data. A transformation step was added to the data flow (see Fig. 5), and a JavaScript function named `loadVisualization()` was written to control the operation. The JavaScript code that creates the *keyword cloud* visualization is given below as an example.

```
<script>
$().ready(function() {
  loadVisualization({
    url : api/dataRequest/person_word_cloud,
    transform : transform_word_cloud_data,
    display : draw_word_cloud,
  });
});
</script>
```

In this example, the visualization is created when the page is loaded into the browser. The visualization loader requests the data with the action name of `person_word_cloud`. When the data is received, the visualization loader calls the JavaScript function `transform_word_cloud_data`, that transforms a SPARQL ResultSet to a JSON structure. Finally, the visualization loader calls the function `draw_word_cloud`, that populates the visualization with the transformed data.

5 Evaluation

To evaluate the proposed visualizations and the linked data navigation methodology, *journal articles*, as a category of scholarly work, was selected. Citations data for more than 11K journal articles, authored by the faculty members and researchers of college of engineering, were uploaded in the VIVO triplestore. A user-based evaluation approach was selected in order to empirically evaluate the usability and the functional suitability of the integrated visualizations.

5.1 Empirical Evaluation

We involved college administrators and librarian to evaluate the visualizations. College administrators highly ranked the zoomable collaboration view. Using different colors for representing distinct colleges, helps a user to understand the breakdown of collaborations between a unit and the collaborating units. The zoom in and out functionality helps to traverse, see visualization from different angles and to answer different questions. Such questions include "where cross-unit collaborations are happening?", "which departments of a college are participating in these collaborations?", "who are the collaborating faculty members and how often do they collaborate?". Further, the visualization also presents the *outcomes* of these collaborations. The outcomes, such as co-authored articles, are presented under the tooltips where a user can click on any of the publication title in order to redirect to a specific page of such scholarly output. This is identical to the traversing of the linked data graph – from one node to the other. Furthermore, having impact view of a scholarly work is very helpful to evaluate the contributions and scholarly outcomes of a unit or person.

Librarians were very much attracted to the person-to-subject-area map and the keyword cloud view. Person-to-subject-area map is helpful for understanding the interests of a faculty member and in learning how these interests overlap with the others. Keyword cloud view is very useful especially for those (e.g. post graduate students) who are not only interested in learning about the research interests of a faculty member, but also in their scholarly works. For example, if a user has interest in the domain of *chemotherapy* and found this (or related) term in the keyword cloud, by clicking on the keyword s/he can view the list of publications, authored by the faculty member, on this topic.

6 Related Work

In last few years, a number of approaches for ontology and linked data visualization have been emerged [2, 4–6, 10, 12, 13]. A similar work on the representation of research expertise of a person is presented in [2]. In contrast to our methodology of usage of subject terms mentioned in the *keyword* category of an article, author proposed a method for the keyword extraction from the text of the document. Similar to our approach, author used word cloud view to present the expert profiling. In [3], author presents the preliminary results on a framework for the representation and visualization of an ontology, extracted from the text of a document. In [4], author presents a rule-based recommendation system that was subsequently developed to help the user choose a suitable ontology visualizer. In [5], author presents a tool named as Visualization Playground (VISU). *VISU* is a SPARQL interface that supports querying and visualizing the university data. The goal here was to create a flexible and user friendly tool for exploring and visualizing data from linked universities. *LinkDaViz* [10] is a framework that supports the automatic binding of linked data to the visualizations. The framework is based on the analysis of the input data structure and a visualization model in order to facilitate automatic bindings. In [9], author presented a model for assessment of research impact beyond citation analysis. Eysenbach [8] found that the number of tweets about a research article, within the first three days of an article's publication, can predict which articles will be highly cited.

7 Discussion and Future Work

In the Scholars@Cornell project, we have developed VIZ-VIVO as a means of bridging the best of the two worlds. The back-end world leverages semantic technologies by providing an ontology-driven data model of the fundamental entities of scholarship (people, publications, organizations). Through automated feeds and selected curation, this data is instantiated in a VIVO triplestore to form a graph that can be traversed, visualized, and subjected to inference. The front-end world uses visualizations driven by sub-graphs of the entire corpus of this scholarly data. This has enabled us to move away from user experiences dominated by lists and directories (e.g., articles and people), towards a visual interactive experience that leverages the power of relationships in the data.

In this paper, we presented our approach on using D3 visualizations in combination with a VIVO-based semantic knowledge base of the scholarly records of Cornell University. Our work enables easy discovery of subject area and domain expertise of faculty members and researchers, as well as the discovery of patterns in the scholarly records. Our approach offers a mean of presenting linked data through dynamic web pages with visualizations that permit a user to navigate a large corpus of semantic data and visually explore interdisciplinary collaborations and emerging trends in scientific publications.

In the future, we will be working with users to explore and evaluate other types of visualizations based on the more complex questions they seek to answer

in VIVO. We will also evolve our data to express common categorization schemes for different disciplines to better support visualizations of research areas. Our user interface will evolve to make all data that underlies each visualization downloadable in common formats such as csv, json, and xml. Our continuing work will focus on making linked data easily discoverable, consumable, visualizable, and downloadable to bring the power of semantic data to users of VIVO and other linked data applications.

Acknowledgements. Other members of the Scholars@Cornell team contributed to backend systems development, data modeling, data feeds, and data curation. We especially acknowledge the efforts of Joe McEnerney, Jill Wilson, Jason Kovari, and George Kozak.

References

1. Krafft, D.B., Cappadona, N.A., Caruso, Corson-Rikert, J., Devare, M., Lowe, B.J., & VIVO Collaboration. VIVO: Enabling national networking of scientists. Paper presented at the Web of Science Conference, raleigh, NC, 2010.
2. Koperwas, J., Skonieczny, L., Kozlowski, M., Andruszkiewicz, P., Rybiński, H., Struk, W.: Intelligent information processing for building university knowledge base. Journal of Intelligent Information Systems, Springer, pages 1–23, 2016.
3. Dasiopoulou, S., Lohmann, S., Codina, J., Wanner, L.: Representing and visualizing text as ontologies: A case from the patent domain.In: Proceedings of the International Workshop on Visualizations and User Interfaces for Ontologies and Linked Data (VOILA 2015). CEUR-WS, vol. 1456, pp. 83–90 (2015)
4. Dudáš, M., Zamazal, O., Svátek, V.: Roadmapping and navigating in the ontology visualization landscape. In 19th International Conference on Knowledge Engineering and Knowledge Management, pages 137–152. Springer, 2014.
5. Alonen, M., Kauppinen, T., Suominen, O., Hyvonen, e.: Exploring the Linked University Data with Visualization Tools. In proceedings of the European Semantic Web Conference (ESWC), 2013.
6. Atemezing G.A., Troncy, R.: Towards a linked-data based visualization wizard. In Proceedings of the 5th International Workshop on Consuming Linked Data (COLD 2014) co-located with the ISWC, 2014.
7. Bostock, M., Ogievetsky, V., Heer, J.: D3 data-driven documents. IEEE Transactions on Visualization and Computer Graphics (TVCG). volume 17, number 12, pages 2301 – 2309, 2011.
8. Eysenbach, G.: Can tweets predict citations? Metrics of social impact based on Twitter and correlation with traditional metrics of scientific impact. Journal of Medical Internet Research (JMIR), volume 13, issue number 4, 2011.
9. Sarli, C.C., Dubinsky, E.K., Holmes, K.L.: Beyond citation analysis: a model for assessment of research impact. Journal of the Medical Library Association (JMLA), volume 98, issue number 1, pages 17–23, 2010.
10. Thellmann, K., Galkin, M., Orlandi, F., Auer, S.: LinkDaViz – Automatic Binding of Linked Data to Visualizations In proceedings of International Semantic Web Conference (ISWC), 2015.

11. Abello, J., van Ham, F., Krishnan, N.:. ASK-GraphView: A Large Scale Graph Visualization System. IEEE Transactions on Visualization and Computer Graphics (TVCG), volume 12, number 5, 2006.
12. Bikakis, N., Liagouris, J., Krommyda, M., Papastefanatos, G., Sellis, T.: Towards Scalable Visual Exploration of Very Large RDF Graphs. In proceedings of the European Semantic Web Conference (ESWC), 2015.
13. Frischmuth, P., Martin, M., Tramp, S., Riechert, T., Auer, S.: Ontowiki–an authoring, publication and visualization interface for the data web. Semantic Web Journal, volume 6, number 3, pages 215–240, 2015.
14. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L. and Su, Z.: Arnetminer: extraction and mining of academic social networks. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data mining. pages 990–998, 2008.
15. Osborne, F., Motta, E. and Mulholland, P.: Exploring scholarly data with Rexplore. In International Semantic Web Conference (ISWC), pages 460–477, Springer Berlin Heidelberg, 2013.
16. Monaghan, F., Bordea, G., Samp, K. and Buitelaar, P.: Exploring your research: Sprinkling some saffron on semantic web dog food. In Semantic Web Challenge at the International Semantic Web Conference (ISWC), vol. 117, pages 420–435, 2010.

A Linked Data Driven Visual Interface for the Multi-Perspective Exploration of Data Across Repositories

Gengchen Mai[1], Krzysztof Janowicz[1], Yingjie Hu[2], Grant McKenzie[3]

[1] STKO Lab, University of California, Santa Barbara, USA
[2] University of Tennessee, Knoxville
[3] University of Maryland, USA

Abstract. As more data from heterogeneous sources become available, interfaces that support the federated exploration of these data are gaining importance to uncover relations between entities across multiple sources. Instead of explicit queries, visual interfaces enable a *follow-your-nose* style of exploration by which a user can seamlessly navigate between entities from different data sources. This requires an alignment of the ontologies used by said sources as well as the coreference resolution of entities across them. Together with Semantic Web technologies, the Linked Data paradigm provides the technological foundations to address these challenges. Nonetheless, the majority of work studies these components in isolation, focusing either on the alignment, coreference resolution, or visualization. Some interesting aspects, however, only arise when all puzzle pieces are in place. Two of these aspects are the seamless transitions between visualization and interaction paradigms as well as the combination of entity and type queries. In this work, we present a multi-perspective visual interface that enables the seamless exploration of major scientific geo-data sources that contain millions of RDF triples.

1 Introduction and Motivation

Linked Data as a paradigm describes how to break up data silos and support the publication, retrieval, reuse, and interlinkage of data on the Web. Together with other Semantic Web technologies, Linked Data shows promise to address many challenges that have affected semantic interoperability between repositories and services within and across domains that are highly heterogeneous in nature, e.g., the broader geosciences [9]. However, making use of the largely machine-oriented global graph of Linked Data also requires human-centric interfaces to query data or to explore it by following links across entities and even repositories. Unsurprisingly, user interfaces, vocabularies for their creation, and visual aids for the construction of SPARQL queries have been an active research area for many years [3,13,5,11,16]. The integration and deployment of such interfaces on top of heterogeneous and conflated sources, however, is still rare. In other words, research on topics such as ontology alignment, coreference resolution, visualization, querying, and so forth, often takes place in isolation. As a consequence, findings that only emerge once the full stack is implemented are frequently overlooked.

One such example is the fact that co-reference resolution without data conflation (fusion) hampers the reuse of data while one would intuitively assume that the opposite

is true. The reason for this lies in the fact that data sources which contain data about the same entities share overlapping information. Consequently, to give a concrete example, establishing that two URIs identify the same entity without fusing the data about them leads to places having more than one (and different) population counts, geographic coordinates, names, and so forth, e.g., for Kobe, Japan in DBpedia and GeoNames.[1]

In this work, we are interested in aspects that arise from exploring Linked Data via graphical user interfaces and more specifically in three observations made when sharing scientific data from several major oceanographic repositories. **(I)** There is no *one size fits all* visualization and interaction paradigms. However, offering multiple perspectives on data works only if users can seamlessly change between these perspectives. **(II)** exploratory interfaces such as implemented by the popular Relfinder [10] benefit from query capabilities that enabled the user to select *entities* as well as *classes* as nodes. **(III)** With an increasing number of data sources (and triples), aspects that may seem like mere convenience function become essential features, e.g., the ability to expand and compress local nodes and edges in a graph view, support for multiple layers (from multiple data sources) in a map view, and so forth.

Graph	**#Triples**
bcodmo	592,467
combined *// (AGU+NSF)*	9,506,867
dataone	25,771,511
gebco	15,212
iodp	108,338
ngdb	5,817,710
r2r	692,873
sesar	2,445,348
wholib	113,977
Total count of triples	**45,064,303**

Table 1 Data repositories made available as Linked Data.

In this paper, we present an interface[2] that supports knowledge exploration across several federated geo-data sources by means of a modular collection of ontology design patterns[3], coreference resolution based on the owl:sameAs and skos:closeMatch predicates, and multiple perspectives including a tabular view (lens), a graph view, and a map view on the data. The used data sources include BCO-DMO, DataONE, IEDA, IODP, LTER, MBLWHOI Library, R2R, and a dataset of AGU abstracts and NSF award [8]; see table 1. Overall, the served data consists of more than 45 million triples about oceanographic (scientific) cruises, research vessels, instrumentation, researchers, research projects, undersea features such as seamounts, physical samples, organizations, and so forth. As the data stems from major repositories and is in use by

[1] Via *dbpedia:Kobe owl:SameAs geodata:Kobe. dbo:populationTotal 1536499. gn:Kobe gn:population1528478.*

[2] http://demo.geolink.org/

[3] http://schema.geolink.org/

the research community, a two-step process was taken for the coreference resolution. OWL:sameAs relations between entities within and across repositories are manually curated by domain experts. In addition, they are enriched with automatically learned skos:closeMatch relations. With respect to the graphical user interfaces, this means that sameAs links will be automatically explored, while an additional checkbox enables the integration of closeMatch results. For performance reasons, the involved repositories are regularly synchronized with a harvesting endpoint.[4]

Nonetheless, our work is not specific to any particular dataset. The multiple views are not merely different ways to represent the data visually but come with their own exploration styles. The tabular view supports classical follow-your-nose exploration. The map view supports a *layered* multi-source exploration of undersea features. Finally, the graph view implements a relation finder [10] access but extends it substantially by offering type-based queries and query compression on top. To the best of our knowledge, this is the first interface that supports layers from different sources and entity-to-type queries.

As a running example, we outline how the tabular view can assist users to get detail information about a researcher, how to relate this researcher to scientific cruises that he participated in, as well as the trajectories that scientific vessels took during these cruises. The initial view of the GeoLink interface is shown in figure 1.

2 Related Work

Visualizations have been widely applied in different research aspects of the Semantic Web. Visual analytic has been used in semi-automatic approached for ontology matching, e.g., AlignmentVis [1]. Visualization is also used for a comprehensive understanding of the evolution of ontologies or time-varying ontologies [4]. A user-oriented visual notation for OWL, VOWL [12], has been proposed to define a mapping from OWL language constructs to graph elements [6]. As for visual user interfaces for Linked Data exploration, lots of work have been proposed to facilitate users without any knowledge of Semantic Web technologies to construct SPARQL queries and explore Linked Data, a spatiotemporal example being the work by Scheider et al. [14].

CS AKTive Space [15] supports an overview of UK University research in Computer Science which includes topic similarity query and geographic representation. A tabular view of direct information of one entity and map representation of similar research topic are enabled. However, due to the lack of enough data, CS AKTive Space has been restricted to support a few services.

Linked Data Scientometrics [7] is another Linked Data-driven user interface to evaluate and analyze scientific works and explore the network of researchers. This interface serves as a middle layer to support users to query the dataset from different perspectives without requiring familiarity with SPARQL. It can be put on top of any Linked Dataset that uses the Bibo ontology.[5]

FedViz [17], a visual interface for SPARQL query and formulation, enables federated and non-federated SPARQL queries from distributed data sources from Life Sci-

[4] http://data.geolink.org/sparql

[5] http://bibliontology.com/

Fig. 1 The initial view of the GeoLink interface.

ence domains. However, FedViz can only support users to ask relatively simple questions which can be formalized as one federated/non-federated SPARQL query. Complex queries which require a combination of results of several different queries, like a path query over two given end nodes are not supported.

RelFinder [10] does support path queries over several nodes through a dataset. The nodes, however, are limited to entities which means it does not support entity-to-type path queries. RelFinder executes queries over only one dataset, e.g., DBpedia.

Based on analyzing several existing Linked Data-driven user interfaces supporting data exploration and querying, we present a novel user visual interface which enables the multi-perspective data exploration of different geo-data sources. A follow-your-nose exploration, map visualization, and path queries between entities as well as entity-to-type are supported.

3 Follow-your-nose Tabular Exploration

The first view of our combined interface supports classical follow-your-nose exploration which is the most common interaction with Linked Data (aside of direct SPARQL queries). In a first step, the user can select a type of entity, e.g., *Cruise* or *Researcher*,

and then use search-while-you-type to select a particular entity of said type. Fig. 2 shows results for the oceanographer Peter Wiebe. The search spans multiple repositories and as long as coreference resolution links (here owl:sameAs) exist, the data will be grouped together in predicate-object style. The user can click on the objects to trigger another query that will select all predicate-object pairs for the newly selected subject, e.g., a specific cruise, thereby revealing information about cruises with Peter Wiebe as a participant. Consequently, further exploring the data will yield results such as the types of instruments used on a cruise in which Peter Wiebe participated. So far, nine core entity types are supported: datasets, cruises, vessels, instruments, physical samples, gazetteer features, researchers, organizations, and awards. Each of them offers different predicates to be explored, e.g., roles played on cruises, affiliations to institutions, trajectories taken by vessels during their cruises, and so forth. Finally, during any stage of the exploration, the user can click on the graph (or map) view icons to *seamlessly* switch to anther perspectives.

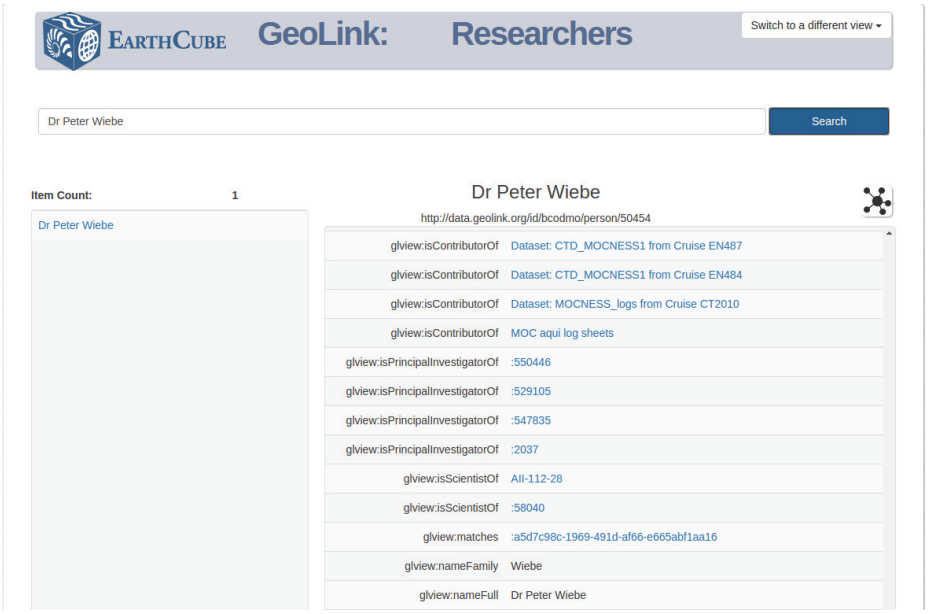


Fig. 2 The tabular view showing detail information of Peter Wiebe.

4 RelFinder Exploration Including Entity-to-type Queries

The second view builds up on the RelFinder system [10] and extends it with various features such as compressing and expanding a path, range queries around nodes, and mixed entity-to-type queries. In contrast to the first view, the user does not navigate step by step through the data but selects a source node, here Peter Wiebe, and a target

node that can either be another entity, e.g., a specific vessel or researcher, or a type of entity such as *Cruise*. Our interface then performs n-degree path queries to uncover all subjects, predicates, and objects that are along the path from source to target. Fig. 3 shows a query from Peter Wiebe to the Cruise type. Depending on the maximum path distance (set to 4 here) the results will contain s-p-o chains such as scientific datasets to which Peter contributed and which were collected during certain cruises. To keep the interface responsive and clean, the user can request more paths (beyond the 10 set as default) and also compress or expand certain paths. Fig. 3 shows some expanded paths while others remain compressed.

While entity-to-entity (e.g., between researchers Wiebe and Chandler) queries will yields results within a reasonable time even for 6-degree queries, these will likely time out for most entity-to-type queries as all entities of the given target type have to be taken into account. Typical use cases for the relfinder-style view include finding all researchers that are using the same instruments as a particular researcher or that went on the same cruises. Right-clicking on any nodes allows the user to switch seamlessly to the table or map view, to visualize the immediate (1-degree) neighborhood of said node, or to set this node as source or target node for further exploration. For compressed paths, their path lengths are shown as numbers. Figure 3 also show owl:SameAs relations between researcher URIs and between cruise URIs.

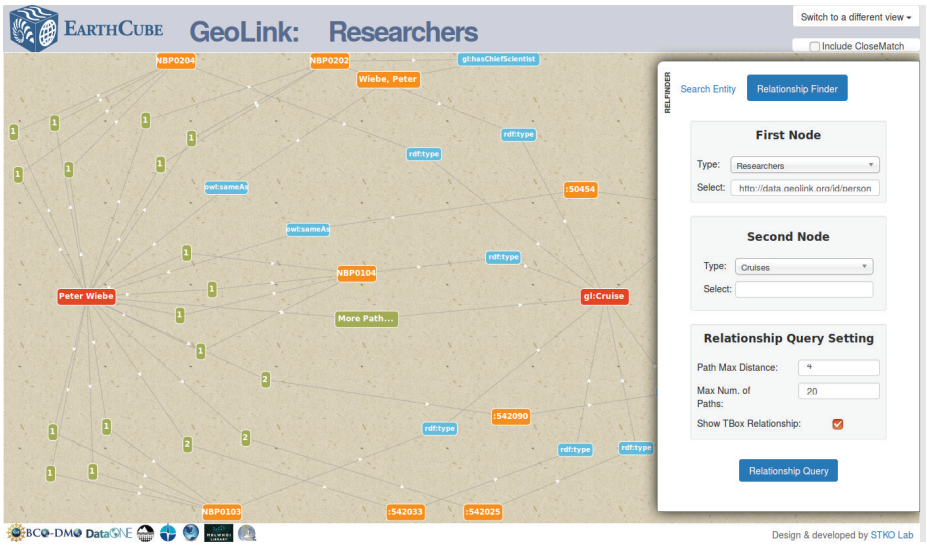


Fig. 3 Graph view showing cruises and datsets related to Peter Wiebe.

5 Multi-Layer Map Exploration

For a geospatial entity like cruise *AL9508*, it may be more appropriate to map out its geometry when a user is exploring the data repositories. For instance, after a user retrieved all cruises related to Peter Wiebe, (s)he can map out the geometries of any cruise by selecting the *Go to Map Visualization* option in the context menu, see Figure 4. Note that only entities which have a GeoSPARQL-conform WKT geometry will display this option in their context menu. The map layer container enables users to organize the geographical data in a map. The user can also map out any other geographical entities of the currently selected entity type using the search bar and the *Map Result* button. This functionality, for instance, can be used to retrieve, the trajectory of all cruises in which a certain researcher took part and then load in oceanographic gazetteer features to determine which of them may have been visited. By selecting such a feature, e.g., the Bahama Escarpment, the user can switch back to the tabular view or the graph view. Multiple layers can be added and enabled/disabled by a checkbox. These data can originate from different repositories and be of different types, e.g., undersea features, buoys, cruise trajectories, and so forth.

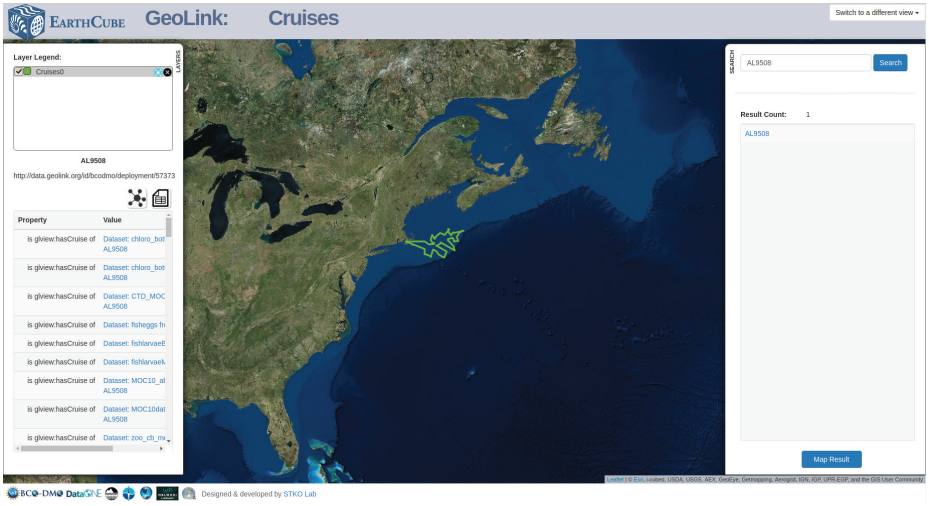


Fig. 4 Map view showing cruise 'AL9508' related to Peter Wiebe.

6 Conclusions

In this work, we introduced a Linked Data driven, multi-perspective interface that allows users to discover data across different repositories from three seamless perspectives, a tabular view, a graph view, and a map view. These perspectives enable users to

discover detailed information about an entity, relationships between entities and between entity types, as well as the spatial distribution of entities. Our work thereby contributes to research on knowledge exploration across repositories. The data stems from 9 (major) oceanographic data sources and includes diverse data about researchers, institutes, research vessels, cruises, physical samples, instruments, datasets, undersea features, and so forth. While the data stems from different repositories, semantic interoperability is enabled via a set of ontology design patterns [8] together with manually curated owl:sameAs links and automatically mined skos:closeMatch relations for coreference resolution. The key challenge for a useful querying of Linked Data by domain experts lies in the realization of features that only become obvious when all the aforementioned components are in places.

Here we focused on three of them, namely the need for seamless changes between multiple perspectives on the data, relation-based exploration queries over entities and types, and convenience functions. For example, a user has to be able to switch from the tabular view about a specific cruise to a graph and a map view without losing focus, i.e., without having to enter the URI or the ID of the cruise again. While such a tabular perspective enables a user to follow his/her nose and explore new data step by step, other paradigms enable the user to explore the relations between two nodes or to map multiple geographic features at the same time. With respect to relation exploration, one interesting finding is that entity-to-type queries are often more useful than entity-to-entity queries. While features such as allowing for multiple layers, local range queries, collapsing property chains, and so forth, seem like mere convenience functions when regarded in isolation or toy examples, they rapidly gain importance for scientific application and when multiple sources are involved. In the future, we plan to add additional data sources and interaction possibilities to further strengthen the interface. A key issue that will define the success of exploratory interfaces is the quality and extent of coreference resolution which is currently ongoing. Finally, we also plan to test the interface by means of a user study.

On a side note, with respect to the underlying data, our work resonates with other current findings of the need for centralization [2] to achieve acceptable query performance and uptime. We believe that this is an issue that needs more attention and an open discussion within the Semantic Web community.

Acknowledgements.

The presented work is partially funded by the NSF award 1440202 *EarthCube Building Blocks: Collaborative Proposal: GeoLink – Leveraging Semantics and Linked Data for Data Sharing and Discovery in the Geosciences.*

References

1. Aurisano, J., Nanavaty, A., Cruz, I.F.: Visual analytics for ontology matching using multi-linked views. In: ISWC International Workshop on Visualizations and User Interfaces for Ontologies and Linked Data (Voila (2015)

2. Beek, W., Rietveld, L., Schlobach, S., van Harmelen, F.: Lod laundromat: Why the semantic web needs centralization (even if we don't like it). IEEE Internet Computing **20**(2), 78–81 (2016). DOI 10.1109/MIC.2016.43
3. Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., Sheets, D.: Tabulator: Exploring and analyzing linked data on the semantic web. In: Proceedings of the 3rd international semantic web user interaction workshop, vol. 2006. Athens, Georgia (2006)
4. Burch, M., Lohmann, S.: Visualizing the evolution of ontologies: a dynamic graph perspective. In: Proceedings of the International Workshop on Visualizations and User Interfaces for Ontologies and Linked Data (VOILA 2015). CEUR-WS, vol. 1456, pp. 69–76 (2015)
5. Dadzie, A.S., Rowe, M.: Approaches to visualising linked data: A survey. Semantic Web **2**(2), 89–124 (2011)
6. Haag, F., Lohmann, S., Siek, S., Ertl, T.: Visual querying of linked data with queryvowl. Joint Proceedings of SumPre pp. 2014–15 (2015)
7. Hu, Y., Janowicz, K., McKenzie, G., Sengupta, K., Hitzler, P.: A Linked-Data-Driven and Semantically-Enabled Journal Portal for Scientometrics. In: The Semantic Web–ISWC 2013, pp. 114–129. Springer (2013)
8. Krisnadhi, A., Hu, Y., Janowicz, K., Hitzler, P., Arko, R., Carbotte, S., Chandler, C., Cheatham, M., Fils, D., Finin, T., Ji, P., Jones, M., Karima, N., Lehnert, K., Mickle, A., Narock, T., O'Brien, M., Raymond, L., Shepherd, A., Schildhauer, M., Wiebe, P.: The geolink modular oceanography ontology. In: Proceedings of The 14th International Semantic Web Conference, Bethlehem, PA., pp. 301–309. Springer (2015)
9. Kuhn, W., Kauppinen, T., Janowicz, K.: Linked data-a paradigm shift for geographic information science. In: Proceedings of The Eighth International Conference on Geographic Information Science (GIScience2014), Berlin., pp. 173–186. Springer (2014)
10. Lohmann, S., Heim, P., Stegemann, T., Ziegler, J.: The relfinder user interface: interactive exploration of relationships between objects of interest. In: Proceedings of the 15th international conference on Intelligent user interfaces, New York, NY, USA, pp. 421–422. ACM (2010)
11. Mazumdar, S., Petrelli, D., Ciravegna, F.: Exploring user and system requirements of linked data visualization through a visual dashboard approach. Semantic Web **5**(3), 203–220 (2014)
12. Negru, S., Lohmann, S.: A visual notation for the integrated representation of owl ontologies. In: WEBIST, pp. 308–315 (2013)
13. Pietriga, E., Bizer, C., Karger, D., Lee, R.: Fresnel: A browser-independent presentation vocabulary for rdf. In: The semantic web-ISWC 2006, pp. 158–171. Springer (2006)
14. Scheider, S., Degbelo, A., Lemmens, R., van Elzakker, C., Zimmerhof, P., Kostic, N., Jones, J., Banhatti, G.: Exploratory querying of sparql endpoints in space and time. Semantic Web (Preprint), 1–22 (2015). DOI 10.3233/SW-150211
15. Shadbolt, N.R., Gibbins, N., Harris, S., Glaser, H., et al.: Cs aktive space: representing computer science in the semantic web. In: Proceedings of the 13th international conference on World Wide Web, pp. 384–392. ACM (2004)
16. Zainab, S., Hasnain, A., Saleem, M., Mehmood, Q., Zehra, D., Decker, S.: Fedviz: A visual interface for sparql queries formulation and execution. In: Proceedings of the International Workshop on Visualizations and User Interfaces for Ontologies and Linked Data co-located with 14th International Semantic Web Conference (ISWC 2015), vol. 1456, pp. 49–60 (2015)
17. e Zainab, S.S., Hasnain, A., Saleem, M., Mehmood, Q., Zehra, D., Decker, S.: Fedviz: A visual interface for sparql queries formulation and execution

A Visual Aide for Understanding Endpoint Data

Fernando Florenzano[1,2], Denis Parra[1], Juan L. Reutter[1,2], and Freddie Venegas[1]

[1] Pontificia Universidad Católica de Chile
[2] Center for Semantic Web research, CL

Abstract. In order to pose queries on SPARQL endpoints, users need to understand the underlying structure of the data that is stored. Unfortunately, and despite the importance of endpoints in the Semantic Web infrastructure, in most (if not all) publicly available endpoints the only way of understanding this structure is by performing a considerable number of probe queries, perhaps inspired in a few examples that are also made available.
This paper looks into the problem of providing additional information for SPARQL-fluent users that need to query a RDF dataset they are not familiar with. We set up to understand what is the essential information that a user needs to query a SPARQL dataset, and then propose a visualisation that can effectively help users learn this information. This visualisation consists of a labelled graph whose nodes are the different types of entities in the RDF dataset, and where two types are related if entities of these types appear related in the RDF dataset. We illustrate our visualisation using the Linked Movie Database dataset.

1 Introduction

SPARQL Endpoints are one of the key elements in the current Semantic Web infrastructure. The idea of these endpoints is to allow users to extract information from a remote RDF dataset; the dataset is made available to be queried over the HTTP protocol, and the information must be obtained using SPARQL queries, according to the latest SPARQL specification [7].

From an algorithmic point of view, the problem of querying endpoints (or at least a single endpoint) has been very well studied, and nowadays there is a considerable number of working endpoints that permit querying databases from numerous different domains. See [15] for a list of working endpoints. Today, one can reasonably state that querying an endpoint is an easy task, assuming of course that the user is familiar in SPARQL and that she is familiar with the structure of the dataset in the endpoint.

Unfortunately, this is a pretty strong assumption: even if the endpoint is up and running, and even if the user is an expert in SPARQL, the task of producing a query that extracts the desired information may end up demanding more resources than those needed to actually compute the query. The problem is the unstructured nature of RDF: as there is no real notion of schema, knowing what and how the RDF data is stored is not an easy task. In contrast with

relational databases, where users can directly consult the schema information for the names and attributes of tables, with RDF data one has almost no alternative but to understand the data by issuing several probe queries. This behaviour has been confirmed when analysing query logs of different endpoints [3, 14].

As an example of the challenges we face, consider and endpoint for *Linked Movie Database* (LinkedMDB [8]), a database storing information about movies: who starred them, who directed them, when and where were they shot, etc. Imagine now a SPARQL expert trying to obtain the information demanded by the query Q below:

> Q: Return the names of all directors that have also acted in one of their movies.

The landing page of our LinkedMDB endpoint of choice would probably not provide any information on the structure of the data residing in the endpoint. Furthermore, the only guidelines for constructing the appropriate SPARQL query would be a small number of general examples that may not even mention the entities we are looking for. In particular, for our query Q above we would need the following information.

- First, how are directors stored in the database? In RDF one typically identifies resources with types, so one would expect that a pattern of the form {`?x a linkedmdb:director`} would match precisely all directors in our database. However, to be able to produce such a query we need to obtain the precise IRI used in the dataset to indicate the director type, in this case http://data.linkedmdb.org/resource/movie/director.
- Next, how is the connection between actors, directors and movies stored? Assuming we already know how to identify actors, directors and movie resources with SPARQL, we still need information about the way these are linked together. In the case of LinkedMDB, directors are connected to entities of type `film` via the same predicate http://data.linkedmdb.org/resource/movie/director, and similarily for actors and films.
- How to understand when a director and an actor are the same person? There could be several possibilities. For example, there could be a linking triple {`A owl:sameAs D`} between an actor resource `A` and a director resource `D`. But in LinkedMDB this is not stored directly, so we need to query for actors `A` and directors `D` that have the same name. Thus, we also need to understand the way in which the names of actors and directors are stored in the database. Furthermore, there is no way of knowing that an `owl:sameAs` link will not work without an explicit SPARLQ query that looks for the existence of triples of this form.

We argue that users new to RDF datasets would benefit tremendously from a graphical interface that would explicitly give them the information they need to start producing meaningful SPARQL patterns. We want a quick and easy lightweight solution, that can be added onto endpoints without much computational overload. This issue has been recognised before, in e.g., [5, 10, 9, 11, 1, 4,

17], and several visualisation proposals have been made to aid users in performing SPARQL queries.

But how can one identify which of these visualisations has the necessary ingredients to be a good aide for querying endpoints? We try to answer this question by proposing 3 basic requirements that must be fulfilled by any system or visualisation if they aim to be a reasonable help for users posing SPARQL queries. Then, as a proof of concept, we present one visualisation that does fulfil all of our requirements, and explain the different choices made when creating this interface. Thorough the paper we assume familiarity with RDF, SPARQL, and the basics of SPARQL endpoint architecture.

2 Looking for the Precise Meta-Information

In RDF databases the term *schema* is used in two completely different contexts. First there is the notion of RDF schema [2], a number of classes and properties with predefined meaning that serve for structuring RDF data. For example, the keywords `rdf:type` and `rdf:subClassOf` are part of the RDF schema specification, and are used as properties within documents to specify, respectively, that a resource is of a certain type or that a type is a subclass of another type.

But in relational databases, a relational schema not only defines the structure of the data, but also gives us guidelines on how to query the data: the names of the relations, which attributes are associated to each relation, the domains of each of those attributes, etc. RDF schema, being stored in RDF itself, does not automatically provide this information, and thus this second notion of schema is not immediately present when dealing with RDF, and has to be obtained by other means.

Thus the natural question: what kind of meta-information is actually needed to write SPARQL queries? Think of the information that a SPARQL expert would need in order to produce a query that fulfils the task at hand, when confronted with a new RDF dataset she doesn't know. This information can be represented in several ways, so we do not discuss how this information should look like, but rather specify what should be expected from it. To this extent, we propose three requirements that any aide for querying SPARQL must have.

> R1: The user should be able to identify all different types of resources in the database, and search for the types she is interested on.

This is the most basic requirement one could think of: if we need to query a certain type of entity (such as actors, directors, or films), we need to know how each of these entities is stored in the database, and how to recognise them by means of patterns. Usually one specifies the types of an entity `e` with triples of the form {`e rdf:type <type>`}, but we we refer to *type* in an abstract way; if RDF schema types are not present then there are other options to classify the resources in a database, such as discovering them as in [13].

The main challenge behind this requirement is that the amount of different types in a database may be so big that simply showing a list of types is useless

(for example DBpedia, as reported in http://wiki.dbpedia.org, has more than a hundred thousand different types). Thus, a way of navigating this information has to be present or included in the interface, so that the user can look for the types she needs without being confronted to a list with thousands of entries.

> R2: When given a pair of types, the user should be able to identify how and when are two entities of these types connected to each other.

Even if we can identify entities of a given type, we still need to know how they link with each other. Thus, any meta-information that deems to be useful towards query formulation needs to be able to produce, for each pair of types A and B, all relations that span between an entity of type A and an entity of type B (and vice-versa). For some types we always know that all entities of the given types are connected to each other (like actors and films in LinkedMDB). But sometimes there might only be a few connections between entities of two given types. Thus, we also need a way of specifying how likely is that two entities are connected to each other.

> R3: For each type, the user should be able to identify which attributes are common to entities of this type, and how common is this attribute amongst all entities of the given type.

To put common terms between RDF and the relational models, we define the attributes of an entity `e` as all the properties `p` present in triples of the form {`e p l`}, where `l` is a literal. Attributes give us specific information of entities: if we want to know the name of an actor `a`, then it is probably stored in a triple of the form {`a actor_name <name>`}, and likewise for any other name or numerical attribute associated to `e`. Again, while some attributes may be common for all entities of a given type (such as the titles of films in LinkedMDB), in other case these attributes may appear only in a subset of the entities (such as the duration of films, as less than a fifth of the films in LinkedMDB has this information).

We also note that there is a *scalability* issue which is orthogonal to all of these requirements: small datasets are of course easier to understand, but bigger datasets (in terms of different types and relationships) are harder, and thus the visualisation must work regardless of the size of the dataset. For this reason, it is almost impossible to develop a static visualisation that satisfies these requirements. On the contrary, we believe that the best option is an interactive page when users can be clearly guided so that they see the details only in the information that they are looking for. We describe our approach at building such a system in the following section.

3 Our Approach

Our approach is to compute, in advance, a graph that contains all the information mandated by requirements R1, R2 and R3, so that users can see this information when attempting to query the endpoint. As we have mentioned, our

visualisation is not static, but instead allow the users to select the level of granularity they wish to see in this graph. We start in section 3.1 with a high-level description of our system, and then proceed, in section 3.2, to specify the more specific choices that were taken when designing our visualisation. As a running example we use LinkedMDB's dataset, but of course the ideas are independent of any particular dataset used[3]. Our visualisation of LinkedMDB is available at http://jreutter.sitios.ing.uc.cl/VisualRDF.html.

3.1 Overview of the System

Our main visualisation is a labelled, undirected graph, where the nodes are the types of the dataset, and where there is a p-labeled edge between nodes t_1 and t_2 if and only if there are entities u_1 and u_2 such that u_1 is of type t_1, u_2 is of type t_2 and the dataset contains the triple $\{u_1\ p\ u_2\}$. That is, two types are related to each other is there are entities of these types that are connected by a triple in the dataset.

However, showing just this graph is probably not enough for users. In a similar research related to the exploration of social networks, Viegas et al. [16], found that solely relying on the traditional graph representation was a disadvantage compared to a user interface that complemented the graph metaphor with additional visual aids. For this reason we include two panels with additional information: a panel on the left side that allows the users to obtain additional information of the semantic graph as a whole and a panel on the right-side with information about the specific node or edge selected on the main pane. Figure 1 presents a screenshot of our visualisation applied to the LinkedMDB dataset. The main pane contains the aforementioned graph, the left-side panel display the top 10 types ranked according to the number of total entities of this type, and the right-side panel shows specific information about the node selected by the user (in this case *performance*).

In what follows we give a brief description of our visualisation, while discussing the satisfaction of requirements R1, R2 and R3 given in the previous section. Further details ara available in the following subsection.

Hierarchical navigation of types. In most RDF graphs the type assignment is hierarchical. Entities can be of different types, but there is usually a hierarchy between them: there can be two types, A and B that are subclass of a higher type C. In turn, C itself can be a subclass of a type D, and so on. We take advantage of the forest-like shape of the RDF type hierarchy[4] and include in our visualisation the ability to *slice* the type graph in different levels of this hierarchy. Coming back to our example, Figure 2(a) describes the same visualization of LinkedMDB as in Figure 1, but where some nodes are hidden and the emphasis is on the Person node. In LinkedMDB there are several types which are a subclass

[3] We do note that computing the necessary files for our visualisation may take several minutes, or even hours, so it is not possible to offer on-demand visualisations.

[4] Note this is not necessarily the case with more powerful ontologies in the OWL profile, this is in fact an interesting direction for future work.

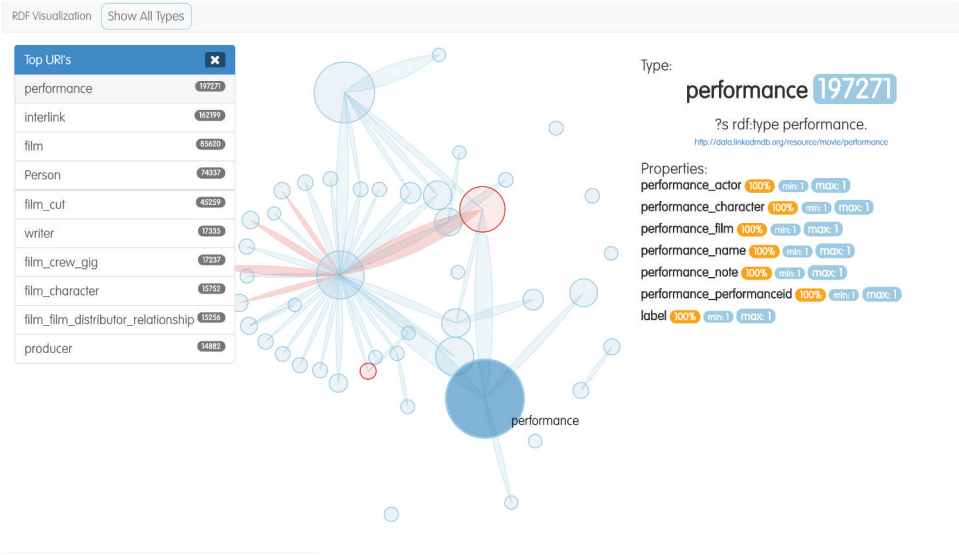


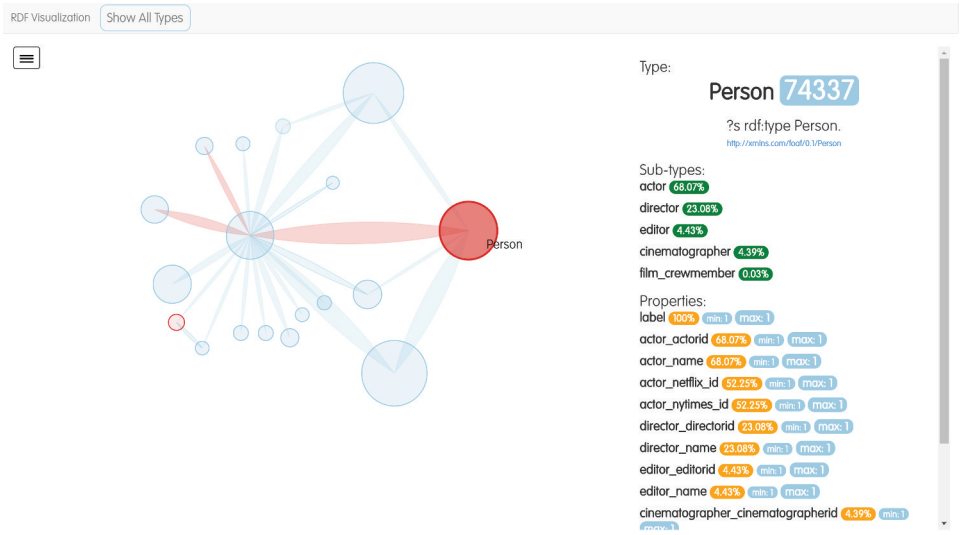
Fig. 1. The top-level visualisation of LinkedMDB's dataset

of Person, and users interested in one of them can drill down on this node, subdividing Person into each of its subclasses, as shown in Figure 2(b).

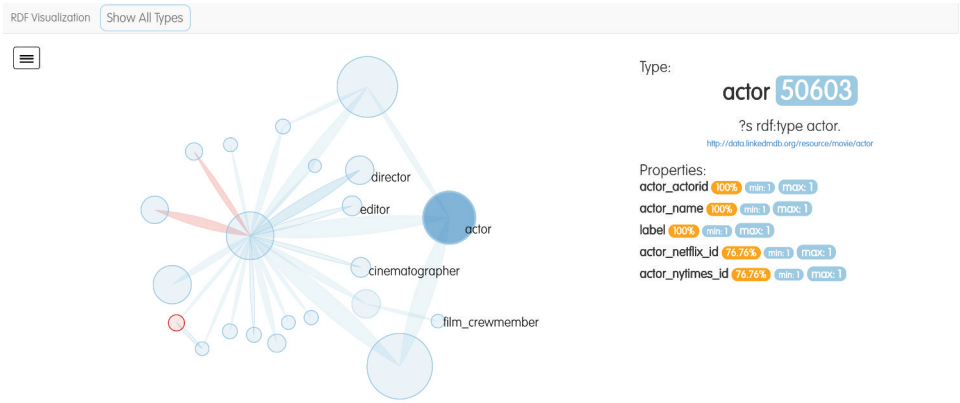
Together with the left panel, we believe this interactive visualisation fully satisfies requirement R1: all types are present in the graph, but we prevent an overload of information by showing only the top-level types first, allowing the user to produce more fined-grained views on demand.

Navigation and summarisation of relations. It is not strange to find different relations amongst entities of the same type in an RDF dataset. For example, there are several relations between persons and films in LinkedMDB. As we did with nodes, we aggregate all relationships into a single set, and only show the more fine-grained relationships whenever they are requested by the user, by hoovering over the relation. This again prevents information overload, and allows users to visualise only those relations that are interesting, thus satisfying requirement R2. Figure 3 shows an example of a relation that is in fact an aggregation of several different properties, shown in the right-side panel.

Summarisation of attributes. Recall that an attribute of a an entity u is a property that relates u with a certain string or value. For example, in Linked MBD, entities of type performance have attributes such as performance_actor, performance_character, etc., according to Figure 1. As we have specified in requirement R3, attributes are vital to users when asking questions meant to retrieve literal values, such as *What is the name of...?* or *In which year did...?*. However, due to the unstructured nature of RDF data, it is presumable that there will be several attributes that could potentially be associated to each type, and possibly not all entities of the same type have the same attributes. More-



(a) Graph with Person node



(b) Result of dividing the Person node

Fig. 2. Visualisation of LinkedMDB when the Person node is subdivided into actor, director, editor, cinematographer and film_crewmember.

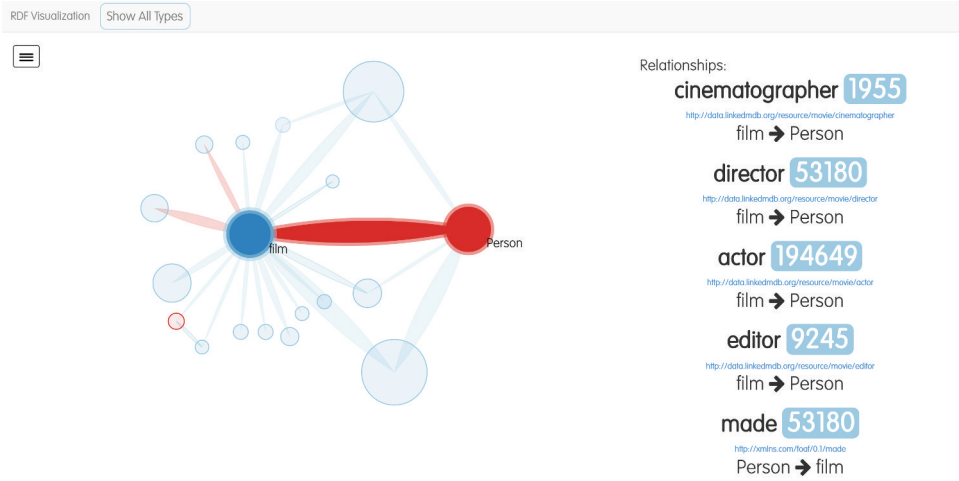


Fig. 3. Highlighting the relations between films and persons in LinkedMDB

over, it is impossible to show each of the types attributes in the same graph at the same time. For this reason we construct our visualisation so that, when users hover over or select a particular node in the graph, the right-side panel displays all the information about the attributes of entities of this type. Additionally, we also show the percentage of entities of the chosen type that posses this attribute.

3.2 Visualization

Finally, we shed light on the specific choices and details of our visualisation.

Architecture: The input to the visualisation is a JSON file that needs to be pre-computed from the dataset wanted to be visualised. This JSON contains the information of all types present in the dataset, as well as the relationships between them. The visualisation is separated into a back end, in charge of retrieving the appropriate data from the JSON file, and a front end built with D3 that displays only the nodes and relations pertinent to the current state of the interaction. This separation allows the server to handle the bigger document with all the pre-processed types, and whereas the computational power of producing the visualisation is shifted to the front end, with the double advantage of reducing the amount of computational resources demanded from the server, and hasting the response time of simple operations in the visualisation.

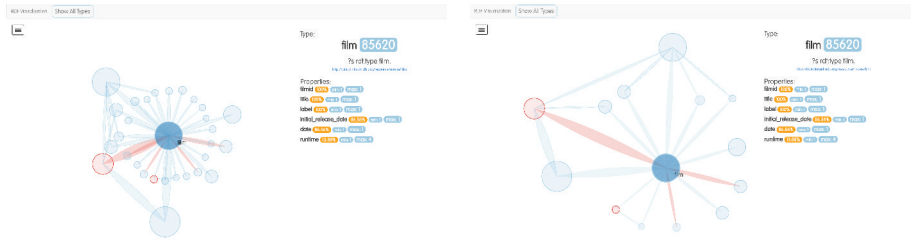
Panes: The visualisation is divided in three panels. The central and main panel is our connection graph. The left-side panel is available on demand, and it shows a box with the 10 most important entities in the current graph. The left-side

panel is connected to the graph, so that a selection of one of the entities in the box it is reflected as if it was selected on the graph. For example, in Figure 1, the user is selecting the node *performance* in the left side box, which appears highlighted in the graph. The right-side panel is where the details of the nodes are shown. There are two different visualisations. When a node is selected, the box shows the URI and name of the corresponding type, and all properties that connect at least one entity of this type with a literal (names are assigned automatically from URIs). In turn, for each property we show how many entities of the selected type have this connection, the minimum number of such connections in an entity of this type, and the maximum number of such connections. For example, looking at the first line after Properties in the right-side box in Figure 1, we see that the 100% of entities of type performance have a connection of the form {`e performance_actor <string>`}. Finally, if the type has any subclasses, then the right-side panel also shows all of these subtypes. This can be seen in Figure 2(a), when the Person node is selected. Next, when a relation is selected, the right-side box shows all the different relations that connect entities of this type, as well as the total number of such connections, their URI, and their direction. For example, Figure 3 highlights all connection between entities of type `film` and type `Person`. The right-side box in this case shows, for example, that there are 1955 triples of the form {`e1 cinematographer e2`} where `e1` is of type `film` and `e2` is of type `Person`.

Color: The graph uses essentially two colours: we use red for nodes and relationships that can be navigated (or split) in several subtypes or that contain different relations, and blue for the ordinary nodes and relations. Coming back to LinkedMDB, we see that the node `Person` is shown in red, because one can divide it (into `actor`, `cinematographer`, etc). On the other hand, nodes such as `film` are instead shown in blue, because `film` has no subclasses. The system highlights nodes and/or relationships by applying a more vivid version of the same colour. Furthermore, the name of the node or relationship is always shown when nodes are highlighted.

User interaction: Apart from displaying the left-side panel and highlighting nodes or relationships, by right-clicking the objects a user can interact with the visualisation in the following ways: First, one can hide a particular node and all the edges that connect this particular node. But also one can hide everything that is not related to a given node (keeping only those nodes connected to the selected node), or show all nodes that are related to a given node (in case they where previously hidden). Finally, there is the subdivide operation: Any red node can be subdivided into their immediate subclasses. When this operation is performed the red node dissapears, and for each subtype of the red node we create the corresponding node with their relations. In our LinkedMDB example, Figures 2(a) and 2(b) show the outcome of dividing the `Person` node, creating nodes `actor`, `director`, `editor`, `cinematographer` and `film_crewmember`.

Size and positioning of objects: In both nodes and edges, the total area is proportional to the number of entities (respectively, triples) of a given type



(a) (b)

Fig. 4. Result of hiding and dragging objects in our graph

(relationship). The positioning is also relative to the area, so that the bigger the nodes the farther away they appear. This is done to avoid visual stress by clogging two big nodes together. We take advantage of a number of D3 libraries to produce a graph that can be dragged and repositioned at will. Together with the ability to hide nodes, this facilitates the exploration and analysis of the graph. Figures 4(a) and Figure 4(b) provide an example of this power: The second graph is built from the first by simply hiding a few unimportant nodes and repositioning the important ones.

3.3 Related Work

The system closest to our ideas is LODSight [5], which works in a very similar way than our visualisation. However, LODSight does not have any hierarchical functionality, and thus the visualisation is rather static. We view our system as an alternative for LODSight that provides more opportunities of interaction. Furthermore, [10] proposes an algorithm and a system that is similar in spirit to ours, but with more emphasis on understanding the used ontologies: the system can describe different namespaces, and is in general oriented at a database with different namespaces. However, there is again no hierarchical navigation on classes or relationships, and cannot deal with summarisation when types are not present. Finally, [11] also provides an interesting alternative to visualise endpoints, but the proposal is based more on structured tables than a graph. It remains to see which approach is the best.

More on the profiling side one can highlight ExpLOD [9], that summarises RDF based on the most frequently occurring patterns. This creates a much more specific and smaller view, but on contrast it hides information inside the shape of the patterns. This is again a problem for users that are not familiar with the dataset, as they would need to spend considerable time understanding how the summarisation works. Loupe [12] and ProLOD++ [1] are systems that create a systematic profile of an RDF dataset. As a system they are much more robust, in the sense that it can create a much bigger set of statistics and relationships. However, bigger also means that there is more information to process,

and it seems these types of profiling tools aims more at maintainers and regular users of endpoints instead of casual users, and thus it might too much for a user that is just looking to produce a few SPARQL queries: in this case we argue that a more lightweight option is better. Finally, LodLive [4] and RDF Pro (http://www.linkeddatatools.com/rdf-pro-semantic-web) are good tools for visualizing particular pieces of an RDF dataset, but they are not focused on providing an aerial view of the entire dataset, and there are also FedViz [6] and ViziQuer [17], interfaces with a much more ambitious goal of helping users which are not even fluent in SPARQL. Instead, we look for the simplest way of presenting not the full information to the user, but only what is enough to produce the desired SPARQL query. This includes information about, for example, how many entities of a certain type have a particular type of label, which is something that ViziQuer does not show.

4 Future Work

There is still much to do in terms of understanding the best way to facilitate SPARQL query answering, and it would be interesting to continue demonstrating the appropriateness of our requirements. There are two main directions that one could take. First, it is important to analyse previous work on visual information about RDF datasets from the lenses of our requirements: what do they satisfy, what don't they satisfy, and how does this reflects at the time of producing SPARQL queries. In this respect, it would also be useful to understand whether a graph visualisation such as ours is the best one can do in order to fulfil these requirements, or whether one should look for other paradigms. The other direction is to demonstrate, via an empirical experiment, the usefulness of our particular visualisation. A possible way of doing this experiment would be to divide a group of SPARQL experts not familiar with a particular dataset, and assign the same set of tasks to both groups, but where only one group is allowed to use our visualisation while the other is not. The feedback from this experiment could then be used to improve our visualisation. We believe this is an interesting line of work and, to our best knowledge, one that is almost unexplored.

Acknowledgements. Work funded by the Millennium Nucleus Center for Semantic Web Research under Grant NC120004. The author Denis Parra was funded by Chilean research agency Conicyt, Fondecyt grant number 11150783.

References

1. Z. Abedjan, T. Gruetze, A. Jentzsch, and F. Naumann. Profiling and mining rdf data with prolod++. In *2014 IEEE 30th International Conference on Data Engineering*, pages 1198–1201. IEEE, 2014.
2. D. Brickley and R. V. Guha. {RDF vocabulary description language 1.0: RDF schema}. 2004.

3. C. Buil-Aranda, M. Ugarte, M. Arenas, and M. Dumontier. A preliminary investigation into sparql query complexity and federation in bio2rdf. In *Alberto Mendelzon International Workshop on Foundations of Data Management*, page 196, 2015.
4. D. V. Camarda, S. Mazzini, and A. Antonuccio. Lodlive, exploring the web of data. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 197–200. ACM, 2012.
5. M. Dudáš, V. Svátek, and J. Mynarz. Dataset summary visualization with lodsight. In *European Semantic Web Conference*, pages 36–40. Springer, 2015.
6. S. S. e Zainab, A. Hasnain, M. Saleem, Q. Mehmood, D. Zehra, and S. Decker. Fedviz: A visual interface for sparql queries formulation and execution.
7. S. Harris, A. Seaborne, and E. Prudhommeaux. Sparql 1.1 query language. *W3C Recommendation*, 21, 2013.
8. O. Hassanzadeh and M. P. Consens. Linked movie data base. In *LDOW*, 2009.
9. S. Khatchadourian and M. P. Consens. Explod: summary-based exploration of interlinking and rdf usage in the linked open data cloud. In *Extended Semantic Web Conference*, pages 272–287. Springer, 2010.
10. S. Kinsella, U. Bojars, A. Harth, J. G. Breslin, and S. Decker. An interactive map of semantic web ontology usage. In *2008 12th International Conference Information Visualisation*, pages 179–184. IEEE, 2008.
11. F. Maali. Sparqture: a more welcoming entry to sparql endpoints. In *Proceedings of the 3rd International Conference on Intelligent Exploration of Semantic Data-Volume 1279*, pages 78–82. CEUR-WS. org, 2014.
12. N. Mihindukulasooriya, M. Poveda-Villalón, R. Garcıa-Castro, and A. Gómez-Pérez. Loupe-an online tool for inspecting datasets in the linked data cloud. In *14th International Semantic Web Conference (ISWC), Posters & Demonstrations Track*, 2015.
13. M.-D. Pham, L. Passing, O. Erling, and P. Boncz. Deriving an emergent relational schema from rdf data. In *Proceedings of the 24th International Conference on World Wide Web*, pages 864–874. ACM, 2015.
14. M. Saleem, M. I. Ali, A. Hogan, Q. Mehmood, and A.-C. N. Ngomo. Lsq: The linked sparql queries dataset. In *International Semantic Web Conference*, pages 261–269. Springer, 2015.
15. P.-Y. Vandenbussche, C. B. Aranda, A. Hogan, and J. Umbrich. Monitoring the status of sparql endpoints. In *Proceedings of the 2013th International Conference on Posters & Demonstrations Track-Volume 1035*, pages 81–84. CEUR-WS. org, 2013.
16. F. B. Viégas and J. Donath. Social network visualization: Can we go beyond the graph. In *Workshop on social networks, CSCW*, volume 4, pages 6–10, 2004.
17. M. Zviedris and G. Barzdins. Viziquer: a tool to explore and query sparql endpoints. In *Extended Semantic Web Conference*, pages 441–445. Springer, 2011.

Starting Ontology Development by Visually Modeling an Example Situation - a User Study

Marek Dudáš[1], Vojtěch Svátek[1], Miroslav Vacura[1,2], and Ondřej Zamazal[1]

[1] Department of Information and Knowledge Engineering,
[2] Department of Philosophy,
University of Economics, W. Churchill Sq.4, 130 67 Prague 3, Czech Republic,
{marek.dudas|svatek|vacuram|ondrej.zamazal}@vse.cz

Abstract. This paper describes a user study aimed at comparing the common approach to developing an OWL ontology, using the Protégé editor alone, with an ontology development workflow starting by building a so-called PURO ontological background model in visual terms, using the tool PURO Modeler. The background model represents a complex example situation to be covered by the ontology, from which a seed of the ontology is semi-automatically generated. The evaluation suggests that starting from the background model might lead to an ontology that better covers the domain and might also alleviate some OWL encoding difficulties such as those tied to n-ary relations. On the other hand, it is more time-consuming and the user interface of the tool supporting it needs much improvement.

1 Introduction

In the semantic web realms, the prevailing practice of formalizing ontologies is creating them, from the onset, in OWL (with editors like Protégé), merely starting from textual specifications and informal charts. The advantages of OWL as uniform representation of ontologies throughout all 'formal' phases of their development lifecycle are its thorough standardization, solid support by authoring tools, and powerful reasoning abilities allowing formal consistency checking of the models. On the other hand, the direct transition from informal specifications to OWL puts quite high demands on ontology engineers. Ontology engineers directly defining OWL entities based on informal specifications have to deal with two problems at the same time: (A) "What are the entities and relations inherently described in the specification?" and (B) "How to represent them with OWL constructs?" Moreover, the latter question often has several possible answers – choosing different *OWL encoding styles*,[3] i.e., representing the same situation with different combinations of OWL constructs.

We have recently proposed a possible solution [4]: starting ontology development by creating a visual model in PURO language [8] representing the real world situation that is to be described by the ontology, thus answering the question *A*, and then configuring its automatic transformation to OWL following the desired encoding style, i.e., dealing with question *B*. The result of the transformation is an ontology *seed* consisting

[3] In previous publications (e.g., [4]), we used the term *OWL modeling styles*.

of classes, properties and domain, range and subClassOf axioms. This seed is then finalized by adding necessary axioms, labels, comments etc. in common IDE like Protégé. Our proposal does not replace common ontology development, it just allows making the first steps more explicit and supported by graphical tools: PURO Modeler [5] for the first step, and OBOWLMorph [4] for the transformation from PURO to OWL.

In this paper, we present an evaluation of PURO Modeler with users. Their performance creating the model in PURO is compared with creating an OWL ontology directly in Protégé.

PURO Language PURO[4] is an ontological modeling language recently drafted as common interlingua for different encoding styles in OWL. A model built in PURO is denoted as *ontological background model* (OBM). PURO inventory is very similar to that of OWL, assuring easy understandability and mappability to OWL. It is based on two distinctions: between particulars and universals and between relationships and objects (hence the PURO acronym). There are six basic entity types: B-object (particular object), B-type (type of object/type), B-relationship (particular relationship), B-relation (type of relationship), B-valuation (particular assertion of quantitative value) and B-attribute (type of valuation). An OBM consists of named entities of these types, plus of subTypeOf and instanceOf relationships. It always represents an example of a specific situation, i.e., the modeling should start from instances.

2 PURO Modeler and Its Possible Benefits

PURO Modeler is basically a web-based diagramming tool[5] designed for the PURO language. Its UI consists of a palette and a canvas. The palette serves for selecting 'tools' for adding instances of PURO terms and relationships between them, represented by nodes of different shapes and links of different styles. It is quite simple, there are 4 types of nodes and 3 types of links, which is enough to cover the whole PURO language. Figure 1 shows a partial screenshot of PURO Modeler including the palette and a part of an OBM.

There are three main differences in OBM-started ontology development compared to creating an ontology directly in OWL ontology editor such as Protégé. First, an OBM represents a specific example from the modeled domain. In other words, it is modeled at the level of instances, but including their types. On the other hand, when developing an ontology directly in OWL, the designer usually focuses on the T-Box. By our experience, ontology engineers think about example situations while creating the T-Box anyway, however, only implicitly. OBM allows to make such example situations explicit, which we assume might lead to achieving the intended coverage of the domain more easily. The results from the evaluation suggest this assumption is valid.

Second, the PURO language abstracts from specific aspects of OWL encoding. The most obvious example are n-ary relations, which have to be represented through reification in OWL. PURO allows to model n-ary relations as single objects, thus making the modeling easier and less error prone, as suggested by the evaluation – encoding of the

[4] Please refer to our previous publications ([8] and [4]) for more information.

[5] Available at `http://protegeserver.cz/puromodeler-v3.5`

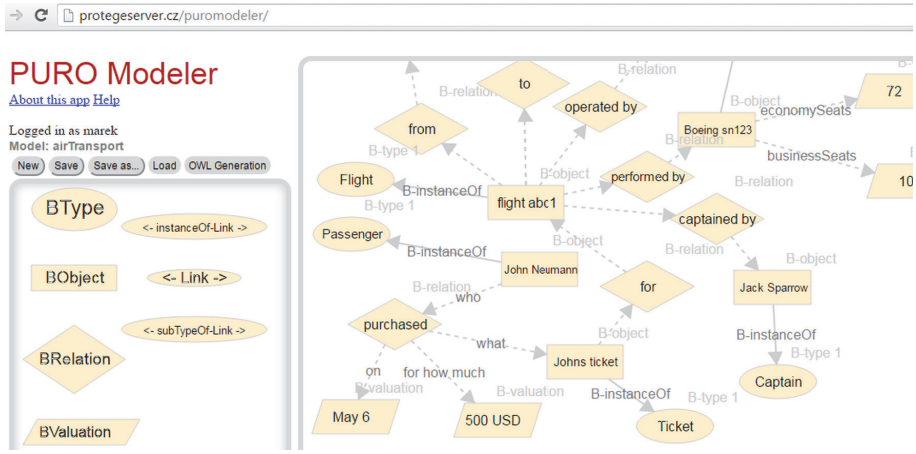


Fig. 1. The palette and a part of an example model in PURO modeler.

n-ary relation into OWL is done later by the automated PURO-to-OWL transformation in OBOWLMorph.

Third, in the OBM, the user can see all entities – types and properties – visualized in one model, while in editors like Protégé each entity type is shown in a separate subwindow. The all-in-one view is preferred by users, as suggested by our evaluation. However, this particular advantage can be achieved also in graphical OWL ontology editors such as OWLGrEd [1], where all entities are also shown and edited in one graph.

3 Evaluation

The evaluation[6] was done with 10 undergraduate students of an ontology engineering course. The students had basic knowledge about OWL (understanding of clasess and their hierarchy, instances, properties and domains/ranges) from the course lectures and had taken 90-minute tutorials about Protégé and PURO Modeler.[7] We prepared 2 example situations to be modeled by the students: an example from air transportation (A) and a human relationships example (B). Five students were asked to model A with Protégé, i.e., to create an ontology that will cover the described example, and to model B with PURO Modeler, i.e., to create an OBM that could be transformed to OWL ontology covering the example. The remaining 5 students modeled A in PURO and B with Protégé. Each student had 45 minutes to accomplish both tasks.[8] Each example consisted of (1) an abstract description of entity and relationship types and (2) an example situation from the domain.

[6] Details about the evaluation are at `http://protegeserver.cz/puroeval`

[7] Based on user and modeling guides, see `http://protegeserver.cz/puroeval`

[8] Due to lack of time, only 2 students finished both tasks. We took even unfinished results into account as they still allowed us to see what errors students made.

To make the results comparable, we instructed students to create only classes, properties and subClassOf and domain/range axioms in OWL, i.e., omit any possible complex classes, restrictions etc. since these are not created by the PURO-to-OWL transformation in OBOWLMorph. We measured the time each student needed to accomplish each task and then we examined the resulting ontologies and OBMs handed in by students. The students were also given a questionnaire focusing on comparison of PURO and direct OWL modeling at the end of the evaluation.

Correctness We classified the errors student made into three levels. Level 1 errors are against the syntax of the language (PURO or OWL). In PURO these are for example missing labels or wrong orientation of links. Level 2 errors are such that are correct syntactically, but do not make sense in the language: for example economy-class being a subclass of plane. Level 3 errors occur when something is modeled differently than in a gold-standard model created by us. We also tried to evaluate the overall severity of errors – whether the errors are critical and affect the whole model, or non-critical where at least part of the model is correct. An example of a critical error in case of PURO is when the students created only the types and did not include the instance-level entities.

There were no level 1 errors in the OWL ontologies as Protégé does not allow to make such errors. PURO Modeler checks for only some such errors and an obvious conclusion is that the application has to check for all syntactic errors as they occur very frequently: 7 out of 10 PURO models contained level 1 errors, in 4 cases critical.

The amount of level 2 errors is quite comparable between the two tools. 6 of 9 ontologies (one student did not hand in the result) and 4 of 6 syntactically mostly correct PURO models[9] contained such errors, which is actually the same percentage. A common critical level 2 error in case of PURO models was modeling only the 'T-Box' part.

There were no critical level 3 errors, i.e., all models without critical level 1 or 2 errors could be used without major changes. There was 1 PURO model without error, however unfinished due to lack of time. No OWL ontology was without errors. The students had problems modeling n-ary relationships in OWL. For example, no one was able to model the "is angry at someone because of something" relationship.

Completeness Models that did not contain any critical errors were evaluated in terms of completeness, i.e., whether they covered all relationships and entities described in the real world situation. The entities or relationships were considered covered even when there were minor errors in the model. Based on that, we found out that only 1 OWL ontology had complete coverage, in contrast with 3 such PURO models.

Time Relevant time measurement is only for the first task (example A), as most students did not finish the second task. The average times were 32 minutes to create an OBM and 26 minutes for ontology created directly in OWL.

Questionnaire The questionnaire contained 9 questions. First question was about how often students hesitated about mapping from the textual example description to PURO

[9] We did not check syntactically incorrect models for level 2 errors as it is meaningless.

term, with 5-level scale of answers between 'never' and 'very often'. Then there were 5 questions comparing PURO Modeler and Protégé in terms of UI-friendliness, fun, speed, easy understanding and discrete views in Protégé (classes and relationships in separate subwindows) vs. all-in-one view in PURO Modeler. These had also 5-level scale from 'definitely PURO Modeler' to 'definitely Protégé'. Next question asked students whether they used rather the general domain description or the concrete example as the main source for the modeling. Eighth question was about how hard it was to understand PURO language. In the last question the students could write in a free-text what would they improve in PURO Modeler.

The answers to the questionnaire[10] were generally pro-PURO, however, the students could have been biased by desire to speak positively about their teacher's research. The students were unsure about the description-to-PURO mapping mostly 'sometimes', considered PURO Modeler more user friendly, more fun and easier to understand than Protégé. They considered work in PURO Modeler faster than in Protégé, even though the reality is the opposite. The students used rather general descriptions as the source for the model and considered the PURO language rather easy to understand. They preferred the all-in-one view in PURO Modeler over separate windows in Protégé.

Summary The evaluation suggests modeling in PURO is a little bit slower and more error prone, partially due to less strict UI, however leads to better coverage of the domain. Both the time consumption and number of errors is quite comparable in OWL and PURO. According to the questionnaire, students generally prefer PURO Modeler over Protégé. Note that the evaluation is focused on PURO Modeler. To obtain an (seed of) actual OWL ontology, the PURO model would have to be transformed in OBOWLMorph. Such transformation, when using default settings, is fully automated and needs literally just three clicks. When not using default settings, i.e., changing the target OWL encoding style in OBOWLMorph, the evaluation would become much more complicated and we wanted to focus on the first step (PURO Modeler) first, leaving OBOWLMorph evaluation for future work.

4 Related Research

Starting ontology development from a simplified model OntoUML [3] is a conceptual modeling language based on UML and grounded in the Universal Foundational Ontology (UFO). OLED, the graphical editor for OntoUML, allows to transform it into OWL fragments. The transformation is hard-coded and each OntoUML element has its single OWL counterpart. Bauman [2] implemented XSLT transformation of conceptual models into XML Schema, while OWL as target is only mentioned as possible future work. The user can choose a sort of encoding style, e.g., whether to transform a concept to an XML attribute or child-element. To allow reusing existing ER diagrams, Fahad [6] designed their rule-based transformation to OWL ontologies. The framework is however not intended as a general ontology development alternative. In all mentioned OWL generation methods, the input model is created at the level of types. In our approach, in contrast, the input model is created as an example situation at the instance level.

[10] The whole questionnaire and answers are available at `https://goo.gl/MR6aS1`

Evaluation of ontology development tools Lambrix et al. [7] did an evaluation of Protégé 2000, Chimaera, DAG-Edit and OilEd. They asked users to perform specific tasks, but the evaluation was based on a questionnaire rather than the actual user performance. Our situation is somewhat different, as we are comparing different approaches, rather than just different tools. Similarly to our evaluation, they admit omitting tests of scalability – the evaluation was done only using small parts of an ontology.

5 Conclusions and Future Work

We compared common ontology development in Protégé with development started from an ontological background model in PURO Modeler in an evaluation with users. Results suggest that users prefer starting from PURO Modeler over Protégé, thanks to its simpler interface and better visualization, and are able to create models better covering the domain. Creating the background model takes however more time and is more error prone in terms of syntactic errors. Future research will include improving PURO Modeler and doing a full-scale evaluation including transformation to OWL in OBOWLMorph and finalizing the ontology in a common ontology editor like Protégé. We will also compare PURO Modeler with graphical ontology editors like OWLGrEd. A specific aspect that we will have to focus on is scalability – we will have to implement advanced visualization techniques and test development of full-scale ontologies started from OBMs.

Acknowledgments *The research is supported by UEP IGA F4/28/2016. Ondřej Zamazal is supported by CSF 14-14076P.*

References

1. Bārzdiņš, J., Bārzdiņš, G., Čerāns, K., Liepiņš, R., Sproģis, A.: OWLGrEd: a UML style graphical notation and editor for OWL 2. In: Proc. 7th International Workshop OWL: Experience and Directions (OWLED-2010) (2010)
2. Bauman, B.T.: Prying apart semantics and implementation: Generating XML schemata directly from ontologically sound conceptual models. In: Proceedings of Balisage: The Markup Conference 2009
3. Benevides, A.B., Guizzardi, G.: A model-based tool for conceptual modeling and domain ontology engineering in OntoUML. In: Enterprise Information Systems, pp. 528–538. Springer (2009)
4. Dudáš, M., Hanzal, T., Svátek, V., Zamazal, O.: OBOWLMorph: Starting ontology development from PURO background models. In: International Experiences and Directions Workshop on OWL. pp. 14–20. Springer (2015)
5. Dudáš, M., Hanzal, T., Svátek, V.: What can the ontology describe? Visualizing local coverage in PURO modeler. In: VISUAL@EKAW (2014)
6. Fahad, M.: Er2owl: Generating owl ontology from er diagram. In: Intelligent Information Processing IV, pp. 28–37. Springer (2008)
7. Lambrix, P., Habbouche, M., Perez, M.: Evaluation of ontology development tools for bioinformatics. Bioinformatics 19(12), 1564–1571 (2003)
8. Svátek, V., Homola, M., Kluka, J., Vacura, M.: Metamodeling-based coherence checking of OWL vocabulary background models. In: OWLED (2013)

LD-VOWL: Extracting and Visualizing Schema Information for Linked Data

Marc Weise[1], Steffen Lohmann[2], Florian Haag[1]

[1] Institute for Visualization and Interactive Systems (VIS), University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany
[2] Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS), Schloss Birlinghoven, 53757 Sankt Augustin, Germany

Abstract. Users currently face the problem that schema information for Linked Data is often not available. If it is available, it tends to be incomplete or does not adequately represent the data. It can therefore be hard for users to get an impression of the data provided by some Linked Data source. In this paper, we introduce LD-VOWL, a web-based tool that extracts and visualizes schema information of Linked Data sources based on the VOWL notation. SPARQL queries are used to infer the schema information from the data of the source, which is then gradually added to an interactive VOWL graph visualization. We tested LD-VOWL on a number of Linked Data endpoints with promising results.

Keywords: Linked Data, Schema Extraction, Visualization, SPARQL, RDF, OWL.

1 Introduction

A huge amount of Linked Data has been published in recent years, and is ready for consumption [2,5]. A large portion of this data is available in RDF format and can be queried using the standardized query language SPARQL [4,5]. The data often does not follow a strict schema, but typically different ontologies and vocabularies are used to describe it in a flexible way. On the one hand, this flexibility is an important characteristic and benefit of Linked Data; on the other hand, it can make it difficult to get an idea of what data is actually provided by a SPARQL endpoint. Visualizations can help to get a better overview of the type and structure of the data and can serve as a useful starting point for further querying and analysis.

In this paper, we introduce LD-VOWL, a tool that extracts and visualizes schema information from Linked Data endpoints, based on a number of SPARQL queries. This schema information is then incrementally added to an interactive graph visualization, using a slightly adapted version of the Visual Notation for OWL Ontologies (VOWL) [13,14].

2 Related Work

There are surprisingly few works concerning the extraction and visualization of schema information from Linked Data. Presutti et al. describe an approach of extracting *core*

knowledge [16] from Linked Data by detecting knowledge patterns. Central types and properties are identified by their betweenness and number of instances. In contrast to our approach, Presutti et al. focus on the detection of patterns in the data but not on the extraction and visualization of schema information.

Peroni et al. developed an approach for the automatic identification of *key concepts* [15]. Different from our work, the approach runs on ontologies and not on Linked Data. They use a couple of metrics, such as the length of concept names and their centrality in the graph structure, to find natural categories in the dataset. The concepts are also weighted by their popularity, which is defined as the number of results found by a search engine.

Another related work is QueryVOWL [9], which is also based on the VOWL notation and enables users without prior knowledge about RDF and SPARQL to query Linked Data. A graph consisting of VOWL elements is gradually constructed by the user and mapped to SPARQL queries which are sent to an endpoint. However, in contrast to our approach, QueryVOWL does not provide an overview visualization of the dataset but assumes that the user has already an idea of the type and structure of the data and knows how to start the querying process—as it is also assumed by many related querying approaches, such as LodLive [7] or the RelFinder [10].

Other works are concerned with the recommendation of concepts based on Linked Data [8,17], or follow general approaches of applying *formal concept analysis* to the Semantic Web [11].

3 Extraction of Schema Information

The schema extraction in LD-VOWL uses a *class-centric perspective*, i.e., the classes are extracted first and define the view on the Linked Data source. The classes are then connected by properties and enriched by datatypes. A class-centric perspective is very common in ontology engineering and fits well with the node-link paradigm of the graph visualization that the VOWL notation is based upon [13].

The extraction is realized by dynamically generated SPARQL queries revealing the schema information from a given dataset based on a couple of assumptions. For these queries, we had to find a trade-off between the number of required requests and the complexity of the queries. Since the SPARQL endpoints of Linked Data sources can have strict limits in terms of execution time, the queries must not be too complex. At the same time, we were aiming for displaying parts of the retrieved schema information as soon as possible, hence short response times were important as well. In addition, short response times were important, as we were aiming for displaying parts of the retrieved schema information as soon as possible to the users to minimize waiting time. Therefore, our priority was on using simple SPARQL queries, while we were also interested in limiting the total number of requests.

The SPARQL queries are sent in a stepwise approach, based on a couple of assumptions that are detailed in the following:

1. **Extraction of classes with the most instances:** A generic SPARQL query asking for the n classes with the most instances is sent to the endpoint first (where n is a user-defined upper limit). Listing 1.1 (Appendix) shows this query for the default

limit of $n = 10$. The results of this query serve as a starting point for further extractions.

This approach is based on the assumption that a dataset is well represented by the classes having the most instances. On the other hand, these classes are often also the more generic ones. Therefore, we integrated three strategies to avoid a too generic visualization:

(a) All built-in classes and properties of RDF, RDFS, OWL and optionally SKOS are contained in a blacklist that is filtered by default.

(b) Users can customize this blacklist by adding or removing classes according to their needs. For instance, they can remove `owl:Thing` from the list to include it in the visualization or add `foaf:Agent` to filter it too.

(c) Users can increase the number n of retrieved classes if the n initially retrieved classes are too generic, by adapting the limit of classes accordingly.

2. **Detection of subclasses, equivalent and disjoint classes:** Based on the n extracted classes with most instances, further SPARQL queries are sent to the endpoint in order to detect classes that can be considered equivalent, subclasses or disjoint classes. This is done by a pairwise comparison of the numbers of shared instances for all n classes, using the following assumptions:

   (a) If the number of shared instances of two classes is equal to the number of instances of each individual class, the classes are assumed to be equivalent.

   (b) If the number of shared instances of two classes is equal to the number of instances of the class having fewer instances, the class with fewer instances is considered a proper subset of the other class, which indicates a subclass relation between the two classes.

   (c) If there are no common instances at all, the two classes are considered to be disjoint.

   All three assumptions are based entirely on the actual data retrieved from the endpoint. For instance, two classes might not be explicitly defined as disjoint; however, if they do not share any instances in the dataset, a disjoint relationship will be inferred following the above assumption. This informs users that any search for individuals in that dataset which belong to both classes will be in vain.

3. **Retrieval of object properties:** In the third step, properties between the instances of the classes are retrieved. As with the classes, we retrieve the most frequently used properties first, i.e., properties having the greatest number of subject individuals (see example in Listing 1.2, Appendix). This also includes property loops, i.e., properties where the subject and object individuals are from the same class.

   As there can be a huge amount of different properties between the instances of two classes, we retrieve the properties in an incremental manner. When using a single SPARQL query, the execution of the query could take a very long time, possibly too long for Linked Data endpoints that have a strict limit for the execution time. Therefore, we choose the following approach in LD-VOWL: Starting with a limit of l properties, this limit is doubled with each SPARQL query sent until all properties are retrieved.

4. **Retrieval of datatype properties:** In the fourth step, LD-VOWL retrieves datatypes linked with the instances of the extracted classes. This step can be performed either after the third step or in parallel to it. LD-VOWL executes it in parallel in order

to avoid the impression that there are no datatypes defined for the retrieved classes due to the delayed retrieval and visualization (remember that LD-VOWL follows a stepwise approach and visualizes the elements as soon as they are extracted).
For each class, LD-VOWL sends queries that retrieve up to m datatypes which are most often used with the instances of that class (Listing 1.3, Appendix). After the datatypes are retrieved, the properties that connect the instances of the classes with these datatypes are fetched in a second step (Listing 1.4, Appendix). The reason for this two-step approach is again the limited execution time of many SPARQL endpoints. In addition, it supports our goal of visualizing the extracted schema information as quickly as possible, even if it is still incomplete. This requires the use of placeholders as labels for the datatype properties in the visualization as long as the actual properties are unknown.

It must be noted that due to the pairwise retrieval in both step two and three of the extraction process, the number of SPARQL requests that need to be sent grows quadratically with the number of classes n retrieved in the first step (i.e., $N_{requests} \in \mathcal{O}(n^2)$). Thus, we recommend to select the number of classes n that are initially retrieved with care and in accordance to the endpoint performance (LD-VOWL currently uses $n = 10$ as default).

4 Visualization Based on VOWL

LD-VOWL uses VOWL [13,14] for the visualization of the extracted schema information. We had to make some minor modifications to VOWL in order to address the peculiarities arising when visualizing information extracted from Linked Data.

In accordance with VOWL, extracted classes are represented as circle nodes in a force-directed layout (see Figure 1). The radii of the circles refer to the number of instances of the classes. Extracted properties are shown as directed and labeled edges linking the nodes. Different from VOWL, multiple properties between instances of the same pair of classes are merged into one edge. The more different properties exist between the instances of a pair of classes, the broader the edge is drawn. If different properties are merged into one edge, the property which occurs most often is considered most important—analogous to the class extraction principle. Therefore, the label of this property is shown on the edge, with the number of properties that have been merged given in brackets. If we would not merge those properties into one edge, this could result in a large number of edges being displayed between two classes, which would quickly clutter the visualization. Datatypes are displayed as yellow rectangles with a black border, like it is specified by VOWL. Accordingly, datatype properties linking the class instances with the datatypes are shown as edges with a green label.

The ontology or vocabulary comprising most of the classes is set as the main namespace of the dataset. The recommended default color of VOWL (light blue) is used as the background color of all elements in this namespace. All other ontologies and vocabularies that are part of the extracted schema have a different background color and inverted font color (white) in accordance with the VOWL specification. The colors used to indicate and group these other namespaces range from dark blue to pink in order to make different namespaces easily distinguishable in the visualization.

5 Implementation and User Interface

LD-VOWL is a web application implemented in JavaScript that sends SPARQL queries via HTTP GET to extract the schema information[3], and uses web standards like HTML5, CSS and SVG to display the extracted information. Furthermore, it makes use of the visualization toolkit D3 [6] for computing and displaying the force-directed graph.[4] The user interface of LD-VOWL is inspired by WebVOWL [12] and consists of three views:

1. The *start view* allows the user to select a Linked Data source by either entering the URL of its SPARQL endpoint or selecting one from a predefined list.
2. The *main view* (see Figure 1) shows the visualization of the extracted schema information. It is complemented by a sidebar with controls and information details.
3. The *settings view* enables the user to adjust the extraction by editing the blacklist or the language of labels, among others.

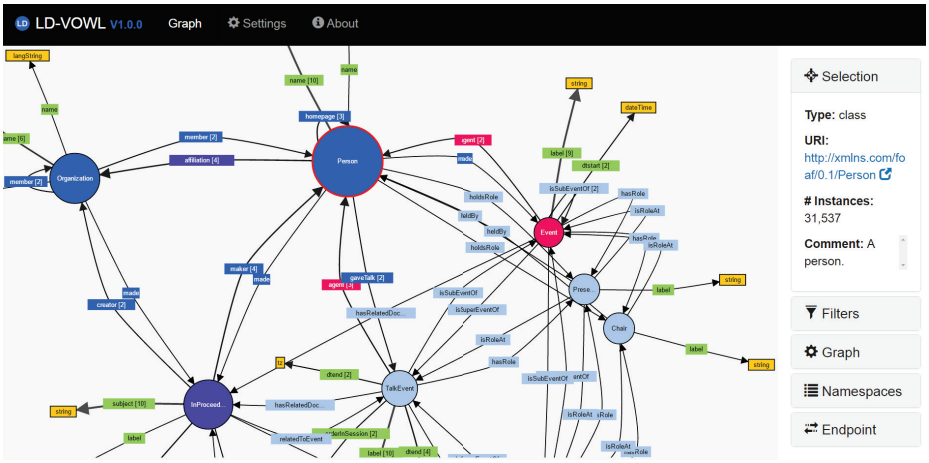


Fig. 1. LD-VOWL applied to the SPARQL endpoint of the Semantic Web Conference Corpus [3].

Users can zoom and pan to adjust the visible area and position of the graph that is shown in the main view. They can also modify the graph layout via drag and drop or by changing the average edge length. Furthermore, LD-VOWL provides options to filter parts of the extracted information in the visualization, such as datatypes, property loops, subclass relations, and disjoint classes. All nodes and edges in the graph visualization can be selected to see details on demand, for instance, the exact number of instances of a class or the list of all properties that connect two classes. URIs are displayed as hyperlinks, i.e., users can click on them to view further information (if available).

[3] Note that there are some endpoint requirements with regard to the supported SPARQL constructs, returned file format, handling of cross-origin requests, etc.

[4] A demo of LD-VOWL is available at: `http://ldvowl.visualdataweb.org`.

Finally, users can control the namespace classification by flagging namespaces as belonging to the main vocabulary or being marked as external. Users can also decide whether different colors should be used for the external namespaces or not.

6 Discussion

To unleash the full potential of Linked Data, it is important that users can get a quick overview of the type and structure of the data provided by a SPARQL endpoint. In this paper, we presented LD-VOWL, which allows to extract and visualize schema information from SPARQL endpoints. It uses a number of SPARQL queries that help to structure the data and reveal how it is described by ontologies and vocabularies, based on a set of assumptions. This schema information is then incrementally added to an interactive graph visualization using the VOWL notation.

We implemented LD-VOWL as a web application and tested it on several SPARQL endpoints. The results of these tests showed that LD-VOWL can create comprehensible overviews of the content and structures of datasets within a few seconds to minutes, depending on the performance of the endpoint (i.e., the used server, middleware, etc.), the extraction parameters selected in LD-VOWL (variables l, m, n, see Section 3) and the size of the dataset (which may affect the query execution time).

However, the results also show that the scalability of LD-VOWL is limited in several regards. As mentioned in Section 3, the number of SPARQL requests that need to be sent grows quadratically with the number of classes n that are initially retrieved. For this purpose, LD-VOWL retrieves only those classes that have the most instances (if not on the blacklist), which comes with benefits and limitations: On the one hand, LD-VOWL intends to provide an *overview visualization*, which implies that not *all* information is shown for datasets that contain a lot of classes and properties. On the other hand, it could be useful to explore certain regions of the overview visualization in more detail by 'expanding' parts of the graph and extracting further information for those parts on demand. Therefore, we could envision to extend LD-VOWL with such an exploration mode, or combine it with related visual querying approaches like QueryVOWL [9]. In general, LD-VOWL can be easily integrated with other tools, as it runs completely on the client side and only requests the server via SPARQL.

A direction for future research would be the extraction of further ontology concepts from the Linked Data sources, such as inverse properties or set operators, by developing corresponding assumptions and extraction patterns. These additional concepts could again be visualized with VOWL, which provides graphical representations for a large number of OWL language constructs [13,14]. LD-VOWL would also benefit from additional interactive features enabling the users to highlight, filter and collapse parts of the graph, as they are implemented in WebVOWL [12]. Such interactive features can improve the visual scalability and support the exploration and analysis of the extracted schema information.

However, the visual scalability of node-link diagrams can only be improved up to a certain extent with interactive features: Although a node-link diagram as used by VOWL is very suitable to depict the structure of some dataset, its visual scalability is inherently limited. The readability usually decreases with the number of nodes and

edges that are visualized. Therefore, another important direction of research is to investigate better means of visualizing a large amount of structured information—as it could potentially be extracted with approaches like LD-VOWL—in a more compact way.

References

1. DBpedia endpoint. http://dbpedia.org/sparql
2. Linked Data. http://linkeddata.org
3. Semantic Web Dog Food endpoint. http://data.semanticweb.org/sparql
4. SPARQL Endpoints Status. http://sparqles.okfn.org
5. Bizer, C., Heath, T., Berners-Lee, T.: Linked data – the story so far. International Journal on Semantic Web and Information Systems 5(3), 1–22 (2009)
6. Bostock, M., Ogievetsky, V., Heer, J.: D^3 data-driven documents. IEEE Transactions on Visualization and Computer Graphics 17(12), 2301–2309 (2011)
7. Camarda, D.V., Mazzini, S., Antonuccio, A.: LodLive, exploring the web of data. In: 8th International Conference on Semantic Systems (I-SEMANTICS '12). pp. 197–200. ACM (2012)
8. Damljanovic, D., Stankovic, M., Laublet, P.: Linked data-based concept recommendation: Comparison of different methods in open innovation scenario. In: 9th Extended Semantic Web Conference (ESWC '12). LNCS, vol. 7295, pp. 24–38. Springer (2012)
9. Haag, F., Lohmann, S., Siek, S., Ertl, T.: QueryVOWL: A visual query notation for linked data. In: ESWC 2015 Satellite Events. LNCS, vol. 9341, pp. 387–402. Springer (2015)
10. Heim, P., Hellmann, S., Lehmann, J., Lohmann, S., Stegemann, T.: RelFinder: Revealing relationships in RDF knowledge bases. In: 4th International Conference on Semantic and Digital Media Technologies (SAMT '09). LNCS, vol. 5887, pp. 182–187. Springer (2009)
11. Kirchberg, M., Leonardi, E., Tan, Y.S., Link, S., Ko, R.K.L., Lee, B.: Formal concept discovery in semantic web data. In: 10th International Conference on Formal Concept Analysis (ICFC '12). LNCS, vol. 7278, pp. 164–179. Springer (2012)
12. Lohmann, S., Link, V., Marbach, E., Negru, S.: WebVOWL: Web-based visualization of ontologies. In: EKAW 2014 Satellite Events. LNAI, vol. 8982, pp. 154–158. Springer (2015)
13. Lohmann, S., Negru, S., Haag, F., Ertl, T.: Visualizing ontologies with VOWL. Semantic Web 7(4), 399–419 (2016)
14. Negru, S., Lohmann, S., Haag, F.: VOWL: Visual notation for OWL ontologies. http://purl.org/vowl/ (2014)
15. Peroni, S., Motta, E., D'Aquin, M.: Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In: 3rd Asian Semantic Web Conference (ASWC '08). LNCS, vol. 5367, pp. 242–256. Springer (2008)
16. Presutti, V., Aroyo, L., Adamou, A., Schopman, B.A.C., Gangemi, A., Schreiber, G.: Extracting core knowledge from linked data. In: 2nd International Workshop on Consuming Linked Data (COLD '2011). CEUR Workshop Proceedings, vol. 782. CEUR-WS.org (2011)
17. Stankovic, M., Breitfuss, W., Laublet, P.: Linked-data based suggestion of relevant topics. In: 7th International Conference on Semantic Systems (I-SEMANTICS '11). pp. 49–55. ACM (2011)

A Examples of SPARQL Queries Used for the Schema Extraction

The following listings provide examples of SPARQL queries used by LD-VOWL to extract schema information from Linked Data sources, based on a couple of assumptions that are described in Section 3.

```
SELECT DISTINCT ?class (COUNT(?instance) AS ?instanceCount)
WHERE {
  ?instance a ?class .
}
GROUP BY ?class
ORDER BY DESC(?instanceCount)
LIMIT 10 OFFSET 0
```

Listing 1.1. SPARQL query retrieving the $n = 10$ classes having the most instances.

```
SELECT (COUNT(?originInstance) AS ?count) ?prop
WHERE {
  ?originInstance a <http://dbpedia.org/ontology/Agent> .
  ?targetInstance a <http://xmlns.com/foaf/0.1/Document> .
  ?originInstance ?prop ?targetInstance .
}
GROUP BY ?prop
ORDER BY DESC(?count)
LIMIT 10 OFFSET 0
```

Listing 1.2. SPARQL query retrieving the $l = 10$ most often used object properties connecting instances of the classes `Agent` and `Document` (run on the DBpedia endpoint [1]).

```
SELECT (COUNT(?val) AS ?valCount) ?valType
WHERE {
  ?instance a <http://dbpedia.org/ontology/Agent> .
  ?instance ?prop ?val .
  BIND(DATATYPE(?val) AS ?valType) .
}
GROUP BY ?valType
ORDER BY DESC(?valCount)
LIMIT 10
```

Listing 1.3. SPARQL query retrieving the $m = 10$ datatypes most often linked to the DBpedia class `Agent`.

```
SELECT DISTINCT ?prop
WHERE {
  ?instance a <http://dbpedia.org/ontology/Agent> .
  ?instance ?prop ?val .
  FILTER (
      DATATYPE(?val) = <http://www.w3.org/2001/XMLSchema#string>
  )
}
LIMIT 10 OFFSET 0
```

Listing 1.4. SPARQL query retrieving properties between instances of the DBpedia class `Agent` and the linked datatype `string`.

Visual Development & Analysis of Coreference Resolution Systems with CORVIDAE

Nico Möller[1] and Gunther Heidemann[1]

Institute of Cognitive Science, University of Osnabrück, 49069 Osnabrück, Germany

Abstract. Communication whether in verbal or written form is part of our daily life. Hence, we as humans have developed a set of skills that enable us to follow a discourse and extract important information from a text quite easily. For a machine however, language understanding is a quite challenging problem and considered to be AI-complete, i.e. a machine must reach human level intelligence in order to solve this task. Recent developments, especially those forming the semantic web, offer new ways of incorporating world knowledge into natural language processing methods. In this paper, we present our latest advancements on CORVIDAE (Coreference Resolution Visual Development & Analysis Environment), a tool for NLP developers to analyse and eventually improve coreference resolution algorithms specially designed for those that interact with world knowledge.

1 Introduction

Coreference resolution (CR) is a subtask of information extraction and describes the task of identifying all mentions in a given document and group those together that refer to the same entity [20]. CR is one of the core tasks in information extraction, making it a necessary preprocessing step before other algorithms can be applied. It has been an active field of research since the 1960s. Whereas research in the early years of CR was dominated by heuristic approaches built on computational theories of discourse [5, 6, 27], methods on based machine learning became more and more popular due to the broader availability and increased processing power of computers in the 1990s. Most common methods are based on supervised learning, using string matching, syntactic, grammatical or semantic features on those mentions. Observing the course of development in this field, a trend becomes visible that starts with local features [1, 17] and goes on towards more global models [15, 24]. The next logical step would be to go beyond global features, i.e. incorporating pieces of information that are not in the document, but can help to solve this task. This includes semantic relatedness features extracted from knowledge bases like *WordNet*, *Wikipedia* or *YAGO* that already have proven to be valuable additions [22, 25]. Additionally, there have been approaches solving subtasks in information extraction like coreference resolution and named entity linking in a joint fashion rather than separately [14, 9]. An elaborated error analysis of the state-of-the-art Stanford *CoreNLP* system has shown that 41.7% of errors can be attributed to the lack of background knowledge of the system [12]. Another motivation behind this shift can be found in the increased interest in information extraction and analysis in the recent years. Besides *Big Data*, another keyword that kept appearing in

the recent years is the one of the *semantic web* [3]. The goal of the semantic web is to increase the exchangeability of data as well as its usability. Web documents should be tagged with additional information that set a context for this document creating a machine-readable knowledge-graph that contains information about persons, organisations, places or events mentioned in the text as well as their connections to other entities. Without proper background knowledge, it is impossible to integrate extracted information correctly into an existing knowledge base. Taking the outlook on data production[1] into consideration as well as the fact that only 4 out of 175 million active domains [7] use the semantic mark-up on their websites[2], it seems a good idea to work on increasing the quality of coreference resolution systems as these play a crucial role in solving problems currently encountered in *Big Data* analysis and fulfilling the dream of the *semantic web*.

2 Related Work

Tools for visualising coreference annotation data can roughly be divided into two groups. The first group of tools focuses on the annotation itself with the aim of creating data that can be used as training input for NLP algorithms like coreference resolution. Most popular along these are *MMAX2* [19], *PAlinkA* [21] and *BRAT* [28]. However, those are mainly text-based with only a very limited capacity of visualising data besides a few visual cues like highlighting mention groups or showing links between mentions. Another way of visualising coreference data was introduced by the *TrEd* annotator using trees to visualise coreference as well as other tree based annotations [8]. The *SUCRE* project in contrast, utilised self-organising maps to visualise coreference features [4]. Additionally, human annotators should be provided with suggestions for possible coreferences in a semi-supervised fashion to speed up the annotation process.

Exploring already annotated data can be done with those tools, but due to their intended purpose, they lack important features that are needed for error analysis. Crucial would be the capability of comparing a data set against a gold standard annotation.

Tools that focus on the NLP developer, on the other hand, are quite rare. A widely used toolkit for error analysis in coreference systems is that of Kummerfeld & Klein [11]. Their approach utilised transformation operations to automatically categorise errors in the output of coreference systems, but also lacks any functionality to visualise their results. Kuhn et al.[10] presented the *ICARUS Coreference Explorer (ICE)*. Specially designed to provide tools for visualisation, search and error analysis for coreference annotations. Besides a tree view similar to *TrEd*, it utilised the entity grid [2], a tabular view of entities in a document to give both a summed up view of mentioned entities as well as show changes of entity descriptions throughout the document. *ICE* however, is focused on the links between pairs of links, neglecting global features on groups of mentions and features beyond that. Complementing those is the tool from Martschat et al. [16], which provides a text-based visualisation similar to *BRAT*. Although the functionality to add world knowledge is mentioned, the system is not yet suitable to handle

[1] According to a recent study conducted by EMC as part of their *Digital Universe* Series, humanity is currently producing about 4.4 Zettabytes of data, which will tenfold by 2020

[2] `http://news.netcraft.com/archives/2015/10/16/october-2015-web-server-survey.html`

analysis on the output of cross-document coreference resolution or entity linking systems. To solve those problems we created *CORVIDAE* a tool for the visual analysis and development of coreference resolution systems that incorporate world knowledge.

3 CORVIDAE

CORVIDAE is a web-based application. The backend is written in *Scala*[3], built upon the the *Play* web application framework[4]. *HTML5* and *JavaScript* are the foundations for the frontend, which uses the *BRAT* library [5] as well as the *D3.js JavaScript* library[6] for interactive visualizations. For more details on the intended workflow with the application and interactions with existing CR systems [23, 13, 14, 9] have a look at our intitial presentation of *CORVIDAE* [18].

In the following subsections we present a few new and improved visualisation modes that focus on different parts of the error analysis. *CORVIDAE* follows Shneiderman's mantra[26], to provide the user with an overlook of the systems strengths and weaknesses first and details for an in-depth analysis later on. All of these visualisation modes are types of circular layouts, a compact drawing style for information visualisation that is especially popular in the area of bioinformatics.

3.1 Radial Sequence Diagrams

The radial sequence diagram is one of the core elements of *CORVIDAE* has undergone a few changes and additions since the last revision. As already pointed out before this visualisation mode is quite versatile and hence can be used in many different ways. It marks the entry point for almost any system analysis the NLP developer might want to perform, as it can be configured to give a quick overview on the most crucial error measures at one glance. Originally used to compare genome sequences, we utilise this technique to quickly compare a broad variety of results. We can use it to:

- visualise and compare different error metrics for one or more documents or system configurations,
- compare annotation results gained by different configurations for a single coreference system or results from different systems on a single document,
- compare different documents to find out if they get linked to the same entities,
- analyse the propagation of errors in the multi-sieve model level wise.

Another big advantage of this view mode is that due to its compact design it allows not only to compare two results to one another but even multiple ones. This feature is especially helpful when we enter the area of cross-document coreference resolution, that cannot be handled properly by the solutions presented in section 2. Figures 1 and 2 show two usage examples for the radial sequence diagram.

[3] http://www.scala-lang.org/
[4] https://www.playframework.com/
[5] http://brat.nlplab.org/embed.html
[6] https://d3js.org/

Fig. 1. Error overview for different system configurations. Outer ring shows the number of possible errors, clustered by mention type and weighted by appearance in document. The inner rings show the error summaries for three different configurations. The order of the inner rings is also sortable, if the user wishes to focus on the worst/best performance for a certain error category. It is also possible split the inner rings, in case the user wants to investigate subcategories.

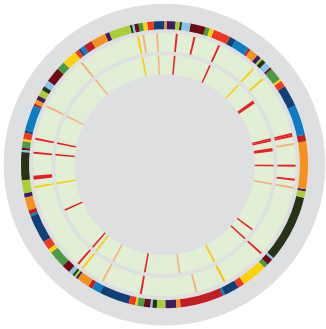


Fig. 2. A radial sequence diagram, comparing the annotation (outer sequence, found mentions, color coded according to cluster membership) from two CR system configurations (inner sequence) against a gold standard annotation. The inner sequence are color coded to show correct (light green), wrongly assigned(red), extra (yellow) and missed mentions (orange). In a similar fashion this view can be used to check the results from an entity linking module.

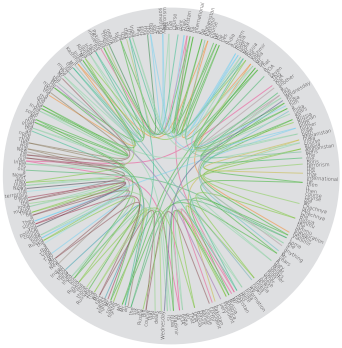


Fig. 3. A radial network diagram, showing links between different mentions within a text. Links are color coded according to cluster membership. This view supports highlighting, filtering and sorting, to enable a more detailed analysis of a certain error type, e.g. by analysing a single coreference chain.

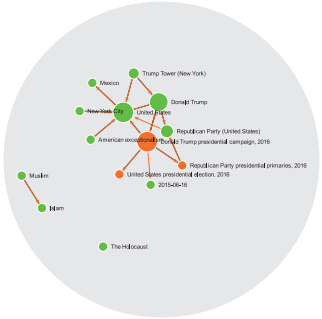


Fig. 4. A radial directed graph diagram, showing entities found within a document. Additional links and entities can be provided by a gold standard annotation as well as querying a linked knowledge base. Size corresponds to the number of in and outgoing links.

3.2 Radial Network Diagram

Figure 3 depicts an example of a radial network diagram, a of visualisation primarily used to display coreference chains throughout a document. Shown on the outer rim are the found mentions within a document, currently in the order of appearance within the document. Entity clusters are depicted by colour coding. Arcs connecting two mentions indicate a coreference between them. The mentions can also be sorted and split into their corresponding entity clusters for further inspection of the individual mentions. As mentioned before the *D3.js* library allows for interactivity, henceforth the visualisations allow for highlighting via hovering or filtering via queries, as well as displaying additional information like the linked real world entity when selecting an entity cluster. Theses functionality is quite essential to simplify the otherwise dense and complex visualisation and isolate single coreference chains the NLP developer wants to analyse.

In order to compare two annotation results or one against a gold standard, a differential view can be computed, highlighting differences in found mentions and links, while fading out the rest, which allows for an easy spotting errors.

3.3 Radial Directed Graph Diagram

The radial directed graph diagram has been incorporated in two different modes.

Mention centred: This mode allows for the visualisation of tree based coreference annotations similar to the ones found in *TrEd* or *ICE*, but instead of a triangular layout we are using a radial one, which allows for a much more compact and cleaner representation. Originating from the inner document root, nodes in the tree correspond to mentions in the text, whereas links indicate coreference between those. Each branch from the root node corresponds to a cluster representing an entity.

Entity centred: The second mode is concerned with the visualisation of semantic background knowledge.

It can visualise information extracted from the documents itself, but is not solely restricted to it. Named entity linking usually uses a knowledge base that serves as an anchor. These can be exploited to provide additional context for the NLP developer, as well as to evaluate and compare extracted information against the knowledge base. The colour of the links indicates that no (yellow/orange), supporting (green) or contradicting (red) information has been found in the knowledge base. For example if a was-born-in relation between entity a and entity b is mentioned within a sentence and this fact can be found in our database the link would be green, if no relation can be found the link would be orange [7] and red in case contradicting information has been found. The size of the dots corresponds to the number of in and outgoing edges. A simple example showing this can be found in figure 4.

The same technique as mentioned in section 3.2 can also be used on this type of visualisation. Mention centred this view allows to compare different sets of annotations for one document, whereas the entity centred view can be used to explore results of one cross-document coreference resolution system over two documents.

[7] if the relation comes from the gold annotation it would be yellow instead

4 Conclusions and Future Work

In this paper we our latest updates on *CORVIDAE* a tool designed for NLP developers for the visual error analysis of coreference systems. This tool offers a variety of circular visualisations to display coreference annotation data, which will help to analyse and debug cross-document coreference resolution algorithms. In its current state *CORVIDAE* supports three different circular visualisations, namely:

- radial sequence diagrams,
- radial network diagrams,
- radial directed graph diagrams.

Each is intended to support the NLP developer in tracking down, isolating and locating errors caused by the CR system. All of these visualisations are interactive and highly customizable, making it easy for the user to adapt the system to his needs. As a starting point for our analysis, we choose the state-of-the-art *CoreNLP* CR system, but *CORVIDAE* can easily be extended to support other systems as well. The next steps will be an extensive analysis of the joint systems mentioned in 2, to further investigate the interaction between named entity linking and coreference resolution with world knowledge and elaborate how this can be exploited to boost the performance in both. More Information on *CORVIDAE* as well as demos will be made available on a separate project website in the near future[8].

References

[1] C. Aone and S. W. Bennett. Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proceedings of ACL 1995*, pages 122–129. Association for Computational Linguistics, 1995.

[2] R. Barzilay and M. Lapata. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34, 2008.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 5 2001.

[4] A. Burkovski, W. Kessler, G. Heidemann, H. Kobdani, and H. Schütze. Self organizing maps in nlp: Exploration of coreference feature space. In *Proceedings of the 8th International Conference on Advances in Self-organizing Maps*, WSOM'11, pages 228–237, Berlin, Heidelberg, 2011. Springer-Verlag.

[5] B. J. Grosz. The representation and use of focus in a system for understanding dialogs. In *Proceedings of IJCAI 1977 - Volume 1*, IJCAI'77, pages 67–76, San Francisco, CA, USA, 1977. Morgan Kaufmann Publishers Inc.

[6] B. J. Grosz, A. K. Joshi, and S. Weinstein. Providing a unified account of definite noun phrases in discourse. In *Proceedings of ACL 1983*, pages 44–50. Association for Computational Linguistics, 1983.

[7] R. V. Guha. Light at the end of the tunnel. Talk at the 12th International Semantic Web Conference (ISWC), Sydney, 10 2013.

[8] J. Hajic, B. Vidová-Hladká, and P. Pajas. The prague dependency treebank: Annotation structure and support. In *Proceedings of the IRCS Workshop on Linguistic Databases*, pages 105–114, 2001.

[8] `https://ikw.uos.de/%7Ecv/publications/VOILA16`

[9] H. Hajishirzi, L. Zilles, D. S. Weld, and L. S. Zettlemoyer. Joint coreference resolution and named-entity linking with multi-pass sieves. In *EMNLP*, pages 289–299. ACL, 2013.

[10] J. Kuhn, M. Gärtner, A. Björkelund, G. Thiele, and W. Seeker. Visualization, search, and error analysis for coreference annotations. *ACL 2014*, page 7, 2014.

[11] J. K. Kummerfeld and D. Klein. Error-driven analysis of challenges in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 265–277, Seattle, Washington, USA, October 2013.

[12] H. Lee, A. Chang, Y. Peirsman, N. Chambers, M. Surdeanu, and D. Jurafsky. Deterministic Coreference Resolution Based on Entity-Centric, Precision-Ranked Rules. *Computational Linguistics*, 39(4):885–916, dec 2013.

[13] H. Lee, Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu, and D. Jurafsky. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, CONLL Shared Task '11, pages 28–34, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[14] H. Lee, M. Recasens, A. Chang, M. Surdeanu, and D. Jurafsky. Joint entity and event coreference resolution across documents. In *Proceedings of EMNLP-CoNLL 2012*, EMNLP-CoNLL '12, pages 489–500, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[15] X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proceedings of ACL 2014*, page 135. Association for Computational Linguistics, 2004.

[16] S. Martschat, T. Göckel, and M. Strube. Analyzing and visualizing coreference resolution errors. In *Proceedings of the 2015 Conference of the North Americal Chapter of the Association for Computational Linguistics: Demonstration Session,* Denver, Col., 31 May – 5 June 2015, pages 6–10, 2015.

[17] J. F. McCarthy and W. G. Lehnert. Using decision trees for coreference resolution. In *Proceedings of IJCAI 1995*, pages 1050–1055, 1995.

[18] N. Möller and G. Heidemann. Corvidae: Coreference resolution visual development & analysis environment. In *Joint Proceedings of the Posters and Demos Track of 12th International Conference on Semantic Systems - SEMANTiCS2016*, Posters&Demos @SEMANTiCS 2016, 2016.

[19] C. Müller and M. Strube. Multi-level annotation of linguistic data with MMAX2. In S. Braun, K. Kohn, and J. Mukherjee, editors, *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*, pages 197–214. Peter Lang, Frankfurt a.M., Germany, 2006.

[20] V. Ng. Supervised Noun Phrase Coreference Research: The First Fifteen Years. *Acl*, (July):1396–1411, 2010.

[21] C. Orăsan. PALinkA: a highly customizable tool for discourse annotation. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialog*, pages 39 – 43, Sapporo, Japan, July, 5 -6 2003.

[22] S. P. Ponzetto and M. Strube. Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 192–199. Association for Computational Linguistics, 2006.

[23] K. Raghunathan, H. Lee, S. Rangarajan, N. Chambers, M. Surdeanu, D. Jurafsky, and C. Manning. A multi-pass sieve for coreference resolution. In *Proceedings of EMNLP 2010*, EMNLP '10, pages 492–501, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[24] A. Rahman and V. Ng. Supervised models for coreference resolution. In *Proceedings of EMNLP 2009: Volume 2-Volume 2*, pages 968–977. Association for Computational Linguistics, 2009.

[25] A. Rahman and V. Ng. Coreference resolution with world knowledge. In *Proceedings of ACL 2011: Human Language Technologies - Volume 1*, HLT '11, pages 814–824, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[26] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343. IEEE, 1996.

[27] C. L. Sidner. Towards a computational theory of definite anaphora comprehension in english discourse. Technical report, Cambridge, MA, USA, 1979.

[28] P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations Session at EACL 2012*, Avignon, France, April 2012. Association for Computational Linguistics.

Advanced UML Style Visualization of OWL Ontologies

Jūlija Ovčiņņikova*, Kārlis Čerāns*

julija.ovcinnikova@lumii.lv, karlis.cerans@lumii.lv
Institute of Mathematics and Computer Science, University of Latvia
Raina blvd. 29, Riga, LV-1459, Latvia

Abstract. The OWLGrEd ontology editor allows graphical visualization and authoring of OWL 2.0 ontologies using a compact yet intuitive presentation that combines UML class diagram notation with textual OWL Manchester syntax for expressions. We describe here the approaches available for ontology presentation fine tuning within the OWLGrEd editor, namely the ontology visualization option framework and the editor plug-in mechanism together with concrete plug-ins aimed to enhance the ontology presenting and editing experience.

Keywords: OWL, OWLGrEd, UML-style ontology visualization, ontology visualization options

1 Introduction

Presenting OWL ontologies [1] in a comprehensible form is important for ontology designers and their users alike. The graphical form in general and UML class diagram notation in particular offers an option of basic visual ontology construct presentation that allows linking together constructs that are related in the ontology (e.g. an object property can be depicted as a line connecting its domain and range class boxes). UML class diagrams need to be extended to cope with full OWL 2.0 construct modeling. This is solved in different graphical notations in different ways. So, ODM [2], defines a UML profile for ontology presentation and OWLGrEd [3] and TopBraid Composer [4] integrate OWL Manchester Syntax [5] for presenting advanced OWL constructs in textual form. The uniqueness of OWLGrEd lies in its combined ability to lay out an ontology that is imported or created in the editor in a compact graphical UML-style form, and make further manual ontology editing/adjustment. VOWL [6] visualizes ontologies by another approach using graphical primitives both for object and data property presentation, so obtaining a more uniform ontology presentation in a dynamic, yet read-only, graph-like form. The VOWL presentation of the same ontology will also typically require more graphical elements, than OWLGrEd.

The real strength of ontology presentation in an editor like OWLGrEd comes from the user's ability to fine-tune the ontology diagram after its automated rendering to obtain documentation-ready ontology presentations. Such a fine tuning may involve the

* Supported, in part, by Latvian State Research program NexIT project No.1 "Technologies of ontologies, semantic web and security".

diagram object repositioning, as well as its re-structuring up to full ontology editing facilities available in the tool (including e.g. manual deletion of irrelevant information).

In order to achieve a high quality ontology presentation in the tool, even in the presence of manual fine tuning options available, the quality of first ontology diagram created upon the import of the ontology into the tool still remains very important. Since the UML diagrams, as well as the OWLGrEd notation allow for different presentations of the same semantic elements (e.g. a graphical and textual one), and different ontologies may correspond to different desirable ontology presentation options, we describe here a re-factored ontology visualization option framework offering a number of choices that the user can make already before importing the ontology into the tool.

We describe here also a number of OWLGrEd plugins that can be used for diagram refactoring services, as well as its structural and semantics extensions. This part of work extends the earlier authors' work on domain specific ontology visualizations [7,8] (this paper describes new re-factoring services plug-in, as well as reviews the plug-in mechanism and concrete plugin architecture from the ontology visual presentation perspective). The editor plugins described here are included in the OWLGrEd tool download and can be activated on-demand for concrete projects. The OWLGrEd editor with preinstalled configuration, as described here, is available at http://owlgred.lumii.lv/pp.

2 OWLGrEd Notation and Editor

OWLGrEd (http://owlgred.lumii.lv/) provides a graphical notation for OWL 2 [1], based on UML class diagrams. OWL classes are typically visualized as UML classes, data properties as class attributes, object properties as association roles, individuals as objects, cardinality restrictions on association domain class as UML cardinalities, etc. The UML class diagrams are enriched with new extension notations, e.g. (cf. [3,9]):

- fields in classes for *equivalent class*, *superclass* and *disjoint class* expressions written in Manchester OWL syntax [5];
- fields in association roles and attributes for *equivalent*, *disjoint* and *super* properties and fields for property characteristics, e.g., *functional*, *transitive*, etc.;
- anonymous classes containing *equivalent class* expression but no name;
- connectors (as lines) for visualizing binary *disjoint*, *equivalent*, etc. axioms;
- boxes with connectors for n-ary *disjoint*, *equivalent*, etc. axioms;
- connectors (lines) for visualizing object property restrictions *some*, *only*, *exactly*, as well as cardinality restrictions.

Fig. 1 contains a simple demonstration fragment of Latvian Medicine Registries ontology [10] in OWLGrEd notation, illustrating the class, data and object property, as well as subclass, sub-property and object property restriction notation, different ways of disjoint classes notations, class-level inline comments and ontology level annotations.

The OWLGrEd tool allows both for ontology authoring (with option to save the ontology in a standard textual format) and ontology visualization that includes automated ontology diagram formation and layouting step, followed by optional manual diagram fine tuning to obtain the highest quality rendering of the ontology.

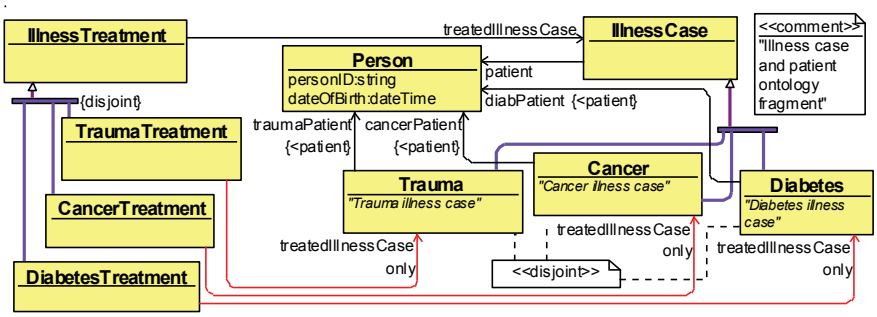


Fig. 1. Demo fragment of Latvian Medicine Registries Ontology

3 Ontology Visualization Parameters

The UML notation provides several options for presenting its semantic elements in different visual ways. This principle is kept also in OWLGrEd by including both graphical and textual notations for such semantic elements as subclass relations, object property relations, annotations and object properties as association roles or as attributes.

The automatic ontology visualization in OWLGrEd by default shall use the graphical notation, if there is no clear reason for switching to textual one. Fig. 2 shows a larger fragment of Medicine Registries Ontology in a graphical notation for all object properties and object property restrictions, and with separate disjoint classes axiom rendering.

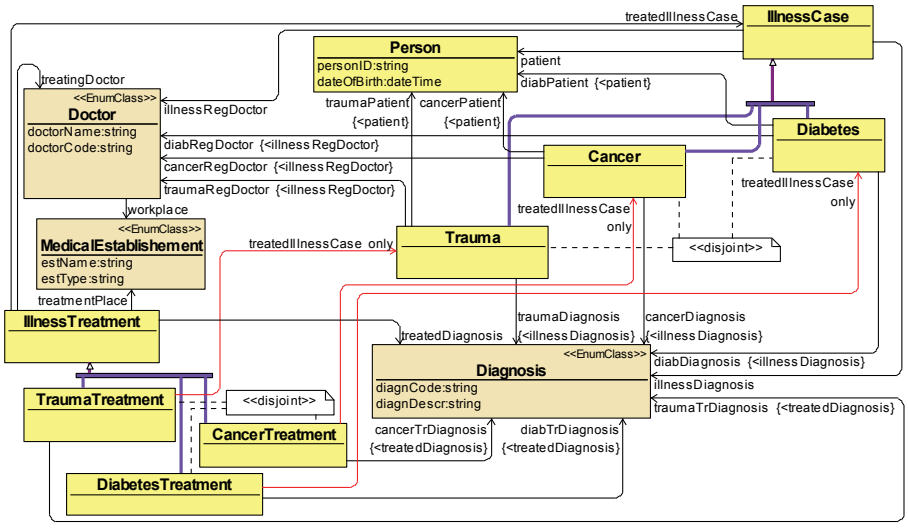


Fig. 2. Medicine Registries Ontology: a larger fragment in a graphical notation

For different ontologies and their parts, the appropriate element visualization methods may differ depending on the ontology content, e.g. if the ontology contains a small

number of object property restrictions, a better way to display object property restrictions would be graphical. However, it would not carry much information to display graphically object property restrictions that are based on owl:Thing as the target class. A textual form object property restriction visualization may be favorable also in the presence of a large number of these restrictions in the ontology.

The ontology visualization parameter framework offers parameters for inclusion / not inclusion of different object types in ontology visualization (e.g. one can choose to visualize ontology without data property annotations or without disjoint class axioms). Most of the framework parameters, however, refer to different ways (e.g. a graphical, or a textual form) of different type entity and axiom information visualization.

Figure 3 illustrates parameter setting interface including the option to represent subclass relations as text or graphically. In graphical mode there is a possibility to draw subclass relations as lines or combine them through forks, if there are more than one subclass for a given class.

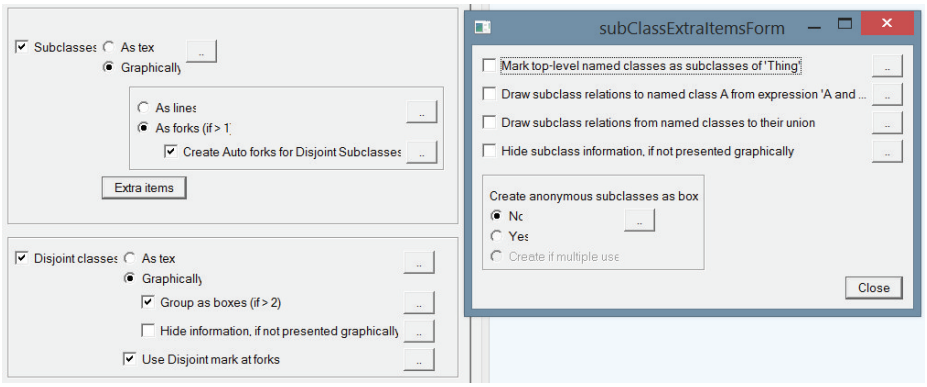


Fig. 3. Ontology Visualization Parameter form illustration

There are also parameters for disjoint class axiom rendering in Fig. 3. Along with disjoint classes representation as text or graphically (in a separate graphical node), there is a possibility to group disjointness relations, into boxes if there are more than two disjoint classes, and it is possible to hide information if it is not presented graphically.

An important option in UML is a possibility to mark the generalization sets as disjoint. This notation is brought also over to OWLGrEd since it can greatly simplify the graphical diagram appearance, if the disjointness axiom matches a level within the class hierarchy. If the 'Use Disjoint mark at forks' box is checked in the import parameter form, the ontology importer is able to identify the cases when all class subclasses are disjoint and add specific {disjoint} mark to the generalization element (cf. the generalization set notation for the subclasses of the *IllnessTreatment* class in Fig. 1).

Figure 3 also shows possible extra choices that can be made for the subclass relations, such as "Mark top-level named classes as subclasses of 'Thing'", "Draw subclass relations to named class A from expression 'A and ...'", "Draw subclass relations from named classes to their union", "Hide subclass information, if not presented graphically" and an option to choose whether to create anonymous subclasses as boxes, or not. The

extra parameters are left unchecked by default to obtain less loaded graphical presentation, however they can be turned on to show more clear class hierarchy, if desired.

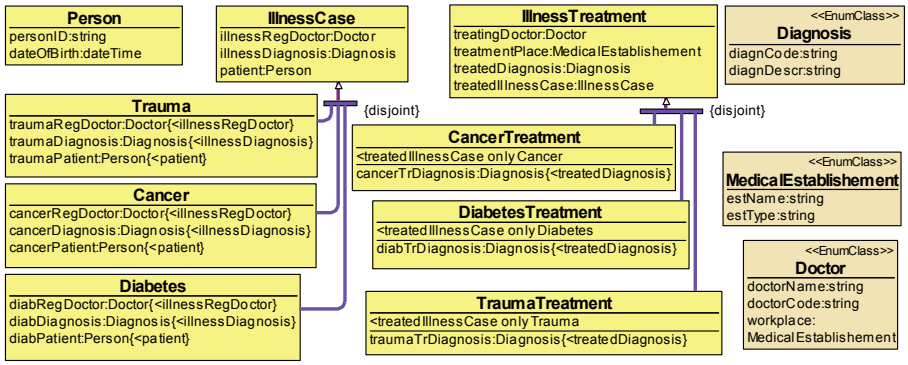


Fig. 4. Demo fragment of Latvian Medicine Registries Ontology: textual notation

Figure 4 illustrates the same Medicine Registries Ontology of Fig.2 with subclass relations presented graphically and disjoint marks at forks, while presenting textually the object properties and object property restrictions. The obtained ontology visualization has less elements that can be a benefit, however, the textual representation of object properties and object property restrictions somewhat hides the ontology structure.

4 Ontology Editor Plugins

To enhance the ontology visualization fine tuning experience after the ontology initial loading into the tool, as well as to support different custom ontology visualization means and the ontology editing work in general, the OWLGrEd ontology editor can be extended by means of plugins. The ontology editor plugins offer editor notation and environment extension means, in a similar manner, as profiles do for UML class diagrams [11,12]. A plugin to the ontology editor, may include structural editor symbol extensions with fields and visual effects, as well as editor behavior extensions.

The OLWGrEd plugin for ontology re-structuring includes transformations for adjusting the ontology visual presentation after its automatic visual rendering. It allows switching between the textual and graphical presentation form for concrete object property and object property restriction presentations within the ontology diagram, as well as re-factoring individual disjoint classes axiom presentation (e.g. from a separate disjoint-box to the 'disjoint' mark at the sub-class form symbol). The plugin can be used to transform e.g. the ontology diagram of Fig. 2 or Fig. 4 into the one of Fig. 5, where certain object properties (in the concrete example – the ones not having super-properties) are displayed in a graphical form to balance the structure presentation and compactness requirements. For an experienced OWLGrEd user it should take about 5 minutes to change manually the ontology diagram of e.g. Fig. 2 into the one of Fig. 5.

The ontology presentation in Fig. 5, as well as the presentation in Fig. 2 and Fig. 4 uses also a custom "enumerated class" boolean user field presentation.

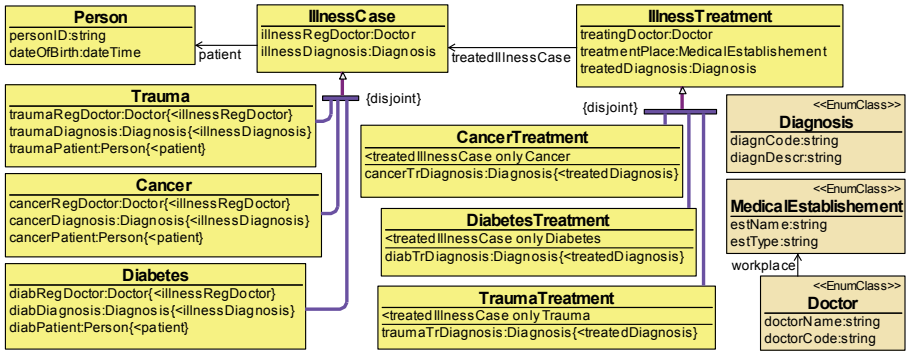


Fig. 5. Demo fragment of Latvian Medicine Registries Ontology: custom notation with plugins

In general, the editor extensions with symbol fields and graphical effects enhance the ontology presentation options by introducing domain-specific notations into the ontology presentation (the extensions can be configured to take into account the domain specific notations also during the ontology import); they are handled by a generic User Fields plugin [8] that is currently part of the default OWLGrEd editor configuration.

The available editor enhancements, supported by the User Fields plugin [8] include:

- custom fields, together with their semantics mappings (e.g. “enumerated class”);
- custom visual effects for text and choice fields and symbols dependent on concrete text or choice field values (e.g. a brown/darker enumerated class color);
- views applying certain visual effects to the entire diagram (e.g. hiding certain information from the presentation). The default OWLGrEd configuration includes views for horizontal and vertical alignment, and hiding all annotations.

An example of ontology editor plugin that involves both the user fields and custom functionality definition is the data ontology support plugin [13]; it contains the above considered “enumerated class” field definition. It also adds extra maximum cardinality 1 axioms to all data and object properties that are rendered in the editor in the textual form and do not have explicit cardinality specification. The plugin also records the class attribute ordering into an annotation so that this ordering could be further used in automated user interface generation for ontology-conformant data browsing (cf. [13]).

The other currently maintained OWLGrEd editor plugins are: (i) Controlled Natural Language (CNL) interface support plugin (used to augment the ontology with lexical information in [14]) and (ii) database to ontology mapping support plugin DBExpr [13].

5 Conclusions

The UML style visualization and authoring of OWL ontologies has been proven to be a successful alternative to other ontology visualization and authoring means. The OWLGrEd editor is able to offer a compact ontology notation by using OWL Manchester syntax [5] for textual presentation of advanced ontology constructs.

We have described three mechanisms that can be applied to enhance the ontology visual presentation and editing experience: (i) ontology visualization parameters; (ii) ontology profiles that may enhance the user experience in creating comprehensible and visually appealing ontology presentations; (iii) ontology visualization transformation means. Each of them can be further developed and enhanced to reflect new ontology rendering patterns, as well as to respond to the needs of concreate use cases.

A future work direction is also to move the OWLGrEd editor to the web environment, this depends on suitable tool definition framework availability providing the necessary editor definition infrastructure; this is work in progress in itself, as well.

References

1. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (2009)
2. ODM UML profile for OWL, http://www.omg.org/spec/ODM/1.0/PDF/
3. Barzdins, J., Barzdins, G., Cerans, K., Liepins, R., Sprogis, A.: OWLGrEd: a UML Style Graphical Notation and Editor for OWL 2. In: Proc. of OWLED 2010 (2010)
4. TopBraid Composer. http://www.topquadrant.com/tools/modeling-topbraid-composer-standard-edition/.
5. OWL 2 Manchester Syntax, http://www.w3.org/TR/owl2-manchester-syntax/
6. Lohmann, S., Negru, S., Haag F., Ertl, T.: Visualizing Ontologies with VOWL. Semantic Web 7(4), 399-419 (2016)
7. Cerans, K., Liepins, R., Sprogis, A., Ovcinnikova, J., Barzdins, G.: Domain-Specific OWL Ontology Visualization with OWLGrEd. In: ESWC 2012 Satellite Events, Springer LNCS, pp. 419-424, (2012)
8. Cerans, K., Ovcinnikova, J., Liepins, R., Sprogis, A.: Advanced OWL 2.0 Ontology Visualization in OWLGrEd. In: Caplinskas, A., Dzemyda, G., Lupeikiene, A., Vasilecas, O. (eds.), Databases and Information Systems VII, IOS Press, Frontiers in Artificial Intelligence and Applications, Vol 249, pp.41-54 (2013)
9. Barzdins, J., Cerans, K., Liepins, R., Sprogis, A.: UML Style Graphical Notation and Editor for OWL 2. In: Proc. of BIR'2010, LNBIP, Springer 2010, vol. 64, pp. 102-113 (2010)
10. Barzdins, G., Liepins, E., Veilande, M., Zviedris, M.: Semantic Latvia Approach in the Medical Domain. Proc. DB&IS2008. Haav, H.M., Kalja, A. (eds.), TUT Press, pp. 89-102, (2008).
11. Unified Modeling Language: Infrastructure, version 2.1. OMG Specification ptc/06-04-03, http://www.omg.org/docs/ptc/06-04-03.pdf
12. Unified Modeling Language: Superstructure, version 2.1. OMG Specification ptc/06-04-02, http://www.omg.org/docs/ptc/06-04-02.pdf
13. Cerans, K., Barzdins, G., Bumans, G., Ovcinnikova, J., Rikacovs, S., Romane, A. and Zviedris, M.: A Relational Database Semantic Re-Engineering Technology and Tools // Baltic Journal of Modern Computing (BJMC), Vol. 3 (2014), No. 3, pp. 183-198.
14. Liepins, R., Bojars, U., Gruzitis N., Cerans, K., Celms, E.: Towards Self-explanatory Ontology Visualization with Contextual Verbalization. In Arnicans, G., Arnicane, V., Borzovs, J., Niedrite, L. (eds.), Databases and Information Systems, Springer, Communications in Computer and Information Science, Vol 615, pp.3-17 (2016)

Exploring Visualization of Geospatial Ontologies Using Cesium

Abhishek V. Potnis, Surya S. Durbha

Centre of Studies in Resources Engineering,
Indian Institue of Technology Bombay, India
abhishekvpotnis@iitb.ac.in, sdurbha@csre.iitb.ac.in

Abstract. In recent years, there has been a substantial increase in the usage of geospatial data, not only by the scientific community but also by the general public. Considering the diverse and heterogeneous nature of geospatial applications around the world and their inter-dependence, there is an impending need for enabling sharing of semantics of such content-rich geospatial information. Geospatial ontologies form the building blocks for sharing of semantics of this information, thus ensuring interoperability. Visualization of geospatial ontologies from a spatio-temporal perspective can greatly benefit the process of knowledge engineering in the geospatial domain. This paper proposes to visually explore and reason over the instances of a geospatial ontology – the geopolitical ontology developed by the Food and Agriculture Organization of the United Nations using Cesium – a WebGL based virtual globe. It advocates the usage of Cesium for visualization of geospatial ontologies in general by demonstrating visualizations of geospatial data and their relationships.

Keywords: geospatial, ontology, visualization, cesiumjs, jowl, spatio-temporal, reasoning

1 Introduction

In recent years, the consumption of geospatial data has increased tremendously[1]. Location based services form a major consumer of the geospatial data consumption demographic. Effective visualization of geospatial ontologies can aid in exploration and understanding of the domain knowledge represented by the ontology and helps reduce information overload on the user. This paper proposes to exploit the 3-dimensional visualization capabilities of Cesium – a WebGL based virtual globe to effectively represent the instances of a geospatial ontology and reason over them using SPARQL queries and render the results on the globe.

1.1 Geospatial Ontologies and Linked Data

Geospatial Ontologies cater to modeling, analyzing and visualizing multi-modal heterogenous information in the geospatial domain[2]. They help establish the semantics of this geospatial information promoting information exchange and interoperability. The geographical data typically involves latitude, longitude and the altitude, supplemented by the temporal information and the feature of interest. Geospatial concepts and relationships are different from those of the non-geospatial data due to their spatio-temporal nature. This calls for a need to focus on effective visualization, analysis and exchange of geospatial data.

Linked Data forms an integral part of the Semantic Web ecosystem. Linked Data can be referred to as the collection of inter-related data-sets on Web[3]. It provides access not only to the data itself, but also to the relationships among the data. It facilitates interoperability of large scale data. It also enables querying and reasoning to performed on this data, thus attempting to fulfill the Semantic Web vision[4]. It helps increase the value of the data itself, by making it more accessible, shareable and understandable. Geospatial Linked Data refers to the Linked Data enriched with a geospatial dimension - spatial and temporal attributes. Advances in the field of Geospatial Linked Data effectively contributes to the vision of the Geospatial Semantic Web first proposed by Max Egenhofer[5].

1.2 Cesium – A WebGL based Virtual Globe

Cesium[1] is a free and open source WebGL based javaScript library that visualizes the earth as a globe in 3 dimensions on the web browser. It was developed by Analytical Graphics, Inc (AGI) in 2011 for dynamic data visualization in space. Over the years it has evolved to help visualize and serve industries from geospatial and oil and gas to agriculture, real estate, entertainment, and sports.

Cesium allows users to pan and zoom around the globe in 3D. It has a timeline feature that can be exploited to visualize multi-temporal data. Cesium allows rendering of 'entities'[6] on its globe to visualize objects in 3D on the earth surface. It allows 'entities' to be interactive, so one can click an entity of interest and be presented with its meta-data. The entities can be placed honoring the geographical coordinates (latitude, longitude and altitude) on the globe.

1.3 jOWL

jOWL[2][7] is a semantic javaScript library for navigating and visualizing OWL-RDFS documents. It extends the jQuery library with its ability to parse and visualize semantic web documents. It also supports use of abstract SPARQL-DL for querying

[1] https://cesiumjs.org/
[2] http://jowl.ontologyonline.org/

and reasoning over ontologies. This paper proposes to use jOWL over Cesium for parsing and reasoning over the Geopolitical Ontology.

2 Visualization of Geospatial Ontologies using Cesium

Cesium can be used to visualize information extracted and parsed from ontologies as 'entities' and render it in 3-D on the virtual globe, thus providing the user an immersive and a riveting experience. The spatial aspect of data can easily be modeled on Cesium, since it supports the cartographic coordinate system. The third dimension in Cesium could be used to visualize a feature of interest such as the temperature, air pressure or wind speed for better user interaction and understanding. For data, that is dependent on the altitude such as the air pressure or the wind speed, using the third dimension could prove to be effective in visualizing the change in this data. The temporal features of data too can be visualized using the Time-line[8] widget of Cesium. Color is another aspect of Cesium that can be exploited to visualize a parameter in Cesium. Cesium entities can be colored depending on the intensity of a predefined parameter from the geospatial ontology. For instance, in case of a Semantic Sensor Network based Ontology[3] of a Weather Station, we could use the temperature data to be visualized using colors on the regions. So a region with greater temperature value can be shown on the higher side of the color spectrum accompanied by the appropriate color legend. The meta-data or other additional information about an 'entity' can be rendered using the 'Info-Box' widget in Cesium.

Along with the visualization benefits that Cesium brings along, the fact that it runs inside a web browser is also a big plus today, where all applications are expected to be ubiquitous. Thus Cesium proves to be an effective tool in visualization of geospatial data-sets.

2.1 Dataset

For this paper, the Geopolitical Ontology[9] developed by the Food and Agriculture Organization of the United Nations has been visualized and reasoned upon. The ontology comprehensively lists all countries of the world with their geographic extent, geographic and socio-economic information and categories and groups they belong to. Geospatial concepts such as the geographic extent and relationships such as "hasBorderWith" have been focused on in the demo implementation of this paper. The instances of "hasStatistics" predicate depicting population and Gross Domestic Product of countries have been visualized over the Cesium globe. The dataset has also been reasoned upon using SPARQL-DL for obtaining the class a country belongs to, the countries that share its border and the groups it is aligned with.

[3] https://www.w3.org/2005/Incubator/ssn/ssnx/ssn

To explore visualization of fine grained data such as population over Cesium, the sample dataset of ORNL's LandScan[4] has been used. The LandScan dataset is available in the .lyr format and cannot be directly consumed by Cesium. It required conversion from the raster to a vector format and was later exported into GeoJSON. The GeoJSON was then parsed and transformed into a simple ontology with the class "Cell" having the following data properties - "latitude", "longitude" and "value". The cell represents a 1km spatial resolution pixel on the earth surface and the value represents the number of people in the cell. The LandScan data consisting of instances from the Sample LandScan Dataset of Cyprus has been published[5] as linked data as a part of this research study.

2.2 Visualization of the Geopolitical Ontology

The implementation demo that supports this paper uses Cesium and jOWL javascript libraries for visualization and reasoning. The geographic extent for each of the countries has been extracted and parsed from the geopolitical ontology to render it on the globe. The bounding boxes of each of the countries from the ontology, visualized on the globe are shown in Fig. 1. The bounding boxes are rectangle geometric entities in Cesium. For this proof of concept implementation, the colors are assigned randomly, with an alpha value of 0.5, thus ensuring transparency for the visibility of underlying Bing Maps imagery. Triples with <hasStatistics> predicate and Population and Gross Domestic Product objects can be selected and visualized on the globe.

Fig. 1. Visualization of the Geopolitical Ontology using Cesium

[4] http://web.ornl.gov/sci/landscan/landscan2011_sample.shtml
[5] http://home.iitb.ac.in/~abhishekvpotnis/voila2016/population_cyprus.owl

The user can pan and zoom around globe with the countries being identified by their bounding boxes. The bounding boxes are clickable entities. Clicking on a bounding box fires up three SPARQL-DL queries over the geopolitical ontology as a proof of concept demonstration.

SPARQL-DL Query 1: *Type(COUNTRY_NAME +",?class)*

It returns the class that the country or region belongs to that the user clicked. The ontology identifies the following classes for a country or region : self governing, non-self governing, disputed and other.

SPARQL-DL Query 2: *PropertyValue(COUNTRY_NAME,hasBorderWith,?country)*

It returns the countries that the country of interest shares its border with. 'hasBorderWith' is a spatial relationship in the geopolitical ontology. This spatial relationship has been effectively visualized in the demo that supports this paper.

SPARQL-DL Query 3: *PropertyValue("COUNTRY_NAME,isInGroup,?group)*

It returns the groups that the country of interest belongs to. The ontology has identified the multiple groups and sub-groups that a country or a region may belong to.

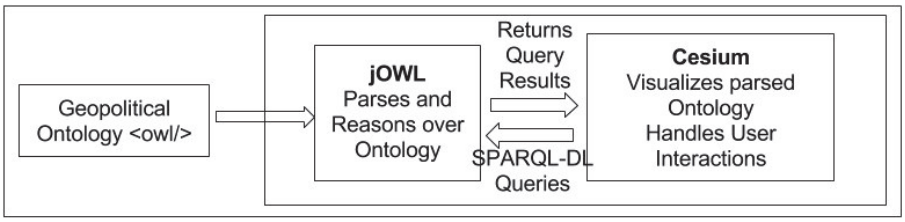


Fig. 2. Architectural Overview of the Proof Of Concept Implementation

The queries are run in real-time on the client-side over the Geopolitical ontology. The results are parsed and rendered in the info-box widget of Cesium. The results of the second query which exploits the 'hasBorderWith' spatial relationship are sent over to Cesium, where the list of countries are mapped to their bounding boxes. The bounding boxes of the neighboring countries and the country of interest are retained on the globe, while other bounding boxes are erased. The info-box at the right, displays the queries along with their results. The queries written in SPARQL-DL are executed over the geopolitical ontology using the jOWL library. Fig 2. shows the architectural overview of the proof of concept demonstration. The Geopolitical Ontology is parsed using jOWL and is rendered on the globe using Cesium. The geographic extent in the form maximum and minimum latitudes and longitudes are extracted from the parsed ontology and relayed to Cesium for rendering the bounding boxes. Cesium handles user interactions and relays them to jOWL for SPARQL-DL query evaluation. The query results are later returned to Cesium for rendering.

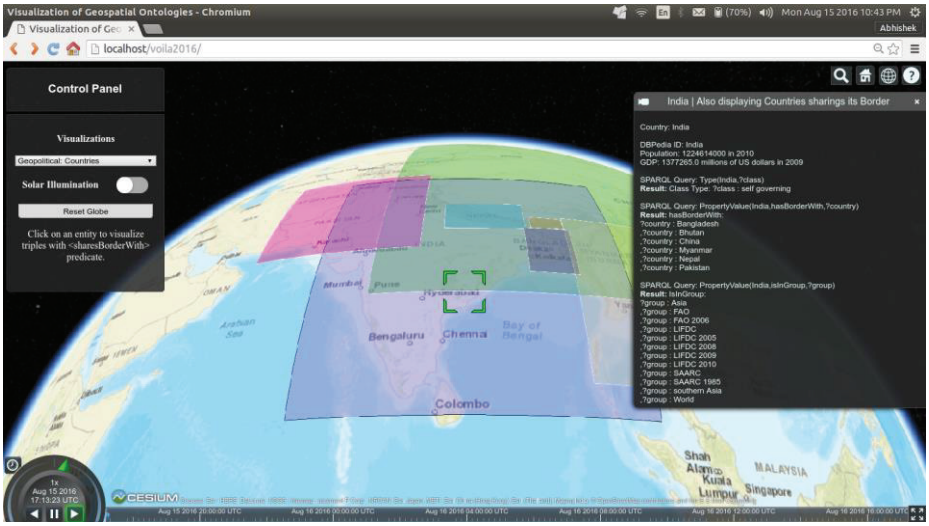


Fig. 3. Visualization of Spatial Relationship 'hasBorderWith' from the Geopolitical Ontology

Fig. 3 shows the country of interest being India. The countries that share their borders with India are shown along with India, while others are erased. The info-box to the right lists the SPARQL-DL queries along with their results. The info-box also lists the Name of the Country, its Population, its GDP and its DBPedia ID, all extracted from the Geopolitical Ontology. The DBPedia ID could later be used to access additional

Fig. 4. Visualization of triples with 'hasStatistics' predicate depicting Countries' Population

information from DBPedia which is not otherwise available in the Geopolitical Ontology. The population of countries has been visualized in Fig. 4. Hovering over a

bar will bring up the population value that the bar represents. From the visualization, it is evident that India and China are the most populous countries.

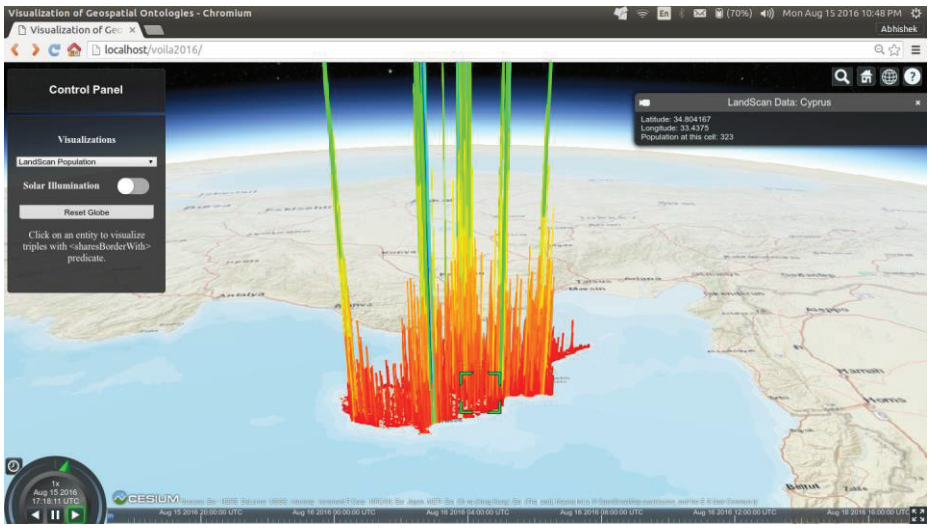


Fig. 5. Visualization of Cyprus LandScan Sample Linked Data

Fine-grained data such as the LandScan Sample Dataset of Cyprus has been visualized in Fig. 5. A simple ontology containing the class 'cell' having data properties – latitude, longitude and value along with instances acts as the input for this visualization. The jOWL library has been used to parse the ontology and its instances. Cesium thus effectively visualizes the population per cell in 3D. Hovering over the data brings up the latitude, longitude and the population at that cell. This allows user to visually explore the LandScan data. From the visualization, one can infer that the coastal region of Cyprus has lesser population as compared to the interiors. One can even point out the population hotspots from such visualizations.

The user can return to the initial state of full coverage of the geopolitical ontology by simply clicking outside a bounding box or by using the 'Reset Globe' button on the Globe Control Panel. There is also a feature to toggle the Solar Illumination on the globe in the Globe Control Panel on the left. The proof of concept implementation can be demoed at http://home.iitb.ac.in/~abhishekvpotnis/voila2016/.

3 Conclusion

This paper demonstrates the visualization capabilities of Cesium virtual globe for geospatial ontologies along with reasoning using the jOWL semantic library. The proof of concept demonstration reasons and visualizes the results of the spatial relationship of the neighboring countries with the country of interest. It does not account for visualization of temporal attributes due to lack of temporal data in the

geopolitical ontology, however they can be visualized effectively using Cesium's Timeline widget. This paper focuses on the visualization of data properties of individual classes. Future work would involve focusing on the object properties to visualize relations between individual classes. This study, in the future, also aims to explore visualization of relationships between population, agricultural production, GDP and HDI using AGROVAC LOD[10] and other sources of linked data using Cesium. Another direction that this research could explore is support for federated SPARQL queries over multiple ontologies to visualize multi-source data instances simultaneously.

Acknowledgments. The authors express their gratitude to the Cesium Community and the developers of jOWL for their time and effort in developing, maintaining and documenting the libraries.

References

1. Chan, Yupo, Visualization and Ontology of Geospatial Intelligence, Data Engineering: Mining, Information and Intelligence. Springer US, 2010
2. Budak Arpinar, I., Sheth, A., Ramakrishnan, C., Lynn Usery, E., Azami, M., & Kwan, M. P. (2006). Geospatial ontology development and semantic analytics.Transactions in GIS,10(4), 551-575.
3. Linked Data (https://www.w3.org/standards/semanticweb/data)
4. Tim Berners-Lee, James A. Hendler, Ora Lassila, The Semantic Web, Scientific American, 284(5):34-43, May 2001.
5. Egenhofer, M.J.: Toward the semantic geospatial web. In: Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems. pp. 1-4. GIS '02, ACM, New York, NY, USA (2002)
6. Visualizing Spatial Data (https://cesiumjs.org/tutorials/Visualizing-Spatial-Data/)
7. jOWL Semantic javaScript Library (http://jowl.ontologyonline.org/)
8. Time-Line – CesiumJS (https://cesiumjs.org/Cesium/Build/Documentation/Timeline.html)
9. Geopolitical Ontology – Food and Agriculture Organization of the United Nations (http://www.fao.org/countryprofiles/geoinfo/geopolitical/resource/organization)
10. AGROVAC Linked Open Data (http://aims.fao.org/standards/agrovoc/linked-open-data)

ViziQuer: Notation and Tool for Data Analysis SPARQL Queries

Kārlis Čerāns[1], Jūlija Ovčiņņikova[1]

karlis.cerans@lumii.lv, julija.ovcinnikova@lumii.lv
Institute of Mathematics and Computer Science, University of Latvia
Raina blvd. 29, Riga, LV-1459, Latvia

Abstract. We present a UML class diagram style notation for data analysis SPARQL query definition and its implementation in the ViziQuer tool that provides query definition environment and query translation into SPARQL 1.1. The notation and its implementation within the tool allows for rich value selection and condition expression language, as well as integrated data aggregation facilities, both essential for data analysis query definition.

Keywords: Visual query creation, SPARQL, RDF, Aggregate queries, Data analysis queries

1 Introduction

SPARQL [1] is de facto query language for RDF [2] databases. Semantic RDF/SPARQL technologies offer a higher-level view on data compared to the classical relational databases (RDB) with SQL query language. Thus, semantic technologies enable more direct involvement of various domain experts in data set definition, exploration and analysis. The database-to-ontology mapping techniques (cf. [3,4]) and ontology-based data access technologies [5,6] create the potential for SPARQL usage also over the massive amount of data stored and maintained in relational databases.

Still, the entirely textual form of SPARQL queries hinders its direct usage for IT professionals and non-professionals alike. A number of diagrammatic query notations to help formulating SPARQL queries have been proposed, including ViziQuer [7,8], OptiqueVQS [9] and QueryVOWL [10]. Their expressivity, however, is limited mostly to basic forms of queries, notably excluding support for aggregate queries included in SPARQL 1.1 [1] (except for authors' earlier demonstration [8]) and means for integrating rich expression language for conditions and selection attributes.

In a real-case scenario [11] it has been identified that users could formulate basic SPARQL queries via graphical notation and be satisfied with it on this query level. Still they lacked expressive query power to calculate different aggregated data that are

[1] Supported, in part, by Latvian State Research program NexIT project No.1 "Technologies of ontologies, semantic web and security" and ERDF project "Rich Visual Queries for Ontology-Based Data Access" (Ref. No. 1.1.1.1/16/A/277)"

important for any data inquiries of statistical nature. The ongoing work of re-engineering the example of [11] makes explicit the need of rich expression language in the queries, as well. The support for aggregation and rich expressions, as presented originally in this paper, makes a visual query language suitable for data analysis query formulation currently served mostly by various business intelligence tools.

We describe here the ViziQuer notation and tool for data analysis query definition and translation into SPARQL 1.1, involving data aggregation facilities and rich expression language integration. The ViziQuer notation, like the one of OptiqueVQS tool [9], is based on UML class diagrams used widely and successfully in engineering; the UML class diagrams have inspired also OWL ontology editor OWLGrEd [12].

The presentation of the ViziQuer language is organized in the form of query patterns covering different query definition aspects and corresponding to the situations naturally arising in the data analysis query creation. The query tool usage can be started just after mastering the simplest query definition patterns, so the language and tool usage is kept low-entry. The advanced query patterns, including the ones for expression language, should not be regarded as prohibitive for motivated end users (similarly, as also non IT-experts can master using expression notation, e.g., in Microsoft Excel).

In the following, Section 2 introduces basic query notation, following by the aggregate query patterns in Section 3 and expression patterns in Section 4. Section 5 concludes the paper. The resources for the example of this paper are available at http://viziquer.lumii.lv/demo/miniUniv.

2 Basic Query Notation

A query in the ViziQuer notation is a graph of class boxes connected with association links. In a typical query both the class boxes and association links will have the class/association names specified and at least some class boxes would contain specified selection attributes. The interpretation of such a query graph is to define a class instance pattern with an instance corresponding to each specified class, its data properties corresponding to the specified attributes, and the object property links between the instances corresponding to the associations linking the classes.

The query diagram shows the local names of classes, associations and attributes, their mapping to the full names is available in the data schema model that has to be loaded in the tool before query creation[2].

We shall demonstrate the query constructs on a generic mini-University data set example involving students, courses and academic programs (cf. e.g. [8] for its brief description). The query in Fig. 1 specifies selection of all names of students together with the names of their taken study courses from this data set. In the ViziQuer notation one of the classes has to be marked as the main query class (specified as orange round rectangle), the others are condition classes (light violet rectangles); the choice of the main query class shall become important in further query patterns.

[2] There are options of loading the data schema from an OWL ontology and from a SPARQL endpoint (actual data schema).

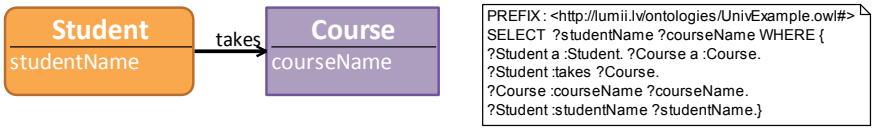


Fig.1. A simple query and its translation into SPARQL

Fig. 2 illustrates the explicit instance reference names introduced and added to the query output (the instance URI is returned). The attribute conditions (marked as purple texts) and the alias option for selection items are illustrated, as well. The instance references and aliases are used for variable name generation, they can also be referred to from other query parts. If a class instance reference name is not explicitly specified, the class name can be used instead of it, however, the implicit instances of the same class appearing in different parts of the query are considered to be different.

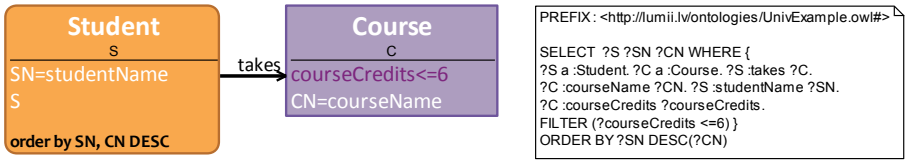


Fig.2. A query with instance references, conditions, aliases and ordering

2.1 Optional and Negation Link Patterns

There can be affirmative (black solid line), optional (blue/light dashed line) or negation (red line with stereotype {not}) links between classes within the query. The presence of optional or negation links in the query require it to have a tree-shaped structure (this shall be relaxed in Section 2.2). The interpretation of optional or negation link is to mark the entire subgraph placed behind the link (from the viewpoint of the main query class) as optional or negated respectively.

Fig. 3 illustrates the optional and negation links among the classes, as well as the optional stereotype for the attributes (we consider the *Nationality* and *Registration* (a student registers for a course) classes, along with the *Student* class for the illustration).

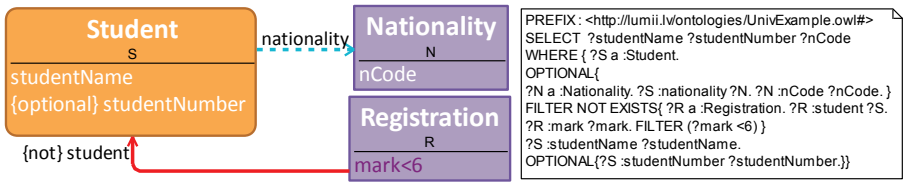


Fig. 3. Optional and negation links, optional attributes

2.2 Condition Link Pattern

The condition links (marked by the {condition}-stereotype) are meant to extend the tree-shaped query structure introduced in Section 2.1. The interpretation of a condition

link is to add a triple connecting the link end nodes to the query pattern in the case of affirmative link and add a respective triple non-existence filter in the case of negation link (notice the difference from marking the entire query part behind the link as negated in the case of non-condition negation links).

Fig. 4 illustrates two queries with condition pattern: one with affirmative links and the other involving negations. Notice that the second query with "double negation" expresses universal quantification: finding names for all students taking all courses included in the academic program they are enrolled in.

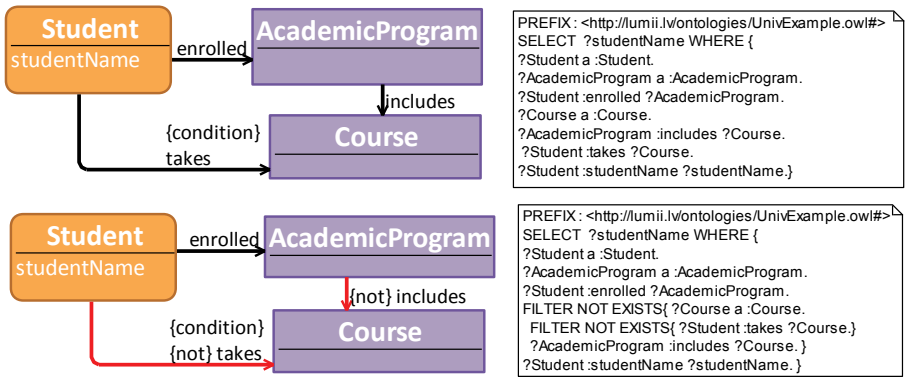


Fig. 4. Condition links: beyond the tree-shaped structure

2.3 Meta-information Query Patterns

The meta-information queries can be obtained by placing explicit variables in the class name and/or link name positions within the query, as illustrated in Fig. 5.

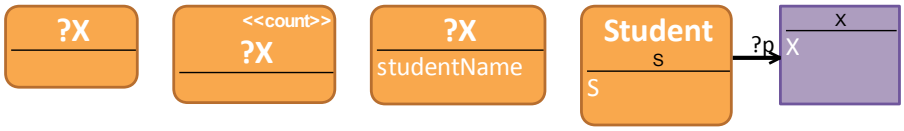


Fig. 5. (i) find all classes, (ii) find all classes together with their instance count, (iii) list class names and *studentName* property values of all instances with this property, (iv) select all student class instances with all object and data properties and their values.

3 Aggregate Query Patterns

The simplest aggregate query pattern is a count of class instances. It can be specified either using a class stereotype <<count>>, or by creating an attribute expression with the count function applied to the class instance reference.

Simple extensions of the basic count pattern allow counting main class instances satisfying conditions specified in either the main class itself, or in a condition class (cf. Fig. 7). In the case of a condition class present in a counting (or other pure aggregation)

query, its semantics by default is just asserting the existence of a respective linked instance with the specified properties. The alternative semantics of computing the aggregation over the query patterns involving possibly multiple condition class instances for a single main class instance (so, a single main class instance could be observed several times in the query) can be achieved by the <<all>> stereotype placed on the condition class.

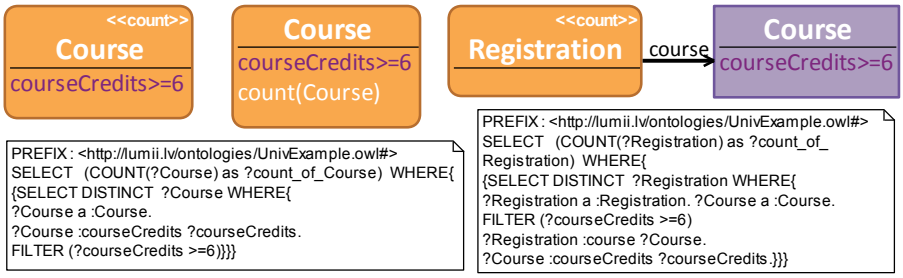


Fig. 6. Two count notation options in single-class query, and count in a query with condition.

3.1 Simple Statistics Patterns

The aggregation options can be included into the queries just by introducing into class instance attribute lists aggregate expressions where an SPARQL aggregate function (e.g. *count*, *sum*, *avg*) is applied to a non-aggregated (i.e. plain) attribute expression, for instance, as in *sum(mark)* in Fig. 7.

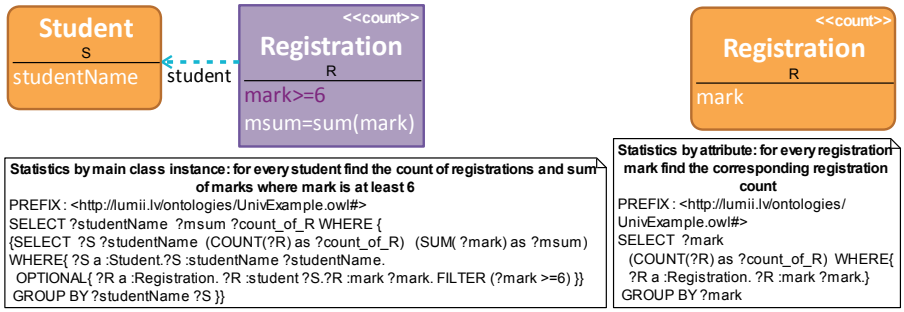


Fig. 7. The statistics by class instance and statistics by attributes patterns

The statistical queries are obtained by including both the scalar (i.e. non-aggregated) and aggregated expressions that are obtained by within the query result set. The aggregate expressions are evaluated by default against the grouping set that includes the main class instance and all non-aggregated attributes included in the query; the main class instance can be excluded from the grouping set by setting a main class stereotype (e.g. <<count>>) in the query. Two important subclasses of simple statistics patterns are statistics by attributes where the <<count>> stereotype is attached to the main query class and statistics by main class instance where a separate statistics row(s) is (are) computed for each main class instance. Fig. 7 illustrates both these patterns.

3.2 Filters over Aggregate Results

The attribute conditions specified in the class nodes are to be evaluated before the aggregation computation and they limit the scope of the aggregation. The filters that compute conditions on aggregate results can be placed in a *having*-compartment within the main query class.

3.3 Existential and Universal Stereotypes

Figure 8 shows two semantically different ways of counting the students receiving any specific mark. They involve either counting each student for each mark at most once, or counting the student as many times, as there are registrations by this student with the specified mark. In general, for any class in the query the <<exists>> and <<all>> stereotypes can be specified to mark, whether only existence of a class instance is to be observed during the aggregation computation, or all instances are to be counted separately; the default stereotype for a condition class is <<exists>>; notice that <<all>> stereotypes for multiple classes within a query can create multiplicative blow-up of aggregation scope, and, therefore, also of aggregate numbers in the query results.

The main class, if not stereotyped, causes creation of separate aggregation scopes for each instance; this corresponds to the effect of the stereotype <<find>>. In general, the <<exists>>, <<all>> and <<find>> stereotypes can be placed on any class, including the query and the condition classes, to change their grouping behavior within the aggregate queries.

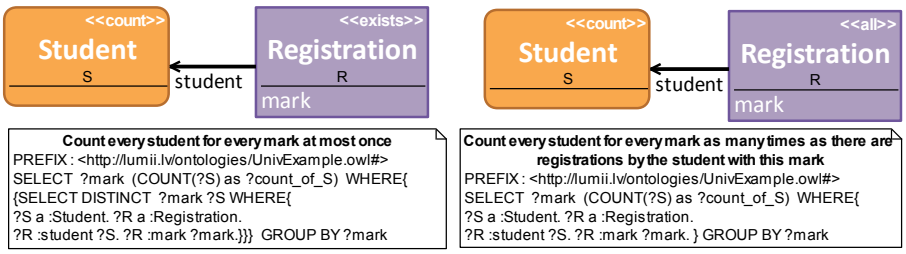


Fig. 8. Existential and universal stereotypes

3.4 Explicit Subquery Pattern

The default aggregate computation rule of using a single grouping set for all computing all aggregate functions within the query is not sufficient, for instance, in the cases of nested aggregation. Therefore, an explicit {subquery}-stereotype is introduced for attributes and links that turns the query fragment within the subquery scope a subquery related to the subquery parent class instances (the subquery parent class is the class containing the {subquery}-attribute, or the class at the end of the {subquery}-link on its main class side). A typical subquery attribute would be a group-concatenation of multiple same named data properties of a class instance. A subquery link example is in Fig. 9, where for every student class instance the minimal registration mark is found

and then all students having the minimal mark at least 7 are counted. Note that since the minimal mark is computed in a subquery, it can be used within the condition (and not the *having*) compartment of the main query class.

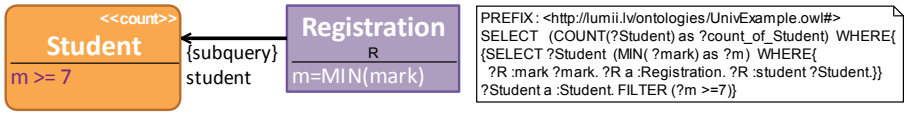


Fig. 9. Explicit subquery pattern example

4 Expression Notation and Patterns

The basic expression pattern in ViziQuer is that of a class attribute specification either for the query selection list, or within a selection or filtering expression. In either case the attribute specification corresponds to:

(i) creating a variable name that is derived from the attribute name (typically, by prefixing the local variable name by '?'; additional decorations can be added to make the variable names unique within the query) and

(ii) linking the class instance variable by the property corresponding to the attribute to the created attribute variable.

If an expression, say `a+b`, is specified in the selection list for the class whose instance variable in the SPARQL translation is `?p`, its translation shall involve `?p :a ?a. ?p :b ?b. BIND(?a+?b AS ?expr_1)` in the SPARQL query pattern part and `?expr_1` in the query selection list; if there were an expression alias specified, as in `c=a+b`, it would be used as the variable name both in the BIND-clause: `BIND(?a+?b AS ?c)` and in the select list.

The general rule for selection and filtering expression forming in ViziQuer is to allow expressions following the SPARQL expression syntax, with the modification expecting a (possibly qualified) attribute name in the place of a variable name within the original SPARQL notation.

The attribute names in the ViziQuer expressions may be qualified by class instance reference names present in the query, or by property path expressions; in either case the qualifications shall use the UML style separator '.' (cf. Fig. 10).

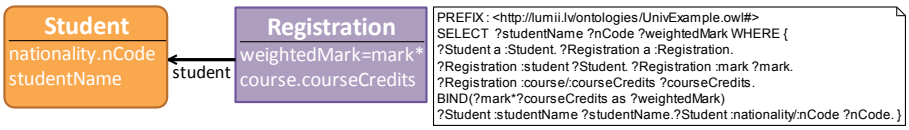


Fig. 10. For all students show the nationality codes (path expression) and weighted marks in all registrations (path expression within arithmetic expression)

4.1 Negated Condition Pattern

Asserting the class instance, say p, attribute value to satisfy a (non-negated) predicate, e.g. mark>=7, implies the existence of the attribute value AND that it satisfies the

predicate: ?p :mark ?mark. FILTER (?mark>=7). So, a negation of the assertion means that either the attribute value dos not exist OR that it exists but not satisfy the predicate, cf. Fig. 11; it should be distinguished from requiring the negation of the condition present in the query. This pattern explains specifics of handling null values in conditions, often important in practical query situations.

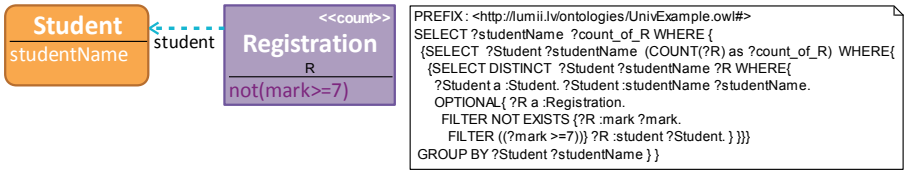


Fig. 11. For all students count the number of registrations not having marks 7 or above

4.2 Specific Expression Patterns

A common specific expression pattern often required in statistic data analysis, however, not supported in standard SPARQL 1.1, is a date value difference. We provide a date value difference functions in ViziQuer (cf. Fig. 12) for accessing vendor-specific SPARQL endpoints; our current implementation is targeted towards OpenLink Virtuoso [13], other target environments can be added based upon the construct availability in the SPARQL endpoint.

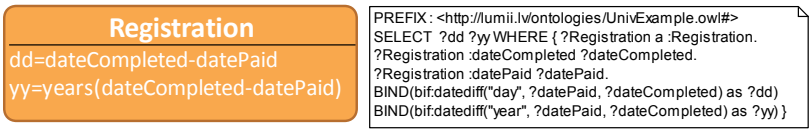


Fig. 12. Date value difference, expressed in days (the default) and in years

4.3 Multiple Statistics Pattern

The expression language incorporated in ViziQuer allows for joining multiple statistical inquiries into single query. Figure 14 shows the way, how to compute for each student simultaneously the count of all taken courses, as well as the counts of taken big (courseCredits>=0) and small (courseCredits<0) courses. We notice that the expressions involved in the query resemble ones that can be used for simple statistical data processing in Microsoft Excel. Should the further practical query tool usage experiments confirm the initially observed importance of this query pattern, a special notation might be designed to ease the query formulation options in accordance to it.

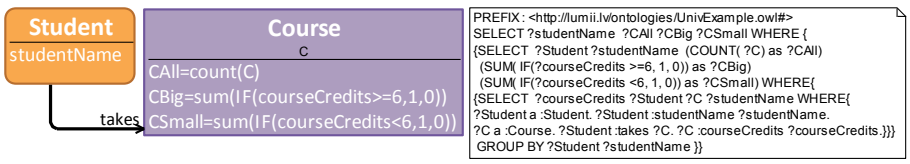


Fig. 13. Counting taken courses from different course sets

5 Discussion and Conclusions

The explained patterns for UML style visual data analysis query definition are implemented in the ViziQuer tool that is freely available at http://viziquer.lumii.lv.

The introduced notation and patterns can be seen also as an attempt to push forward the UML style diagrammatic SPARQL query definition in general with the aim of covering data analysis queries that are currently in practical situations handled by business intelligence suites with data residing in relational databases. The presented notation can be criticized, updated, extended and offered alternative implementations.

The initial practical experience with defining queries for Latvian Medicine Registries example [11] show that the notation can be near to sufficient for the end user statistical needs; the practical application of the notation as well as its further fine tuning shall be continued. At the same time the initial usage of the notation with simplest basic and aggregate query patterns can be kept low-entry.

The future work plans include re-implementing the tool within the web environment, as well as adding result visualization component to it.

References

1. SPARQL 1.1 Overview. W3C Recommendation 21 March 2013, http://www.w3.org/TR/sparql11-overview/
2. Resource Description Framework (RDF), http://www.w3.org/RDF/
3. R2RML: RDB to RDF Mapping Language, http://www.w3.org/TR/r2rml/
4. D2RQ. Accessing Relational Databases as Virtual RDF Graphs, http://d2rq.org/
5. Optique. Scalable End-User Access to Big Data, http://optique-project.eu
6. Calvanese, D., Cogrel, B., Komla-Ebri, S., Lanti, D., Rezk, M., Xiao, G.: How to Stay Ontop of Your Data. In: Databases, Ontologies and More. ESWC (Satellite Events) 20-25, (2015)
7. Zviedris, M., Barzdins, G.: ViziQuer: A Tool to Explore and Query SPARQL Endpoints. In: The Semantic Web: Research and Applications, LNCS, Volume 6644/2011, pp. 441-445, (2011)
8. Cerans, K., Ovcinnikova, J., Zviedris, M.: SPARQL Aggregate Queries Made Easy with Diagrammatic Query Language ViziQuer. In: Proceedings of the ISWC 2015 PD, CEUR Vol. 1486, (2015), http://ceur-ws.org/Vol-1486/paper_68.pdf
9. Soylu, A., Giese, M., Jimenez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I.: OptiqueVQS: Towards an Ontology based Visual Query System for Big Data. In: MEDES. (2013)
10. Haag, F., Lohmann, S., Siek, S., Ertl, T.: QueryVOWL: Visual Composition of SPARQL Queries. In: The Semantic Web: ESWC 2015 Satellite Events. LNCS, Vol.9341, pp. 62-66. Springer, (2015), http://vowl.visualdataweb.org/queryvowl/
11. Barzdins, G., Liepins, E., Veilande M., Zviedris M.: Semantic Latvia Approach in the Medical Domain. In: Proc. of 8th International Baltic Conference on Databases and Information Systems. Haav, H.M., Kalja, A. (eds.), TUT Press, pp. 89-102. (2008)
12. Barzdins, J., Cerans, K., Liepins, R., Sprogis, A.: UML Style Graphical Notation and Editor for OWL 2. In: Proc. of BIR'2010, LNBIP, Springer, vol. 64, pp. 102-113, (2010)
13. Blakeley, C.: RDF Views of SQL Data (Declarative SQL Schema to RDF Mapping), OpenLink Software, (2007)