

Bo Fu
Patrick Lambrix
Catia Pesquita (Eds.)

VOILA! 2022

Proceedings of the 7th International Workshop on

Visualization and Interaction for Ontologies and Linked Data

Co-located with ISWC 2022, Virtual, October 23, 2022.

Title: Visualization and Interaction for Ontologies and Linked Data (VOILA! 2022)

Editors: Bo Fu, Patrick Lambrix, Catia Pesquita

ISSN: 1613-0073

CEUR Workshop Proceedings
(CEUR-WS.org)

Copyright © 2022 for the individual papers by the papers' authors. Copyright © 2022 for the volume as a collection by its editors. This volume and its papers are published under the Creative Commons License Attribution 4.0 International (CC BY 4.0).

Organizing Committee

Bo Fu, California State University Long Beach, USA

Patrick Lambrix, Linköping University and University of Gävle, Sweden

Catia Pesquita, LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

Program Committee

Kārlis Čerāns, University of Latvia, Latvia

Christophe Debruyne, University of Liège, Belgium

Anastasia Dimou, KU Leuven, Belgium

Roberto García, Universitat de Lleida, Spain

Anika Groß, University of Leipzig, Germany

Mayank Kejriwal, University of Southern California, USA

Huanyu Li, Linköping University, Sweden

Albert Navarro Gallinad, Trinity College Dublin, Ireland

Alexander O'Connor, Autodesk, USA

Declan O'Sullivan, Trinity College Dublin, Ireland

Evan Patton, Massachusetts Institute of Technology, USA

Emmanuel Pietriga, INRIA, France

Harald Sack, FIZ Karlsruhe, Leibniz Institute for Information Infrastructure
& KIT Karlsruhe, Germany

Cogan Shimizu, Kansas State University, USA

Ahmet Soylu, OsloMet – Oslo Metropolitan University, Norway

Markel Vigo, The University of Manchester, UK

Preface

The Semantic Web enables intelligent agents to create knowledge by interpreting, integrating and drawing inferences from the abundance of data at their disposal. It encompasses approaches and techniques for expressing and processing data in machine-readable formats. All these tasks demand a human-in-the-loop; without them, the great vision of the Semantic Web would hardly be achieved. Meanwhile, visual interfaces for modeling, editing, exploring, integrating, etc., of semantic content have not received much attention yet.

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. The evaluation of such interfaces and techniques poses another relevant concern given the specific challenges of visualizing data imbued with semantic complexity. 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 seventh edition of VOILA! was co-located with the 21th International Semantic Web Conference (ISWC 2022) and took place as a half-day virtual event on October 23, 2022. It was organized around scientific paper presentations and discussions.

The call for papers for VOILA! 2022 attracted 8 submissions in different paper categories. At

least three reviewers were assigned to each submission. Based on the reviews, we selected 6 contributions for presentation at the workshop.

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 Marta Sabou and Raghava Mutharaju, the workshop chairs of ISWC 2022, for their continuous support during the workshop organization.

October 2022

Bo Fu,
Patrick Lambrix,
Catia Pesquita

Contents

Regular papers	1
Shacled Turtle: Schema-based Turtle Auto-Completion <i>by Julian Bruyat, Pierre-Antoine Champin, Lionel Médini, Frédérique Laforest</i>	2
Enabling Exploratory Search on Manufacturing Knowledge Graphs <i>by Kevin Haller, Fajar J. Ekaputra, Marta Sabou, Florina Piroi</i>	16
VOWLExpain: Knowledge Graph visualization for Explainable Artificial Intelligence <i>by Filipa Serrano, Susana Nunes, Catia Pesquita</i>	29
Toward a Methodology Comparison Framework for Interactive Ontology Enrichment <i>by Jarno Vrolijk, Ioannis Reklos, Mahsa Vafaie, Arcangelo Massari, Maryam Mohammadi, Sebastian Rudolph</i>	41
Short paper	51
Multi-level Visual Tours of Weather Linked Data <i>by Nadia Yacoubi, Damien Graux, Catherine Faron</i>	52
Demonstration paper	58
Experience with Visual SPARQL Queries over DBPedia <i>by Kārlis Čerāns, Jūlija Ovcinnikova, Mikus Grasmanis, Lelde Lace</i>	59

. Regular papers

Shacled Turtle: Schema-Based Turtle Auto-Completion

Julian Bruyat¹, Pierre-Antoine Champin^{1,2,3}, Lionel Médini¹ and Frédérique Laforest¹

¹Univ Lyon, INSA Lyon, CNRS, UCBL, LIRIS, UMR5205, F-69621 Villeurbanne, France

²W3C, Sophia Antipolis, France

³University Côte d'Azur, Inria, CNRS, I3S (UMR 7271), France

Abstract

Producing RDF documents has always been a tedious task. To make it easier, most approaches propose an abstraction like forms to produce the data. In this paper, we propose Shacled Turtle, a method and a tool to ease the editing of RDF documents with auto-completion, based on RDFS and/or SHACL schemas. We also describe an experiment with volunteers to evaluate the usefulness of the tool, and we discuss its results.

Keywords

RDF, Auto-completion, SHACL

1. Introduction

Producing RDF data is a widely studied problem. In real life, most RDF datasets are either generated by tools to convert other formats to RDF or from data filled in a form by the user. However, the lack of good RDF editors is identified by the EasierRDF group as one of the many issues that can refrain developers from using RDF¹. Indeed, many usages require to write relatively small pieces of RDF by hand: teaching the basics of RDF, describing ontologies, building graph patterns in SPARQL queries, declaring R2RML mappings [1]... Meta-ontologies, such as RDF Schema [2] (RDFS) or the Web Ontology Language [3] (OWL), have always played a central role in the RDF ecosystem. Recently, the necessity to validate RDF data appeared, giving birth to SHACL [4] and ShEx² to check the validity of graphs. But to the best of our knowledge, the literature has barely explored the ability to use inferential schemas or validating schemas to help users explicitly write RDF graphs.


Our research hypothesis in this work is that schemas can be used to provide useful suggestions to users when writing an RDF document. We developed Shacled Turtle³ [5], a tool that uses RDFS and SHACL schemas to provide auto-completion for writing Turtle documents [6]. The

VOILA! 2022: 7th International Workshop on Visualization and Interaction for Ontologies and Linked Data, October 23, 2022, Virtual Workshop, Hangzhou, China, Co-located with ISWC 2022.

✉ julian.bruyat@liris.cnrs.fr (J. Bruyat); pierre-antoine@w3.org (P. Champin); lionel.medini@liris.cnrs.fr (L. Médini); frederique.laforest@liris.cnrs.fr (F. Laforest)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

¹<https://github.com/w3c/EasierRDF/issues/35>

²<https://shex.io>

³<https://github.com/BruJu/shacled-turtle>

purpose of this work is not the validation of documents, for which tools already exist, but only the assistance to the users in writing them.

The rest of this paper is structured as follows. In Section 2, we review the different available tools to produce RDF data. In Section 3 we present Shacled Turtle through a concrete example to give an intuition about the tool. In Section 4 we describe the architecture used to provide suggestions based on the chosen schema graph that is then detailed in Sections 5 and 6. In Section 7 we evaluate Shacled Turtle style of suggestions against the kind of suggestions provided by other similar tools from a user perspective. In Section 8 we discuss our contribution, the results and the experimental protocol. Finally, in Section 9 we conclude with the possible improvements.

2. Where do RDF Triples come from?

To produce RDF data, most approaches use some kind of abstraction, convert existing data or directly write programs that output RDF data.

2.1. RDF data production

The most popular abstractions are forms. Protégé [7] requires the user to fill forms to build their ontology and then generate the corresponding RDF file. The SHACL specification explicitly mentions the possibility to generate forms from property shapes, which has been implemented by systems like Schimatos [8]. Form generators have also been developed for the other shape language ShEx.

Converting non-RDF data is also a popular way to produce RDF data. Many tools have been developed: R2RML maps relational databases and tabular data to RDF by using mappings provided by the user, RML [9] extends the latter to support other kinds of data sources, RDF123 [10] aims to produce RDF data by using spreadsheets as an abstraction, JSON-LD [11] transforms JSON documents to RDF and is the way recommended by Google to add metadata to a website in order to improve its SEO (Search Engine Optimisation).

Even by using these approaches, users may still have to write RDF documents: the R2RML mappings must be described in RDF, users may want to fine tune the ontology produced by Protégé.

2.2. Current editors

Plugins for popular code editors have been developed, like *LNKD. tech Editor*⁴ and *RDF and SPARQL*⁵ for the JetBrains suite. A language server for Turtle has recently been developed by Stardog Union⁶. But all these plugins mainly focus on syntactic checking and coloration.

In [12], Rafes et al. list some of the expected features from a SPARQL auto-completion module. They identify 3 major categories: suggestion of snippets, prefix declaration and auto completion for Internationalized Resource Identifiers (IRIs). Snippets suggestion is described as being mostly

⁴<https://plugins.jetbrains.com/plugin/12802-lnkd-tech-editor>

⁵<https://sharedvocabs.com/products/rdfandsparql/>

⁶<https://marketplace.visualstudio.com/items?itemName=stardog-union.vscode-langserver-turtle>

requested by experienced users “and can be seen as the step after suggesting terms”. Prefix auto-completion is deployed by most editors, through the use of the prefix.cc API⁷.

To the best of our knowledge, IRI suggestions in all RDF document editors, like *RDF and SPARQL* and *Yasgui* [13] are limited to proposing all the terms that exists in a given ontology. *Yasgui* filters the list of suggestions depending on the position: for example if the current term is a predicate, all properties in the ontology are displayed and other terms are discarded. This approach is best suited for small ontologies, as for big ontologies, like schema.org⁸, the number of suggestions can reach hundreds, making it impractical for users.

Some SPARQL editors like the Flint Sparql Editor⁹ or the one presented by Gombos and Kiss in [14] uses intermediate SPARQL queries to help users write their queries. Sparqlis [15] also uses this approach but goes further by exposing an interface in natural language, removing the need for the end-user to know SPARQL. In [16], De la Parra and Hogan first compute the relationships between all types and predicates in the graph, and use the result of this computation to provide auto-completion when the user builds their SPARQL query. All these approaches resort on using the actual data to produce the effective schema of the graph. But in our case, as we are interested in writing new data, these kinds of approaches are not applicable. Hence instead of using the effective schema of the graph we will rely on the expected schema as specified by an RDFS ontology or a set of SHACL shapes.

3. Shacled Turtle usage example

Shacled Turtle is implemented as a Code Mirror 6¹⁰ extension that provides support for the Turtle language.

The selling key-point of Shacled Turtle is the auto-completion module. While most advanced editors suggest all terms from the ontology, Shacled Turtle narrows the list to the parts of the ontology that are related to the currently edited resource. We consider that most resources in an RDF graph must be typed. When the type of a resource is known, it is likely that the predicates related to the known types will be used to describe it. Figure 1 shows a concrete usage example: we defined that `ex:alice` is a `s:Person` and then we start to write a new triple. The suggestion engine considers that we probably want to use a predicate related to persons, like `s:nationality` or `s:name`, and does not suggest terms like `s:numTracks` or `s:measurementTechnique` that are related to other types.

4. Shacled Turtle general architecture

Figure 2 shows the general architecture of Shacled Turtle.

0. Before the interaction starts, a preprocessing phase is performed. The content of a *schema graph* is converted into *inference rules* and *suggestion rules* by the *schema to rules converter*. This schema can be written either in RDFS, in SHACL or a mix of both.

⁷<https://prefix.cc>

⁸<https://www.schema.org>

⁹<http://fr.dbpedia.org/sparqlEditor>

¹⁰<https://codemirror.net/6/>

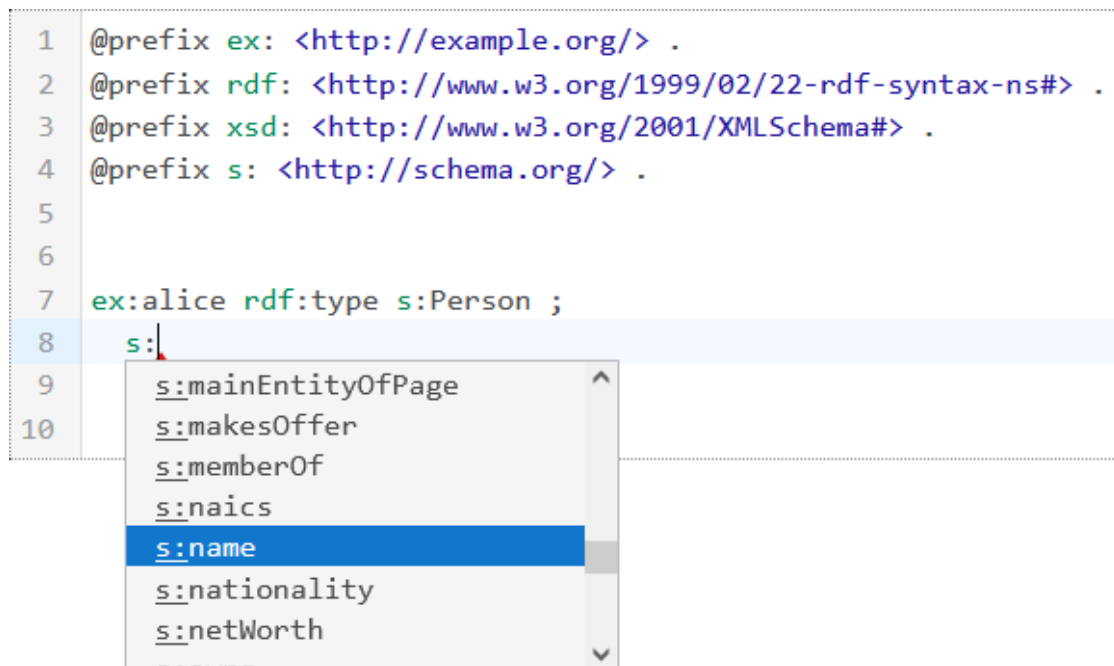


Figure 1: For a subject of type Person, Shacled Turtle only suggests predicates related to this type in the Schema ontology

1. When the user is writing an RDF graph, the *inference engine* uses all *complete triples* to deduce the types of all resources and the list of shapes that they should comply with. These results are stored in the *meta graph*.
2. When the user is writing a new *incomplete triple*, after the subject has been written, *i.e.* on writing the predicate or on writing the object, the *suggestion engine* queries the *meta graph* for the list of all the types and shapes of the subject. It will then return to the user:
 - If the incomplete triple only has a subject, the list of all predicates related to the types and shapes of the subject.
 - If the incomplete triple has a subject and a predicate, a list of resources depending on the types and shapes of the subject and the predicate¹¹.

We will first describe the basics of all the components used by the *interaction loop* in Section 5, in particular describe how the rule systems used by the *inference engine* and the *suggestion engine* work. Then we will describe how the *preprocessing* translates the schema graph into *inference and suggestions rules* in Section 6.

5. The interaction loop

The **interaction loop** comprises all operations performed when the user uses the editor.

¹¹Note that for some predicates like `rdf:type`, the suggestion may be independent of the list of types and shapes of the subject.

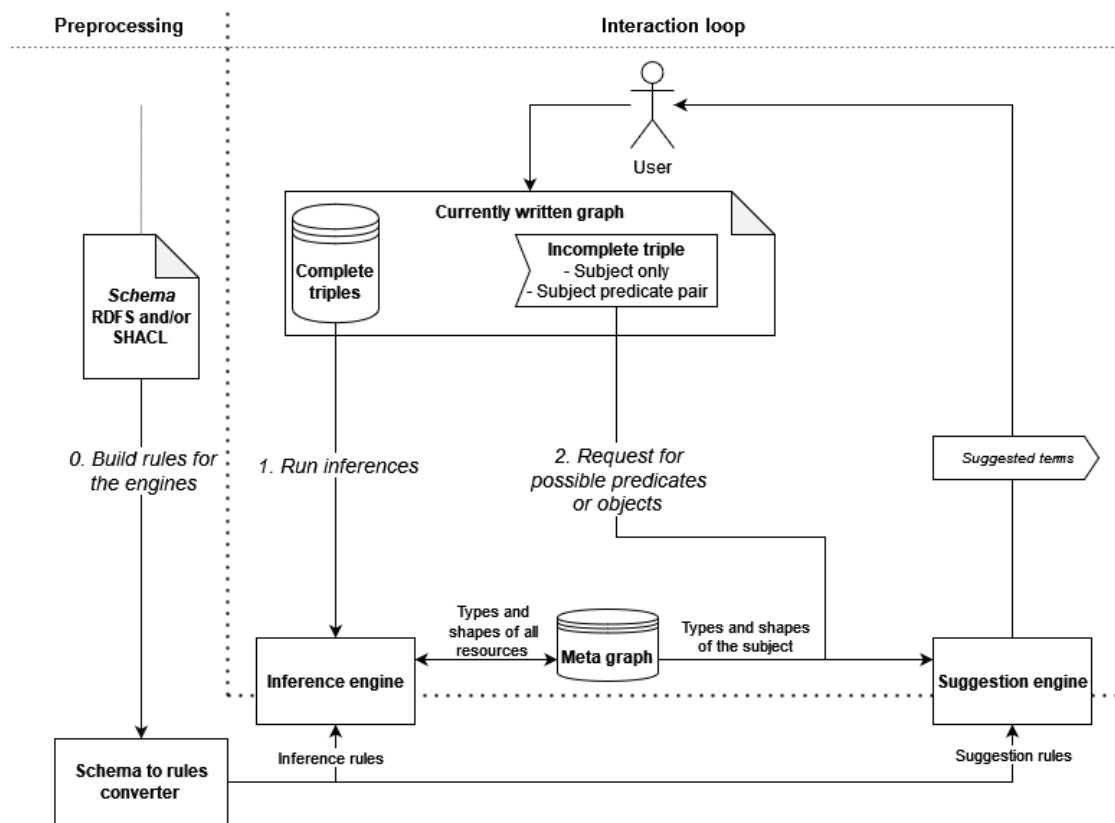


Figure 2: The different components of Shacled Turtle

5.1. The graphs

During the interaction loops, two different graphs are used:

- The *currently written graph* is the graph in the text editor. It is composed of two different parts:
 - The *completed triples*, triples for which the subject, the predicate and the object are known. These triples are used to power up the *inference engine* and produce triples for the *meta graph*.
 - The *incomplete triple* that the user is currently editing. If the subject of this incomplete triple is known, the *suggestion engine* and the *meta graph* will be requested to provide a list of *suggested terms*. If other triples are incomplete, they are ignored by the engine.
- The *meta graph* is the graph that stores triples produced by the *inference engine*. Its role is to store, for each resource, the list of all known types, and the list of shapes that the resource should comply with. The content will then be used by the *suggestion engine*, in particular to know the list of types and shapes of the subject of the *incomplete triple*.

5.2. The inference engine

We want our system to be able to deduce both the deducible types from an RDFS ontology and to be able to list the shapes a resource must comply with.

These inferences are specified by *inference rules* (see Table 1 and 3). These rules go beyond the ones traditionally used for RDFS, but do not need to capture the full semantics of SHACL as we are not aiming at validating the graph.

Each inference rule has the form:

$$\frac{DataTriple? \quad SourceMetaTriple?}{ProducedMetaTriple}$$

where:

- The body can require
 - *DataTriple?*: 0 or 1 complete triple from the *currently written graph*.
 - *SourceMetaTriple?*: 0 or 1 triple from the *meta graph*.
- The head must be a triple *ProducedMetaTriple* that will be stored in the *meta graph* and its predicate must either be `rdf:type` or `:pathsOf`¹².

5.3. The suggestion engine

In the same way, the system relies on a set of rules to deduce the possible suggestions at run-time, from the meta graph and the incomplete triple.

We suppose that *?s*, *?p*, *?o* are variables and *A* and *P* are IRIs.

Each suggestion rule has the form:

$$\frac{IncompleteTriple \quad MetaTripleCondition?}{Suggestion}$$

where:

- The *IncompleteTriple* is either
 - (*?s*, ..., ...) for applying the rule when only the subject of the *incomplete triple* is known.
 - (*?s*, *A*, ...) for applying the rule when both the subject and predicate are known.
- The *MetaTripleCondition?* is optional, and either
 - A triple pattern of the form (*?s*, *P*, *?o*) that is searched in the *meta graph*, where *?s* is the subject occurring in *IncompleteTriple*.
 - “No info on *?s*”, for applying the rule only if there are no types or shapes known for the resource *?s*.
 - *none* when no condition holds on the meta graph content.
- The *Suggestion* is either
 - *suggest(A)* to add *A* to the list of suggested terms.
 - *suggestAll(?p, ?o)* to add to the list of suggested terms all resources *α* such that (*α*, *?p*, *?o*) is in the *meta graph*.

¹²The triple (*u*, `:pathsOf`, *s*) means that for the resource *u* we should suggest the paths specified by the shape *s*.

Table 1

Transformation of triples in the schema graph into inference and suggestion rules.

<i>Triple in schema graph</i>	<i>Inference rules</i>	<i>Suggestion rules</i>
	$\frac{(?u, rdf:type, ?t) \quad none}{(?u, rdf:type, ?t)}$ $\frac{none \quad (?u, rdf:type, ?t)}{(?t, rdf:type, rdfs:Class)}$	$\frac{(?u, \dots, \dots) \quad \text{No info on } ?u}{suggest(rdf:type)}$ $\frac{(?u, rdf:type, \dots) \quad none}{suggestAll(rdf:type, rdfs:Class)}$
$(P', rdfs:domain, T)$ $\forall P = P'$ or P' subproperty of P	$\frac{(?u, P, ?v) \quad none}{(?u, rdf:type, T)}$	$\frac{(?u, \dots, \dots) \quad \text{No info on } ?u}{suggest(P)}$ $\frac{(?u, \dots, \dots) \quad (?u, rdf:type, T)}{suggest(P)}$
$(P', rdfs:range, T)$ $\forall P = P'$ or P' subproperty of P	$\frac{(?u, P, ?v) \quad none}{(?v, rdf:type, T)}$	$\frac{(?u, P, \dots) \quad none}{suggestAll(rdf:type, T)}$
$(P, s:domainIncludes, T)$		$\frac{(?u, \dots, \dots) \quad (?u, rdf:type, T)}{suggest(P)}$
$(P, s:rangeIncludes, T)$		$\frac{(?u, P, \dots) \quad none}{suggestAll(rdf:type, T)}$
$(S, rdf:type, sh:NodeShape)$	$\frac{none \quad (?u, rdf:type, S)}{(?u, :pathsOf, S)}$	
$(S, sh:targetNode, U)$	$\frac{none \quad none}{(U, :pathsOf, S)}$	
$(S, sh:targetClass, T)$	$\frac{none \quad (?u, rdf:type, T)}{(?u, :pathsOf, S)}$	$\frac{(?u, rdf:type, \dots) \quad none}{suggest(T)}$
$(S, sh:subjectsOf, P)$	$\frac{(?u, P, ?v) \quad none}{(?u, :pathsOf, S)}$	$\frac{(?u, \dots, \dots) \quad \text{No info on } ?u}{suggest(P)}$ $\frac{(?u, \dots, \dots) \quad (?u, :pathsOf, S)}{suggest(P)}$ $\frac{(?u, \dots, \dots) \quad none}{suggest(P)}$
$(S, sh:objectsOf, P)$	$\frac{(?u, P, ?v) \quad none}{(?v, :pathsOf, S)}$	$\frac{(?u, P, \dots) \quad none}{suggestAll(:pathsOf, S)}$
$(S1, sh:node, S2)$ and $S1$ is a node shape	$\frac{none \quad (?u, :pathsOf, S1)}{(?u, :pathsOf, S2)}$	

6. The preprocessing

We now describe the *preprocessing*, which is the step where the *schema to rules converter* converts the *schema* graph into *inference* and *suggestion rules* for the eponymous engines. It uses two kinds of transformations: rules that are built by searching all triples with a certain pattern in

the schema graph, and SHACL paths whose recursive nature will be handled by using finite state automata.

6.1. Rules built by looking up some triples pattern

Table 1 exposes the list of inference and suggestion rules that are generated from the schema graph. Note that the purpose of this tool is neither to infer all possible suggestions, nor to validate the graph, but to make suggestions that are as relevant as possible. This is a subjective criterion, as having either too few or too many suggestions would make the tool less useful. We will discuss this further in Section 8.

6.2. Rules built from SHACL Paths

Similarly to SPARQL paths, a SHACL path can be either a predicate path (an out-coming triple with a given predicate) or a composition of other paths with one of the following operators: inverse, sequence, alternative, repetition, kleene, and optional.

One issue with paths is that we want to be able to process complex paths, and provide suggestions at any point in the path. For example, for the sequence path (:a :b), :b should be a suggested predicate for nodes targeted by :a.

Unit paths and virtual shapes Our solution is to split composite paths into what we consider unit paths. Unit paths are either predicate paths, e.g. :owns, or inverse paths of a predicate path, e.g. [sh:inversePath :ownedBy]. These unit paths are connected with *virtual shapes*, shapes that do not explicitly exist in the composite shape graph. The chain of all the unit paths through the virtual shapes is equivalent to the original composite path for the purpose of our suggestion engine.

Let us consider the shape graph on Listing 1. This shape graph means any node ?postalAddress extracted from the SPARQL request on Listing 2 must comply with the node shape s:PostalAddress. For our purpose, it is equivalent to the shape graph on Listing 3 where we introduced a new shape, ex:VirtualShape that will act as the shape of all the matches for ?o in the SPARQL request.

Overview on transforming SHACL Paths into rules. To process SHACL paths, we assume that:

- We can decompose any path into unit paths connecting virtual shapes.
- Processing a chain of unit predicate paths is similar to processing a string with a regex. Hence we can use finite-state automaton (FSA) to recognize if a chain of triples is recognized by a path.
- The only difference between a predicate path and an inverse predicate path is whenever the subject or the object variable is bound to a known value.

Based on these assumptions, to parse the SHACL path into a list of inference and suggestion rules, we first transform the path into an FSA, then we transform the FSA into rules.

Listing 1: An example shape

```
s:Person rdf:type sh:NodeShape ;
  sh:targetClass s:Person ;
  sh:property [
    sh:path (
      s:worksFor
      s:address
    ) ;
    sh:node s:PostalAddress
  ] .
```

Listing 2: SPARQL query to get all the resources targeted by the property shape contained by s:Person

```
SELECT ?postalAddress WHERE {
  # All resources targeted by the shape
  ?person rdf:type s:Person .
  # Travel the path
  ?person s:worksFor ?o .
  ?o s:address ?postalAddress .
}
```

Listing 3: The same shape graph with a virtual shape

```
s:Person rdf:type sh:NodeShape ;
  sh:targetClass s:Person ;
  sh:property [
    sh:path s:worksFor ;
    sh:node ex:VirtualShape01
  ] .

ex:VirtualShape01
  rdf:type sh:NodeShape ;
  sh:property [
    sh:path s:address ;
    sh:node ex:PostalAddress
  ] .
```

From SHACL paths to FSA. We build the FSA that describes the path P by composition. Predicate paths produce an FSA with two states and only one transition. The FSA of other paths, that are composite paths, are built by combining the automaton of their components in some way. The transition symbol used by all the produced automaton are composed by combining either the sign $+$ for out-going edges or $-$ for incoming edges, with the predicate to travel. Table 2 describes all the composition rules, where we consider that p is any predicate, P , $P1$ and $P2$ are paths.

From FSA to rules. After minimization and determination, an FSA can be defined as one initial state, a set of final states and a set of transitions (*StartState*, *Symbol*, *EndState*).

- We define m a total function from all states s of the FSA to RDF Nodes. For each state s , $m(s)$ is a fresh RDF node, *i.e.* it is not used elsewhere.
- The virtual shape mapped from the initial state of the FSA is a super-shape of the starting shape of the property shape
- If a destination shape is known for the property shape, it is declared as a sub-shape of all the final states of the FSA.
- Table 3 describes how to convert the transitions to inferences rules.

7. Evaluation

Shacled Turtle uses schemas to reduce the number of suggestions proposed to users, keeping only the most relevant ones. The underlying assumption is that this is more helpful for users

Table 2

Mapping from all kinds of path to automata

<i>Kind and SHACL Syntax</i>	<i>Regex equivalent</i>	<i>Built automaton</i>
Predicate pred	p	
Inverse [sh:inversePath P]	None	Take automaton P Inverse all transitions Transform all + into - Transform all - into +
Sequence (P1 P2)	P1 P2	
Alternate [sh:alternatePath (P1 P2)]	(P1 P2)	
Zero or one [sh:zeroOrOnePath P]	P?	
One or more [sh:oneOrMorePath P]	P+	
Zero or more [sh:zeroOrMorePath P]	P*	Equivalent to (P+)?

than a less selective suggestion engine. In order to evaluate the validity of this assumption, we asked volunteers to translate two texts into Turtle documents by using a given ontology. The produced documents were expected to be constituted of approximately 10 triples. One of the documents had to be written by using our auto-completion engine, the other by using an auto-completion engine similar to the one used by YASGUI, *i.e.* that displays all the terms of the ontology. The order of the two different documents and of the two auto-completion engines was randomized.

We used two different schemas:

- The Schema.org ontology. For this session, we used the RDF schema graph published on Github by Schema.org¹³. We slightly altered the graph to transform the cases where a pred-

¹³<https://github.com/schemaorg/schemaorg/blob/main/data/releases/14.0/schemaorg-all-https.ttl>

Table 3

Converting the transitions of the produced FSA to rules

Transition	Inference rules	Suggestion rules
$(S, +P, E)$	$\frac{(?u, P, ?v) \quad (?u, :pathsOf, m(S))}{(?v, :pathsOf, m(E))}$	$\frac{(?u, \dots, \dots) \quad (?u, :pathsOf, m(S))}{suggest(P)}$ $\frac{(?u, P, \dots) \quad (?u, :pathsOf, m(S))}{suggestAll(:pathsOf, m(E))}$
$(S, -P, E)$	$\frac{(?u, P, ?v) \quad (?v, :pathsOf, m(S))}{(?u, :pathsOf, m(E))}$	

icate only had one value for `schema:domainIncludes` or `schema:rangeIncludes` to `rdfs:domain` and `rdfs:range` to help the *inference engine* of Shacled Turtle. This alteration has no impact on the naive suggestion engine. As said previously, Schema.org is a big ontology with thousands of terms. For this session, we had 23 volunteers, 21 of them were Semantic Web experts with more than 3 years of usage and 6 had already used the Schema.org ontology.

- *Friend of a friend* (foaf)¹⁴. As this ontology is defined by using mostly RDFS, it benefits fully from the *inference engine*. Moreover, it is a small ontology, with a few dozen of terms. For this session, we had 11 volunteers, 7 of them were Semantic Web experts and none declared to already have used the ontology.

After writing the two different RDF documents, one with Shacled Turtle and one without it, they were asked to grade on a Likert scale [17] their feeling about the usefulness of both completion engine (naive and Shacled Turtle) and if they preferred an auto-completion engine over another one. We also let users explain in a free field why they preferred one engine, if any; and another free field to collect general feedback. Finally, we measured how much time each volunteer took to write each document.

The whole evaluation was conducted online. We published the source code of the platform and the anonymized collected results on Github at <https://github.com/BruJu/shacled-turtle-evaluation>.

Of the 34 volunteers, 17 declared to have no preference towards an engine or the other. Six volunteers even admitted to have seen no difference between the two engines. The number of people that prefer one engine over another is almost equal for both engines.

When asked separately, all volunteers gave a similar rank to both engines, the worst case being a strong appreciation on an engine and a neutral appreciation on the other; but 21 users gave the same appreciation to both.

Using Shacled Turtle does not enable the user to complete the task faster: 20 volunteers were faster to complete the second task than the first, regardless of if Shacled Turtle is the first engine or the second, and 14 took about the same time.

¹⁴<https://xmlns.com/foaf/spec/>

8. Discussion

In this section, we study some of the most recurring comments made by the volunteers about the tool to have a better understanding of what can be improved in Shacled Turtle.

Relevance of suggestions. Five volunteers showed a high enthusiasm about the approach and their comments showed that they understood well the purpose of the tool. In particular, two of them appreciated that the tool leads to less errors, feeling more confident about the produced graph.

However, in Section 6.1, we mentioned that the choice of which suggestions to filter out is, to some extent, arbitrary, and could lead to false negatives.

The question arises especially in the case of SHACL shapes: we suggest only predicates that are mentioned in the shape(s) of the subject, but unless these shapes are flagged with `sh:closed true`, they actually do not disallow *other* predicates. Similar issues may apply with RDFS classes, because an instance of a class might still be an instance of another one.

Indeed, three volunteers complained about the fact that Shacled Turtle produced less suggestions than the other engine. 21 volunteers ranked both engines similarly, and six of them explicitly reported that they did not notice any difference, suggesting that there is no clear benefit in reducing the overall number of suggestions.

Other filtering strategy. Most auto-completion engines enable users to filter the list of suggestions by name. In Code Mirror, and therefore in Shacled Turtle, when the user types for example `s:na`, the system will only show the terms that contain the characters `s:na` in that order (e.g. `s:familyName` or `s:eventStatus`). A common practice to find a desired term is to opportunistically reduce the list of terms using the filtering by name. Then when the user considers the list of terms to be short enough, they look further at the displayed terms. The responses of the volunteers indicate that they proceeded that way.

Therefore it might be more valuable to *promote* the suggestions we deem relevant than to filter out the others, and leave it to the user reduce the number of suggestions using filtering by name. Once a suggestion list is filtered out by the user, we think that Shacled Turtle could provide an efficient strategy to help the user pick the right term, in conjunction with manual filtering by name.

The importance of good documentation. Shacled Turtle shows, with each suggested term, a description (`rdfs:comment`) of that term when provided by the schema. While the query GUI of Wikidata does the same, because Wikidata IRIs are opaque, many other suggestions engine do not. During our experiment, seven volunteers reported that the descriptions of the terms are important, as they complained when descriptions were missing or incomplete, either because of bugs during the early stages of the experiment or because of the used schema. Five volunteers reported to have consulted the ontology online documentation to check how to use the ontology and have a better idea of the usage of the terms and their links. At the opposite, a volunteer reported that thanks to this tool, they fortunately did not feel the necessity to consult the ontology documentation.

One of the volunteers explained that the domain and the range of a property can be more informative than a description. The *schema to rules converter* could also be used to enrich the descriptions to add the links between the predicates and the different types and shapes.

As mentioned previously, Shacled Turtle should not be used to filter out choices from contextual data, but to enrich the documentation. This could be changed by using Shacled Turtle to *promote* terms that we deem relevant, either by displaying them first in the list, by highlighting them, or both: it would solve the issue of users not seeing a difference. To increase the perceived reliability of the tool, the decision made should be explained to the users, *i.e.* in case of an incomplete triple with only a subject, displaying which type or shape of the subject is used to suggest each relevant predicate; and for an incomplete triple with only a missing object, which type or shape of the suggested objects is used to suggest them depending on the subject and the predicate.

9. Conclusion

In this paper, in order to tackle the problem of writing RDF documents by hand, we proposed Shacled Turtle, an auto-completion engine that resorts to a schema graph to suggest terms related to the types and shapes of the subject of the triple that the user is writing. The system relies on two different rule engines : an *inference engine* that deduces the list of types and shapes of all resources in the *currently written graph* and a *suggestion engine* that provides possible following terms. However, in our experiments, the users barely saw any difference between a naive approach, proposing all terms that are in the ontology, and our approach: to find appropriate terms, they preferred to rely on other strategies like filtering by name and reading the ontology online documentation. We explain this by the inability of our method to display explicit insights: the difference between Shacled Turtle and the naive approach is implicit, as it consists in showing less options.

As users are in quest of information, four aspects can be considered:

- Enriching the descriptions of the terms, both with information extracted from the *schema to rules converter* like the links between the predicates and the types and shapes, and with contextual information to explain why the system thinks a term may be relevant in the current *incomplete triple*.
- Instead of using Shacled Turtle to filter out irrelevant terms, promote these relevant terms in the list of all existing terms.
- Running an inference engine to provide the list of types of the resources when the user hovers the resource. While this is currently done for RDFS, it could be expanded to any inference rule-set like OWL.
- Using a SHACL validation report to report errors, *i.e.* as a linting tool. This would lead to more accurate information and more visible error.

Another perspective that is to propose snippets, *i.e.* complete set of triples, instead of simple paths. SHACL sequence paths are paths composed of other paths: instead of requiring the user to chain blank nodes for each path that composes the sequence path, a snippet could be suggested that would build all the intermediate blank nodes at once. This approach would better benefit from SHACL paths and offer a higher level of suggestion.

Acknowledgments

We would like to thank all the volunteers for their time and their very valuable feedback.

References

- [1] S. Das, S. Sundara, R. Cyganiak, R2RML: RDB to RDF Mapping Language, W3C Recommendation, W3C, 2012. <https://www.w3.org/TR/2012/REC-r2rml-20120927/>.
- [2] R. Guha, D. Brickley, RDF Schema 1.1, W3C Recommendation, W3C, 2014. <https://www.w3.org/TR/2014/REC-rdf-schema-20140225/>.
- [3] D. L. McGuinness, F. Van Harmelen, et al., Owl web ontology language overview, W3C recommendation 10 (2004) 2004.
- [4] D. Kontokostas, H. Knublauch, Shapes Constraint Language (SHACL), W3C Recommendation, W3C, 2017. <https://www.w3.org/TR/2017/REC-shacl-20170720/>.
- [5] J. Bruyat, Bruju/shacled-turtle: Shacled turtle 0.0.3, 2022. URL: <https://doi.org/10.5281/zenodo.6907388>. doi:10.5281/zenodo.6907388.
- [6] E. Prud'hommeaux, G. Carothers, RDF 1.1 Turtle, W3C Recommendation, W3C, 2014. <https://www.w3.org/TR/2014/REC-turtle-20140225/>.
- [7] M. A. Musen, The protégé project: a look back and a look forward, AI matters 1 (2015) 4–12.
- [8] J. Wright, S. J. Rodríguez Méndez, A. Haller, K. Taylor, P. G. Omran, Schímatos: a shacl-based web-form generator for knowledge graph editing, in: International Semantic Web Conference, Springer, 2020, pp. 65–80.
- [9] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, R. Van de Walle, Rml: a generic language for integrated rdf mappings of heterogeneous data, in: Ldow, 2014.
- [10] L. Han, T. Finin, C. Parr, J. Sachs, A. Joshi, Rdf123: from spreadsheets to rdf, in: International Semantic Web Conference, Springer, 2008, pp. 451–466.
- [11] P.-A. Champin, G. Kellogg, D. Longley, JSON-LD 1.1, W3C Recommendation, W3C, 2020. <https://www.w3.org/TR/2020/REC-json-ld11-20200716/>.
- [12] K. Rafees, S. Abiteboul, S. Cohen-Boulakia, B. Rance, Designing scientific sparql queries using autocompletion by snippets, in: 2018 IEEE 14th International Conference on e-Science (e-Science), IEEE, 2018, pp. 234–244.
- [13] L. Rietveld, R. Hoekstra, Yasgui: not just another sparql client, in: Extended Semantic Web Conference, Springer, 2013, pp. 78–86.
- [14] G. Gombos, A. Kiss, Sparql query writing with recommendations based on datasets, in: International Conference on Human Interface and the Management of Information, Springer, 2014, pp. 310–319.
- [15] S. Ferré, Sparklis: An expressive query builder for sparql endpoints with guidance in natural language, Semantic Web 8 (2017) 405–418.
- [16] G. de la Parra, A. Hogan, Fast approximate autocompletion for sparql query builders (2021).
- [17] R. Likert, A technique for the measurement of attitudes., Archives of psychology (1932).

Enabling Exploratory Search on Manufacturing Knowledge Graphs

Kevin Haller¹, Fajar J. Ekaputra^{1,2}, Marta Sabou^{2,1} and Florina Piroi¹

¹TU Wien, Vienna, Austria

²WU Wien, Vienna, Austria

Abstract

Knowledge graphs have been recognized in manufacturing as a suitable technology for integration of multidisciplinary knowledge from heterogeneous data sources. The effective reuse of this knowledge can better inform stakeholders in their decision making processes and consequently, establish a competitive advantage. In contrast to the utilization of knowledge graphs for autonomous decision making systems, less attention in production research has been given to the creative participation of humans in the exploration of manufacturing knowledge graphs. Exploratory search systems are a promising solution to facilitate this participation. However, most exploratory search systems focus on general knowledge graphs for which common knowledge is sufficient. We argue that within the complex environment of manufacturing, closer attention has to be paid to particular exploratory search features. In this paper, we therefore present a configurable and adaptive exploratory search system, which implements three special features. Firstly, adaptability of the system to multiple (engineering) perspectives. Secondly, visibility of provenance details about statements to simplify investigative work. And finally, a tree view for browsing deep hierarchical structures.

Keywords

knowledge graph, exploratory search, industry 4.0

1. Introduction

The new paradigm shift in manufacturing, which is commonly referred to as the *fourth industrial revolution*, is guided by the fusion of traditional manufacturing technology with modern information and communication technology [1]. This vision is investigated by several strategic initiatives such as *Industrie 4.0* in Germany [2]. Core to all of these initiatives is the digitization of multidisciplinary information about production systems and processes.

Knowledge graphs (as defined by Galkin et al. [3]) are one solution to facilitate this digital transformation. As shown by the survey of Li et al. [4], knowledge graphs have received considerable attention in production research over the last years. In fact, it has been widely recognized as an important component of the next generation of information systems for

VOILA! 2022: 7th International Workshop on Visualization and Interaction for Ontologies and Linked Data, October 23, 2022, Virtual Workshop, Hangzhou, China, Co-located with ISWC 2022.

✉ kevin.haller@tuwien.ac.at (K. Haller); fajar.ekaputra@tuwien.ac.at (F.J. Ekaputra); marta.sabou@wu.ac.at (M. Sabou); florina.piroi@tuwien.ac.at (F. Piroi)

ORCID 0000-0001-8949-610X (K. Haller); 0000-0003-4569-2496 (F.J. Ekaputra); 0000-0001-9301-8418 (M. Sabou); 0000-0001-7584-6439 (F. Piroi)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

manufacturing [4]. Moreover, industrial enterprises started to construct knowledge graphs related to manufacturing such as Bosch [5] or Siemens [6].

Problem. The effective reuse of manufacturing knowledge can better inform stakeholders in their decision making and consequently, establish a competitive advantage [7]. Much work in production research has been dedicated to the utilization of knowledge graphs for autonomous decision making systems, e.g., Rožanec et al. [8]. However, less attention has been given to the creative participation of human stakeholders in the exploration of manufacturing knowledge graphs. In particular, we present two concrete use cases in which this need arises:

(UC1) *OntoTrans*¹ is an EU project (H2020) which aims to design an environment of tools for translators who are working on innovation challenges to manufacturing processes. Three companies take part in this project with their individual innovation challenge to improve a certain manufactured product in respect to several key performance indicators. An innovation challenge requires the interactive collaboration of stakeholders from different disciplines who have diverging perspectives and conceptualizations of a problem. A translator has to provide a communication-based glue between the involved stakeholders. The innovation challenge as well as the corresponding manufacturing information are converted into an ontological form. A tool to explore this interdisciplinary knowledge graph could assist translators in their tasks.

(UC2) *Aspern pilot factory*² is one of several Austrian "learning and experimentation factories". It currently hosts a series of valuable collaborative and industrial robotic arms as well as a wide range of supporting tools (grippers, 3D cameras, projectors, etc.), which can be used by students, researchers, and companies for their own purposes. However, interested students, researchers, and companies are often unaware of the availability and capabilities of the manufacturing technology in this pilot factory, which contributes to a relatively low usage degree of these expensive, state of the art production equipment. Furthermore, best practises and design patterns in software engineering are a valuable guide for writing robotics software. A knowledge graph was built from this interdisciplinary knowledge and a tool to explore it could help users of the pilot factory to learn about how to realize their individual projects with the available equipment.

Solution. Exploratory search systems are a promising solution to support human stakeholders in their decision and sense making. Exploratory search is an open-ended and multi-faceted information-seeking activity. It is commonly used in the context of scientific discovery, learning and decision making [9]. Most exploratory search systems focus on general knowledge graphs for which common knowledge is sufficient. We argue that within in the complex environment of manufacturing, closer attention has to be paid to particular exploratory search features.

Contribution. To that end, we provide in Section 2, a list of exploratory search features towards which we want to draw special attention, because they were commonly requested in interviews with stakeholders of our two use cases. Section 3 proposes a configurable and adaptive exploratory search interface, which considers the special search features for manufacturing

¹<https://ontotrans.eu/>

²<https://www.pilotfabrik.at/language/en/>

Common Features (from Marie et al. [10])	Manufacturing Features
(1) Overview and analysis feature (2) Faceted interface (3) Result clustering (4) Facilitator for back and forth navigation (5) Query-suggestions and refinement (6) Serendipitous discovery enforcement (7) Result explanation generator (8) Memorization feature	(9) Multiperspectival exploration (10) Provenance visibility (11) Hierarchical browsing

Table 1

Exploratory search features for the domain of manufacturing.

domain. The system architecture of our exploratory search is briefly introduced in Section 4. Finally, the preliminary evaluation of our exploratory search system is presented in Section 5.

2. Features for Manufacturing

A survey about widespread features in KG-based exploratory search systems was conducted by Marie et al. [10] (see left-side of Table 1). Three additional features that are important to manufacturing professionals were identified from informal interviews (see right-side of Table 1). Two translators to an industry partner in OntoTrans (UC1), two smart manufacturing researchers working in the pilot factory (UC2) with industry and university background respectively, and one simulation expert of production plants were among the interviewees. They were given an introduction to a basic exploratory search system in beforehand, and were then asked to discuss their requirements towards such a search system. The remaining part of this section elaborates on the three extracted features from these collected requirements.

Multiperspectival exploration enables stakeholders to select and switch between different perspectives on a manufacturing knowledge graph. In a manufacturing environment, stakeholders from multiple disciplines work together and they might have different conceptualization of manufacturing entities. Moreover, different disciplines might have a distinct perspective on what information is relevant about a manufacturing entity, and don't want to be overloaded with information that is irrelevant to them. A machinery could for instance be viewed in a impressively detailed manner by mechanical engineers, but only be seen as a black box with certain wiring requirements by electrical engineers.

Provenance visibility is a valuable feature in a manufacturing environment, where interdisciplinary knowledge is integrated from many heterogeneous sources. If a stakeholder is interested in investigating the properties of a physical component, then it might be necessary to know the original data source in which these values for a property have been reported in order to resolve ambiguities for proper decision making. The value of a parameter could have been gathered from promotional material of the manufacturer, or could have been measured by a local engineer. The visibility of provenance information allows stakeholders to quickly reason about the trustworthiness of presented statements.

Hierarchical browsing is a frequent search task over manufacturing knowledge graphs in which the digital twin of production plants or machines might be comprised of deep containment relationships between components. Furthermore, manufacturing processes and bill of materials

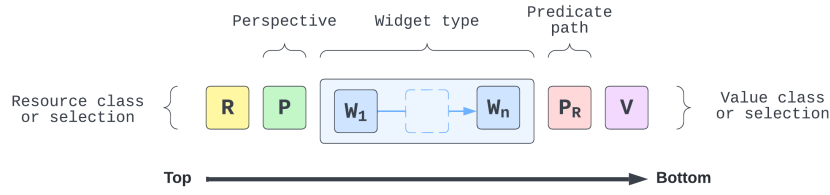


Figure 1: Scope hierarchy for the adaptive UI engine.

of products can have deep hierarchies. Thus, stakeholders must be able to quickly navigate through these hierarchies, and be able to cognitively focus on important parts and not be overwhelmed by the whole hierarchy.

3. Adaptive Exploratory Search Interface

Based on our interviews with stakeholders from the two manufacturing use cases, exploratory search interfaces in manufacturing ought to be designed with the requirements from Section 2 in mind. Preferences about the presentation method of entities and arrangement of interface elements might differ from use case to use case. Bootstrapping a new exploratory search system is however a time-consuming and costly endeavour. Thus, we propose in this section a configurable and adaptive search interface, which aims to be flexible and reusable. The architecture of the whole system is outlined in Section 4.

This adaptive search interface is based on the concept of *scopes* and *configurations*, which was introduced by the Linked Data Reactor (LD-R) [11]. A scope is in LD-R a hierarchical permutation of *dataset*, *resource*, *property* and *value*, where *dataset* is at the top and *value* at the bottom of this hierarchy. A presentation template (i.e. a *configuration*) can then be written in JSON format for a particular scope, which tells the web application how to render entities in the knowledge graph that match this scope. Each scope has a specificity, and if entities belong to multiple scopes, then the configuration of the most specific scope overwrites the others. While we adapt this mechanism for our adaptive search interface, the scope components and their hierarchy were changed to integrate multiperspectival exploration.

Scope components are organized in the hierarchy that is depicted in Figure 1.

- At the top of this hierarchy is the *resource class or selection* **R**. This can either be the IRI of a class of resources or an IRI matching one single specific resource. The class `rdfs:Resource` can be used as a wildcard to match all resources in a knowledge graph.
- The following scope component is the *perspective* **P**, which allows to adapt metrics powering the search interface as well as the presentation of widgets and entities to the

currently selected perspective. The underscore (i.e. '_') is a wildcard and represents all perspectives.

- A list of *widgets* $\mathbf{W}_1, \dots, \mathbf{W}_n$ is the next component in the hierarchy. Widget refers here to an interface element with an unique name. A list always starts with the outermost widget and ends with the innermost. The distinct presentation of entities in different widgets (e.g. bookmarks and result overview) is made possible with this scope component.
- *Predicate path* \mathbf{P}_R is one level below in the hierarchy. A predicate path is a subset of a property path in SPARQL, only allowing sequences (i.e. '/'), alternatives (i.e. '|') and inversion (i.e. '^'). A predicate path makes it possible to configure how a matching property itself is presented and how a collection of it's values should be visualized.
- At the bottom is the *value class or selection* \mathbf{V} . This scope component has the purpose to configure the presentation of matching values. In order to facilitate the design of UI components for visualizing provenance details, metadata about relevant named graphs is passed to values. In our solution, it is assumed that the *single-triple named graphs* approach is used to state provenance information in favor of *RDF reification*, *singleton properties* [12] and *RDF-star* [13].

Templates are aggregated in files written in HCL³, which is easy to edit and read by humans. Listing 1 shows a snippet of such a configuration file for the class `RobotType`. UI components are assigned with 'handler' to widgets, properties, and values. These UI components can moreover be customized by passing properties with a 'config' object. Line 29-39 in Listing 1 states that every value for the property "reach" of a `RobotType` shall be rendered as ordinary text literal as long as it isn't a quantity value from the QUDT ontology, which needs some additional parsing. Listing 2 shows how the recommendation section for instances of `RobotType` is configured to use LDSO [14] in general, but is limited to specific classes for the two mentioned perspectives. Similarly, distinct ranking metrics could be chosen.

³Hashicorp configuration language - <https://github.com/hashicorp/hcl>

Listing 1: Configuration of RobotType.

```

(10) class = "cobot:RobotType"
(11)
(12) perspective _ widget infobox {
(13)   handler = "GeneralInfoBox"
(14)   config {
(15)     sections = ["prop_table",
(16)                 "recommendations"]
(17)   }
(18) }
(19) perspective _ widget infobox section prop_table {
(20)   handler = "ProvenanceTableSection",
(21)   config {
(22)     neighbourhood {
(23)       include = ["cobot:degreesOfFreedom",
(24)                 "cobot:handlingPayload", "cobot:reach",
(25)                 "cobot:skills"],
(26)     }
(27)   }
(28) }
(29) perspective _ widget infobox {
(30)   property "cobot:reach" {
(31)     handler = "LinkedProperty"
(32)     value _ {
(33)       handler = "TextValue"
(34)     }
(35)     value "qudt:Quantity" {
(36)       handler = "QudtQuantityValue"
(37)     }
(38)   }
(39) }

```

Listing 2: Different recommendation configurations per perspective.

```

(40) perspective _ widget infobox {
(41)   section recommendations {
(42)     handler = "SimilaritySection"
(43)     config {
(44)       number = 4
(45)       ranking = {
(46)         ldsd {
(47)           step = "esm.exploit.sim.ldsds"
(48)           weight = -1.0
(49)         }
(50)       }
(51)     }
(52)   }
(53) }
(54) perspective RoboticsEngineer widget infobox {
(55)   section recommendations {
(56)     config {
(57)       classes = ["cobot:RobotType"]
(58)     }
(59)   }
(60) }
(61) perspective SoftwareEngineer widget infobox {
(62)   section recommendations {
(63)     config {
(64)       classes = ["cobot:HandlingFunction",
(65)                 "star:ArchitecturalElement"]
(66)     }
(67)   }
(68) }

```

Search interface is provided by a rendering engine that interprets these templates. The interface is intended to be similar to major search engines on the Web in order to lower the learning curve. The main entry point for starting an exploration is a keyword search as depicted in Figure 2. The result set of a keyword search is listed vertically, and an info box is shown for the first entry of the result list. Nonetheless, a user can switch to a different search paradigm, e.g. a tree view.

4. System Architecture

Many KG-based exploratory search systems report to only rely on a SPARQL interface to the target knowledge graph, which has the big advantage that these systems can directly be applied to virtually every RDF-based knowledge graph. An integral part of our system is the computation of centrality metrics (e.g. PageRank) for the ordering of resources according to their "importance" and similarity metrics (e.g. LDSD [14]) for recommendations. A reasonable responsiveness is however an important non-functional requirement towards search interfaces, and we claim that this is hard to achieve without precomputing metrics or sophisticated indexing techniques. Thus, we introduce a *middleware* application which sits on top of the storage solution for the target knowledge graph. The conceptual overview of our whole exploratory search system is depicted in Fig. 3.

The adaptive search interface presented in Section 3 is provided by our **web application**, which is designed to be a thin single-page application. ReactJS⁴ is used to implement the UI

⁴JavaScript library for building user interfaces - <https://reactjs.org>

SPARQL

TREE VIEW

ROBOTICS ENGINEER

EN

universal robot

ALLE

PRODUCTION PLANT

ROBOT

END EFFECTOR

ROBOT TYPE


END EFFECTOR TYPE

SKILLS

10 results (29 ms)

UNIVERSAL ROBOTS UR10


RobotType



While the largest robot arm in the UR family and the one with the most muscle power, the UR10 does not compromise on precision. The collaborative robot arm will automate heavier-weight process tasks with payload requirements of up to 10 kg.

UNIVERSAL ROBOTS UR3


RobotType



The UR3 collaborative robot is a smaller collaborative table-top robot, perfect for light assembly tasks and automated workbench scenarios. The compact table-top cobot weighs only 24.3 lbs (11 kg), but has a payload of 6.6 lbs (3 kg), ±360-degree rotation ...

UNIVERSAL ROBOTS UR5


RobotType



The slightly bigger UR5 is ideal for automating low-weight processing tasks like picking, placing and testing. The medium-sized robot arm is easy to program, fast to set up and, just like the other collaborative members of the UR family, offers one of the ...

Neobotix MP-400


RobotType



The mobile robot MP-400 was designed for daily use in industrial applications. Its construction is based on the best elements of our well proven robots but also includes many innovative ideas. The MP-400's primary task is the flexible material transport ...

KUKA LBR IIWA 7 R800


RobotType



The LBR iiwa is the world's first series-produced sensitive, and therefore HRC-compatible, robot. LBR stands for "Leichtbauroboter" (German for lightweight robot), iiwa for "intelligent industrial work assistant". This signals the beginning of a new era in ...

UNIVERSAL ROBOTS UR10


RobotType



While the largest robot arm in the UR family and the one with the most muscle power, the UR10 does not compromise on precision. The collaborative robot arm will automate heavier-weight process tasks with payload requirements of up to 10 kg.

Country: Denmark


Compatible End Effectors:



Schunk Gripper EGA

EndEffectorType


Electric 2-finger parallel gripper with lightweight profile rail ...



Robotiq Gripper Hand-E

EndEffectorType

Robotiq's Hand-E Gripper is simple to integrate in your producti ...



Robotiq Gripper 2F-85


EndEffectorType

The 2F-85 and 2F-140 Adaptive Grippers are the world's best-sell ...

Property Table (Provenance)

	Pilot factory interviews	Cobot List from Robotics World
Handling Function	Move, Approach, Depart, Retract	
Skill	Wait, Retrieve, Store	
Reach	1300 mm	
Handling Payload	10 kg	
Degrees Of Freedom	6	


Related Robot Types



UNIVERSAL ROBOTS UR5

RobotType


The slightly bigger UR5 is ideal for automating low-weight proce ...



UNIVERSAL ROBOTS UR3

RobotType


The UR3 collaborative robot is a smaller collaborative table-top ...



FANUC CR7IA

RobotType

I'm small, flexible and can work right by your side. I take care ...



KUKA LBR IIWA 7 R800

RobotType

The LBR iiwa is the world's first series-produced sensitive, and ...

Figure 2: Entry point for exploration (keyword search).

components, and the state is managed with Redux⁵. In order to be able to render the UI, the *web application* needs to load the UI configuration (see Section 3). The required data for rendering the UI components is then fetched by assembling corresponding *exploration flows*, and sending them to the *middleware*. The *web application* is only responsible for the correct rendering and isn't handling any RDF data itself.

⁵State container for JavaScript applications - <https://redux.js.org>

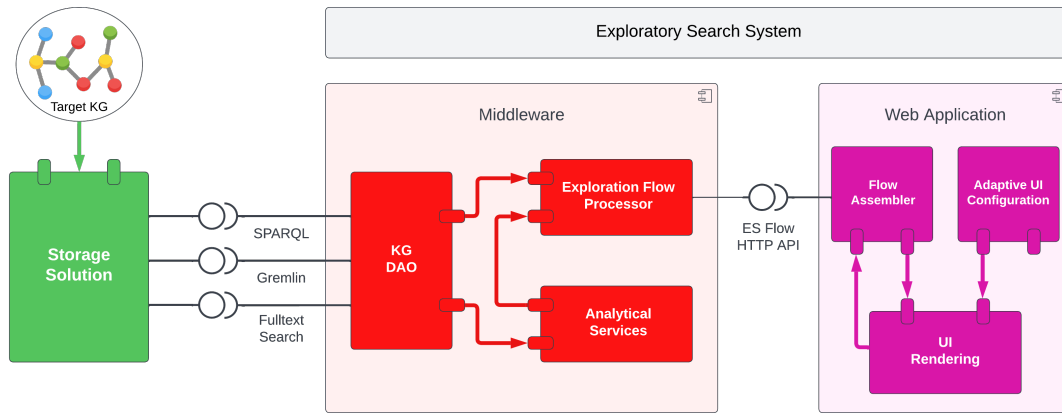


Figure 3: Conceptual overview of the exploratory search system.

An *exploration flow* is a sequence of steps which all execute a single operation. Listing 3 shows a flow that computes the weighted sum of LDSD [14] and a general peer pressure metric for all pairs between a particular gripper and all other resources in the knowledge graph. Then, the collection of pairs is ordered by this weighted sum, and only the top 10 pairs are eventually returned. The goal of an *exploration flow* is to abstract the more complex part of the Semantic Web technologies, and only expose the basic concepts of RDF to the *web application*.

Listing 3: Top 10 recommendations for a certain gripper (Python).

```
(1) f = Single(resource='ex:gripper04') \
(2)   >> PairWith(flow=All()) >> (LSDS() | PeerPressure()) \
(3)   >> WeightedSum('sum', {Sim.lsd: -1, Sim.peerpressure: 1}) \
(4)   >> OrderBy('sum', strategy=Order.DESC) >> Limit(n=10)
(5) resp = FlowAPI("http://localhost:8080").execute(f)
```

The *middleware* is a Spring Boot application written in Java and it has two main tasks. Firstly, it provides an interpreter for *exploration flows*, which parses flows and executes their steps one by one in the correct order. Secondly, it orchestrates the computation of analytical services such as centrality and similarity metrics among others. However, we can't solely rely on SPARQL for these tasks, because:

- (a) Graph traversal is limited in the specification of SPARQL 1.1 [15]. While property paths add the ability to check whether a route of arbitrary length exists between two nodes in the knowledge graph, a wider range of path operations aren't supported such as computing the shortest distance between two nodes. Query languages for property graphs usually don't have this limitation, and RDF-based knowledge graphs can easily be represented in property graphs. However, at the moment of writing no query language is accepted as a standard for property graphs. GQL is one of the emerging attempts to establish such a standard [16]. Cypher and Gremlin are nevertheless two prominent choices.
- (b) Full-text searches are a feature of many triplestores, but it is not included in the specifi-

TASK A	TASK B
Imagine that you are a member of a team which is working on a manufacturing project with collaborative robots and you have access to the equipment located at the pilot factory in Aspern. In this project, you have to move a 4 kg heavy and cubic object with a length of 20cm from a conveyor belt to a manufacturing cell that is 60cm away. Goal You have been asked to design a hardware setup for this project with equipment that is available at the pilot factory. Your team would be glad about a brief presentation of your findings.	Imagine that you are a member of a team which is working on a collaborative robot picking up a small transistor from a storage box and placing it on a certain position on a circuit board. The robot works in proximity of human workers and it might have to interrupt it's task, due to them coming too close. However, we don't want the robot to remain in this state, and proceed with the task as soon as possible. Goal: You have been asked to explore design patterns that could be lend from software engineering for this kind of error handling and eventually present promising design patterns to your team.

Table 2

Exploratory search tasks for evaluating the system.

cation of SPARQL 1.1 [15]. The SPARQL syntax for issuing a full-text search is vendor-specific, and the configurability of search indices varies from vendor to vendor. Alternatively, one might want to use an external solution for creating full-text indices.

Due to these limitations of SPARQL, the *middleware* expects three interfaces to the stored knowledge graph: (1) SPARQL 1.1, (2) Gremlin as query language for property graphs, and (3) simple full-text search API. The *middleware* includes plugins, which implement these three interfaces for a number of popular triplestores (Blazegraph, GraphDB, Stardog and Virtuoso). And given that barely any triplestore supports the Gremlin query language, the *middleware* provides also a mechanism to clone the knowledge graph over the SPARQL interface into an embedded JanusGraph⁶ instance.

5. Preliminary Evaluation

Our exploratory search system⁷ was evaluated on a small scale with five participants for the pilot factory use case (UC2), which is why we want to set a heavy focus on qualitative analysis. All five participants were non-experts in smart manufacturing and robotics, but they were knowledgeable in software engineering. Participants took in our experiments the role of an information seeker and explored the domain of collaborative robotics. The evaluation aimed to assess the ability of our system to allow a user to interactively investigate, learn and make sense of a topic from this manufacturing domain.

The two tasks listed in Table 2 were designed to elicit exploratory search behaviour from participants. The experiment started with a brief survey to assess participant's a-priori knowledge about robotics and software engineering. Then, they were given both exploratory search tasks one-by-one, and had 20 minutes time with our search system for each task. Afterwards, their gained knowledge was assessed by a new survey. Moreover, we asked the participants

⁶Open source distributed graph database - <https://janusgraph.org/>

⁷Online deployment - <https://data.ifs.tuwien.ac.at/cobot>

Source code for local deployment - <https://gitlab.tuwien.ac.at/kevin.haller/cobot-playground>

to rate the "usefulness" of single features for completing their tasks, and conducted informal interviews to gather feedback. All the sessions were recorded for later video analysis.

Overall, all participants gained some additional knowledge and were able to present a reasonable solution for both tasks to their fictional team. Regarding the exploratory search features, the participants thought that info boxes were the most useful, which isn't surprising given that it plays a prominent role in our search system. On the other hand, the tree view for browsing deep hierarchies wasn't rated highly. A video analysis of the recorded sessions showed that the participants were overburdened by the current design of the tree view and the click rate was low. Moreover, it showed that they preferred to use browser tabs as well as back and forth navigation over our in-built memorization feature.

6. Related Work

Little work has been published for exploratory search in manufacturing and production. Metaphactory describes in [17] their platform for knowledge graph management and briefly outlines use cases in the engineering and manufacturing domain at Siemens. While exploratory search is not discussed, it touches on faceted navigation, query building assistance as well as customizable search experience and result visualization. Sabou et al. [18] show an approach to transform a central repository of architectural knowledge at Siemens into a knowledge graph, and furthermore, present a search system with an user interface similarly to major search engines on the Web. The search interface includes info boxes with faceted navigation and a KG-based recommendation engine. Yet, these systems don't address the manufacturing features discussed in this paper.

Much work has been published about utilizing the semantics of knowledge graphs to render configurable presentations of resources. Fresnel [19] is an ontology that provides a vocabulary to annotate resources, classes and properties with presentation knowledge. A browser understanding this vocabulary can then visualize these entities accordingly. Rutledge et al. [20] propose an extension of this Fresnel ontology that allows to define presentation knowledge for provenance information as well. LESS [21] introduces on the other side a new template language based on Smarty⁸ to define the presentation of resources. A LESS processor is then taking the designed template and applies it either to a specified RDF document or the result set of a specified SPARQL query. Uduvudu [22] utilizes templates in a similar manner, but additionally proposes algorithms for the automatic selection of templates based on the input data. These solutions focus however exclusively on the visualization of entities in a knowledge graph and thus, not all aspects of configuring an adaptive search interface are covered.

Linked Data Reactor (LD-R) [11] is a faceted explorer with a configurable and adaptive user interface. It proposes the concept of *scopes* and *configurations* as outlined in more detail in Section 3. LD-R enables the configuration of facets following these newly introduced concepts. Moreover, it implements a mechanism to change its UI components based on the persona of a user. Nonetheless, LD-R exclusively focuses on faceted exploration. Furthermore, we argue that simple perspectives are a more accessible mechanism for stakeholders in manufacturing than user personas in LD-R. KG-Explorer [23] proposes similarly a configurable and adaptive

⁸PHP template engine - <https://www.smarty.net>

search interface that mainly focuses on faceted exploration, but not exclusively. Some aspects of the overall search interface can be adapted in a configuration file as well as templates can be created for editorial pages about entities. However, the search behaviour and interface elements can't be customized for different user perspectives. Moreover, these systems don't address the other two manufacturing features discussed in this paper.

7. Conclusion and Future Work

KG-based exploratory search systems share a common set of features. In this paper, we identified and reported three particular features from interviews that are additionally relevant for enabling exploratory search on manufacturing knowledge graphs. The adaptability of the search system to *multiple (engineering) perspectives* is one of these features. Moreover, the *visibility of provenance* information supports stakeholders in manufacturing in their investigative work. Finally, *browsing deep hierarchical structures* of containment relationships is a frequent task on manufacturing knowledge graphs.

We presented an adaptive and configurable exploratory search system, which implements solutions to these features among others. A new scope component representing *perspectives* was introduced to allow the *multiperspectival* configuration of visualizations as well as the underlying search system. The *visibility of provenance* details was addressed by passing information about the named graphs for each statement to the corresponding UI components and hence, enabling the implementation of UI components that visualize this knowledge. The requirement of *browsing deep hierarchies* was addressed with the implementation of a simple tree view (as known from Protegé for example) in combination with info boxes.

A small-scale evaluation of our system showed the usefulness of the implemented features for participants for completing their search tasks, with the exception for the tree view and memorization feature. Participants were overburdened by our current design of the tree view. Moreover, our implemented memorization feature was mainly ignored by participants in favor of maintaining multiple browser tabs. Overall, the participants were able to present a reasonable solution for their search tasks.

Hence, as future work we want to improve on the tree view for *browsing deep hierarchies*. Furthermore, we want to make our rendering engine compatible with RDF-star [13], due to its increasing popularity. Eventually, we aim to evaluate the improved system on the OntoTrans use case (UC1) with a larger number of participants.

Acknowledgments

This work is supported by European Union's Horizon 2020 research project OntoTrans under Grant Agreement No 862136.

References

- [1] S. Haag, R. Anderl, Digital twin – Proof of concept, Manufacturing Letters 15 (2018) 64–66. doi:10.1016/j.mfglet.2018.02.006, Industry 4.0 and Smart Manufacturing.

- [2] T. Bauernhansl, M. T. Hompel, B. Vogel-Heuser, *Industrie 4.0 in Produktion, Automatisierung und Logistik: Anwendung-Technologien-Migration*, Springer, 2014. doi:10.1007/978-3-658-04682-8.
- [3] M. Galkin, S. Auer, M.-E. Vidal, S. Scerri, *Enterprise Knowledge Graphs: A Semantic Approach for Knowledge Management in the Next Generation of Enterprise Information Systems*, in: ICEIS, 2017, pp. 88–98. doi:10.5220/0006325200880098.
- [4] X. Li, M. Lyu, Z. Wang, C.-H. Chen, P. Zheng, *Exploiting knowledge graphs in industrial products and services: A survey of key aspects, challenges, and future perspectives*, *Computers in Industry* 129 (2021) 103449. doi:10.1016/j.compind.2021.103449.
- [5] J. Strötgen, T.-K. Tran, A. Friedrich, D. Milchevski, F. Tomazic, A. Marusczyk, H. Adel, D. Stepanova, F. Hildebrand, E. Kharlamov, *Towards the Bosch Materials Science Knowledge Base*, in: ISWC 2019 Satellite Tracks, volume 2456, CEUR Workshop Proceedings, 2019, pp. 323–324.
- [6] T. Hubauer, S. Lamparter, P. Haase, D. M. Herzig, *Use Cases of the Industrial Knowledge Graph at Siemens*, in: ISWC 2018 P&D/Industry/BlueSky Tracks, volume 2180, CEUR Workshop Proceedings, 2018.
- [7] C. Palmer, Z. Usman, O. C. Junior, A. Malucelli, R. I. Young, *Interoperable manufacturing knowledge systems*, *International Journal of Production Research* 56 (2018) 2733–2752. doi:10.1080/00207543.2017.1391416.
- [8] J. M. Rožanec, J. Lu, J. Rupnik, M. Škrjanc, D. Mladenčić, B. Fortuna, X. Zheng, D. Kiritsis, *Actionable cognitive twins for decision making in manufacturing*, *International Journal of Production Research* 60 (2022) 452–478. doi:10.1080/00207543.2021.2002967.
- [9] R. W. White, R. A. Roth, *Exploratory Search: Beyond the Query-Response Paradigm*, *Synthesis Lectures on Information Concepts, Retrieval, and Services* 1 (2009). doi:10.2200/s00174ed1v01y200901icr003.
- [10] N. Marie, F. Gandon, *Survey of Linked Data Based Exploration Systems*, in: 3rd International Workshop on Intelligent Exploration of Semantic Data, volume 1279, CEUR Workshop Proceedings, 2015.
- [11] A. Khalili, A. Loizou, F. van Harmelen, *Adaptive Linked Data-Driven Web Components: Building Flexible and Reusable Semantic Web Interfaces*, in: European Semantic Web Conference, Springer, 2016, pp. 677–692. doi:10.1007/978-3-319-34129-3_41.
- [12] V. Nguyen, O. Bodenreider, A. Sheth, *Don't like RDF reification? Making statements about statements using singleton property*, in: International Conference on World Wide Web (WWW), Association for Computing Machinery, 2014, p. 759–770. doi:10.1145/2566486.2567973.
- [13] O. Hartig, P.-A. Champin, G. Kellog, A. Seaborne, *RDF-star and SPARQL-star*, W3C Community Group Report (2021). URL: <https://w3c.github.io/rdf-star/cg-spec>.
- [14] A. Passant, *Measuring semantic distance on linking data and using it for resources recommendations*, in: AAAI Spring Symposium Series, 2010.
- [15] S. Harris, A. Seaborne, E. Prud'hommeaux, *SPARQL 1.1 query language*, W3C recommendation (2013). URL: <https://www.w3.org/TR/sparql11-query/>.
- [16] ISO/IEC JTC 1/SC 32, *Information Technology — Database Languages — GQL*, Standard ISO/IEC CD 39075, International Organization for Standardization, 2019.
- [17] P. Haase, D. M. Herzig, A. Kozlov, A. Nikolov, J. Trame, *Metaphactory: A platform for*

- knowledge graph management, *Semantic Web 10* (2019). doi:10.3233/SW-190360.
- [18] M. Sabou, F. J. Ekaputra, T. Ionescu, J. Musil, D. Schall, K. Haller, A. Friedl, S. Biffl, Exploring Enterprise Knowledge Graphs: A Use Case in Software Engineering, in: *European Semantic Web Conference*, 2018, pp. 560–575. doi:10.1007/978-3-319-93417-4_36.
 - [19] E. Pietriga, C. Bizer, D. Karger, R. Lee, Fresnel: A Browser-Independent Presentation Vocabulary for RDF, in: *International Semantic Web Conference*, Springer, 2006, pp. 158–171. doi:10.1007/11926078_12.
 - [20] L. W. Rutledge, P. Mellema, T. Pietersma, S. M. M. Joosten, Displaying triple provenance with extensions to the Fresnel vocabulary for semantic browsers, in: *Workshop on the Visualization and Interaction for Ontologies and Linked Data*, volume 3023, *CEUR Workshop Proceedings*, 2021, pp. 103–114.
 - [21] Raphael, D. S. A. Sören, Doebling, LESS - Template-Based Syndication and Presentation of Linked Data, in: *Extended Semantic Web Conference*, Springer, 2010, pp. 211–224. doi:10.1007/978-3-642-13489-0_15.
 - [22] M. Luggen, A. Gschwend, B. Anrig, P. Cudré-Mauroux, Uduvudu: a graph-aware and adaptive UI engine for linked data, in: *Workshop on Linked Data on the Web (LDOW)*, volume 1409, *CEUR Workshop Proceedings*, 2015.
 - [23] T. Ehrhart, P. Lisena, R. Troncy, KG Explorer: a Customisable Exploration Tool for Knowledge Graphs, in: *Workshop on the Visualization and Interaction for Ontologies and Linked Data*, volume 3023, *CEUR Workshop Proceedings*, 2021, pp. 63–75.

VOWLExplain: Knowledge Graph Visualization for Explainable Artificial Intelligence

Filipa Serrano, Susana Nunes and Catia Pesquita

LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

Abstract

Ontologies and Knowledge Graphs are a potential solution to the problem of lack of explainability in Artificial Intelligence, and are especially suited to explain how a given prediction fits with existing knowledge in a domain. Communicating these semantic explanations to end users in a correct, clear and trustworthy fashion is crucial to support the adoption of artificial intelligence in critical and complex domains such as healthcare. We developed VOWLExplain, a tool that supports the visualization of semantic post-hoc explanations for predictions made by AI black-box models. We performed a small-scale user study comparing text-based and graph-visualization based explanations in a case study for personalized medicine. The results highlighted the diversity of how users perceive explanations, and demonstrated that although users indicate a slight preference for graph based representations, they generally rate them as correct and as trustworthy as text-based explanations, but do consider them clearer.

Keywords

Explainable AI, Knowledge Graph, Visualization

1. Introduction

Artificial Intelligence (AI), specifically Machine Learning (ML) algorithms, have been gaining more importance due to the development of powerful models, such as Deep Learning (DL). There are several applications in which DL models are being used, with great potential and promising results [1].

However, the application of black-box AI in critical use cases is hindered by their lack of explainability. Black-box models are opaque models whose internal mechanism is unknown or uninterpretable to humans. Explainability, the ability of a user to understand, evaluate and eventually trust a specific prediction made by a machine learning model is essential for applying these models in sensitive fields, where decisions highly impact people's lives [2, 3].


The concept of explainable AI (XAI) is not new and has been used since the beginnings of artificial intelligence. There have been efforts to clearly define XAI terminology, distinguishing concepts such as transparency, interpretability and explainability [2]. Explainability approaches allow users to have a clearer understanding of why certain AI predictions were made, which may help increase their trust and acceptance in these predictions.

Explanations are usually divided into two categories: post-hoc and ante-hoc explanations. Ante-hoc systems are interpretable by design, which includes decision trees and linear regression.

VOILA! 2022: 7th International Workshop on Visualization and Interaction for Ontologies and Linked Data, October 23, 2022, Virtual Workshop, Hangzhou, China, Co-located with ISWC 2022.



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

Post-hoc systems, on the other hand, suggest possible explanations for specific predictions made by ML models, maintaining high fidelity to the original model and producing helpful explanations instead of trying to explain the original model itself [4]. An additional dimension of explainability is how the model's predictions fit with prior knowledge. This is especially relevant in areas where the body of knowledge cannot be fully understood by users due to its size and complexity, with a prime example being the clinical domain [5]. The need for knowledge representations to support AI in clinical and biomedical settings is recognized by the scientific community, but their effective use is still an open challenge [4].

Ontologies and Knowledge Graphs (KGs) afford a potential solution to this need [3]. KGs are graph-based representations of knowledge that use nodes to represent entities and edges to represent relations. The possible types of elements and relations can be described by an ontology [6]. The structured and connected form of modelling domains in an ontology allows a facilitated integration, as well as an extensive vocabulary and clear identifiers [7], which allows a shared common knowledge [8].

The abundance and diversity of ontologies and KGs in the biomedical and clinical domains are an opportunity that can be explored by XAI. One particular area where explanations are crucial is personalized medicine. Personalized medicine tries to answer the question of "*What is the right drug at the right dose for the right patient?*" by integrating and analysing very large volumes of diverse and heterogeneous data coming from a variety of sources and different scientific and clinical domains. Black-box algorithms are leading in the field [9], but their opaque nature is a recognized challenge. Moreover, a lack of consideration of users' expectations is among the chief reasons for the limited adoption of ML systems in critical and complex domains [10], so effective user interfaces are also a requirement [4]. If KGs are to be the backbone of an explanation for an AI prediction, then the visualization of the KG can be the communication means of such an explanation. Ontology and KG visualization are active research areas, with several existing tools, but the challenges of visualizing large graphs and adapting to specific use cases remain [11].

We are then faced with two challenges: (1) Can KGs be used to craft semantic explanations of how a particular AI prediction fits with existing knowledge?; (2) Are KG-based visualizations an effective means to communicate such an explanation? In this work, we focus on the second challenge, building on the following definition of a semantic explanation: an explanation for a specific AI prediction for a given instance that corresponds to a subgraph extracted from the KG that includes a representation of the instance, the prediction and a path in the KG that connects both. This type of explanation requires that both the instance for which the prediction was made and the prediction itself to be encoded in the KG. To address this challenge, we require a tool that enables the visualization of a semantic explanation, and a user study that compares this visualization with other communication approaches, e.g. text-based.

The main goal of this work is to develop a visualization tool, that given a semantic explanation of how a particular AI prediction fits with prior knowledge represents this explanation in a visual manner. The main contributions of this work are: (1) the extension of the VOWL language to represent additional KG elements required for semantic explanations; (2) the adaptation of the WebVOWL tool to represent semantic explanations; (3) the development of representative semantic explanations to evaluate the tool; (4) the design of a user study; (5) a small-scale user study.

2. Related Work

This work is related to two complementary domains: the use of KGs for XAI and the visualization of KGs. Below we provide a brief overview of relevant works in these domains.

KGs for XAI We focus on the exploration of KGs for post-hoc explanations both in supervised learning and pattern mining settings. Explanations that use background knowledge are likely to be closer to human conceptualizations and thus more useful in applications.

Trepan Reloaded [12] has been a recent extension of Trepan [13], an algorithm that creates a decision tree that tries to replicate deep neural network model predictions and employs ontologies to select the most general concepts, determined through the hierarchy of the ontology, to then be used as tree nodes. The authors consider that the more general concepts will provide the most understandable explanation, which is a reasonable although semantically poor criterion.

Lécué and Wu [14] developed a method that uses ontologies to help explain predictions of classification models. It selects representative data points and their semantic context is then built by characterizing them with their respective concepts using an ontology. The concepts can then be divided into positive concepts, if they characterize points in a certain class, and negative concepts if they describe points in the opposite class. An algorithm is then used to select the most useful positive and negative concepts for explaining each class, which are preferably the more general ones. This results in a list of ordered informative explanations, which are based on the contrasting concepts of each class.

Ontologies can also help in filtering and organizing results of pattern mining techniques. Jay and D'Aquin (2013) developed a tool to interpret results obtained from data mining with the use of Linked Open Data (LOD) [15]. Their approach is applied to results from pattern mining techniques, which are sequential patterns regarding hospitalized patients' trajectories. This approach makes use of linked data to extract information about the result patterns and to organize them in a hierarchical way. The tool also allows the linkage of the patterns to their terminology, making their interpretation of patterns easier.

These representative works differ from ours in their definition of an explanation. In [12], an explanation is the model itself, which is built with input from an ontology. In [14], the explanations are the most common classes that represent positive and negative examples. In [15] the explanations are the semantic representations of the extracted patterns. In all works, the presentation of the explanations is addressed, however both [12] and [14] disregard the contextual and semantic properties of the ontology they explore to generate the explanations when presenting them to users. [12] presents the decision tree where although nodes correspond to ontology classes, their semantic properties are ignored, while [14] merely proposes to present a list of relevant classes, without providing any other semantic information. On the other hand, [15] explore the semantic properties of the linked data and ontologies they use, allowing patterns to be navigated according to the ontology hierarchy.

Visualization of ontologies and KGs The majority of tools to visualize ontologies employ two-dimensional node-link visualizations with a focus on class hierarchies and are rarely use case oriented [11]. 10 out of the 33 tools surveyed by [11] are plugins for the popular ontology editor Protégé [16]. Protégé affords a visualization of an ontology as an indented list, but a variety of plugins cover other layouts, such as trees and graphs (e.g., OWLViz [17], OntoGraf [18]). Other popular tools are browser-based. Ontodia [19] supports quick visualization of RDF

datasets and OWL ontologies on the Web. WebVOWL [20] uses VOWL (Visual Notation for Ontologies) to support web-based ontology visualization aiming at a better and more intuitive experience for the user. VOWLMap [21] targets the visualization of ontology alignments. Graph databases that include visualization interfaces can also be used to visualize KGs, such as Gruff for AllegroGraph ¹.

Orthogonally, a recent study [22] performed a comparative evaluation of state-of-the-art linked data visualization tools based on a number of use cases including the ability to *visualize the paths that connect different instances*. Only one of ten tools was able to accomplish this use case [23], however it failed on other relevant use cases such as visualizing the information related to a class or a property.

There is not one-size-fits all solution to the problem of ontology and KG visualization and it is clear that different use cases demand different visualization and interaction techniques. For the specific use case of semantic explanation visualization no existing tool is capable of answering all requirements.

3. Methodology

Figure 1 represents two semantic explanations that illustrate the fit between prior knowledge encoded in the KG and the AI prediction. In this case, the instance is John Doe and the predicted drug to treat this patient's disease is Sunitinib, an antineoplastic agent, and they are connected through two paths that link the semantic representation of John Doe to the semantic representation of Sunitinib. These paths provide two possible explanations of why Sunitinib was predicted for this patient, one of them more generic (grey) and one more specific (colors). The generic explanation states that John Doe has a mutation MET T540 that is related to renal cell carcinoma, which is a type of cancer and cancers can be treated by the antineoplastic agents, of which Sunitinib is an example. The more specific explanation declares that the patient has a specific mutation MET T540 that promotes the transcription of the MET gene that is related to tyrosine kinase activity which is inhibited by Sunitinib. Both explanations are valid, but the specific one provides more information to understand the possible link between a patient feature (the mutation) and the drug effect.

To visualize semantic explanations we need to fulfil the following requirements:

1. load a semantic explanation, i.e., a KG subgraph
2. visualize the instance and its properties, i.e, the KG individual for whom the AI prediction was made
3. visualize the predicted class and its properties, i.e., the KG class that represented the predicted class
4. visualize the path between instance and predicted class composed by individuals, classes and properties

¹<https://allegrograph.com/products/gruff/>

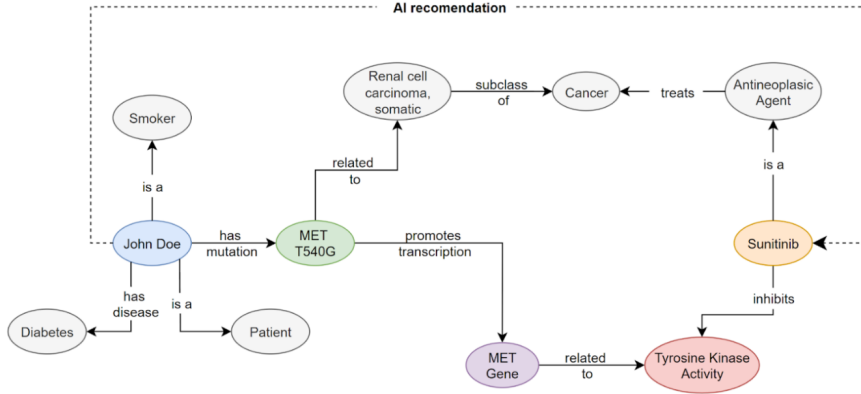


Figure 1: Example of two semantic explanations for the recommendation of sunitinib to patient John Doe. A more specific semantic explanation is represented in color and a more generic one represented in grey.

5. expand the neighbourhood of nodes in the path to include neighbouring regions of the KG

The visualization of ontologies can be supported by visual languages, such as VOWL [24] and its associated visualization tool WebVOWL [25]. In this work, we extended VOWL to support the visualization of individuals and adapted WebVOWL to the visualization of semantic explanations.

3.1. Extending VOWL

The VOWL notation was extended to represent new elements required for the visualization of individuals and their properties, as presented in Table 1. We defined the representation of Named Individuals and their relations (both the “Instance of” relation to their corresponding class, as well as the object properties that connect the individuals between themselves).

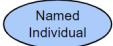
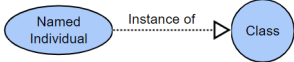
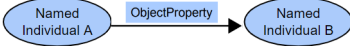
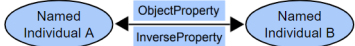
3.2. VOWLExplain

VOWLExplain was developed over WebVOWL. Previous works have already demonstrated the potential to adapt WebVOWL to develop new visualization tools adapted to specific tasks [21].

WebVOWL takes as input a JSON file with the desired ontology. This JSON file has a specific format that describes the different elements to be interpreted and represented by WebVOWL and is generated by the OWL2VOWL tool. However, this tool does not allow the representation of individuals, so we developed a tool to process the KG subgraph and generate a JSON representation that follows the structure of WebVOWL but contains the extensions required to represent individuals. We then modified the WebVOWL code to include the representation of the new VOWL elements. The first part of this adaptation included recognizing and processing these elements from the JSON file. Then, we guaranteed their accurate representation, with the new VOWL notation, by creating the new graphical elements

Table 1

Extension of the OWL notation.

VOWL Element	Notation
Individual	
Instance of	
Object Property	
Object Property and Inverse Property	

in the VOWLExplain code. The addition of the new OWL elements also included the adaptation of all of the useful original features of WebVOWL for each element, such as moving, selecting, and showing the details in the lateral menu: Name, Type, other characteristics, and domain and range (in case of relations). The overall appearance and functionalities remained the same, with the addition of the new OWL elements, as well as a new feature for collapsing and expanding the neighborhood of nodes, in order to facilitate the visualization of the explanation paths.

3.3. Evaluation strategy

The evaluation of VOWLExplain was grounded in the specific case of personalized medicine for renal cancer².

Ontologies, Data and Explanations To evaluate VOWLExplain we built a KG based on a network of aligned ontologies and simulated drug recommendations based on data describing real patients and the drugs that were used for their clinical case.

The ontology network comprises a set of 28 biomedical ontologies aligned to each other to build the semantic backbone of the KG [26, 27]. The ontologies cover a wide range of domains, including clinical data, clinical trial data and 'omics data, such as immunopeptidomics and transcriptomics and proteomics.

The patient data (clinical features, gene mutations and administered drug) was obtained from The Cancer Genome Atlas (TCGA), which contains rich metadata, such as the clinical characterizations of patients, and transcriptomics data from the work by Braun *et al.* [28], which describes gene activity and mutations in renal cancer.

We developed semantic explanations for the drug recommendations for patients by creating paths in the KG between patient's gene mutations or clinical characteristics and the recommended drug using Protégé [16]. We created six semantic explanations, four which represented specific explanations where the mechanism of a genetic mutation and the effect of a drug are

²in the context of European Commission funded KATY project <https://katy-project.eu/>

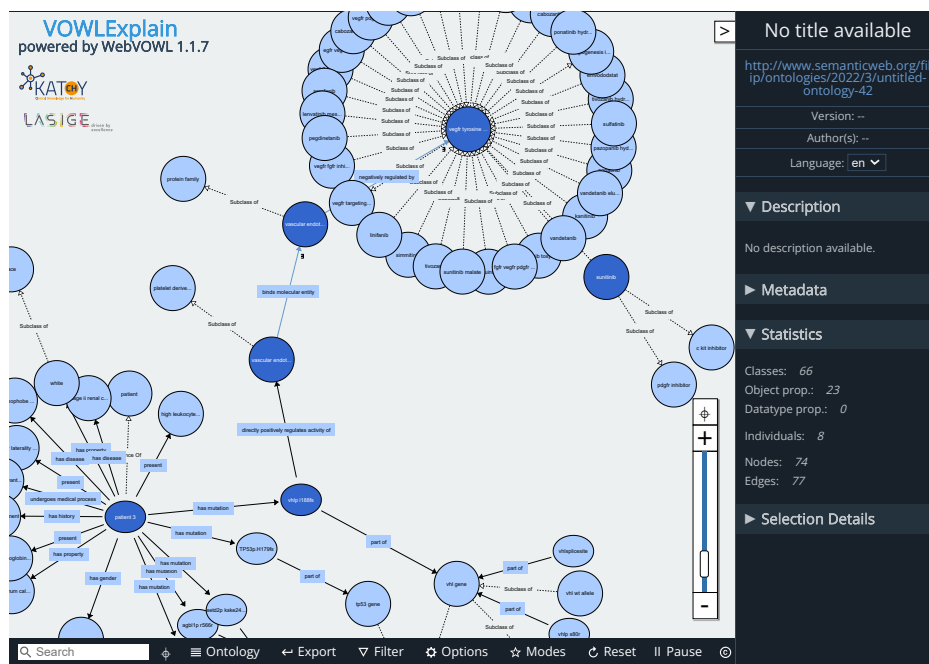


Figure 2: Example of semantic explanation loaded in VOWLExplain. The explanation shows a path (represented in dark blue) that connects the patient (patient 3) to the recommended drug (sunitinib), as well as some neighbors that provide context to the explanation (represented in light blue).

represented (an example of a specific graph-based explanation is presented in Figure 2) and two based on the generic anti-cancer effect of drugs (example of a generic text explanation is presented in Figure 3). Text explanations are simple transformations of class and property names in the explanation path into grammatically correct sentences.

Patient 2 has the disease Metastatic Renal Cell Carcinoma. Nivolumab is capable of inhibiting or preventing the pathological process of Metastatic Renal Cell Carcinoma.

Figure 3: Example of a text explanation presented in the user study. This is a generic explanation.

Ontologies, Data and Explanations We performed a preliminary user study, to gather feedback from a small pool of users, before embarking on a large-scale study. The goal of the user study was to evaluate the usefulness of visual semantic explanations and is based on comparing textual representations, handcrafted based on the semantic explanations and graph-based representations of semantic explanations using VOWLExplain.

We recruited four users with a background in health informatics. The study was both observational (online video call) and questionnaire based. The evaluation was task-based: users were given information about a patient and its corresponding AI recommendation (Figure 4) and then asked a number of questions about either a textual or a graph-based semantic explanation (SEQ) for the given patient and prediction:

Patient 3	
Clinical Data	Male Birth Year: 1935 White Number of packs smoked a year: 48 Renal cell carcinoma, chromophobe type No neoadjuvant therapy given No prior treatment High Leukocyte count Low hemoglobin and serum calcium levels Stage II renal cell cancer Tumor Laterality Left Diagnosed at 71
Mutated Genes	VHL gene AGBL1 gene DST gene SETD2 gene TP53 gene
Prediction	Sunitinib

Figure 4: Example of table presented in the user studies with characterization of a patient and its corresponding AI recommendation.

- (SEQ1) Rate the explanation in terms of Correctness (I don't know or 1-Not at all to 4-Completely)
- (SEQ2) Rate the explanation in terms of Clarity (I don't know or 1-Not at all to 4-Completely)
- (SEQ3) Rate the explanation in terms of Trustworthiness (I don't know or 1-Not at all to 4-Completely)
- (SEQ4) How would you improve this particular explanation? (free text - optional)

After rating six different semantic explanations (three text and three graph-based), users were also asked the following general questions (GQ):

- (GQ1) Do you think explanations, either graph or text based, are useful? (1-Not useful 5-Very useful)
- (GQ2) Which explanations do you prefer? (1-Graph, 3 corresponds to no preference, 5-Text)
- (GQ3) Adding context to the explanations (neighborhood in VOWLExplain) is useful? (1-Not useful 5-Very useful)
- (GQ4) Any suggestions or comments to improve the graph-based explanations? (free text - optional)
- (GQ5) Any suggestions or comments to improve the VOWLExplain tool? (free text - optional)
- (GQ6) Any suggestions or comments to improve the textual explanations? (free text - optional)
- (GQ7) Any suggestions or comments to improve the explanations, overall? (free text - optional)

Users' screens were recorded while using VOWLExplain to elucidate which features were used. Half the users were first presented with textual explanations followed by graph explanations, and the other half vice-versa. Users were never shown the same explanation in both forms.

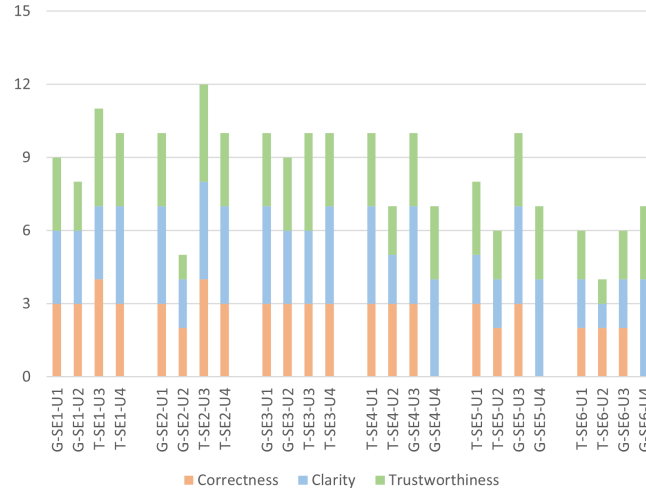


Figure 5: User (U) ratings for each semantic explanation (SE) in graph (G) or text (T) format, in terms of correctness, clarity and trustworthiness

4. Results and Discussion

4.1. VOWLExplain

Figure 2 depicts a semantic explanation loaded in VOWLExplain. It presents the path that connects the patient to the predicted drug, as well as the neighborhood of this path, for more context.

4.2. Preliminary User Studies

All users rated explanations as very useful (rating=5), regardless of type, all showed a slight preference for graph explanations (rating=2), and rated the context of graph explanations as useful (rating=4) or very useful (rating=5). However, the specific ratings on Correctness, Clarity and Trustworthiness reflect different opinions (see Figure 5). This preliminary user study was helpful in understanding the diversity of how users perceive explanations. For instance, U4 struggles with rating the correctness of all graph-based explanations (rating=I don't know), but shows no hesitancy in rating textual explanations. Moreover, U4 rates all explanations regardless of type with the same level of clarity and trustworthiness. On the other hand, U2 rates graph-based explanations generally higher in terms of clarity and trustworthiness, except for the generic explanation (G-SE2) which is rated considerably lower on par with the generic textual explanation (T-SE6). U1 rates graph-based explanations generally higher, while U3 rates the textual ones higher. Looking at the profiling information, these preferences make sense, since U3 rates their knowledge of KGs as *Novice* while U1 rate themselves as *Competent*.

In Table 2, we can see the median of all the answers regarding the Correctness, Clarity and Trustworthiness of the explanations. The scores for Correctness and Trustworthiness are equivalent for both text and graph visualization based explanations, but in Clarity the graph

Table 2

Median scores for correctness, clarity and trustworthiness of explanations.

Explanation Type	Correctness	Clarity	Trustworthiness
Graph Visualization	3	4	3
Text	3	3	3

explanations received a higher median score. It is possible the perceived increase in clarity comes with the additional context that the KG visualization affords.

Users provided some suggestions for improvements, such as fitting the entire explanation path on the screen and marking the instance and the prediction nodes with a different color.

5. Conclusions

In recent years, ontologies and KGs have been proposed as a fundamental piece of the XAI puzzle. In complex and critical domains, such as healthcare, they are widely recognized as essential [29]. This work presented VOWLExplain, a tool for visualizing semantic explanations for AI predictions that are based on elucidating how a prediction fits with existing knowledge encoded in the KG. A small-scale user study comparing text representations of semantic explanations with graph-visualization representations revealed a diversity of user perceptions, and although users stated a preference for the graph-based visualization, they did not rate them as more correct or trustworthy than text based ones. They did rate them as generally more clear. The user study also highlighted some limitations of VOWLExplain, including the lack of distinct representations for instance and prediction. In future work, we will address user suggestions and also integrate text explanations into VOWLExplain by exploring tools that translate OWL constructs into natural language such as NaturalOWL [30]. We believe presenting both types of explanation will make the tool more versatile and easier to pick up for users without familiarity with ontologies or KGs. We will also conduct a larger-scale user study, with users recruited from both clinical, biomedical and health informatics backgrounds.

Data and Source Code Availability

Data, code, video tutorial and user study form are openly available at: <https://github.com/liseda-lab/VOWLExplain>

Acknowledgements

This work was partially supported by the KATY project funded the European Union’s Horizon 2020 research and innovation program (GA 101017453). It was also supported by FCT through the LASIGE Research Unit (UIDB/00408/2020 and UIDP/00408/2020). The authors are also grateful to Daniel Faria for the discussions and advice on transforming OWL into a JSON representation.

References

- [1] S. Min, B. Lee, S. Yoon, Deep learning in bioinformatics, Briefings in bioinformatics 18 (2017) 851–869. doi:10.1093/bib/bbw068. arXiv:1603.06430.
- [2] M. A. Clinciu, H. F. Hastie, A survey of explainable AI terminology, NL4XAI 2019 - 1st Workshop on Interactive Natural Language Technology for Explainable Artificial Intelligence, Proceedings of the Workshop (2019) 8–13. doi:10.18653/v1/w19-8403.

- [3] S. Chari, D. M. Gruen, O. Seneviratne, D. L. McGuinness, Foundations of Explainable Knowledge-Enabled Systems (2020). [arXiv:2003.07520v1](#).
- [4] A. Holzinger, C. Biemann, C. S. Pattichis, D. B. Kell, What do we need to build explainable AI systems for the medical domain? (2017) 1–28. URL: <http://arxiv.org/abs/1712.09923>. [arXiv:1712.09923](#).
- [5] K. N. Vokinger, S. Feuerriegel, A. S. Kesselheim, Mitigating bias in machine learning for medicine, *Communications medicine* 1 (2021) 1–3.
- [6] Heiko Paulheim, Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods, *Semantic Web* 8 (2016) 489–508.
- [7] J. J. Cimino, X. Zhu, The practical impact of ontologies on biomedical informatics., *Yearbook of medical informatics* (2006) 124–135. doi:10.1055/s-0038-1638470.
- [8] R. Hoehndorf, P. N. Schofield, G. V. Gkoutos, The role of ontologies in biological and biomedical research: A functional perspective, *Briefings in Bioinformatics* 16 (2015) 1069–1080. doi:10.1093/bib/bbv011.
- [9] N. J. Schork, Artificial intelligence and personalized medicine, in: *Precision medicine in cancer therapy*, Springer, 2019, pp. 265–283.
- [10] C. J. Cai, E. Reif, N. Hegde, J. Hipp, B. Kim, D. Smilkov, M. Wattenberg, F. Viegas, G. S. Corrado, M. C. Stumpe, et al., Human-centered tools for coping with imperfect algorithms during medical decision-making, in: *Proceedings of the 2019 chi conference on human factors in computing systems*, 2019, pp. 1–14.
- [11] M. Dudáš, S. Lohmann, V. Svátek, D. Pavlov, Ontology visualization methods and tools: a survey of the state of the art, *The Knowledge Engineering Review* 33 (2018).
- [12] R. Confalonieri, T. Weyde, T. R. Besold, F. Moscoso del Prado Martín, Using ontologies to enhance human understandability of global post-hoc explanations of black-box models, *Artificial Intelligence* 296 (2021) 103471. URL: <https://doi.org/10.1016/j.artint.2021.103471>. doi:10.1016/j.artint.2021.103471.
- [13] M. W. Craven, J. W. Shavlik, Extracting Tree-Structured Representations of Trained Networks, *Advances in Neural Information Processing Systems* 8 (1996) 24–30. URL: citeseer.ist.psu.edu/craven96extracting.html.
- [14] F. Lecue, J. Wu, Semantic Explanations of Predictions (2018). URL: <http://arxiv.org/abs/1805.10587>. [arXiv:1805.10587](#).
- [15] N. Jay, M. D'Aquin, Linked data and online classifications to organise mined patterns in patient data., *AMIA ... Annual Symposium proceedings / AMIA Symposium*. AMIA Symposium 2013 (2013) 681–690.
- [16] N. F. Noy, R. W. Ferguson, M. A. Musen, The knowledge model of protégé-2000: Combining interoperability and flexibility, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 1937 (2000) 17–32. doi:10.1007/3-540-39967-4_2.
- [17] M. Horridge, Owlviz, Available on: <http://protegewiki.stanford.edu/wiki/OWLViz> (2010).
- [18] S. Falconer, Ontograf, Protégé Wiki (2010).
- [19] D. Mourmoutsev, D. S. Pavlov, Y. Emelyanov, A. V. Morozov, D. S. Razdyakonov, M. Galkin, The Simple Web-based Tool for Visualization and Sharing of Semantic Data and Ontologies., in: *ISWC (Posters & Demos)*, 2015.
- [20] S. Lohmann, S. Negru, F. Haag, T. Ertl, Visualizing ontologies with VOWL, *Semantic Web* 7 (2016) 399–419. doi:10.3233/SW-150200.
- [21] A. Guerreiro, C. Pesquita, D. Faria, VOWLMap: Graph-based Ontology Alignment Visualization and Editing ?, *CEUR Workshop Proceedings* 2980 (2021) 1–12.
- [22] F. Desimoni, N. Bikakis, L. Po, G. Papastefanatos, A comparative study of state-of-the-art linked data visualization tools, in: *5th International Workshop on Visualization and Interaction for Ontologies and Linked Data, VOILA 2020*, volume 2778, CEUR-WS, 2020, pp. 1–13.
- [23] P. Heim, S. Hellmann, J. Lehmann, S. Lohmann, T. Stegemann, Relfinder: Revealing relationships in rdf knowledge bases, in: *International Conference on Semantic and Digital Media Technologies*, Springer, 2009, pp. 182–187.
- [24] S. Lohmann, S. Negru, F. Haag, T. Ertl, VOWL 2: User-oriented visualization of ontologies, in: *International Conference on Knowledge Engineering and Knowledge Management*, Springer, 2014, pp. 266–281.
- [25] S. Lohmann, V. Link, E. Marbach, S. Negru, Webvowl: Web-based visualization of ontologies, in: *International Conference on Knowledge Engineering and Knowledge Management*, Springer, 2014, pp. 154–158.
- [26] M. C. Silva, D. Faria, C. Pesquita, Integrating Knowledge Graphs for Explainable Artificial Intelligence in Biomedicine (2021). [arXiv:2003.07523](#).
- [27] M. C. Silva, D. Faria, C. Pesquita, Matching multiple ontologies to build a knowledge graph for personalized medicine, in: *European Semantic Web Conference*, Springer, 2022, pp. 461–477.

- [28] D. A. Braun, Y. Hou, Z. Bakouny, M. Ficial, M. Sant'Angelo, J. Forman, P. Ross-Macdonald, A. C. Berger, O. A. Jegede, L. Elagina, et al., Interplay of somatic alterations and immune infiltration modulates response to pd-1 blockade in advanced clear cell renal cell carcinoma, *Nature medicine* 26 (2020) 909–918.
- [29] C. Pesquita, Towards semantic integration for explainable artificial intelligence in the biomedical domain., in: *HealthInf Conference - BIOSTEC 2021*, volume 5, 2021, pp. 747–753.
- [30] I. Androutsopoulos, G. Lampouras, D. Galanis, Generating natural language descriptions from owl ontologies: the naturalowl system, *Journal of Artificial Intelligence Research* 48 (2013) 671–715.

Toward a Comparison Framework for Interactive Ontology Enrichment Methodologies

Jarno Vrolijk^{1,*}, Ioannis Reklós², Mahsa Vafaie³, Arcangelo Massari⁴,
Maryam Mohammadi⁵ and Sebastian Rudolph⁶

¹University of Amsterdam (UvA), Plantage Muidergracht 12, Amsterdam, 1018 TV, Netherlands

²Department of Informatics, King's College London, Bush House, 30 Aldwych, London, WC2B 4BG, United Kingdom

³FIZ Karlsruhe - Leibniz Institute for Information Infrastructure, Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany

⁴Research Centre for Open Scholarly Metadata, Department of Classical Philology and Italian Studies, University of Bologna, Bologna, Italy

⁵Department of Advanced Computing Sciences, Faculty of Science and Engineering, Maastricht University, Minderbroedersberg 4-6 6211 LK Maastricht, The Netherlands

⁶Computational Logic Group, TU Dresden, 01062 Dresden, Germany

Abstract

The growing demand for well-modeled ontologies in diverse application areas increases the need for intuitive interaction techniques that support human domain experts in ontology modeling and enrichment tasks, such that quality expectations are met. Beyond the correctness of the specified information, the quality of an ontology depends on its (relative) completeness, i.e., whether the ontology contains all the necessary information to draw expected inferences. On an abstract level, the Ontology Enrichment problem consists of identifying and filling the gap between information that can be logically inferred from the ontology and the information expected to be inferable by the user. To this end, numerous approaches have been described in the literature, providing methodologies from the fields of Formal Semantics and Automated Reasoning targeted at eliciting knowledge from human domain experts. These approaches vary greatly in many aspects and their applicability typically depends on the specifics of the concrete modeling scenario at hand. Toward a better understanding of the landscape of methodological possibilities, this position paper proposes a framework consisting of multiple performance dimensions along which existing and future approaches to interactive ontology enrichment can be characterized. We apply our categorization scheme to a selection of methodologies from the literature. In light of this comparison, we address the limitations of the methods and propose directions for future work.

Keywords

Formal Semantics, Ontology Design, Human-Computer Interaction, Knowledge Elicitation

VOILA! 2022: 7th International Workshop on Visualization and Interaction for Ontologies and Linked Data, October 23, 2022, Virtual Workshop, Hangzhou, China, Co-located with ISWC 2022.


*Corresponding author.

✉ j.vrolijk@uva.nl (J. Vrolijk); ioannis.reklos@kcl.ac.uk (I. Reklós); mahsa.vafaie@fiz-karlsruhe.de (M. Vafaie); arcangelo.massari@unibo.it (A. Massari); m.mohammadi@maastrichtuniversity.nl (M. Mohammadi); sebastian.rudolph@tu-dresden.de (S. Rudolph)

DOI 0000-0003-0409-4924 (J. Vrolijk); 0000-0002-2747-579X (I. Reklós); 0000-0002-7706-8340 (M. Vafaie); 0000-0002-8420-0696 (A. Massari); 0000-0003-4850-8068 (M. Mohammadi); 0000-0002-9421-8566 (S. Rudolph)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

1. Introduction

In practical knowledge management scenarios, ontologies need to be modified and updated on a regular basis. It is therefore important to aid human domain experts in exploring, understanding, and modifying domain-specific ontologies [1]. Providing human domain experts with intuitive interaction techniques can significantly support comprehension and adaptation of domain representations, ultimately resulting in higher quality ontologies. However, there is no *one-size-fits-all* solution; rather, different use cases demand different interaction techniques to foster user engagement and deliver better performance.

From a general logical perspective, an ontology can fail to meet requirements in two different ways. First, an ontology can contain wrong information (correctness). Second, an ontology can lack information (completeness). In incorrect ontologies, wrong conclusions may be derived, while in incomplete ontologies, valid conclusions may be missed. For example, it was shown that semantically-enabled querying of PubMed¹ using MeSH² with one piece of background knowledge removed would lead to a 55% drop in the result [2].

Ontology enrichment addresses the incompleteness problem. We define Ontology Enrichment to be the procedure that enables the addition of novel or missing relations, concepts and rules to an existing ontology [3]. The identified missing information is represented by the set of missing axioms that are correct according to the human and should be added to the ontology. Also, we define the human domain expert to be equivalent to the *limited all-knowing oracle* as defined by Lambrix [2], i.e., the expert knows part of the domain well, however, it may not know the answer to all questions.

In this paper, we focus on designing a comparison framework for interactive ontology enrichment methodologies. In particular, we focus on four different dimensions, namely, expressiveness, comprehensiveness, initiative, and scalability, based on which we can categorize existing interaction techniques. Our aim is to compare the possibilities the fields of Formal Semantics and Automated Reasoning have to offer for the interaction between human domain expert and ontologies. We argue that by involving the user and eliciting their knowledge, we can improve ontologies and expand them to include missing inferences. We demonstrate the usefulness of our comparison framework by evaluating four different ontology enrichment methods that elicit knowledge from human domain experts, and we underline the characteristics of these methods on the proposed dimensions. Addressing these points allows us to outline some of the remaining issues and open questions that implementations of formal methodologies face.

This paper is organized as follows: In Section 2, we define qualitative metrics for comparison of ontology enrichment approaches. In Section 3, we define and explain a selection of existing approaches and their contributions towards ontology enrichment through Formal Semantics. In Section 6, we summarize the characteristics of each of the approaches along the four dimension in a table. Lastly, in Section 7, we discuss the limitations of these approaches and suggest directions for future work.

¹<https://pubmed.ncbi.nlm.nih.gov/>

²<https://www.nlm.nih.gov/mesh/meshhome.html>

2. Comparison Framework

The process of ontology enrichment is essentially a continuous interaction between humans and machines. Therefore, we propose scoping the *repairing* phase as defined by Lambrix [2] to better fit this continuum. First, we argue that the term *expansion* better fits the concept than *repairing*, since *repairing* implies there is something broken; whereas, we are solely focusing on missing information to draw inferences from, rather than correcting incorrect axioms. Furthermore, to draw a line between the different tasks in the enrichment process, and as such, the interaction between human domain expert(s) and the machine (i.e. reasoners), we added the *validation* phase to better follow this interplay. In short, we believe elicitation to consist of the following steps:

1. **Detection:** Identifying which expected inferences are missing;
2. **Expansion:** Updating the existing knowledge base by adding axiom(s);
3. **Validation:** Checking the consistency of the added axiom(s) with the rest of the knowledge base.

A comparative analysis of the different techniques in order to get a better grasp of the pros and cons of each methodology is better enabled with the introduction of specific metrics. For this comparison, we defined the following dimensions:

1. **Expressiveness:** To what extent is the technique able to represent the breadth of the human domain expert's ideas (i.e. what constraint does the technique impose on the human domain expert's comprehensiveness).
2. **Comprehensiveness:** The degree to which the technique is capable of finding all the missing expected inferences (i.e. the expected inferences of the human domain expert) in interaction with the human domain expert.
3. **Initiative:** The degree to which the input requested from the human domain expert is pre-determined (i.e. initiated and put forward by the human domain expert vs. governed by the technique).
4. **Scalability:** What are the possible complexities with regard to scaling the methodology.

3. Formal Semantic Methods

In this section, we compare and contrast different interaction techniques which can be used to elicit knowledge from human domain experts, to enrich existing knowledge bases. Specifically, we examine the methodologies referred to as *Abductive Completion* [4], *Reasoning-Supported Interactive Revision of Knowledge bases* [5], *Advocatus Diaboli* [6] and *Relational Exploration* [7].

The **Abductive Completion** method (see Fig. 1) is based on abductive reasoning over on-

tologies [4]. In this method the domain expert is iteratively prompted to provide inferences that they expect of the ontology. If the inference cannot be entailed from the ontology, the reasoner suggests expansions of the ontology that would entail the desired consequence. The domain expert is then asked to select the most appropriate enrichment according to their domain knowledge and that axiom is added to the ontology. Through this method, the knowledge of the domain expert is elicited both when the domain expert provides the expected inference, as well as when the domain expert selects the most appropriate expansion of the ontology.

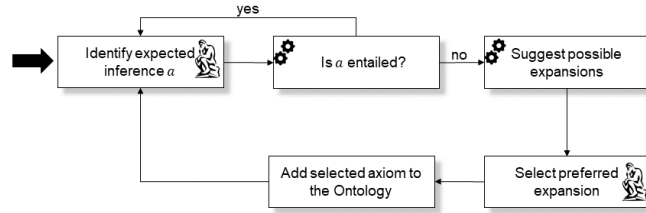


Figure 1: A flowchart of the *Abductive Completion* method. The cog icon denotes involvement of the reasoner and the human figure denotes involvement of the domain expert.

Reasoning-Supported Interactive Revision of Knowledge Bases (RSIR of KB) [5], as seen in Fig. 2, supports ontology revision based on logical criteria. In this approach, a set of candidate axioms are provided, from which the axiom that allows the automatic evaluation of the highest number of unevaluated axioms, i.e., the most *impactful* axiom, is presented to the domain expert. If the expert accepts the axiom, it is added to the knowledge base (i.e. the knowledge base is enriched). Otherwise, the axiom is added to an unintended consequence set.

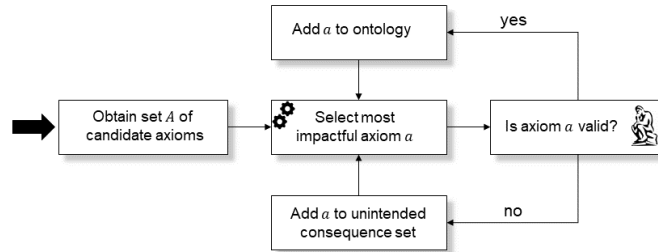


Figure 2: A flowchart of the *Reasoning-Supported Interactive Revision of Knowledge Bases* method. The cog image denotes reasoner involvement and the human figure denotes expert involvement.

The **Advocatus Diaboli** methodology [6] (see Fig. 3) introduces a system that allows domain experts to enrich an ontology by adding negative constraints, which are often overlooked despite their effectiveness in causing inconsistencies, finding modeling errors [8], repairing the mapping between ontologies [9], and iteratively revising ontologies [10]. The main idea behind the *Advocatus Diaboli* methodology is to allow the domain expert to show that the given ontology is underconstrained by actively constructing class expressions that are satisfiable according

to the current ontology³, but impossible according to the expert's knowledge. Following this process, domain experts can add negative constraints which invalidate the impossible class expressions and thus, make the ontology more complete.

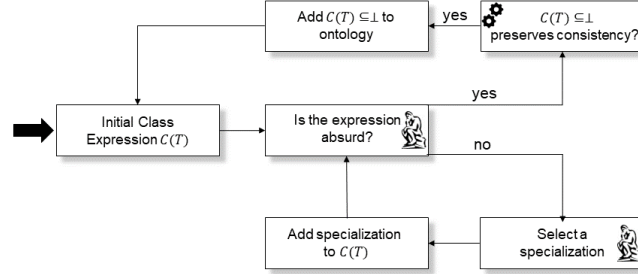


Figure 3: A flowchart of the *Advocatus Diaboli* method. The cog image denotes reasoner involvement and the human figure denotes expert involvement.

The **Relational Exploration** approach [7] (see Fig. 4) describes a formal process that aims to produce complete domain specifications by iteratively generating hypotheses which are processed by a reasoner that evaluates if they are entailed or rejected based on the existing ontology. If a generated hypothesis is not entailed or rejected, it is then presented as a question to a domain expert who either accepts or rejects it. This methodology ensures that, upon completion, the resulting domain specification is complete and that the domain expert never has to answer redundant questions, thus, minimizing the burden placed on them. The knowledge elicited from the domain experts results in the enrichment of incomplete ontologies.

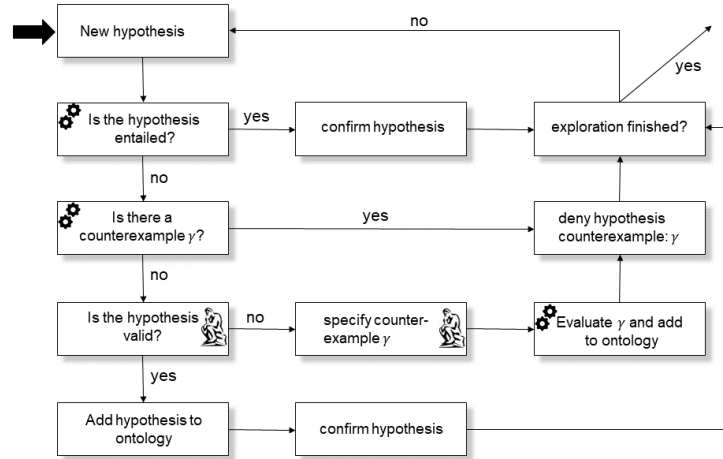


Figure 4: A flowchart of the *Relational Exploration* method. The cog image denotes reasoner involvement and the human figure denotes expert involvement.

³Preservation of satisfiability is ensured by the way the class expressions are constructed in a navigation-like process similar to faceted browsing

4. Results

In this section, we utilize our proposed framework for comparison of four ontology enrichment methodologies that elicit domain expert knowledge through structured interaction with a human. The results of this comparison are shown in Table 1.

Table 1

Table comparing the different methods for elicitation of knowledge from human domain experts. Human domain experts are denoted as H, and calls to the reasoner as R.

	Expressiveness	Comprehensiveness	Initiative	Scalability
Abduction	• Depending on given set of abducibles	• No completeness guarantee.	• Prio H > R. • H provides required inference.	• R at each step.
RSIR of KB	• class hierarchy and disjointness	• No completeness guarantee.	• Prio R > H • H provides required inference.	• Ranking limits R. • R at initialization. • R after H. • Patronizes H
Advocatus Diaboli	• Negative constraints wrt. cognitively plausible class expressions	• Every reachable situation is possible. • All possible situations are reachable. • Complex quantifiers not possible. • No completeness guarantee.	• Prio on H > R. • H can explore worlds. • H can exclude worlds.	• Expensive for H. • R at initialization. • R at each navigation step.
Relational Exploration	• General class inclusions with respect to a specified logical fragment.	• Terminates after finite steps. • Completeness upon termination. • H can stop it beforehand.	• Prio on R > H. • H if and only if R fails. • H can specify counter-examples. • H can complete assertions.	• R at initialization. • R after H. • Patronizes H • R for counter-examples.

5. Discussion

Of the four methods discussed in this work, the only two that allow situations/hypotheses to be generated dynamically are the *Advocatus Diaboli* and the *Relational Exploration* methods. Because of this, they are the most comprehensive ones and they have the potential to identify the highest number of missing expected inferences through interaction with the human expert. Indeed, the *Relational Exploration* method guarantees the completeness of the knowledge base upon completion. While this guarantee serves to highlight the comprehensiveness of the method, it remains theoretical since it may require the human expert to answer exponentially many questions before completion. As such, in real-world applications, we can expect both methods to perform similarly in terms of comprehensiveness, with the *Advocatus Diaboli* methodology being better at allowing the expert to guide the process toward the situations that are of interest to them, whereas the *Relational Exploration* methodology has the ability to automatically generate new hypotheses that the domain expert may not have thought of.

The interaction techniques use a variety of ways to represent the breadth of the human domain expert's ideas. For example, both *RSIR of KB* and *Advocatus Diaboli* use class expressions to turn axioms unsatisfiable if their consequence is unintended. However, the largest difference between them is on initiative and scalability (see Table 1 above). *Relational Exploration* deals only with conjunction on atomic classes. However, it is possible to leverage ontological background

by having complex definitions for named classes [11]. In contrast, Ferré and Rudolph [6] aid the human domain expert in the construction of intuitive satisfiable class expressions, which – if found to be absurd – can be turned unsatisfiable by adding a corresponding negative constraint to the knowledge base.

With respect to initiative, the methods shown in this work are evenly divided with the reasoner leading the process in *Advocatus Diaboli* and *RSIR of KB* and the expert initiating the process in *Abductive Completion* and *Relational Exploration*. An important drawback of the *Abductive Completion* method is that the expected inference must be provided by the domain experts, which places an undue burden on them, since they have to both generate the expected inferences and formulate them in formal logic. Similarly, the *Relational Exploration* method requires the user to input a counterexample for invalid hypotheses which assumes that the domain expert can not only identify the correct counterexample, but also describe it in logical formulae. Given that such familiarity with formal logic cannot be expected in most cases, these methods are prone to inserting wrong information in the ontology and deteriorating its quality. Furthermore, it is important to note that while *Relational Exploration* and *RSIR of KB* are similar in terms of the workflow, the major difference is that in *Relational Exploration* the axioms are not pre-specified but created on the go and therefore, the exploration may require exponentially many human decisions [5].

Heavy reliance on the reasoner at different stages in the elicitation process may negatively affect the scalability of the methodologies, making them unfit for larger knowledge bases. Vice versa, heavy reliance on the human domain expert will greatly reduce the efficiency and could potentially result in the loss of quality in the enrichment of the ontology.

The possible complexities with regard to scaling the methodology are largely intertwined with the number of calls to the reasoner. *RSIR of KB* is the only interaction technique that computes and updates decision spaces to bring down the the number of calls to the reasoner by up to 75% [10]. The axiom's *impact* as defined by Nikitina [10] determines a beneficial order of evaluation that none of the other interaction techniques use.

Naturally, the involvement of a human domain expert is required for the enrichment of ontologies; particularly, in the context of knowledge elicitation, where Formal Semantic methodologies are seldom enough for adequate representations. Yet, often formal reasoning can be leveraged to keep the necessary human interaction at a manageable level.

We argue that our preliminary results show that comparison over the four dimensions allows the identification of the strengths and weaknesses of each methodology. Furthermore, the comparison framework highlights the appropriate application scenario for each of the chosen methods. Additionally, our method facilitates evaluation, hence it helps create a movement toward more effective enrichment processes that allows users more utility using semantic and formal axiom enrichment methods. For example, creating better (i.e., more explainable) user interfaces to make the underlying mechanics more understandable to human domain experts unfamiliar with Formal Semantics.

All the studies reviewed so far, however, suffer from the fact that human domain experts could make mistakes in their assumptions of the domain knowledge, which can cause a loss of quality in the enrichment of the ontology. Likewise, unfamiliarity of the human domain experts with Formal Semantics and logical inferences could result in the enrichment of the ontology with false axioms.

The scope of this study was limited in terms of the compared methods (i.e., selected methods all focus on a subset of enrichment methods). The study does not consider the plethora of "newer" approaches that incorporate external resources through machine- and/ or deep learning (e.g., through recommendations based on natural language processing). However, we argue that the comparison framework as suggested in this study can also be applied to those.

6. Conclusion

In this position paper, we have reviewed a variety of methodologies for ontology enrichment through interaction with human domain experts. We have provided a comparative qualitative analysis on a selection of existing Formal Semantic techniques and their constituent phases (i.e. detection, expansion, and validation) on four dimensions; namely: i) Comprehensiveness, ii) Expressiveness, iii) Initiative, and iv) Scalability.

Involvement of human domain experts in ontology enrichment is required for the maintenance and upkeep of high quality ontologies. However, finding the correct interplay between formal reasoning and human involvement depends on the size of the ontology, the availability of resources, and the requirements of the use case. The task-specific nature of ontologies also forces certain constraints on the human domain experts i.e. the quality of the enrichment is directly intertwined with the domain knowledge of the human domain experts.

We further argue that the provided comparison framework can also help steer the movement towards a more effective enrichment process. Indicating that user interfaces can help improve the explainability of the underlying mechanics, and as such improve the quality of the interaction between the human domain expert and the ontology.

7. Future Work

In this paper, we focused on identification and definition of four dimensions for comparison of ontology enrichment methodologies. A natural progression of this work would be to develop quantitative measures, to increase robustness, for the introduced dimensions. Finally, more work needs to be done to link the Expressiveness, Comprehensiveness, Initiative, and Scalability dimensions to method performance.

In order to increase reliability and confidence in the quality of the ontology, we propose the creation and use of a collaborative framework in which multiple domain experts can communicate and share their understanding of the concepts and agree on conceptual models in the elicitation process. Furthermore, using an intermediate language such as Manchester OWL syntax [12] to translate syntactically challenging logical elements into a simplified version for human domain experts could improve the robustness of the knowledge elicitation process. Another interesting venue would be research into the different combinations of methods for the different steps of the enrichment process, as described in Section 2 of this work. Moreover, research into axiom ranking and axiom choosing strategies, as demonstrated in Nikitina et al. [5], can reduce the amount of manual effort and automated reasoning.

Using the evaluation dimensions described in this work, current and future ontology enrichment methodologies can be evaluated and scored. The scores obtained in each dimension will

highlight the strengths and weaknesses of the methodology and, by extension, the scenario that it is best suited for. As such, a web framework can be created where the domain experts can input the ontology that they desire to enrich and specify the importance of each dimension for their enrichment task using appropriate input methods e.g. a slider. The system can then automatically evaluate the ontology based on its entities, relations and other characteristics and suggest the most appropriate enrichment methodology for the task.

In the movement towards a more effective enrichment processes, improvements in the explainability of the underlying mechanics are of imminent importance. Implementations of the examined methodologies rely on keyboard and mouse input which may not be optimal. Therefore, further research in Human-Computer Interaction methodologies needs to be conducted to elucidate which interaction method is best for eliciting the required information from the domain expert while minimizing the burden placed on them.

References

- [1] P. Rodler, K. M. Shchekotykhin, P. Fleiss, G. Friedrich, RIO: minimizing user interaction in ontology debugging, in: ISWC (Posters & Demos), volume 914 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2012.
- [2] P. Lambrix, Completing and debugging ontologies: state of the art and challenges, CoRR abs/1908.03171 (2019).
- [3] G. Petasis, V. Karkaletsis, G. Paliouras, A. Krithara, E. Zavitsanos, Ontology population and enrichment: State of the art, in: Knowledge-Driven Multimedia Information Extraction and Ontology Evolution, volume 6050 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 134–166.
- [4] C. Elsenbroich, O. Kutz, U. Sattler, A case for abductive reasoning over ontologies, in: OWLED, volume 216 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2006.
- [5] N. Nikitina, S. Rudolph, B. Glimm, Reasoning-supported interactive revision of knowledge bases, in: IJCAI, IJCAI/AAAI, 2011, pp. 1027–1032.
- [6] S. Ferré, S. Rudolph, Advocatus diaboli - exploratory enrichment of ontologies with negative constraints, in: EKAW, volume 7603 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 42–56.
- [7] S. Rudolph, Relational exploration: combining description logics and formal concept analysis for knowledge specification, Ph.D. thesis, Dresden University of Technology, Germany, 2006.
- [8] P. Haase, F. van Harmelen, Z. Huang, H. Stuckenschmidt, Y. Sure, A framework for handling inconsistency in changing ontologies, in: ISWC, volume 3729 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 353–367.
- [9] C. Meilicke, H. Stuckenschmidt, A. Tamilin, Supporting manual mapping revision using logical reasoning, in: AAAI, AAAI Press, 2008, pp. 1213–1218.
- [10] N. Nikitina, Reasoning-Supported Quality Assurance for Knowledge Bases, Ph.D. thesis, Karlsruhe Institute of Technology, 2012.
- [11] J. Völker, S. Rudolph, Lexico-logical acquisition of OWL DL axioms, in: ICFCA, volume 4933 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 62–77.

- [12] M. Horridge, N. Drummond, J. Goodwin, A. L. Rector, R. Stevens, H. Wang, The manchester OWL syntax, in: OWLED, volume 216 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2006.

. Short paper

Multi-Level Visual Tours of Weather Linked Data

Nadia Yacoubi, Damien Graux and Catherine Faron

Inria, Université Côte d’Azur, CNRS, I3S, France

Abstract

The recent trend of adopting linked-data principles to integrate and publish semantically described open data using W3C standards has led to a large amount of available resources. In particular, meteorological sensor data have been uplifted into public weather-focused RDF graphs, such as WeKG-MF which offers access to a large set of meteorological variables described through spatial and temporal dimensions. Nevertheless, these resources include huge numbers of raw observations that are tedious to explore by lay users. In this article, we aim at providing them with visual exploratory “tours”, benefiting from RDF data cubes to present high-level aggregated views together with on-demand fine-grained details through a unified Web interface.

Keywords

Weather data, Spatio Temporal data, Visualisation, RDF Knowledge Graphs, SPARQL endpoints

1. Introduction

The recent trend of adopting linked-data principles to integrate and publish semantically described open data using W3C standards has led to a large amount of available resources. In particular, meteorological sensor data have been uplifted into public weather-focused RDF graphs, such as WeKG-MF graph which offers access to a huge number of sensor observations related to different weather variables, described through spatio-temporal dimensions. Hence, supporting lay users to browse, analyze, consume and reuse sensor data transformed and integrated into LOD datasets is challenging. In this article, we present the first release of a Web interface that enables users to visualize weather observational data at different levels of spatio-temporal granularity. We show how the WeKG-MF principles and the adoption of RDF data cubes can provide users with visual multi-level “tours”. Our main objective is to provide users with interactive exploration means to navigate the WeKG-MF, leveraging RDF data cubes to present high-level aggregated views as well as fine details on demand through a unified Web interface.

2. The WeKG Spatio-Temporal Model

In this section, we present the WeKG spatio-temporal model and use-cases identified in the context of the D2KAB research project which highlighted the need to build a knowledge graph from historical records published as open data by the French weather service provider Météo-France.

VOILA! 2022: 7th International Workshop on Visualization and Interaction for Ontologies and Linked Data, October 23, 2022, Virtual Workshop, Hangzhou, China, Co-located with ISWC 2022.


✉ nadia.yacoubi-ayadi@inria.fr (N. Yacoubi); damien.graux@inria.fr (D. Graux); catherine.faron@inria.fr (C. Faron)

🌐 <https://dgraux.github.io/> (D. Graux); <https://www.i3s.unice.fr/~faron/> (C. Faron)

🆔 0000-0002-6132-8718 (N. Yacoubi); 0000-0003-3392-3162 (D. Graux); 0000-0001-5959-5561 (C. Faron)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

SPARQL endpoint → https://weakg.i3s.unice.fr/sparql			
Number of RDF Triples	123.413.015	Weather stations	62
Total Observations	16.433.031	Weather properties	22
Observations per weather property	≈ 416.762	Meteorological features	6
		Links to Wikidata	92

Table 1

Key Statistics of the WeKG-MF Knowledge Graph.

2.1. From weather Observations archives to an RDF Knowledge graph

In our previous work [1], we have presented a self-contained semantic model that re-uses and extends standard ontologies, among which the GeoSPARQL ontology for spatial features and their relations [2], the Time ontology [3] for temporal entities, the Sensor, Observation, Sample, and Actuator (SOSA) [4] and Semantic Sensor Network (SSN) ontologies [5] for sensors and observations, and the RDF Data Cube ontology [6] for aggregation and multidimensionality features. The WeKG model captures the semantics of atomic and fine-grained weather observations by reusing and extending SOSA classes as well as spatio-temporal time series of aggregate values using the RDF vocabulary of the data cube. The proposed model is generic enough to be adopted and extended by meteorological data providers to publish and integrate their sources while complying with Linked Data principles.

Then, we built the WeKG-MF knowledge graph [7], based on this model, considering the open weather observations published by Météo-France¹. The SPARQL WeKG-MF endpoint allows users to retrieve weather observations recorded every 3 hours by different sensors hosted by weather stations and related to different parameters (air temperature, humidity, wind speed, precipitation, atmospheric pressure, etc.). The Table 1 summarises some key statistics of WeKG-MF. WeKG-MF includes meteorological data from the period 2016-2021 and is continuously evolving to include new, newer and older data. The knowledge graph is intended to serve different use case scenarios in several domains, including agriculture, biodiversity and climate studies.

2.2. Use Case Scenarios for WeKG-MF

WeKG-MF was initially created to answer expert's needs in the context of the D2KAB French project². A preliminary analysis revealed several competency questions that express the needs of experts to retrieve weather observations at different levels of granularity. For instance, an expert may be interested by the exact time of a day at which the minimum/maximum temperature was recorded and in this case, he is querying a fine-grained temporal entity represented by a `XSD:DATETIME` literal in WeKG-MF. In several other situations, experts are more interested in aggregate values of some weather parameters, such as the daily total precipitation, the number of days with precipitation greater than 1 mm over a time period, or the monthly mean values of maximum, minimum and mean temperatures.

To address these needs, we reused the RDF Data Cube Vocabulary (DCV) [6] to create multidimensional RDF slices, that are pre-calculated by fixing temporal and spatial dimensions and by applying aggregation functions such as min/max/avg/sum on fine-grained observation. Thus,

¹<https://www.meteofrance.com/>

²<http://www.d2kab.org>

```

1 SELECT distinct ?groupDate (SUM(?vp) as ?sum_precipitation) WHERE {
2   ?obs a weo:MeteorologicalObservation;          sosa:hasSimpleResult ?vp;
3   sosa:observedProperty wevp:precipitationAmount; sosa:resultTime ?date;
4   wep:madeByStation <http://ns.inria.fr/meteo/weatherstation/07434> .
5   BIND (day(?date) as ?day) BIND (month(?date) as ?month) BIND (year(?date) as ?year)
6   BIND (if (datatype(?year/4)=xsd:integer && ((?year/100)*100 != 0 ||
7     (?year/400)*400 = 0) , 1, 0) as ?bissexYear)
8   BIND ( if (?day = 1, if (?month in (1, 2, 4, 6, 8, 9, 11), 31,
9     if (?month in (5, 7, 10, 12), 30,
10      if (?bissexYear = 1, 29, 28))), ?day - 1) AS ?previousDay)
11  BIND (if (?day = 1, if (?month=1, 12, ?month - 1), ?month) as ?previousMonth)
12  BIND (if (?day = 1 && ?month=1, ?year - 1, ?year) as ?previousYear)
13  BIND ( xsd:date(if(hours(?date)<=6, concat(?previousYear, "-",
14    if (?previousMonth<10, concat("0", ?previousMonth), "-"),
15    if (?previousDay<10, concat("0", ?previousDay), ?previousDay)), concat(?year, "-",
16    if (?month<10, concat("0", ?month), ?month), "-"),
17    if (?day<10, concat("0", ?day), ?day)))) AS ?groupDate)
18 } GROUP BY ?groupDate ORDER BY ?groupDate

```

Figure 1: SPARQL Query for Daily values of Total Precipitation according to WMO documentation.

```

1 SELECT ?label_station ?date ?avg_temp WHERE {
2   { # Query weather stations located in "Nouvelle Aquitaine" region.
3     SELECT ?statURI ?label_station WHERE {
4       ?statURI a weo:WeatherStation; rdfs:label ?label_station .
5       dct:spatial [ wdt:P131 [rdfs:label ?label ; wdt:P2585 '75']] . }
6   }
7   # Query slices for each statURI.
8   VALUES ?year { "2021"^^xsd:gYear "2020"^^xsd:gYear "2019"^^xsd:gYear }
9   ?slice a qb:Slice ; wes-dimension:station ?statURI ; wes-dimension:year ?year ;
10    qb:observation [ a qb:Observation ; wes-attribute:observationDate ?date ;
11    wes-measure:avgDailyTemperature ?avg_temp ] . }

```

Figure 2: Query to retrieve avg. daily temp. timeseries computed from the observation in WeKG-MF recorded by weather stations located in “Nouvelle Aquitaine” French region.

a set of observations which applies to a spatio-temporal dimensions (e.g. a region, a weather station, a year, a time interval) is represented by the DCV class `qb:SLICE` such as the attributes and measures attached to these observations are previously semantically described in a DCV `qb:DATASTRUCTUREDEFINITION` class. This class enables to represent the slice’s metadata along with the specification of dimensions, attributes, and measures. An example of a DSD definition of annual times series of min/max/avg air temperatures is available³.

We have experimented different strategies to generate the RDF materialized slices according to a given DSD. A first strategy consists in relying on a unique SPARQL query of the CONSTRUCT form, enabling to create homogeneous RDF slices that include only aggregate values of one unique weather parameter (e.g., air temperature). A second strategy consists in combining several SPARQL queries of the SELECT form, whose results sets are integrated into the same slice. As

³<https://github.com/Wimmics/weather-kg/blob/main/meteo/dataset-metadata/DSD-Definition.ttl>

an example, Figure 1 illustrates the SPARQL query to calculate the total precipitation following the WMO documentation [8] which indicates that it is calculated for the day R_{day-j} as the accumulated precipitation of a specific day j from 6 *am* till 6 *am* of the following day j . Hence, six hours of the following UTC day shall be considered together with the current UTC day.

3. SPARQL-based Visual Tours

Aiming at simplifying the exploration of large RDF observational data available in WeKG-MF, we developed a Web application demo accessible at <https://nadiaya2019.github.io/DemoKGViz/> using the D3 JavaScript library. The webpage provides different visualisations offering lay-users visual “tours” at different levels of granularity. Thanks to the WeKG spatio-temporal model and the incorporation of pre-calculated RDF slices, data retrieved from our SPARQL endpoint can be visualized with no additional transformations involved, while most approaches for Linked Data visualisation include pre-processing steps that can be time-consuming (see Section 4).

3.1. Retrieving Salient Information

In order to retrieve the WeKG-MF graph, we rely on three categories of SPARQL query patterns that could be easily adapted.

- An initial pattern allows to retrieve the Météo-France weather stations (and their geo-spatial coordinates) grouped by French regions.
- The second pattern follows up, retrieves materialised RDF slices and collects values of at least one aggregate parameter pre-calculated for each station over a period of time.
- The third pattern enables users to extract fine-grained, atomic observations based on results provided before. It provides detailed data giving information about aggregated value provenances.

3.2. Visual Features

From a graphical point-of-view, we have developed several features to enrich the WeKG-MF users’ experience. As illustrated in Figure 3, we provide an interactive map allowing users to interact with the SPARQL endpoint by clicking on a French region (Figure 3.a). This action leads to the execution of an initial SPARQL query like the one depicted in Figure 2 that retrieves weather stations and their corresponding timeseries of a specific parameter, e.g., air temperature during 3 years (2019-2021). Markers indicating geo-spatial locations of weather stations are added to the map and the timeseries are represented through two interactive line charts, which x-axis represents time and y-axis represents the daily average air temperatures, while the color encodes the different weather stations for the selected region. The first chart (Fig.3.b – top-right) supports a brushing interaction allowing the user to select a time period to further explore the timeseries in the second chart (Fig.3.c – middle-right), which x-axis is updated according to the time selection. The brush selection is represented by a gray rectangle that can be resized at any time to expand/reduce the time span and by consequence update the view of the middle chart. Moreover, the brush selection can be handled through a click-and-drag movement to modify the time period while keeping the same time span. This chart (Fig.3.c – middle-right) supports interaction through a hovering technique, which displays a tooltip with detailed information

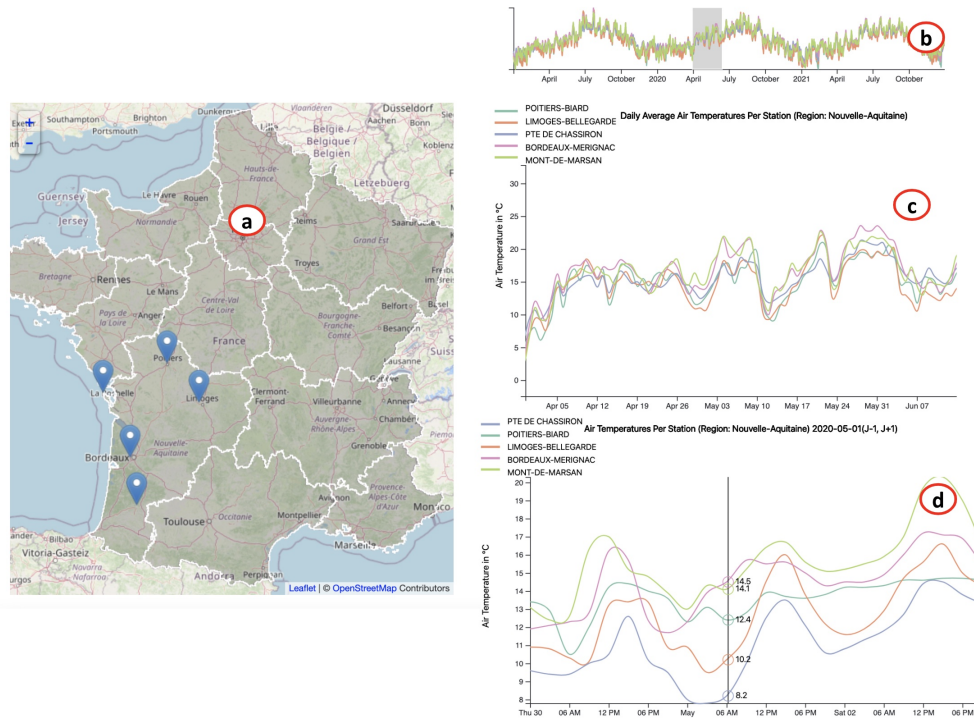


Figure 3: WeKG-MF exploration and navigation interface.

on the temperatures of a specific weather station and at the same time a third chart appears (Fig.3.d – bottom-right). Indeed, this chart offers a fine-grained view on WeKG-KG observations by displaying for a specific date $day - j$ un-aggregated atomic observations including those of the previous and following days. It supports also interaction through a hovering technique which displays a vertical line that the user could move to visualize values at the specific time of the day. Therefore, through this view, a user may easily explore the WeKG-MF knowledge graph from high-level aggregated timeseries to the elements from where the timeseries were calculated.

4. Related Work

Several research projects have focused on providing visualisation and exploration tools for LD datasets. Indeed, exploring, browsing and querying these datasets through space and time is very relevant for users but not straightforward for developers in order to transform RDF data into meaningful visualizations that suit users' needs. For an extensive review on LD exploration and visualization tools, we refer interested readers to [9]. While most existing approaches focus on how to shift pipelines to import/map/transform RDF data into data suitable for visualisations [10, 11], few of them highlight the importance of RDF modeling to easily support the generation of meaningful visualisations. Indeed, research works such as CubeViz [12] or OpenCube [13] aim to provide users with data cubes visualization and interactive analysis tools. However, to the best of our knowledge, multi-visualisation interfaces that combine high-level views on aggregated data using the RDF data cube vocabulary [6] and fine-grained views of un-aggregated values do not exist.

5. Conclusion and Future Works

We presented the first release of a Web application that offers interactive multi-level tours based on high-level aggregated views together with on-demand fine-grained data, and this through a unified multi-visualisations interface. In near future, we aim to work on a user evaluation study of our system to provide advanced analysis functionalities enabling experts to compare climatic conditions across geospatial and temporal dimensions. Moreover, we plan to enrich the interface to track data quality issues such as missing values across timeseries of weather parameters.

Acknowledgements. This study was carried out within the D2KAB project “From Data to Knowledge in Agronomy and Biodiversity”, financed by the French National Research Agency (ANR-18-CE23-0017). We are also grateful to Aline Menin and Olivier Corby for their valuable help.

References

- [1] N. Yacoubi Ayadi, C. Faron, F. Michel, F. Gandon, O. Corby, A model for meteorological knowledge graphs: Application to Météo-France observational data, in: ICWE, 2022.
- [2] R. Battle, D. Kolas, Enabling the geospatial semantic web with parliament and geosparql, *Semantic Web 3* (2012) 355–370.
- [3] S. Cox, C. Little, Time ontology in OWL, 2020. URL: <https://www.w3.org/TR/owl-time/>.
- [4] K. Janowicz, A. Haller, S. J. D. Cox, D. L. Phuoc, M. Lefrançois, SOSA: A lightweight ontology for sensors, observations, samples, and actuators, *J. Web Semant.* 56 (2019) 1–10.
- [5] A. Haller, K. Janowicz, S. J. D. Cox, M. Lefrançois, K. Taylor, D. L. Phuoc, J. Lieberman, R. García-Castro, R. Atkinson, C. Stadler, The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation, *Semantic Web 10* (2019) 9–32.
- [6] D. Reynolds, R. Cyganiak, The RDF Data Cube Vocabulary, W3C Recommendation, W3C, 2014. URL: <https://www.w3.org/TR/2014/REC-vocab-data-cube-20140116/>.
- [7] N. Yacoubi Ayadi, C. Faron, F. Michel, F. Gandon, O. Corby, WeKG-MF: A knowledge graph of observational weather data, in: ESWC 2022 Satellite Events, 2022.
- [8] World Meteorological Organization., Handbook on climat and climat temp reporting, 2009. URL: https://library.wmo.int/doc_num.php?explnum_id=9253.
- [9] L. Po, N. Bikakis, F. Desimoni, Linked Data Visualization Tools, 2020, pp. 47–72.
- [10] A. Menin, C. Faron, O. Corby, C. Freitas, F. Gandon, M. Winckler, From linked data querying to visual search: Towards a visualization pipeline for LOD exploration, in: Proceedings of the 17th International Conference on Web Information Systems and Technologies, 2021.
- [11] H. Raissya, F. Darari, F. Ekaputra, VizKG: A framework for visualizing SPARQL query results over knowledge graphs, volume 3023, CEUR-WS, 2021, pp. 95–102.
- [12] M. Martin, K. Abicht, C. Stadler, S. Auer, A.-C. Ngonga Ngomo, T. Soru, CubeViz – exploration and visualization of statistical linked data, in: WWW, 2015.
- [13] E. Kalampokis, A. Nikolov, P. Haase, R. Cyganiak, A. Stasiewicz, A. Karamanou, M. Zotou, D. Zeginis, E. Tambouris, K. Tarabanis, Exploiting linked data cubes with opencube toolkit, in: ISWC Poster & Demonstration, 2014.

. **Demonstration paper**

Experience with Visual SPARQL Queries over DBPedia

Kārlis Čerāns, Jūlija Ovčinnikova, Mikus Grasmanis and Lelde Lāce

Institute of Mathematics and Computer Science, University of Latvia, Riga, Latvia

Abstract

We describe a development of tooling for and experience with visual UML-style presentation and visual-based creation of SPARQL queries over *DBPedia*. The tool for visual query creation offers query auto-completion based on the actual *DBPedia* schema; we add to the tool an index-based service to start a query from an individual. We discuss end-user experience in creating visual SPARQL queries over *DBPedia* and evaluate the possibilities of auto-mated visual query generation from their SPARQL form in the context of the public QALD-9 query dataset.

Keywords

SPARQL, DBPedia, Visual queries, Query visualization

1. Introduction

Visual presentation of information artefacts can help their perception. The tools for visual creation of SPARQL queries over RDF data endpoints (cf. e.g., [1-4]) introduce the visual cognitive aspect into the query perception and query composition. Visual method has been successfully used for SPARQL query formulation by domain experts [1]. *ViziQuer* [4] has shown the possibilities of using a UML-style notation to visually create [5] and visualize [6] complex SPARQL queries, involving e.g., basic graph patterns, aggregation and subqueries, complex data expressions and filters. This paper reports on novel options to use visual queries over *DBPedia* [7,8] - one of the central Linked Data resources of fundamental importance to the entire Linked Data ecosystem. A solution for visual query creation over *DBPedia* has been out-lined in [9], where the services for query auto-completion, based on class-to-property and property-to-property relations, as well as some client-side enhancements over the generic visual query solution, based on available property ordering and filtering, have been presented. ***The current work reports on expanding the visual DBPedia query environment*** by means for efficient query starting from individual resources, as well as it discusses findings from an ***early user study in formulating the queries*** over *DBPedia* in the context of the expanded visual environment.

The study of creation of visual *DBPedia* queries is accompanied in this work with a ***study of visualization of existing SPARQL queries*** (cf. [6]) over the *DBPedia* data endpoint. We

VOILA! 2022: 7th International Workshop on Visualization and Interaction for Ontologies and Linked Data, October 23, 2022, Virtual Workshop, Hangzhou, China, Co-located with ISWC 2022.

✉ karlis.cerans@lumii.lv (K. Čerāns); julija.ovcinnikova@lumii.lv (J. Ovčinnikova); mikus.grasmanis@lumii.lv (M. Grasmanis); lelde.lace@lumii.lv (L. Lāce)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

consider the queries from the QALD-9 test dataset¹ used originally in the context of natural language query answering [10] (we look at 150 queries that have textual formulations in English); we provide visual presentations of these queries. For 146 out of 150 queries (over 97%) the visual presentation can be computed automatically, allowing to conclude that there exist contexts (e.g., the QALD-9 queries), where the visual method can be used for the SPARQL query visualization purposes.

The *gallery of the visually presented queries* is available in a working visual tool environment, where each visual query can be translated back into SPARQL and executed over the *DBPedia* data endpoint². The user can also use the visual environment to create new queries over *DBPedia*, and to visualize other existing SPARQL queries. The environment can be accessed from the paper's supporting site available at <http://viziquer.lumii.lv/examples/dbpedia2022>.

The concept of query auto-completion, essential for a satisfactory user experience in the visual query creation, has been implemented over large data sets in various settings, including the Wikidata query service [11], or FAAS [12] solution to support query creation in RDF Explorer [2]. Our solution is the first one that offers the in-instance name lookup for the *DBPedia* data set in the context of a visual query environment.

2. Visual Query Notation Review

A UML-style visual query in *ViziQuer* notation [5,13] consists of nodes describing variables or resources, each node can have a possible class name and attribute specification. One of the nodes is marked as the main query node (orange round rectangle). The edges that connect the nodes correspond to links among the node variables or resources (there can be “same-instance” links, labelled by ‘==’, and “empty” links that do not specify a data connection, labelled by ‘++’, as well). Textual condition/filter fields, along with aggregation and query nesting links are available, as well.

Figure 1 shows six simple visual queries over the *DBPedia* SPARQL endpoint, they are chosen to match the visualizations of SPARQL queries from the QALD-9 dataset, as well as to illustrate different query building constructs appearing in the examples. Just four of the nodes in the six queries have their class names specified (the default *dbo:* namespace classes *Volcano*, *Book*, and *Film*, as well as the *yago:* namespace class *WikicatJapaneseMusicalInstruments*). The instances, matching the query nodes are marked by resources (e.g., *dbr:Constitutional_monarchy*, *dbc:Countries_in_Africa* and *dbr:Taiko*), or by variables (e.g., *uri*, *area*, *x*). Each of the presented queries includes a comment stating the QALD-9 ID and the textual purpose of the query.

Since *DBPedia* typically provides English-based human-readable URIs, the results of the queries are often obtained in the form of URIs matching the variables used in the query; the selection of the variables corresponding to the query nodes are marked by the (*select this*) notation within the query node attribute list. The attribute expressions can be specified for selection, as well; *rdfs:label@en* (choose the label in English) and *dbp:birthName* are some examples.

¹<https://github.com/ag-sc/QALD/blob/master/9/data/qald-9-test-multilingual.json>

²<https://dbpedia.org/sparql/>

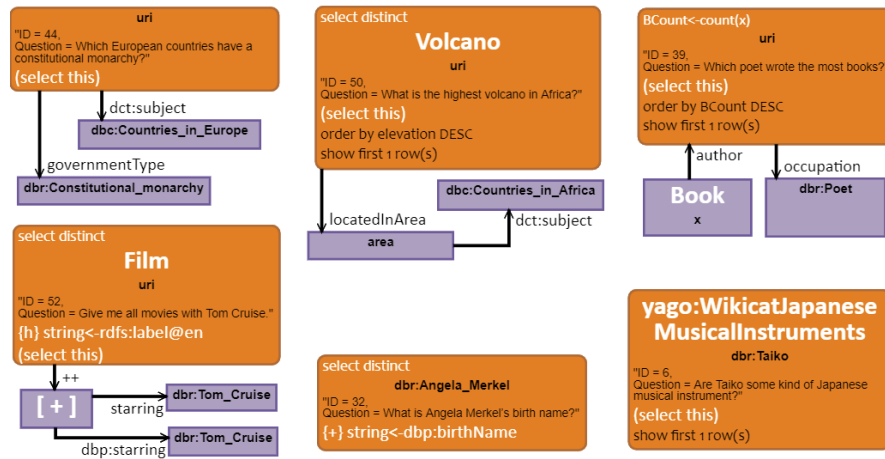


Figure 1: Six QALD-9 queries over *DBPedia* SPARQL endpoint.

An attribute, marked by $\{+\}$, is required to have value for the corresponding row to be included in the result set (in *ViziQuer* the attributes are optional by default); the mark h specifies finding the attribute but not including it into the selection list. The query (QALD-9 ID=52) finding all movies with Tom Cruise features a union construct (marked by $[+]$).

A description of other *ViziQuer* notation features (e.g., the notation for subqueries) can be found in [13] and on the tool website.

3. Visual Query Creation

The visual query environment assists the user in query creation by offering searchable lists of entities (classes, properties and individuals) for starting (seeding) the query, as well as similar lists for extending (growing) a partial query with entities relevant in its context. Since the actual schema of the *DBPedia* endpoint involves over 480 thousand classes and over 50 thousand properties, a relational database support for query completion over *DBPedia* has been introduced in [9] and involves (partial) storing of both the class-to-property and property-to-property relations.

Since a variety of natural SPARQL queries over *DBPedia* would involve some individual, whose properties are to be retrieved or analyzed (this applies also to the considered QALD-9 queries), an efficient service for starting a query from an individual is essential. We have introduced such a service, based on an index of all *DBPedia* resources and categories (all individuals, except the documents), and have integrated it into the query environment. Starting a query from an individual within the *ViziQuer* tool is performed by typing a part of the individual's URI (full or pre-fixed form) in a search-box to the right of the visual diagram pane.

The data for the index are currently collected from the *DBPedia* data dump, and are stored into the same Postgres database, as the data schema, holding the *DBPedia* class and property information. It should in principle be possible to also invoke an external service as e.g., *DBPedia Lookup* [14] to fill the instance auto-completion list; this is seen as a task for a future work.

The extended auto-completion functionality of the visual query environment over *DBPedia* has allowed to carry through a preliminary user study on the suitability of the visual query environment for creation of queries over *DBPedia*. The study has been performed with technically literate end users (IT master's degree students) with limited previous experience of *ViziQuer* (about 15 queries composed over a simpler data endpoint). The study indicates that the environment can be used by the considered end user group, ***provided the encoding of the information in the data endpoint is clear.***

The students were given textual formulations of nine queries similar in spirit to the QALD-9 queries, with some advanced constructs involved, as shown in Table 1.

1. How many films are there?
2. Find (list) all films starring Tom Cruise.
3. Find the 10 youngest tennis players (list the player resource URI and the birth date).
4. Find all soccer players that are born on or after January 1, 2007. Note: use "yyyy-MM-dd"^^xsd:date as the format for date literals (replace yyyy, MM and dd by the year, the month and the date, respectively, e.g., as in "2007-01-01"^^xsd:date).
5. How many grandchildren did Thomas Jefferson have?
6. Find the person with the largest grandchildren count (list the person resource URI and the grandchildren count).
7. Find the three top countries with largest volcano counts in the country (list the country resource URI and the volcano count).
8. List all persons that each have 100 or more films starring them (list the person resource URI and the respective film count).
9. Find all politicians, born on the same date as Tom Cruise (list the politician's resource URI and the birth date).

Table 1

Queries for the user study.

Out of the 10 study participants all 10 were successful on queries #1-#4 and 9 were successful on the "more advanced" queries #9 and #10³. The queries #5 and #6, related to grandchildren count, had 6 successful completions (using a chain of *dbo:child* relations), while the rest attempted to use *dbp:grandchildren* property that does not provide the necessary data. The query #7 however, had only 3 of 10 successful submissions based on the *dbo:locatedInArea* property (the others have used the *dbo:country* property, which provided much less information). Further information on the user study is available on the paper's support page.

The study results show that the size of the *DBPedia* data set schema, as presented within the visual query environment, does not impede the query creation by the considered user group, as the queries have generally been successfully composed. The present query composition failures in queries #5, #6 and #7 are clearly related to the lack of the knowledge of the study participants on the ways, how the information has been encoded in the *DBPedia* resource. Clearly, the visual method for the query creation alone would not be sufficient to overcome this difficulty. Some possible approaches to the solution might involve encouraging the participants to try different candidate approaches for the information extraction and then choose the one that apparently gives the most relevant results. Some help on the existing variations of the information encoding within the data set could perhaps be offered by the query environment, however, this task would already go into the realm of natural language question answering (cf. e.g., [10]) and is beyond the scope of this short paper.

³The criterion for success of a query formulation work is producing a query that returns the results that are equivalent to the expected ones.

4. SPARQL Query Visualization

The visual presentation of a textual SPARQL query provides an additional perspective on a query presentation by invoking the visual cognitive aspect in query perception, so potentially easing the task of understanding a query (note that the visual query form can be used in addition to the textual form, not necessarily replacing it). The visual SPARQL query presentation using **UML-style** notation (cf. [6]) can be said to have benefits of (i) presenting the **classification and attribute selection triples** in a compact notation (that involves just a class or property name within a query node), and (ii) splitting the query over **multiple visual elements** to reduce the local complexity of any visually separated query element.

We used the automated visualization of SPARQL queries [6] (a slightly optimized and fine-tuned version of it) to create a visual query library from the QALD-9 test queries over *DBPedia* to see, how the SPARQL query visualization methods work in practice, as well as to demonstrate the capabilities of the visual notation on a set of externally available queries. It can be argued that the visual presentation of these particular queries do provide insights into different ways, how a factual information may happen to be encoded into *DBPedia* (via class names, via linked resources (*dbr*: namespace entities) or categories (*dbc*: namespace), or even via fragments within class names (within *yago*: namespace); the use of the default *dbo*: namespace and the *dbp*: namespace for properties is to be considered, as well).

There are 4 SPARQL queries in the QALD-9 query set that are of the ASK format. Since the *ViziQuer* notation currently supports only SELECT queries, we visualize an ASK query into a SELECT query that returns a single line in the case of a result *true*, and an empty result set in the case of the result *false*.

There are 150 queries in the considered QALD-9 query subset, all of them are presented in the visual notation within the visual QALD-9 query library (Figure 1 depicts 6 of these visualizations). In the case of 146 queries (involving the 4 ASK queries) the visual presentation can be produced automatically (followed by a manual tuning of the query placement)^{4 5}. 4 queries have been drawn or corrected manually: 2 queries involving HAVING construct (expected to be modeled in the current visual notation via a filter in an enclosing query) and 2 queries involving complex UNION constructions (query optimization and implementation fine-tuning would resolve these issues).

The visual *DBPedia* query library is available as a project within the *ViziQuer* environment, accessible from the paper's support page.

5. Conclusions

The presented work shows the possibility to use UML-style visual presentation both in visual creation of SPARQL queries over *DBPedia*, and in visualization of existing SPARQL queries. The provided technical solutions of the data schema and instance information pre-computation and

⁴The non-standard SPARQL expressions as *SELECT xsd:date(?var)* found in 10 QALD-9 queries need to be re-written into *SELECT (xsd:date(?var) AS ?v)* before the visualization.

⁵A query is successfully visualized, if it produces equivalent results to the original query, when executed. The structural equivalence is considered as a primary visualization success criterion for queries that do not produce any results, when run over the *DBPedia* endpoint.

storage allow to ensure a smooth auto-completion experience both for the initial query seeding and its further context-based growing. An interesting future work would be to fine-tune the auto-completion process, by providing the class-to-individual connections (to suggest efficiently the names of individuals in a class context) based on *DBPedia Lookup* [14] or some FAAS-style component [12].

The presented user study could be seen as preliminary confirming that the size and complexity of the *DBPedia* query environment are not limiting the use of the schema and data elements in formulating a query. It has shown clearly, however, also that the visual method alone does not necessarily indicate which elements of the data schema should be used to find answers to the textually formulated queries. An interesting future work would be to develop and test an eco-system of semi-automated extracting the information from a SPARQL endpoint, like *DBPedia*, based on different knowledge encoding patterns, known for the data set.

The visual presentation of all QALD-9 example SPARQL queries demonstrates the capability of the automated query visualization method to handle queries from an externally existing data set. This result complements an earlier work on authors on *Wikidata* example query visualization [15] to show the maturity level of the visual query notation (as well as to indicate the directions of its further improvements).

The SPARQL query visualization provides a visual structure for any considered SPARQL query, thus adding the visual perception benefits to the query understanding. Considering the set of visualizations for the entire QALD-9 dataset allows also for visual conceptual evaluation of patterns of information encoding into *DBPedia*, up to possible discussion in what ways *DBPedia* can be used as an *ad hoc* information source in different knowledge areas.

Acknowledgements This work has been partially supported by a Latvian Science Council Grant lzp-2021/1-0389 “Visual Queries in Distributed Knowledge Graphs”.

References

1. Soylyu, A., Giese, M., Jimenez-Ruiz, E., Vega-Gorgojo, G., Horrocks, I.: *Experiencing OptiqueVQS: A Multi-paradigm and Ontology-based Visual Query System for End Users*. Universal Access in the Information Society, March 2016, Volume 15, Issue 1, pp 129–152.
2. Haag, F., Lohmann, S., Siek, S., and Ertl, T. *QueryVOWL: Visual Composition of SPARQL Queries*. In: The Semantic Web: ESWC 2015 Satellite Events. Springer LNCS, Vol.9341, pp. 62-66. (2015).
3. Vargas, H., Buil-Aranda, C., Hogan, A. and Lopez, C. *RDF Explorer: A Visual SPARQL Query Builder*, Proc. of ISWC 2019, Springer LNCS, Vol. 11778, pp. 647-663. (2019).
4. Čerāns, K., Šostaks, A., Bojārs, U., Ovčiņņikova, J., Lāce, L., Grasmanis, M. Romāne, A., Sprogis, A., Bārzdiņš, J. *ViziQuer: A Web-Based Tool for Visual Diagrammatic Queries Over RDF Data*. In: ESWC 2018 Satellite Events. Springer LNCS, vol 11155, pp. 158-163. (2018).
5. Čerāns, K., Šostaks, A., Bojārs, U., Bārzdiņš, J., Ovčiņņikova, J., Lāce, L., Grasmanis, M., Sprogis, A.: *ViziQuer: A Visual Notation for RDF Data Analysis Queries*. In Proc. of MTSR’2018, Springer CCIS, Vol.846, pp.50-62 (2019)

6. Čerāns, K., Ovčinnikova, J., Grasmanis, M., Lāce, L. and Romāne, A. *Visual Presentation of SPARQL Queries in ViziQuer*. In Proc. of VOILA! 2021, CEUR Workshop Proceedings Vol. 3023, pp. 29-40 (2021). <http://ceur-ws.org/Vol-3023/paper12.pdf>
7. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S. *DBpedia - A crystallization point for the Web of Data*. Web Semantics: Science, Services and Agents on the World Wide Web. 7 (3): pp. 154–165. (2009).
8. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S. and Bizer, C. *DBpedia - a large-scale, multilingual knowledge base extracted from Wikipedia*. Semantic Web 6(2), 167–195. (2015)
9. Čerāns, K., Lāce, L., Grasmanis, M. and Ovčinnikova, J. *A UML-style Visual Query Environment over DBPedia*. In Proc. of MTSR2021, Springer CCIS, Vol. 1537 (2022).
10. Usbeck, R., Gusmita, R.H., Ngonga Ngomo, A.-C., Saleem, M. *9th challenge on question answering over linked data (QALD-9)*. CEUR Workshop Proceedings Vol. 2241, pp. 58-64. (2018). <http://ceur-ws.org/Vol-2241/paper-06.pdf>
11. Wikidata Query Service. <https://query.wikidata.org/>
12. de la Parra, G., and Hogan, A. *Fast Approximate Autocompletion for SPARQL Query Builders*. CEUR Workshop Proceedings Vol. 3023, pp.41-55. (2021). <http://ceur-ws.org/Vol-3023/paper10.pdf>
13. Čerāns, K., Bārzdiņš, J., Šostaks, A., Ovčinnikova, J., Lāce, L., Grasmanis, M. Sproģis, A.: *Extended UML Class Diagram Constructs for Visual SPARQL Queries in ViziQuer/web*. In Voila!2017, CEUR Workshop Proceedings, Vol.1947, pp.87-98. (2017) <http://ceur-ws.org/Vol-1947/paper08.pdf>
14. DBPedia Lookup, <https://lookup.dbpedia.org/>
15. Čerāns, K., Ovčinnikova, J., Grasmanis, M. and Lāce, L. *Towards UML-style Visual Queries over Wikidata*. In ESWC 2022: The Semantic Web: ESWC 2022 Satellite Events, Springer LNCS volume 13384, pp.11-15. (2022).