An Interactive Dashboard for Ontology Quality Monitoring

Avetis Mkrtchian*, Petr Křemen

Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Computer Science

Abstract
Monitoring ontology quality is a key activity during the whole ontology lifecycle that requires adequate tools to provide overview over changing ontologies. In this paper we present an interactive dashboard framework for monitoring ontology metrics and quality indicators. While the framework is designed as generic, we present a prominent use-case for OBO Foundry ontologies, backing by ROBOT metrics and quality indicators.

Keywords
OBO Foundry, ROBOT tool, ontologies, dashboard, OWL

1. Introduction

Ontologies are a well-known paradigm of explicit shared formal conceptualizations. They have been traditionally strongly supported by medicine, or biology [1] that used them as large and rich reference taxonomies. Yet, nowadays, ontologies have become important also for sharing meaning of enterprise and open data, becoming a key piece of data-centric architecture [2].

As a result, communities have been created to monitor and supervise the quality of ontologies of a particular domain, like OBO Foundry [1], Industrial Ontologies Foundry (IOF) [3].

Yet, creating a proper monitoring solution for ontology quality is a challenge. Although most ontologies have been based on the semantic web standards (like RDFS [4], or OWL [5]), their structure can significantly differ, ranging from flat taxonomies to strongly axiomatized logical structures. This complicates creating a reusable solution for monitoring ontology quality, sentencing the communities to develop their own proprietary solutions.

In our work, we offer a framework for building interactive dashboards over a set of ontologies. Our approach to serve different requirements by different communities supervising ontology evolution for monitoring ontology quality and metrics by keeping the generic solution easily and quickly configurable. Having this said, we present here a work in progress. We did a single experiment with the framework so far – creating an interactive dashboard for OBO Foundry ontologies, their quality, metrics and evolution over time.

Proceedings of the International Conference on Biomedical Ontologies 2023, August 28th-September 1st, 2023, Brasilia, Brazil
*Corresponding author.
mkrtcave@fel.cvut.cz (A. Mkrtchian); petr.kremen@cvut.cz (P. Křemen)
0009-0003-1678-6773 (A. Mkrtchian); 0000-0001-6299-4766 (P. Křemen)

Section 2 presents some existing solutions, and in section 3 we present the architecture of our solution together with its features. In section 4 we evaluate our work on a set of OBO Foundry ontologies and discusses its usability with some members of the OBO Foundry community and section 5 presents our conclusions and lessons learnt.

2. Related Work

A traditional view on ontology quality analysis is given in [6]. In [7], an overview of ontology metrics and quality assessment approaches is further elaborated. Various ontology metrics and quality checks can be computed by the ROBOT tool , which is a general-purpose swiss-knife tool for general OWL ontologies, heavily used in the OBO Foundry. This community uses a periodically updated preconfigured dashboard in a tabular form over fundamental quality issues over all OBO Foundry Ontologies [9]. The objective of the OBO Dashboard is to offer a collection of automated tests aimed at defining a baseline level of conformity with OBO Principles and best practices. These principles encompass openness, a common format, URI/identifier space management, versioning, defined scope, textual definitions, relationships, comprehensive documentation, acknowledgment of diverse user communities, clear locus of authority, naming conventions, maintenance, and responsiveness within ontology development and management. However, as stated by OBO Foundry themselves, the outcome of the OBO Foundry Dashboard does not indicate the quality of an ontology's content.

Ontology quality dashboard could be configured also using general-purpose BI solutions over RDF, like SANSA[10].

3. Interactive Dashboard Framework

The Interactive Dashboard Framework is designed to deliver dynamic dashboards over a set of ontologies that can change over time. Its architecture is depicted in Figure 1. One of the key components is ROBOT [8], which provides ability to generate metrics and violation reports for the ontologies. However, the output generated by ROBOT is available in various formats, excluding RDF. To handle RDF data, the RDF4J [11] and Apache Jena [12] libraries are used. The ontological data is stored in the GraphDB [13] database, which communicates with the aforementioned libraries via an API.

For the frontend, the existing dashboard solution Kibana is utilized. Kibana [14] allows visualization, exploration and analysis of data from the Elasticsearch search engine. However, Elasticsearch does not support RDF data, an RDF Indexer is introduced as intermediary between Elasticsearch and GraphDB. The RDF Indexer facilitates the integration of RDF data into Elasticsearch, enabling querying and visualization within the Kibana frontend.

3.1. Main features

One of the main features is the approach to obtaining quality data on ontologies. Since the ROBOT tool is a key component, its capabilities were analyzed in detail, the most suitable commands were: `robot measure` to generate metrics and `robot report` to get a report on

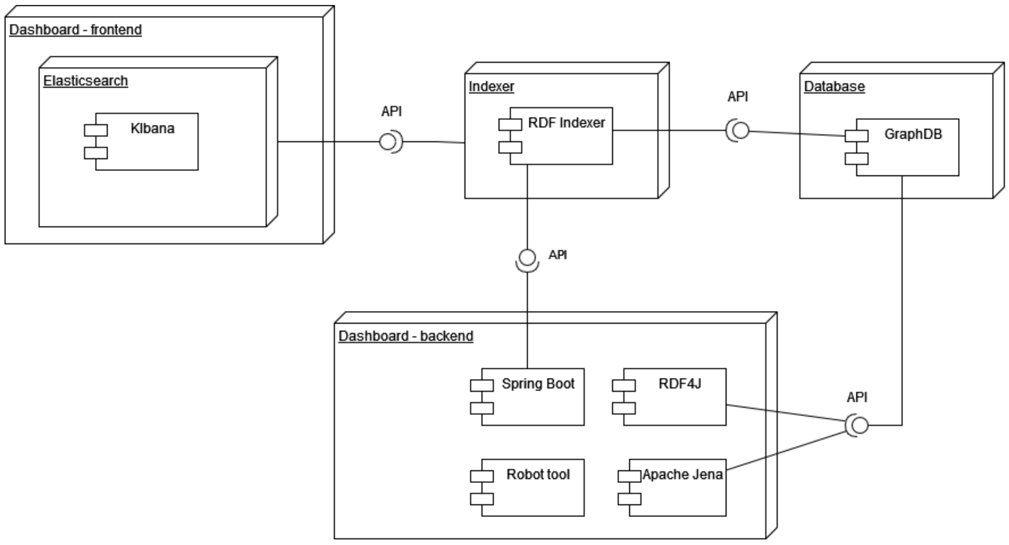


Figure 1: Component diagram

violations in ontologies. It was important for us to represent the outputs of these commands in RDF format in order to save them in triple-store. In order to implement this, the first idea was to describe these two commands using a suitable vocabulary. The results of the research of suitable vocabulary were the finding of DQV(Data Quality Vocabulary)[15] to describe metrics generated by ROBOT. In Figure 2 you can see the part of the data model that is used to represent output metrics of the `robot measure` in RDF format.

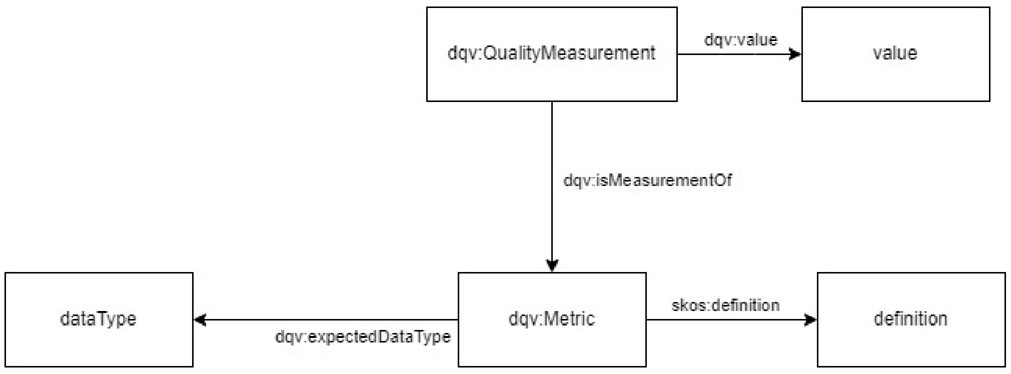


Figure 2: Part of DQV data model

To standardize validation and generation of violation reports and keep them independent of the ROBOT tool we use SHACL (Shapes Constraint Language) [16]. ROBOT offers ontology validation using a set of SPARQL [17] queries which we translated into SHACL.

Here is an example of translating the ROBOT rule called "lowercase definition":

```
PREFIX obo: <http://purl.obolibrary.org/obo/>
```

```
SELECT DISTINCT ?entity ?property ?value WHERE {
  VALUES ?property { obo:IAO_0000115 obo:IAO_0000600 }
  ?entity ?property ?value .
  FILTER (!regex(?value, "^[A-Z0-9]"))
  FILTER (!isBlank(?entity))
}
ORDER BY ?entity
```

This rule can be represented in SHACL as follows:

```
@prefix ex: <http://example.com/ns#> .

ex:lowercase_definition
    a sh:NodeShape ;
    sh:targetClass owl:ObjectProperty , owl:AnnotationProperty ,
    owl:Class ;
    sh:property [
        sh:path obo:IAO_0000115 ;
        sh:severity sh:Info;
        sh:message "lowercase_definition" ;
        sh:pattern "^[A-Z0-9](.*)" ;
    ] ;
    sh:property [
        sh:path obo:IAO_0000600 ;
        sh:severity sh:Info;
        sh:message "lowercase_definition" ;
        sh:pattern "^[A-Z0-9](.*)" ;
    ] .
```

Not all the rules were described exactly as above, complexity of some rules made us replace `sh:property` with `sh:sparql` and adopt the original SPARQL query to its SHACL version. Nevertheless, as many SHACL rules as possible were left without using SPARQL in order to improve validation efficiency. Thus all the predefined queries of the ROBOT report command were transformed in this way, with the exception of two: "deprecated class reference" and "deprecated property reference" due to their complexity.

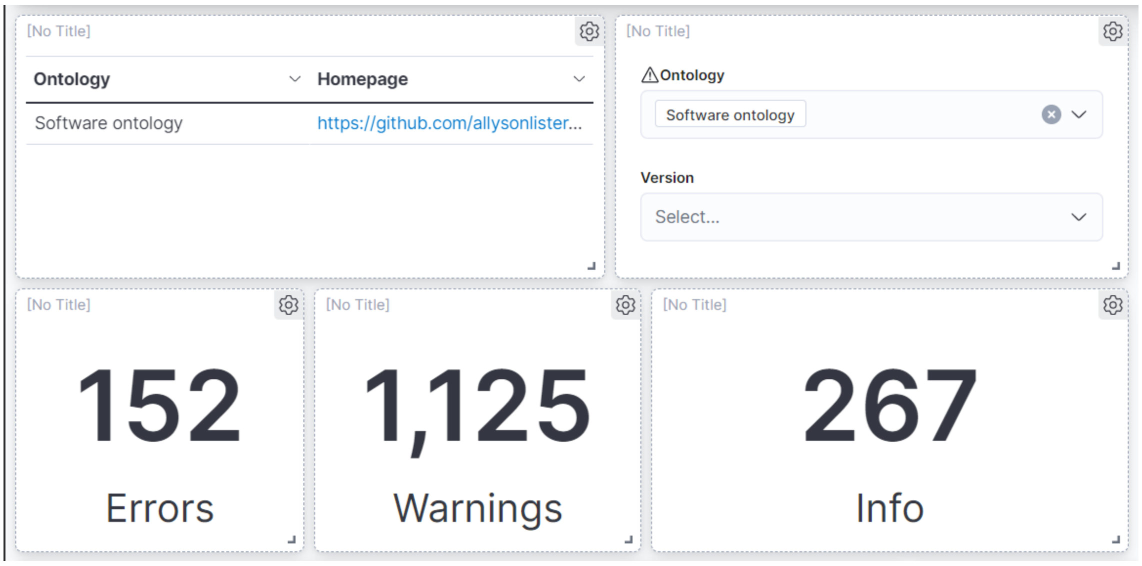
The Interactive Dashboard Framework solves the issue of ontology version tracking, which is currently a significant concern. Many ontology creators either overlook the using of specific OWL attributes like `owl:version` and `owl:versionInfo`, or they they use different schemata as numeric identifiers or date stamps, or a combination of thereof. This inconsistency can lead to difficulties in effectively tracking.

To solve this problem, the Interactive Dashboard Framework works in update mode. With each update, the framework performs a check to determine if the ontology contains date stamp attribute. If such an attribute is absent, the framework assigns its own version to the ontology data, which is date of update. This approach eliminates the reliance on inconsistent versioning practices and ensures ability to track changes over time.

By adopting this strategy, the framework enables the monitoring of variations in the number of violations or specific violations themselves across different ontology versions.

4. Evaluation

In order to test our framework, it was decided to make a prominent use case for OBO Foundry ontologies. The dashboard (available at http://tinyurl.com/obodashboard) currently contains a subset of all OBO Foundry ontologies, mainly to keep the experiment limited and running on a common hardware. While OBO Foundry provides its own dashboard solutions based on its principles, these solutions offer more general information without delving into specifics, such as detailed data on the types of violations. Our dashboard for OBO Foundry ontologies offers easy configuration and possibility to monitor ontology evolution in time. It consists of three main sections: "All ontologies", "Single ontology" and "Specific ontologies". In Figure 3 you can see part of "Single ontology" section, which demonstrates the possibility of selecting an ontology and its version, as well as information about violations of the ontology.



Ontology	Homepage
Software ontology	https://github.com/allysonlister...

Software ontology

152 Errors

1,125 Warnings

267 Info

Figure 3: Selection of one of the ontologies and violations information

Thus, using the example of a dashboard for OBO Foundry ontologies, you can easily configure a dashboard, for this you need to perform only three steps:

1. Configure a SPARQL query tailored to your specific requirements. This query will retrieve the necessary data from GraphDB. By carefully designing your query, you can extract relevant data you wish to visualize.
2. Index data from GraphDB to Elasticsearch with RDF Indexer using the SPARQL query from the previous point.

3. Use Kibana to design customized visualizations, such as graphs, charts, and tables, to effectively represent the ontology data indexed in Elasticsearch.

4.1. User feedback

We received feedback from the OBO Community by conducting four test scenarios that test usefulness where subjects gave comments and rated the scenario on a scale. And also ten questions on usability. Three subjects took part in the tests, and another subject gave his feedback with the format of the discussion.

The test results revealed that most of the comments were focused on the dashboard's UI rather than the data it contains. The primary drawback highlighted by all participants was the height of the dashboard box, which required constant scrolling down to access the most relevant information. This was caused by the navigation bar taking up a large percentage of the available area. Additionally, Kibana has its own UI elements like data filtering fields that further reduce the available space for the dashboard content. The separation of sections in the navigation panel also raised doubts, most participants would like to see the "All ontologies" and "Specific ontologies" sections in one. Almost all participants had problems with filtering data from the first time, this is due to the fact that filtering in Kibana is quite specific and takes a little time to sort it out, as with all other things, since Kibana is a bit technical UI.

As for the positive qualities of the dashboard, the main feature that the community appreciated was the version of ontologies with the possibility of tracking the number of violations in chronological order. One participant was surprised at how we approached the method of generating a violation report, i.e. by transforming ROBOT rules into SHACL syntax, as well as the description of metrics generated by ROBOT using DQV. Another thing appreciated by the participants were the links to the ROBOT rules website and to the ontology repositories.

5. Conclusions

The presented solution shows in a flexible and configurable way to configure an interactive dashboard over ontologies stored in a triple-store (or external files), while still providing useful outputs to the domain experts, as our experiment showed.

In the future, we would like to address several research directions. First, making indexing of large ontologies more efficient and robust. Also, incorporating other types of ontology metrics and quality control rules than those provided by ROBOT. Last, but not least, creating a set of predefined quality control widgets for a default dashboard ready to provide basic quality-related information about any OWL ontology.

This plan for future development will also include integration with the OBO Dashboard, further enhancing its utility and accessibility, as well as trying to apply the dashboard solution to other ontology communities.

References

[1] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. Goldberg, K. Eilbeck, A. Ireland, C. Mungall, N. Leontis, P. Rocca-Serra, A. Ruttenberg, S.-A. Sansone, R. Scheuermann, N. Shah, P. Whetzel, S. Lewis, The obo foundry: Coordinated evolution of ontologies to support biomedical data integration, Nature biotechnology 25 (2007) 1251–5. doi:10.1038/nbt1346.

[2] D. McComb, The Data-Centric Revolution: Restoring Sanity to Enterprise Information Systems, Technics Publications, 2019.

[3] Industrial Ontologies Foundry, https://www.industrialontologies.org/, 2023. [Accessed 5-September-2023].

[4] D. Brickley, R. Guha, RDF Schema 1.1, W3C Recommendation, W3C, 2014. Https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[5] S. Rudolph, P. Patel-Schneider, B. Parsia, M. Krötzsch, P. Hitzler, OWL 2 Web Ontology Language Primer (Second Edition), Technical Report, W3C, 2012. Https://www.w3.org/TR/2012/REC-owl2-primer-20121211/.

[6] S. Tartir, I. B. Arpinar, M. Moore, A. P. Sheth, B. Aleman-Meza, OntoQA: Metric-based ontology quality analysis, in: KADASH, 2005.

[7] R. S. I. Wilson, J. S. Goonetillake, W. A. Indika, A. Ginige, Analysis of ontology quality dimensions, criteria and metrics, in: O. Gervasi, B. Murgante, S. Misra, C. Garau, I. Blečić, D. Taniar, B. O. Apduhan, A. M. A. C. Rocha, E. Tarantino, C. M. Torre (Eds.), ICCSA 2021, Springer International Publishing, Cham, 2021, pp. 320–337.

[8] R. Jackson, J. Balhoff, E. Douglass, N. Harris, C. Mungall, J. Overton, Robot: A tool for automating ontology workflows, BMC Bioinformatics 20 (2019).

[9] OBO Dashboard, https://dashboard.obofoundry.org/dashboard/index.html, 2023. [Accessed 5-September-2023].

[10] J. Lehmann, G. Sejdiu, L. Bühmann, P. Westphal, C. Stadler, I. Ermilov, S. Bin, N. Chakraborty, M. Saleem, A.-C. N. Ngonga, H. Jabeen, Distributed semantic analytics using the sansa stack, in: ISWC'2017, Springer, 2017, pp. 147–155. URL: http://svn.aksw.org/papers/2017/ISWC_SANSA_SoftwareFramework/public.pdf.

[11] E. R. developers, Welcome · Eclipse RDF4J™ | The Eclipse Foundation — rdf4j.org, https://rdf4j.org/, 2023. [Accessed 13-July-2023].

[12] Apache Jena, 2023. URL: https://jena.apache.org/, [Accessed 13-July-2023].

[13] GraphDB Downloads and Resources — graphdb.ontotext.com, 2023. URL: https://graphdb.ontotext.com, [Accessed 13-July-2023].

[14] Kibana: Explore, visualize, Discover Data, 2023. URL: https://www.elastic.co/kibana/, [Accessed 13-July-2023].

[15] R. Albertoni, A. Isaac, Introducing the data quality vocabulary (DQV), Semantic Web 12 (2021) 81–97.

[16] Shapes constraint language (SHACL), Technical Report, W3C, 2017. URL: https://www.w3.org/TR/shacl/, [Accessed 13-July-2023].

[17] E. Prud'hommeaux, A. Seaborne, SPARQL Query Language for RDF, W3C Recommendation, W3C, 2008. URL: https://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/, [Accessed 15-July-2023].