




# ABECTO: An ABox Evaluation and Comparison Tool for Ontologies

Jan Martin Keil<sup>(✉)</sup> 

Heinz Nixdorf Chair for Distributed Information Systems,  
Institute for Computer Science, Friedrich Schiller University Jena, Jena, Germany  
[jan-martin.keil@uni-jena.de](mailto:jan-martin.keil@uni-jena.de)

**Abstract.** Correctness and completeness of ontologies on the schema and the instance level are important quality criteria in their selection for an application. Due to the general lack of gold standard data sources, the determination of these criteria, especially on the instance level, is challenging. The direct comparison of candidate data sources enables the approximation of these criteria. We introduce ABECTO, an ABox evaluation and comparison tool for ontologies. ABECTO provides a framework for the comparison of different semantic data sources in the same domain on the instance level.

**Keywords:** Ontology ABox · Ontology comparison · Ontology evaluation · Ontology quality · Ontology selection

## 1 Introduction

Ontologies can be valuable sources of domain knowledge for various applications. However, the selection of appropriate ontologies requires particular attention. The ontologies must provide a sufficient degree of entity coverage (*population completeness* in [1, 2]) and a sufficient level of detail (*schema completeness* in [1, 2]) [3]. Besides that, the faultless operation of applications also relies on the correctness (*accuracy* in [1, 2]) and sufficient value coverage (*property completeness* and *interlinking completeness* in [1] or *column completeness* in [2]) of the ontologies. To verify the correctness and completeness of a candidate ontology, modeled facts must be compared to actual facts. These actual facts would be contained in a gold standard data source. Classical ontology engineering methodologies use competency questions to specify requirements and to verify requirement compliance. Expected answers to competency questions for the verification of correctness or completeness of facts in ontologies would implicitly also represent a gold standard data source. However, the existence of a gold standard data source for real world data is almost impossible. Due to this general lack of gold standard data sources, we proposed the direct comparison of multiple independent candidate ontologies to approximate their correctness

---

The original version of this chapter was revised: this chapter was previously published non-open access. The correction to this chapter is available at

[https://doi.org/10.1007/978-3-030-62327-2\\_48](https://doi.org/10.1007/978-3-030-62327-2_48)

© The Author(s) 2020, corrected publication 2023

A. Harth et al. (Eds.): ESWC 2020 Satellite Events, LNCS 12124, pp. 140–145, 2020.

[https://doi.org/10.1007/978-3-030-62327-2\\_24](https://doi.org/10.1007/978-3-030-62327-2_24)

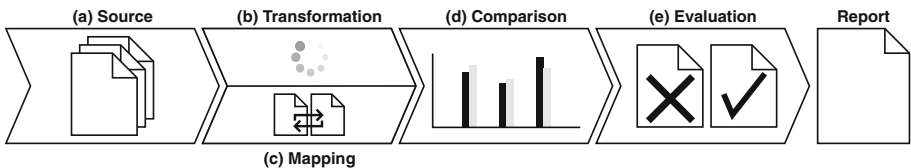
and completeness [4]. Measures of the correctness and completeness could then support the selection of appropriate ontologies that fulfill the requirements of a certain project. As an example, consider two candidate ontologies containing 100 and 150 relevant entities respectively for a text annotation service. The first contains 100 English and 100 Spanish labels, the second 140 English and 130 Spanish labels. A comparison reveals that 90 of the entities are contained in both ontologies and detects deviations between the ontologies in 40 Spanish labels, caused by errors in the second ontology. Depending on the focused languages, this allows a more profound choice of the ontology.

The term *ontology comparison* is used with different meanings in the literature: (a) The comparison of entire ontologies regarding certain aspects to evaluate or select ontologies, (b) the comparison of different versions of one ontology to highlight changes, (c) the comparison of single entities or sets of entities to calculate recommendations of entities, or (d) the calculation of the similarity of single or a few entities from different ontologies to match or merge these ontologies [4]. In this paper, we focus on Variant (a), only.

We introduce ABECTO, an ABox evaluation and comparison tool for ontologies. ABECTO implements a framework for comparing multiple ontologies in the same domain. To the best of our knowledge, this is the first software tool for the comparison of ontologies on ABox level to approximate their correctness and completeness. In the remainder of this article, we will introduce the functionality of ABECTO in Sect. 2, explain our strategy to handle different modeling approaches in Sect. 3, describe the implementation of ABECTO in Sect. 4, and describe the demonstration in Sect. 5.

## 2 System Overview

ABECTO implements our framework for ontology ABox comparison described in [4]. The framework consists of five components, as shown in Fig. 1: (a) A *source* component to load ontologies, (b) a *transformation* component to add deduced axioms to the ontologies in preparation of further processing, (c) a *mapping* component to map the resources of the ontologies, (d) a *comparison* component to provide measurements of the ontologies, and (e) an *evaluation* component to identify potential mistakes in the ontologies.



**Fig. 1.** Schematic of the comparison framework implemented in ABECTO. The order of the transformation and mapping processes is up to the user.

For each component, ABECTO provides a couple of *processors*, which provide a specific functionality. These processors can be arranged by the users into a processing pipeline to define the comparison process.

### 3 Handling of Different Modeling Approaches

The comparison of the ABoxes requires identifying corresponding facts of the ontologies. However, different ontologies of the same domain might use different approaches to model certain aspects of this domain. For example, there might be (a) properties corresponding to a chain of properties, (b) anonymous individuals corresponding to named individuals, (c) data properties corresponding to annotation properties, or (d) classes corresponding to individuals [4].

To meet this challenge, the sets of resources and their comparable properties are described with so-called *categories*. A category is defined by a SPARQL GroupGraphPattern [5] (the WHERE clause) for each ontology. The variable with the same name as the category represents the resource to compare. The bindings of all other equally named variables will be compared. This enables the definition of the facts to compare in a way that meets all mentioned cases: (a) Resource and variables can be linked by properties as well as complex property paths, (b) unambiguous IRIs can be created using key properties values, (c) resource and variables can be linked by data properties as well as annotation properties, and (d) the resource might represent a class as well as an individual. In the further processing, these patterns will be used to obtain the facts for ontology comparison.

### 4 Implementation

ABECTO is implemented as a Java HTTP REST service based on Apache Jena<sup>1</sup> and Spring<sup>2</sup> to provide a convenient interface for user interfaces or other applications. The size of compared ontologies is mainly limited by the memory required to represent the ontologies. Therefore, we expect ABECTO to be able to process large ontologies on appropriate hardware. A Python module provides handy functions to use ABECTO inside a Jupyter notebook<sup>3</sup>, hiding the raw HTTP requests. This allows an easy setup of reproducible ontology comparison projects. However, the result presentation in the Jupyter Notebook interface for ABECTO is only suitable for smaller ontologies. To support large ontologies, an independent interface like a stand-alone web application would be needed. The sources of ABECTO are publicly available under the Apache 2.0 license [6].

In ABECTO, the ontologies will be compared inside of a project. A project consists of several ontologies and a processing pipeline. Each node of the pipeline represents a processor with a particular configuration. A processor is a Java class with specified methods to generate an output RDF model. The start nodes of the pipeline are the nodes representing a *source processor*, which loads an RDF model from an external source. To support modularized ontologies, multiple source nodes might belong to one ontology. Nodes of other processors require at least one input node. These processors can be divided into (a) *transformation*

<sup>1</sup> <https://jena.apache.org/>.

<sup>2</sup> <https://spring.io/>.

<sup>3</sup> <https://jupyter.org/>.

*processors*, which extend the input RDF model, (b) *mapping processors*, which provide resource mappings of the input RDF models of different ontologies, and (c) *meta processors*, which calculate comparative meta data from the input RDF models. The comparative meta data include *measurements*, like resource counts, identified *deviations* of mapped resources, *issues*, like an encountered literal when a resource was expected, and *categories*, which define the sets of resources and their properties to compare. The output RDF models of source and transformation processors belong to a certain ontology, whereas the output RDF models mapping and meta processors do not belong to a certain ontology. Therefore, they will be treated differently by the subsequent processors. The following processors are already available in ABECTO:

- RdfFileSourceProcessor:** Loads an RDF document from the local file system.
- JaroWinklerMappingProcessor:** Provides mappings based on Jaro-Winkler Similarity [7] of `string` property values using our implementation from [8].
- ManualMappingProcessor:** Enables users to manually adjust the mappings by providing or suppressing mappings.
- RelationalMappingProcessor:** Provides mappings based on the mappings of referenced resources.
- OpenIetReasoningProcessor:** Infers the logical consequences of the input RDF models utilizing the OpenIet Reasoner<sup>4</sup> to generate additional triples.
- SparqlConstructProcessor:** Applies a given SPARQL Construct Query to the input RDF models to generate additional triples.
- CategoryCountProcessor:** Measures the number of resources and property values per category.
- LiteralDeviationProcessor:** Detects deviations between the property values of mapped resources as defined in the categories.
- ManualCategoryProcessor:** Enables users to manually define resource categories and their properties.
- ResourceDeviationProcessor:** Detects deviations between the resource references of mapped resources as defined in the categories.

We plan to add further processors in the near future, including:

- A mapping processor that employs the well known matching libraries using the Alignment API [9].
- A mapping processor that reuses mappings contained in the ontologies.
- A mapping processor that provides transitive mappings based on results of other mappings.
- A meta processor that utilizes mark and recapture techniques [10] to measure the completeness of ontologies.
- A source processor that loads an RDF document from a URL.
- A source processor that imports triples of a specified scope from a SPARQL endpoint.
- A source processor that utilizes SPARQL Generate [11] to load comparison data from non-RDF documents.

<sup>4</sup> <https://github.com/Galigator/openIet>.

Category Count Report				
Category	Variable	Ontology 1	Ontology 2	Ontology 3
person		2	3	4
person	boss	2	2	
person	label	2	3	4
person	pnr	2		4

Deviation Report				
Category: person				
Ontology 1		Ontology 2		
http://example.org/a/alice	boss	<http://example.org/a/bob>	<http://example.org/b/alice>	boss
http://example.org/a/bill	label	Bill	William	label
Ontology 1		Ontology 3		
http://example.org/a/alice	pnr	45678	12345	pnr

Issue Report		
Ontology: Ontology 2		
Issue Type	Affected Entity	Message
UnexpectedValueType	http://example.org/b/william	Value of property "boss" is not a resource.

Fig. 2. Screenshot of an example report generated in a Jupyter Notebook. The report shows one type of measurement (number of resources and property values per category), encountered deviations, and encountered issue of a comparison of three ontologies.

The meta data models generated by the nodes can be used to generate reports. These reports might contain the calculated *measurements*, *deviations* and *issues*. Figure 2 shows an example report generated in a Jupyter Notebook.

## 5 Demonstration

We will demonstrate how users can utilize ABECTO to compare and evaluate ontologies. We will provide sets of real world RDF documents with prepared project definitions and category descriptions. The projects are managed inside of Jupyter notebooks. A tutorial notebook is available and can be executed online<sup>5</sup> using Binder [12]. Users will be able to manipulate and execute the project pipelines and examine the resulting comparison and evaluation reports.

**Acknowledgments.** Many thanks to the three anonymous reviewers, to my supervisor Birgitta König-Ries, and to my colleagues Alsayed Algerawy, Felicitas Löffler, and Samira Babalou for very helpful comments on earlier drafts of this manuscript.

<sup>5</sup> <https://mybinder.org/v2/zenodo/10.5281/zenodo.3786194/?filepath=abecto-tutorial.ipynb> (live preview loading might take a few minutes).

## References

1. Zaveri, A., Rula, A., Maurino, A., et al.: Quality assessment for linked data: a survey. *Seman. Web* **7**(1), 63–93 (2016). <https://doi.org/10.3233/SW-150175>
2. Färber, M., Bartscherer, F., Menne, C., Rettinger, A.: Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. *Seman. Web* **9**(1), 77–129 (2018). <https://doi.org/10.3233/SW-170275>
3. Heist, N., Hertling, S., Ringler, D., Paulheim, H.: Knowledge graphs on the web - an overview. *arXiv: 2003.00719v2* [cs.AI] 2 March 2020
4. Keil, J.M.: Ontology ABox comparison. In: Gangemi, A., et al. (eds.) *ESWC 2018*. LNCS, vol. 11155, pp. 240–250. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-98192-5\\_43](https://doi.org/10.1007/978-3-319-98192-5_43)
5. SPARQL 1.1 Query Language. Recommendation. W3C, 21 March 2013. <https://www.w3.org/TR/2013/REC-sparql11-query-20130321/>
6. Keil, J.M.: ABECTO. <https://doi.org/10.5281/ZENODO.3786194>
7. Winkler, W.E.: String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. In: *Proceedings of the Section on Survey Research*, pp. 354–359. American Statistical Association (1990). <http://eric.ed.gov/?id=ED325505>
8. Keil, J.M.: Efficient bounded Jaro-Winkler similarity based search. In: Grust, T., Naumann, F., Böhm, A., et al. (eds.) *BTW 2019*. Gesellschaft für Informatik, Bonn, pp. 205–214 (2019). <https://doi.org/10.18420/btw2019-13>
9. David, J., Euzenat, J., Scharffe, F., dos Santos, C.T.: The alignment API 4.0. *Seman. Web* **2**(1), 3–10 (2011). <https://doi.org/10.3233/SW-2011-0028>
10. Razniewski, S., Suchanek, F.M., Nutt, W.: But what dowe actually know? In: Pujara, J., Rocktäschel, T., Chen, D., Singh, S. (eds.) *Proceedings of the 5th Workshop on Automated Knowledge Base Construction, AKBC@NAACL-HLT 2016*. The Association for Computer Linguistics, pp. 40–44 (2016). <https://doi.org/10.18653/v1/W16-1308>
11. Lefrançois, M., Zimmermann, A., Bakerally, N.: A SPARQL extension for generating RDF from heterogeneous formats. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) *ESWC 2017*. LNCS, vol. 10249, pp. 35–50. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-58068-5\\_3](https://doi.org/10.1007/978-3-319-58068-5_3)
12. Project Jupyter, Bussonnier, M., Forde, J., et al.: Binder 2.0 - reproducible, interactive, sharable environments for science at scale. In: Akici, F., Lippa, D., Niederhut, D., Pacer, M. (eds.) *Proceedings of the 17th Python in Science Conference 2018*, pp. 113–120. <https://doi.org/10.25080/Majora-4af1f417-011>