Automatic synchronization of RDF graphs representing ontologies and Wikibase instances*

Alejandro González Hevia[2][0000−0003−1394−5073], Guillermo Facundo Colunga[2][0000−0003−1283−2763], Emilio Rubiera Azcona[2][0000−0002−0292−9177], and Jose Emilio Labra Gayo[1][0000−0001−8907−5348]

[1] Dpt. of Computer Science, University of Oviedo, Spain labra@uniovi.es
[2] WESO Research Group, University of Oviedo, Spain
{alejgh.weso thewilly.work emilio.rubiera}@gmail.com

Abstract. Version control systems provide ways to ease collaboratively working on a set of source files. Due to the nature of RDF models that represent ontologies, there is a clear benefit on using those kinds of systems, and many well-known datasets and vocabularies are already being worked on collaboratively. However, there are domain experts who could provide feedback and add new knowledge to the data, but do not know how to work with version control systems or with RDF files directly. Wikibase provides a human-friendly interface with an underlying Linked Data model, and can be used as a publication service for the data where those experts can easily browse and add their knowledge. In this paper, we propose a system that automatically synchronizes RDF files hosted in a version control system with a Wikibase instance. We describe the system from an architectural point of view, and explain the main components needed for the synchronization of data. Some of the challenges for an effective synchronization of RDF files with Wikibase are also addressed. This system is currently being used as part of the Hércules project to synchronize a research ecosystem ontology with a Wikibase instance.

Keywords: wikibase · RDF synchronization · RDF diff · wikidata · ontology

1 Introduction

Version Control Systems (VCS) have been used by software developers to facilitate the process of working collaboratively on a set of source files. Developers can greatly benefit from the discussion of individual changes and the establishment of an acceptance workflow. Due to the nature of RDF models representing ontologies, they are dynamic entities which may change as a consequence of different reasons[6]:

– Changes in the conceptualization of concepts.

– Appearance of new information about concepts.
– Changes in the domain being described.

This makes RDF files a perfect fit for use with version control systems. However, some domain experts may not know how to work with Version Control Systems or with RDF files directly, but could still contribute useful information to the ontology creation and conceptualization. These users can work on an easy-to-use RDF browsing and publication service like Wikidata, where they can make modifications to the underlying content through a web interface. The contents of both the VCS and the publication service need to be synchronized, so both ontology engineers and domain experts can contribute to the growth and evolution of the ontology.

In this paper we propose a synchronization system between RDF files, which are hosted on GitHub, and a Wikibase instance. As part of the Hércules project[3], we have implemented this system and it is being used to synchronize an ontology which represents the research ecosystem[4] with a custom Wikibase instance. This instance is not publicly accessible but we have deployed a public copy at https://herc-core.wiki.opencura.com.

We have structured this paper as follows: we will start by briefly defining some of the technologies used by the synchronization system and related work. In section 4, we will give an overview of the synchronization system from an architectural point of view. After this, we will illustrate the flow and operations made on the data in section 5. In section 6, we will go over some of the functionality that we are planning to add as future work to the system. Finally, we will give our final conclusions in section 7.

2 Background

2.1 Wikibase

Wikibase is an open-source knowledge base software-suite which drives Wikidata, and makes collaboration easy for humans and machines alike. Since it uses MediaWiki as its front-end, it provides users with an intuitive and easy to use interface to interact with the underlying data model, based on semantic web concepts[5].

The underlying infrastructure of Wikibase contains a SPARQL engine, which makes use of a BlazeGraph triplestore and allows the exposure of the data within Wikibase as Linked Data. Content negotiation mechanisms are also set up, so a Wikibase entity can be either viewed through the default HTML format displayed by the browser, or in other multiple formats, such as *.json*, *.rdf*, *.ttl* or *.nt*.

Therefore, one of the main strengths of Wikibase is that users working directly with it may not know the details of the underlying data model and infras-

[3] https://www.um.es/web/hercules/
[4] This ontology can be accessed at https://github.com/weso/hercules-ontology

tructure, but can still collaboratively work with the linked data and add new information to it.

2.2 Wikidata Integrator (WDI)

Wikidata Integrator is a Python library that serves as a wrapper around the Wikidata API, providing tools to programmatically modify its contents. It consists of the common code modules from a series of computer programs that uploaded biomedical knowledge to Wikidata, and attempts to simplify the process of creating Wikidata bots [7]. The synchronization system that we have proposed relies on WDI to easily communicate with the Wikibase API and update its underlying contents.

2.3 WebHooks

WebHooks are POST requests that are sent to a given URL, which can be configured by the user, in response to a certain event that has occurred [4]. GitHub provides support for WebHooks[5], which can be triggered when certain GitHub-related events occur. This allows external systems to subscribe to certain events from organizations or repositories, and to act accordingly when they receive notifications from them. Our system makes use of WebHooks to receive information about the RDF files when a new release is made on the repository where they are hosted.

3 Related work

There are several ontology editors like the popular Protégé[6], which also offers a collaborative web environment called WebProtégé[7]. WebProtégé provides useful features for collaborative ontology development, like user permissions and change history. A difference between that approach and the proposal presented in this paper is that we can rely on GitHub project management tools like issues, releases and pull requests. Furthermore, with the use of Wikibase as a publication platform for the ontology we can also have access to the additional functionality from the software suite defined previously. With the use of both GitHub and Wikibase we aim to facilitate the collaboration of external contributors to the evolution of the ontology.

The problem of making an effective and efficient diff between two given RDF graphs has been for a long time a field of study in the Semantic Web community. In [1], Berners Lee and Connolly give an overview on the problems that may arise when comparing two RDF graphs and obtaining their diff. To avoid arbitrary choices in the serialization of RDF graphs - specially regarding blank nodes - there is a need to perform a canonicalization algorithm on the given graphs.

[5] https://developer.github.com/webhooks/
[6] https://protege.stanford.edu/
[7] https://webprotege.stanford.edu/

The W3C Semantic Web wiki maintains a list with various tools that can be used to compare RDF graphs[8]. In our system we are using one of those tools, RDFLib [3], to build and compare the RDF graphs obtained from the Version Control System. RDFLib provides the method `rdflib.compare`, which implements a port of the RDF graph isomorphism tester made by Sean B. Palmer[9], to perform the comparison.

4 System overview

Fig. 1 shows an overview of the synchronization system presented in this paper. The initial step is the modification of RDF files by an ontology engineer, and the push of those changes to a VCS (in our case, GitHub). After a new release is created, the WebHook associated with the repository is called with the changes regarding this new release. These changes are passed to the synchronization system, which will be in charge of updating this information to a Wikibase.

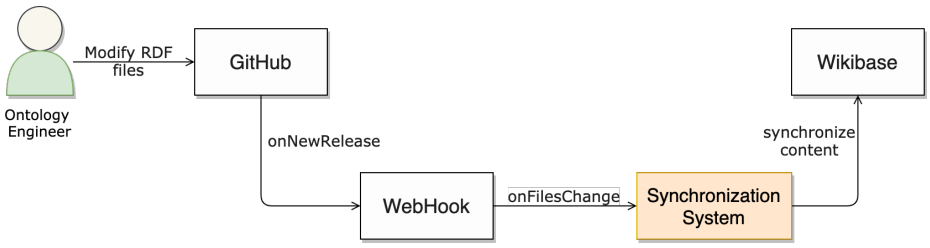

Fig. 1. Complete overview of the synchronization system.

Although the complete workflow involves the use of GitHub and WebHooks to synchronize the content to a Wikibase, it is important to note that this is not necessary, and the synchronization system can be invoked independently to synchronize any given RDF data to a Wikibase[10].

5 Data flow

In this section we are going to go through the flow of data and inner workings of the synchronization system, from the modification of an RDF file to its synchronization to a Wikibase.

[8] https://www.w3.org/2001/sw/wiki/How_to_diff_RDF

[9] The code for this algorithm can be found at https://www.w3.org/2001/sw/DataAccess/proto-tests/tools/rdfdiff.py

[10] For instance, the UniTest Wikibase, available at https://unitest.wiki.opencura.com, has been populated using just the synchronization system and an RDF file.

5.1 Modifying an RDF file

The overall process begins with the modification of an RDF file. These changes in the file may be distributed amongst many commits in the VCS. In Fig. 2 we provide an example where a subset of an original file is changed multiple times. Some new triples are added to the graph, while others are removed or modified.

Original file

```
:knows owl:inverseOf :isKnownBy ;
       a owl:ObjectProperty
:alice :knows :bob, :carol .
```

Commit A

```
:knows owl:inverseOf :isKnownBy ;
       rdfs:range :Person ;
       a owl:ObjectProperty .
:alice :knows :bob, :carol ;
       rdfs:label "Alice" .
```

Commit B

```
:knows owl:inverseOf :isKnownBy ;
       rdfs:range :Person ;
       a owl:ObjectProperty .
:alice :knows :bob, :carol ;
       rdfs:label "Alice" .
```

Final file

```
:knows owl:inverseOf :isKnownBy;
       rdfs:range :Person ;
       a owl:ObjectProperty .
:alice :knows :bob ;
       rdfs:label "Alice" .
```

Fig. 2. Example evolution of an RDF file.

5.2 Making a new release

After a number of commits have been performed and the administrators of the repository decide that the new data is ready to be synchronized and published, a new release in GitHub can be created[11]. After that, the WebHook will be triggered and will notify the synchronization system with information about this new event.

5.3 Obtaining the graph diff

Once the synchronization system has received this notification from the WebHook, it will download the original and final RDF files from the repository. These files will then be parsed as graphs using the *RDFLib* library, and compared to obtain the diff between them.

5.4 Creating the synchronization operations

After the diff between the two graphs is known, the next step is the creation of the operations that need to be executed on Wikibase to synchronize the changes.

An important step when creating the synchronization operations is to infer the types of each resource that will be added to the Wikibase. First of all, when

[11] https://help.github.com/en/enterprise/2.13/user/articles/creating-releases

synchronizing an IRI node we need to know if that node represents a Wikibase property (Px) or an item (Qx) before executing any operations on the Wikibase. Sometimes it is trivial to know when a node corresponds to a given entity type: for example, all the IRIs that are used as a predicate in an RDF triple are properties. However, there are cases when some kind of type of inference needs to be performed (e.g. exploring `rdfs:subPropertyOf` predicates in the graph to identify the properties).

After we know which IRIs correspond to classes and properties, we also need to know which are the ranges of the properties. When creating a new property in Wikibase it is necessary to indicate its data type. Modifying it after its creation, although possible, is not an expected operation and should be done with care. When a property is used as a predicate in a triple, we rely on the type of the object to infer the datatype of the property. If the property is not used as a predicate in the whole graph, we also rely on the `rdfs:range` predicates available for the property. There may be times when the datatype of a property may not be inferred with the information available in the graph. For those cases, we expect its range to be '`wikibase-item`' by default.

5.5 Performing the operations on Wikibase

Once the operations have been created, the next step is to execute them on the target Wikibase. Since those operations provide general information to be executed on multiple triplestores, this step also includes some Wikibase-specific tasks:

- **Handling labels, descriptions and aliases**. An important aspect of the Wikibase data model to take into account for the synchronization is its internal entity representation[12]. The description of a node is composed of the following mappings to RDF:
  - Labels are defined as `rdfs:label`, `schema:name` and `skos:prefLabel` predicates.
  - Aliases are defined as `skos:altLabel` predicates.
  - Descriptions are defined as `schema:description` predicates.

  All of the predicates above have as objects language-tagged literals. When we encounter any of those predicates in the synchronization process, they are used as label, alias or description values for the given entity.
- **Mapping datatypes**. Although in the creation of synchronization operations we have already annotated the properties with their respective datatypes, there still needs to be a mapping between those datatypes and the ones from the Wikibase data model. For example, coordinates expressed in an RDF file using the `geo:wktLiteral` datatype are mapped to the Wikibase `globe coordinate` datatype, and the values from literals must be processed to fit with the format specified by Wikibase.

[12] For more information, see https://bit.ly/2PBU6Fo

– **Blank nodes**. Blank nodes can be defined as existential variables, representing the existence of some unnamed resource [2]. The approach taken for the synchronization of blank nodes consists of representing each blank node as a Wikibase item, with its label being the UID generated by `rdflib` in the graph canonicalization phase. However, it is important to note that the time to canonicalize blank nodes may increase exponentially on large graphs.

6 Future work

6.1 Synchronizing changes from Wikibase to the RDF files

We have seen so far the point of view of ontology engineers making changes directly on the files in the GitHub repository and reflecting those changes on the Wikibase system. However, the inverse direction is also possible.

An initial approach to solve this type of synchronization is illustrated in Fig. 3. Tasks related to the inverse synchronization are marked in blue, while other tasks already present in the system are greyed out. After the initial modification from a domain expert has been saved, we could make use of the available Wikibase hooks[13] and notify the synchronization system with the changes made to the item from Wikibase. After that, the system would create a new pull request in the original repository that administrators can review before synchronizing the new changes.

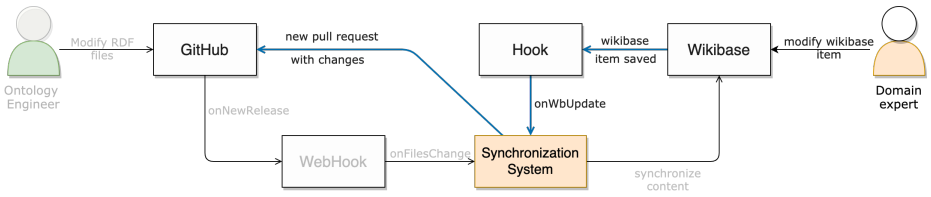

Fig. 3. Complete overview of the system with inverse synchronization functionality.

6.2 Qualifiers and references

Both qualifiers and references are important parts of the Wikibase data model: the former allow the refinement of values from properties in a given statement, while the latter are used to state which are the sources of a claim. Although the synchronization of both elements is not implemented yet, we are currently working on defining a model that can be used to synchronize them.

[13] https://doc.wikimedia.org/Wikibase/master/php/md_docs_topics_hooks-js.html

6.3 Support additional datatypes

Although there is already a initial mapping of XML Schema datatypes to Wikibase ones, some of them are not yet mapped. To be specific, there is still no support for external identifiers, mathematical expressions, musical notations and lexemes.

7 Conclusions

In this demo paper we have presented a synchronization system between RDF files and a Wikibase instance. The system allows to browse ontologies defined as RDF data through the Wikibase user-friendly interface, and allows domain experts to add knowledge to them without knowing the underlying data model. The system is already being used as part of the Hércules project to synchronize a research management ontology with a Wikibase, and has also been used to populate several Wikibase instances hosted on WBStack.

However, there is still additional work that needs to be done to allow a complete synchronization between the Wikibase instance and the repository where the RDF files are stored. The complete documentation of the system from an architectural point of view can be accessed at http://www.weso.es/hercules-sync.

Acknowledgements

The HÉRCULES Semantic University Research Data Project is backed by the Ministry of Economy, Industry and Competitiveness with a budget of 5.462.600,00 euros with an 80% of cofinancing from the 2014-2020 ERDF Program. This work has been partially funded by the Spanish Ministry of Economy and Competitiveness (Society challenges: TIN2017-88877-R).

References

1. Berners-Lee, T., Connolly, D.: Delta: an ontology for the distribution of differences between rdf graphs. World Wide Web, http://www. w3. org/DesignIssues/Diff **4**(3), 4–3 (2004)
2. Hogan, A., Arenas, M., Mallea, A., Polleres, A.: Everything you always wanted to know about blank nodes. Journal of Web Semantics **27**, 42–69 (2014)
3. Krech, D.: Rdflib: A python library for working with rdf (2006)
4. Leggetter, P.: What are webhooks and how do they enable a real-time web? (2012)
5. Malyshev, S., Krötzsch, M., González, L., Gonsior, J., Bielefeldt, A.: Getting the most out of wikidata: semantic technology usage in wikipedia's knowledge graph. In: International Semantic Web Conference. pp. 376–394. Springer (2018)
6. Noy, N.F., Klein, M.: Ontology evolution: Not the same as schema evolution. Knowledge and information systems **6**(4), 428–440 (2004)
7. Waagmeester, A., Stupp, G., Burgstaller-Muehlbacher, S., Good, B.M., Griffith, M., Griffith, O.L., Hanspers, K., Hermjakob, H., Hudson, T.S., Hybiske, K., et al.: Science forum: Wikidata as a knowledge graph for the life sciences. ELife **9**, e52614 (2020)