A *Protégé* Plugin with *Swift Linked Data Miner*

Jedrzej Potoniec and Agnieszka Ławrynowicz

Faculty of Computing, Poznan University of Technology
ul. Piotrowo 3, 60-965 Poznan, Poland
{jpotoniec,alawrynowicz}@cs.put.poznan.pl

Abstract. We present a *Protégé* plugin implementing *Swift Linked Data Miner*, an anytime algorithm for extending an ontology with new subsumptions. The algorithm mines an RDF graph accessible via a SPARQL endpoint and proposes new SubClassOf axioms to the user.

1 Introduction

It is not uncommon for a Linked Data dataset to provide only a very shallow ontology [3], which does not cover more complex aspects of the underlying conceptual model. On the other hand, the data in the dataset follows the model and thus the model is reflected in the data in a form of patterns. Such patterns can be detected using pattern mining techniques and used to extend the ontology with new knowledge.

To address this use case, we developed *Swift Linked Data Miner* (SLDM) [8]. SLDM is a pattern mining algorithm, which can discover new partial definitions, i.e. SubClassOf axioms, for a given class. The mined axioms are expressed in OWL 2 EL [6]. SLDM is an anytime algorithm, which means that it delivers patterns once they are mined and then refines them. In other words, the longer the algorithm works, the more complex patterns are mined. The algorithm does not require an access to the whole RDF graph at the same time. Instead, it downloads on-demand necessary parts by querying the SPARQL endpoint. To avoid issues with high load put on the endpoint by a complex query, the queries are very simple, consisting only of a single triple pattern and a VALUES clause. Direct usage of a SPARQL endpoint without overloading it is the main difference between SLDM and former approaches to mining ontologies from RDF graphs (e.g. [9,2]), which require accessing the whole graph at the same time or posing complex SPARQL queries to the endpoint.

2 Plugin Description

Swift Linked Data Miner was implemented as a *Java* library using *Apache Jena* [5] to interact with a SPARQL endpoint and *OWL API* [4] to deliver mined axioms. On top of the library, we built a plugin for *Protégé* [7] simplifying the use

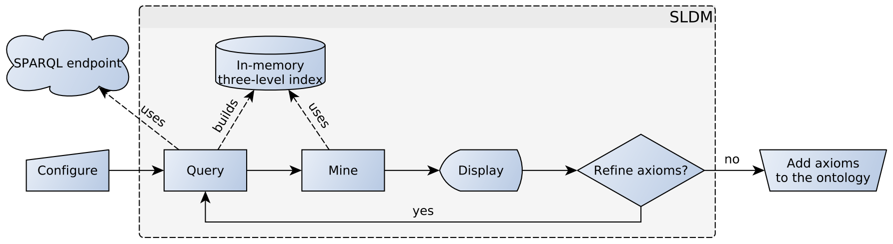


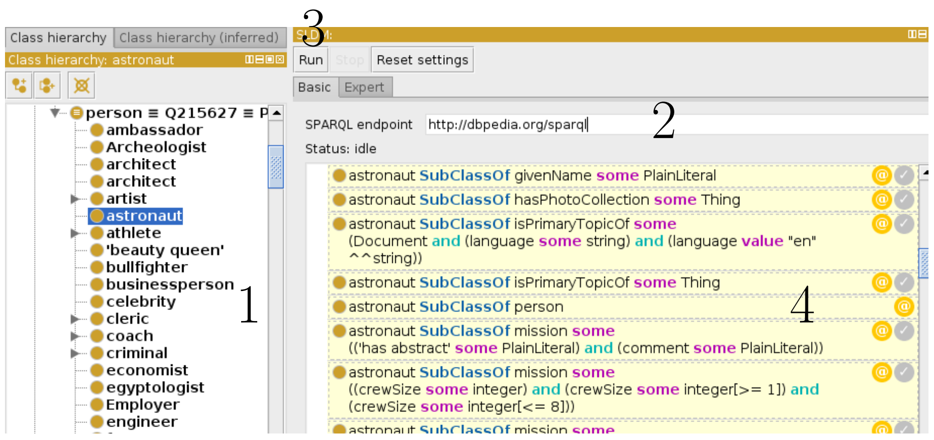
Fig. 1. A typical workflow with the plugin. The user configures the plugin by specifying a class and an address of a SPARQL endpoint, then SLDM is run with alternating phases of querying the endpoint and mining obtained triples. The axioms are displayed right after they are mined, and the user can add them to the ontology.

of SLDM to just few clicks. Both the source code and a JAR file with the plugin are available in a *Git*[1] repository at `https://bitbucket.org/jpotoniec/sldm`.

A typical workflow with the plugin is presented in Figure 1. First, the user configures SLDM with the basic interface of the plugin (Figure 2) by choosing a class in the class hierarchy view and entering the address of the SPARQL endpoint, and starts SLDM with *Run* button. SLDM constructs a set of URIs belonging to the selected class, e.g. `Astronaut`, by posing a SPARQL query with a triple pattern `?x a Astronaut` and then operates in two alternating phases of querying the endpoint and mining the obtained triples. In the first phase, the endpoint is queried about all the triples having an URI from the set in a subject position, by using the following WHERE clause: `?s ?p ?o . VALUES ?s` {`<<URIs>>`}. The triples are then organized into a three-level index, with predicates in the first level, objects in the second and subjects in the third. Such an order enables efficient access e.g. to all URIs occurring in triples with predicate `rdf:type` and object `Person`. During the second phase, the index is scanned to discover the axioms, e.g. if for predicate `rdf:type` and object `Person` there are many subjects in the third level, an axiom `Astronaut` SUBCLASSOF `Person` is mined. If, for a given predicate, no pattern can be found, the corresponding subjects are used as an input to the first phase of SLDM, to mine more complex axioms. The mined axioms are displayed using a standard *Protégé* interface. An axiom is accompanied there by two buttons: the one with `@` symbol to display additional information about the axiom (e.g. corresponding value of the measure used by SLDM) and the one with ✓ to add the axiom to the ontology.

Figure 3 presents the *Expert* interface of the plugin, where the user can fine-tune parameters of SLDM. Field *Minimal support* sets up a minimal value of support (i.e. the measure used during the mining) which an axiom must achieve in order to be presented to the user. Field *Maximum level* specifies a maximal number of nested SOME expressions in a mined pattern. Field *Ignored properties* enables the user to use regular expressions to specify a set of predicates to ignore,

[1] `https://git-scm.com/`



```
person ≡ Q215627 ≡ P
  ambassador
  Archeologist
  architect
  architect
  artist
  astronaut
  athlete
  'beauty queen'
  bullfighter
  businessperson
  celebrity
  cleric
  coach
  criminal
  economist
  egyptologist
  Employer
  engineer
```

```
SPARQL endpoint  http://dbpedia.org/sparql
Status: idle

astronaut SubClassOf givenName some PlainLiteral
astronaut SubClassOf hasPhotoCollection some Thing
astronaut SubClassOf isPrimaryTopicOf some
(Document and (language some string) and (language value "en"
^^string))
astronaut SubClassOf isPrimaryTopicOf some Thing
astronaut SubClassOf person
astronaut SubClassOf mission some
(('has abstract' some PlainLiteral) and (comment some PlainLiteral))
astronaut SubClassOf mission some
((crewSize some integer) and (crewSize some integer[>= 1]) and
(crewSize some integer[<= 8]))
astronaut SubClassOf mission some
```

Fig. 2. A basic view of SLDM Tab in *Protégé*. The user selects a class in the class hierarchy (1), enters an address of the SPARQL endpoint (2) and clicks *Run* (3). The mined axioms are presented in the main part of the window (4).

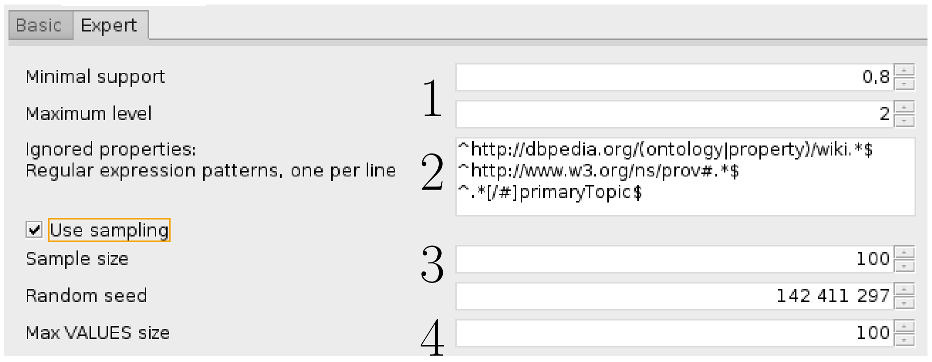
e.g. if a predicate conveys provenance information and is not directly relevant to the semantics of the selected class. If the graph in the SPARQL endpoint is big, it may be useful to use random sampling to decrease the load and increase the speed of the mining. Field *Sample size* allows the user to set how many different URIs from a given class will be considered. To ensure repeatability of sampling, field *Random seed* enables the user to set a seed for a random number generator. Field *Max VALUES size* limits the number of URIs specified in a VALUES clause of a single query.

3 Proposed Demo

During the demo we will present how to use the plugin using the *DBpedia* ontology [1] and a SPARQL endpoint with the *DBpedia* dataset loaded. To ensure smooth operation, we will provide our own SPARQL endpoint. We will also discuss how various settings of the parameters affect the obtained axioms. The interested attendees will be able to use the plugin to mine an ontology of their choice. A short video similar to what will be presented during the demo is available at `https://www.youtube.com/watch?v=ENdNQ8ES1Ek`.

4 Conclusion

In this paper, we presented a plugin to *Protégé* enabling the user to discover new SubClassOf axioms directly from an on-line Linked Data dataset accessible via a SPARQL endpoint using *Swif Linked Data Miner* [8]. The basic configuration of the plugin is very simple and does not require prior expertise in data mining from the user. The plugin along with its source code is freely available at `https://bitbucket.org/jpotoniec/sldm`.



Minimal support: 0,8
Maximum level: 2
Ignored properties:
Regular expression patterns, one per line:

```
^http://dbpedia.org/(ontology|property)/wiki.*$
^http://www.w3.org/ns/prov#.*$
^.*[/#]primaryTopic$
```

☑ Use sampling
Sample size: 100
Random seed: 142 411 297
Max VALUES size: 100

Fig. 3. An expert of view of SLDM Tab in *Protégé*. The user can configure the algorithm (1), list of ignored predicates (2), sampling (3) and a maximal number of URIs in a VALUES clause of queries (4).

Acknowledgement. Jedrzej Potoniec acknowledges the support from the Polish National Science Center (Grant No 2013/11/N/ST6/03065). This work was partially supported by the PARENT-BRIDGE program of Foundation for Polish Science, co-financed from European Union, Regional Development Fund (Grant No POMOST/2013-7/8). Agnieszka Ławrynowicz acknowledges the support from the Polish National Science Center (Grant No 2014/13/D/ST6/02076).

References

1. Bizer, C., Lehmann, J., et al.: DBpedia - A crystallization point for the Web of Data. J. Web Sem. 7(3), 154–165 (2009)
2. Bühmann, L., Lehmann, J.: Pattern based knowledge base enrichment. In: Alani, H., Kagal, L., et al. (eds.) The Semantic Web - ISWC 2013. Lecture Notes in Computer Science, vol. 8218, pp. 33–48. Springer (2013)
3. Glimm, B., Hogan, A., Krötzsch, M., Polleres, A.: OWL: yet to arrive on the web of data? In: LDOW. CEUR Workshop Proceedings, vol. 937. CEUR-WS.org (2012)
4. Horridge, M., Bechhofer, S.: The OWL API: A java API for OWL ontologies. Semantic Web 2(1), 11–21 (2011)
5. McBride, B.: Jena: A semantic web toolkit. IEEE Internet Computing 6(6), 55–59 (2002)
6. Motik, B., Grau, B.C., Horrocks, I., Fokoue, A., Wu, Z.: OWL 2 web ontology language profiles (second edition). W3C recommendation, W3C (Dec 2012), http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/
7. Noy, N.F., Sintek, M., et al.: Creating Semantic Web Contents with Protégé-2000. IEEE Intelligent Systems 16(2), 60–71 (2001)
8. Potoniec, J., Jakubowski, P., Ławrynowicz, A.: Swift Linked Data Miner: Anytime Algorithm for Mining OWL 2 EL Class Expressions Directly from On-Line Linked Data, submitted to the J. of Web Semantics, available at `https://goo.gl/HFghXp`
9. Völker, J., Niepert, M.: Statistical schema induction. In: Antoniou, G., Grobelnik, M., et al. (eds.) The Semantic Web: Research and Applications. Lecture Notes in Computer Science, vol. 6643, pp. 124–138. Springer (2011)