Web Semantics: Science, Services and Agents on the World Wide Web ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Swift Linked Data Miner: Mining OWL 2 EL class expressions directly from online RDF datasets

Jędrzej Potoniec *, Piotr Jakubowski, Agnieszka Ławrynowicz

Faculty of Computing, Poznan University of Technology, ul. Piotrowo 3, 60-965 Poznań, Poland

ARTICLE INFO

Article history:
Received 6 July 2016
Received in revised form 15 May 2017
Accepted 1 August 2017
Available online xxxx

Keywords:
Linked Data
Online linked data mining
Ontology learning
OWL 2 EL
RDF Data Shapes
Protégé plugin

MSC:
68T05
68T10
68T27
68T30

ABSTRACT

In this study, we present Swift Linked Data Miner, an interruptible algorithm that can directly mine an online Linked Data source (e.g., a SPARQL endpoint) for OWL 2 EL class expressions to extend an ontology with new SubClassOf: axioms. The algorithm works by downloading only a small part of the Linked Data source at a time, building a smart index in the memory and swiftly iterating over the index to mine axioms. We propose a transformation function from mined axioms to RDF Data Shapes. We show, by means of a crowdsourcing experiment, that most of the axioms mined by Swift Linked Data Miner are correct and can be added to an ontology. We provide a ready to use *Protégé* plugin implementing the algorithm, to support ontology engineers in their daily modeling work.

1. Introduction

It is reasonable to assume that a Resource Description Framework (RDF [1]) graph is not a random set of triples. There is some reason for the graph to have this specific shape: some underlying data model or a process generating the graph. An element of the model is reflected as a pattern in the graph. The idea of Swift Linked Data Miner (SLDM) is to make use of this observation backward: if there is a pattern in the graph, then there is a good amount of chance that there should be a corresponding element in the model.

One of possible ways to express such a model is to generate an ontology. To express an ontology for an RDF graph, one can employ RDF Schema [2] or aim for some variant of OWL 2 (Web Ontology Language [3]). RDF Schema offers just a tiny bit of expressive power, so we decided to choose OWL 2 instead. One may argue that OWL 2 has exactly the opposite problem, as it is a very expressive language and thus reasoning complexity is unbearable. Fortunately, there are three OWL 2 profiles: OWL 2 QL, OWL 2 RL, and OWL 2 EL, all of these provide polynomial-time reasoning algorithms. According to the OWL 2 specification [4], the first two are suitable for lightweight ontologies. However, OWL 2 EL is a profile tailored specifically to deal with very large ontologies. It is our intuition that one can easily get a large ontology if they do not have to develop it by hand, but has it data mined instead. It must be noted here that the OWL Lite profile, defined for OWL [5] and inherited by OWL 2 has exponential reasoning complexity [6]. Taking all these into account, we decided that our primary way to express the model is an OWL 2 EL ontology, which we introduce in Section 3.

In the age of Big Data, we cannot assume anymore that we can have a direct access to a graph all the time and fit the whole graph to the RAM. We must be able to process the graph chunk at a time, sample it, and retrieve the chunks from a remote location. Swift Linked Data Miner was developed with all this in mind. In Section 4, we show how we retrieve only a part of an RDF graph at a time and organize the retrieved part into a smart structure to facilitate pattern mining.

In Section 5, we describe SLDM, which can mine an RDF graph to extend an OWL 2 EL ontology. SLDM is composed of many small algorithms, each fitted to mine particular type of patterns. Section 6 provides a theoretical analysis of the algorithm, including both analyses of memory complexity and worst-case computational complexity.

Some users may not be interested in reasoning. Maybe the logical inference in their application is unnecessary and instead they

* Corresponding author.
E-mail addresses: jpotoniec@cs.put.poznan.pl (J. Potoniec), pjakubowski@cs.put.poznan.pl (P. Jakubowski), alawrynowicz@cs.put.poznan.pl (A. Ławrynowicz).

https://doi.org/10.1016/j.websem.2017.08.001
1570-8268/

would like to understand their graph better or validate if a new graph they just received is compatible with the model. Especially to address these issues there is an ongoing work on RDF Data Shapes,[1] and we provide an alternative way to express patterns mined with SLDM. In Section 7, we show how the patterns can be transformed to RDF Data Shapes expressed in Shapes Constraint Language (SHACL) [7].

To facilitate incremental research and enable early adopters to use our ideas, we provide an implementation of SLDM. In Section 8, we present a plugin to *Protégé*, which enables ontology engineer to use SLDM in just few clicks.

To validate SLDM, we planned and conducted a crowdsourcing experiment using arguably the most popular Linked Data resource: *DBpedia* [8]. We used SLDM to mine patterns for multiple classes used in *DBpedia*, translated the patterns to English and asked the contributors of a crowdsourcing platform if these sentences correctly describe the classes. They decided that most of them are indeed correct. Details of the experiment are described in Section 9.1.

In Section 9.2, we show another use-case, based on myExperiment RDF dataset and the associated ontology [9]. We compare the mined patterns with documentation and pragmatics of the dataset. Finally, in Section 9.3, we discuss run-time properties of the algorithm, such as CPU time in function of various parameters of the algorithm.

2. Related work

The idea of automated and semi-automated creating and extending ontologies has been studied by many researchers. These studies can be roughly divided into five areas: (i) ontology learning from text, (ii) interactive ontology learning, (iii) concept learning, (iv) learning from local data, (v) learning from online data. The first area is also the farthest from our work, and thus we will not discuss it. The interested reader is refereed to a comprehensive compendium on ontology learning [10].

One of the prominent ways of interactive ontology learning is an application of formal concept analysis to the Description Logics [11–14]. The aim of these algorithms is to complete an ontology with respect to all subsumptions of a given type, for example, between named classes [11] or between named classes and domain and range restrictions [12]. The algorithms generate all possible subsumption axioms that are consistent with the ontology but do not follow from it, and for each axiom, the user is asked to either add the axiom to the ontology or to provide a counterexample.

In the same area fits the idea of games with a purpose, where a player receives rewards for completing tasks like creating new entities in an ontology, adding types or aligning an ontology with another one. The consensus is obtained by posing the same task to multiple players. A classical example of such a system are *OntoGames* [15]. Similar to the games with a purpose is the idea of using crowdsourcing services like *CrowdFlower*[2] or *Amazon Mechanical Turk*.[3] For example, Hanika et al. present a *Protégé* plugin that enables a user to post microtasks related to her ontology development process directly to a crowdsourcing website [16].

Concept learning by itself is not necessarily an ontology learning approach, as it is concerned with learning class expressions given a set of positive and negative examples. Nevertheless, such an approach can be used to extend an existing ontology with missing definitions. Fanizzi et al. propose DL-FOIL algorithm, which allows for learning class expressions in Description Logics underlying OWL-DL [17,18]. The learning algorithm is based on sequential covering and employs two refinement operators, one for specialization and the other for generalization. Lehmann describes a concept learning software *DL-Learner*,[4] which also employs refinement operators, but with different search strategy [19]. He also describes the application of *DL-Learner* specifically to learning ontologies, along with a *Protégé* plugin implementing the idea [20]. An approach using a refinement operator with background knowledge to refine SPARQL queries was proposed by awrynowicz and Potoniec [21]. These queries are further used as binary features for a classical machine-learning classification algorithm. Due to certain properties of these SPARQL queries, they can be immediately transformed to OWL class expressions, and thus also used in ontology learning, as discussed in a previous study [22]. Galárraga et al. employ techniques known from Inductive Logic Programming on top of their in-memory RDF store to mine association rules with variables [23]. Such rules could be then transformed into an ontology.

The idea of learning ontological axioms from a static, fully-available dataset is related to various forms of data mining in relational databases. For example, Fu and Han propose a framework for mining association rules in relational databases guided by constrains on a shape of mined rules [24]. By setting these constrains appropriately, one could obtain an ontology. Völker et al. propose an algorithm for mining an OWL 2 EL ontology from scratch [25,26]. The input to the algorithm is an RDF graph, which is first transformed into a set of database tables, and then association rule mining is employed to discover the ontological axioms. Instead of considering an RDF graph and an ontology describing it, one could tackle the problem of extending an ontology using individuals contained in it. Another study provides a method for learning general class inclusions (GCIs) that takes into account results of reasoning with the ontology [27].

Finally, one may want to extend an ontology using data that are not entirely available at hand but are available only in for example, remote SPARQL endpoints [28]. To the best of our knowledge, there is the only one work that has addressed this problem so far [29]. It is a top-down method, which first performs data mining on a repository of ontologies to build a library of patterns and then the patterns are used to form SPARQL queries, which are posed to a SPARQL endpoint to discover axiom candidates. Unfortunately, the queries are computationally expensive for the endpoint, due to the heavy usage of GROUP BY and COUNT DISTINCT clauses.

3. Preliminaries

3.1. OWL 2 EL

OWL 2 is a language designed to describe information about entities and relations between them. The language provides formally defined semantics and decidable reasoning procedures [3]. OWL 2 EL is a subset of OWL 2, tailored to support applications employing very large ontologies while having typical reasoning tasks tractable [30]. For example, this subset is used by a large clinical health ontology SNOMED CT [31].

Throughout this work we write OWL expressions in Manchester Syntax [32]. We also use a set of well-known prefixes: `rdf:` for the namespace http://www.w3.org/1999/02/22-rdf-syntax-ns#, `rdfs:` for http://www.w3.org/2000/01/rdf-schema#, `owl:` for http://www.w3.org/2002/07/owl# and `xsd:` for http://www.w3.org/2001/XMLSchema#.

We start by defining an *OWL 2 EL class expression*. Let A be a named class, DT a datatype, p a named object property, and r a named data property. Moreover, a denotes an individual and l a

[1] https://www.w3.org/2014/data-shapes/charter.

[2] http://www.crowdflower.com.

[3] http://www.mturk.com.

[4] http://dl-learner.org.