HANNE - A Holistic Application for Navigational Knowledge Engineering

Sebastian Hellmann, Jörg Unbehauen, Jens Lehmann

AKSW Research Group, http://aksw.org, Universität Leipzig, Germany
lastname@informatik.uni-leipzig.de

Abstract. Although research towards the reduction of the knowledge acquisition bottleneck in ontology engineering is advancing, a central issue remains unsolved: Light-weight processes for collaborative knowledge engineering by a massive user base. In this demo, we present HANNE, a holistic application that implements all necessary prerequisites for Navigational Knowledge Engineering and thus reduces the complexity of creating expressive knowledge by disguising it as navigation. HANNE enables users and domain experts to navigate over knowledge bases by selecting examples. From these examples, formal OWL class expressions are created and refined by a scalable Iterative Machine Learning approach. When saved by users, these class expressions form an expressive OWL ontology, which can be exploited in numerous ways: as navigation suggestions for users, as a hierarchy for browsing, as input for a team of ontology editors.

1 Introduction

Over the past years, structured data has become widely available. Still, the retrieval of dedicated knowledge for given applications or research questions out of these data sources remains a tedious process. A domain expert might have a very precise idea of the concepts she would like to retrieve from a knowledge source. Yet, she faces a number of challenges when trying to retrieve corresponding examples out of a particular data set.

Due to their sheer size, users of these knowledge bases can hardly know which identifiers are used and are available for the construction of queries. Furthermore, domain experts might not be able to express their queries in a structured form at all, but they often have a very precise imagination what kind of results they would like to retrieve. A historian, for example, searching in DBpedia [2] for ancient Greek law philosophers influenced by Plato can easily name some examples and if presented a selection of prospective results she will be able to quickly identify false results. However, she might not be able to efficiently construct a formal query adhering to the large DBpedia knowledge base a priori.

The construction of queries asking for objects of a certain kind contained in an ontology, such as in the previous example, can be understood as a class construction problem: We are searching for a class expression which subsumes

exactly those objects adhering to our informal query (e.g. ancient Greek law philosophers influenced by Plato [1]).

In recent years, several methods have been proposed for constructing ontology classes by means of Machine Learning techniques from positive and negative examples (see [3] for an overview). Due to their dependency on reasoning methods, these techniques are tailored for small and medium size knowledge bases and cannot be directly applied to large knowledge bases. The scalability of the algorithms is ensured, however, by reasoning only over "interesting parts" of a knowledge base for a given task [1]. As a result users of large knowledge bases are empowered to construct queries by iteratively providing positive and negative examples to be contained in the prospective result set.

In this paper, we present HANNE - a Holistic Application for Navigational kNowledge Engineering. HANNE allows for the extraction of formal definitions of user-defined concepts and the corresponding examples out of arbitrary and possibly large RDF data sets. Based on initial examples given by the user, HANNE learns a formal OWL Class Expression of the concept that the user is interested in. This expression is converted into a SPARQL query[2] and passed to a triple store database with reasoning capabilities. The results are gathered and presented to the user to choose more examples, to refine the query, and to improve the formal definition at will.

Our tool, available online at `http://hanne.aksw.org`, addresses and circumvents the barriers to the acquisition of knowledge out of data sets: (1) it does not need any deployment and provides a user interface in a familiar surrounding, the browser, (2) the meaning of the identifiers used in the knowledge source is made explicit by the tool, and, finally, (3) the application uses OWL; the results are thus represented in a readable, portable and sustainable way.

2 Example Usage

At the time of writing, the DBpedia ontology class `http://dbpedia.org/ontology/Country` contained 2505 instances, including all current countries as well as all historic countries, most of which ceased to exist nowadays.

On April 27th, 2010, there has been a discussion on the DBpedia mailing list[3] on how to retrieve (via SPARQL) a list of current countries only, as the coverage of the OWL class was obviously too imprecise (or its definition was too broad). One suggested solution[4] by a DBpedia expert was to manually include a filter for *dbo:dissolutionYear*[5] within the SPARQL query.

[1] technically we mean OWL class expressions such as *AncientGreekPhilosopher and influencedBy value Plato* in Manchester OWL Syntax `http://www.w3.org/TR/owl2-manchester-syntax/`

[2] `http://www.w3.org/TR/rdf-sparql-query/`

[3] `http://www.mail-archive.com/dbpedia-discussion@lists.sourceforge.net/msg01652.html`

[4] `http://www.mail-archive.com/dbpedia-discussion@lists.sourceforge.net/msg01658.html`

[5] `http://prefix.cc/dbo`

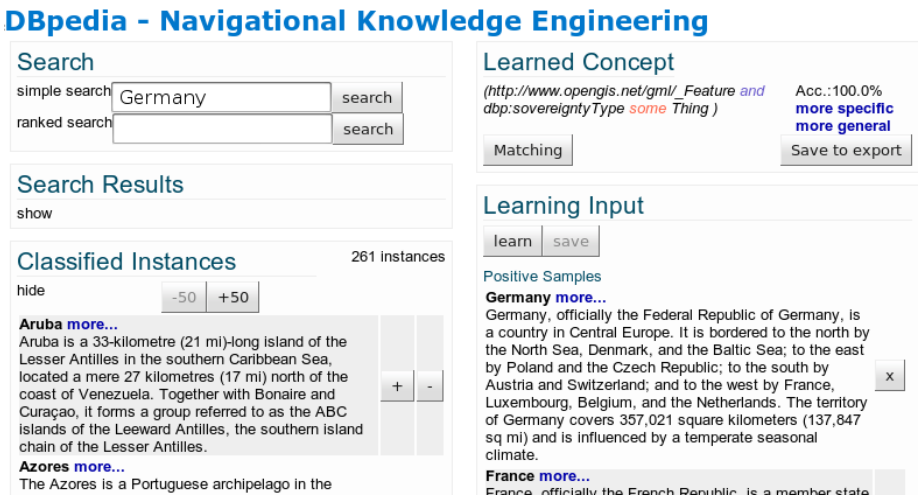

Fig. 1. Screenshot of the left and middle part of `http://hanne.aksw.org`: Real countries in DBpedia. (The right part containing a list of stored concepts and additional features is omitted for a larger image and readability)

Although, the request (originally posted by a DBpedia user) was answered, two shortcomings remain: 1. The answer was not recorded or documented in a sustainable way (e.g. incorporated as OWL class within the ontology) 2. The process of finding the answer was very tedious for the user. He had to wait several days and required the help of an ontology expert that was familiar with the existing vocabulary.

In the following, we will explain step by step, how an OWL class (named e.g. *Real_Country*) can be created without hardly any effort and previous knowledge with HANNE. On the left side of Figure 1, a full text search over the DBpedia data set can be conducted. This represents the entry point, as initial examples have to be chosen to bootstrap the learning process. In our case, a user could start by searching for "Germany". From the search result, she picks *Germany* as a positive example and *East Germany*, *West Germany*, *Nazi Germany* as negatives. After she has pressed the *learn* button (middle, above given examples) a formal OWL definition (in Manchester OWL Syntax) is presented in the top middle (*Learned Concept*) in this case *http://www.opengis.net/gml/_Feature and dbp:sovereigntyType some Thing* . She now has two options on how to proceed: 1. if she finds the learned concept adequate, she can label (e.g. *Real_Country*), comment (*Countries, which are officially accepted and still exist*) and save it to export a complete list of instances 2. Alternatively, she can retrieve instances matching the learned OWL class, which are then displayed on the left side *Classified Instances*. These instances can be further evaluated and more positive and negative examples can be chosen to iterate the process. In our case, a total of 261

instances adhere to the class definition, a quite accurate list (manually checked, including some cases, such as the Azores or the Isles of Man, which are arguable).

3 Overview of the Application

The application[6] realizes a holistic approach to Navigational Knowledge Engineering, as it combines navigational features with knowledge engineering capabilities. It is implemented in Java based on the Google Web Toolkit[7] and is made up of highly configurable and extensible Spring components, so that it can be customized and tailored for certain data sets. The default implementation of the component interfaces is held generic and works on arbitrary SPARQL endpoints with RDFS-Reasoning capabalities[8]. The full text search (Figure 1 left side) is based on a configurable SPARQL template engine. For learning OWL class expressions, DL-Learner[9][3] is used. [1] describes the underlying technique of machine learning on large knowledge bases and contains performance measurements (especially on DBpedia) with acceptable speed for a web scenario.

To help users understand the meaning of learned class expression, labels and comments are displayed in a tooltip, when hovering over a named class or property. Advanced users or ontology editors can also manually alter the class expression by selecting suggested classes, which are either more special or more general than the currently learned example. Whenever the learned class expression changes an additional reasoner is queried and shows related concepts from the formerly saved class expressions, which are either sub-, super-, or sibling classes. All saved classes can be browsed and loaded by all users to further refine searches. If all classes are exported in bulk, they form a class hierarchy, which can be utilized as additional schema for browsing or as input for a team of ontology engineers.

At the time of writing, we configured the Web demo for DBpedia and a Linguistic data set, but plan to increase the number of available knowledge bases.

References

1. Sebastian Hellmann, Jens Lehmann, and Sören Auer. Learning of OWL class descriptions on very large knowledge bases. *IJSWIS*, 5(2):25–48, 2009.
2. Jens Lehmann, Chris Bizer, Georgi Kobilarov, Sren Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009.
3. Jens Lehmann and Pascal Hitzler. Concept learning in description logics using refinement operators. *Machine Learning journal*, 78(1-2):203–250, 2010.

[6] source code available at `http://nlp2rdf.googlecode.com`

[7] `http://code.google.com/webtoolkit`

[8] our local mirror of DBpedia used in the demo is Virtuoso based `http://virtuoso.openlinksw.com/`

[9] `http://dllearner.org`