LOD2 Deliverable D3.3.2: Release of Knowledge Base Enrichment User Interface

Lorenz Bühmann, Jens Lehmann

Abstract: This prototype deliverable consists of a software release of a JavaScript widget for a knowledge base enrichment user interface and an accompanying deliverable report. The software is open source and can be downloaded at https://github.com/AKSW/EnrichmentUIWidget. A description of the usage of the widget in the ORE project is provided. The deliverable describes the purpose and usage of the widget in detail
Collaborative Project
LOD2 - Creating Knowledge out of Interlinked Data
Project Number: 257943  Start Date of Project: 01/09/2010  Duration: 48 months

Deliverable 3.3.2
Release of Knowledge Base Enrichment User Interface

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due Date of Deliverable</td>
<td>Month 24, 31/08/2012</td>
</tr>
<tr>
<td>Actual Submission Date</td>
<td>31/08/2012</td>
</tr>
<tr>
<td>Work Package</td>
<td>WP3, Knowledge Base Creation, Enrichment and Repair</td>
</tr>
<tr>
<td>Task</td>
<td>T 3.3</td>
</tr>
<tr>
<td>Type</td>
<td>Prototype</td>
</tr>
<tr>
<td>Approval Status</td>
<td>Approved</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Number of Pages</td>
<td>14</td>
</tr>
<tr>
<td>Filename</td>
<td>deliverable-3.3.2.pdf</td>
</tr>
</tbody>
</table>

Abstract: This is a deliverable accompanying a software release on a user interface widget for knowledge base enrichment algorithms.

The information in this document reflects only the author's views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided “as is” without guarantee or warranty of any kind, express or implied, including but not limited to the fitness of the information for a particular purpose. The user thereof uses the information at his/her sole risk and liability.

Project funded by the European Commission within the Seventh Framework Programme (2007 - 2013)
History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Reason</th>
<th>Revised by</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>10/07/2012</td>
<td>Initial Version</td>
<td>Lorenz Bühmann</td>
</tr>
<tr>
<td>0.5</td>
<td>30/07/2012</td>
<td>General Description of Enrichment</td>
<td>Lorenz Bühmann</td>
</tr>
<tr>
<td>0.9</td>
<td>15/08/2012</td>
<td>Peer review</td>
<td>Norman Heino</td>
</tr>
<tr>
<td>1.0</td>
<td>30/08/2012</td>
<td>Address peer review comments</td>
<td>Jens Lehmann</td>
</tr>
</tbody>
</table>

Author List

<table>
<thead>
<tr>
<th>Organization</th>
<th>Name</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>ULEI</td>
<td>Jens Lehmann</td>
<td><a href="mailto:lehmann@informatik.uni-leipzig.de">lehmann@informatik.uni-leipzig.de</a></td>
</tr>
<tr>
<td>ULEI</td>
<td>Lorenz Bühmann</td>
<td><a href="mailto:buehmann@informatik.uni-leipzig.de">buehmann@informatik.uni-leipzig.de</a></td>
</tr>
</tbody>
</table>
Executive Summary

This prototype deliverable consists of a software release of a JavaScript widget for a knowledge base enrichment user interface and an accompanying deliverable report. The software is open source and can be downloaded at https://github.com/AKSW/EnrichmentUIWidget. A description of the usage of the widget in the ORE project is provided. The deliverable describes the purpose and usage of the widget in detail.
# Table of Contents

1 Introduction and Motivation .................................................. 5

2 A General Method for Enrichment with OWL Axioms ................. 6

3 General Description .......................................................... 8
   3.1 Availability ................................................................ 8
   3.2 Implementation ......................................................... 8
   3.3 Layout ........................................................................ 8
   3.4 Configuration Parameters ............................................. 8
   3.5 Scenario ...................................................................... 9

4 Manual .............................................................................. 11
   4.1 Obtaining the Widget ................................................... 11
   4.2 Integrating the Widget .................................................. 11
   4.3 Reporting Bugs .......................................................... 12

5 Pointers ............................................................................ 13

References ............................................................................ 13
List of Figures

1 3-Phase Enrichment Workflow ........................................... 6
2 Integration of the Enrichment Widget into the ORE tool. ............... 9
1 Introduction and Motivation

The Semantic Web has recently seen a rise in the availability and usage of knowledge bases, as can be observed within the DataHub¹ and other repositories. Despite this growth, there is still a lack of knowledge bases that consist of sophisticated schema information and instance data adhering to this schema. Several knowledge bases, e.g. in the life sciences, only consist of schema information, while others are, to a large extent, a collection of facts without a clear structure, e.g. information extracted from databases. The combination of sophisticated schema and instance data would allow powerful reasoning, consistency checking, and improved querying. Schema enrichment, as described in this article, allows to create schemata base based on existing data.²

Example 1.1 As an example, consider a knowledge base containing a property birthPlace and subjects in triples of this property, e.g. Brad Pitt, Angela Merkel, Albert Einstein etc. Enrichment algorithms could, then, suggest that the property birthPlace may be functional and has the domain Person as it is encoded via the following axioms in Manchester OWL syntax³:

ObjectProperty: birthPlace
  Characteristics: Functional
  Domain: Person
  Range: Place
  SubPropertyOf: hasBeenAt

Adding such axiom to a knowledge base can have several benefits: 1.) The axioms serve as documentation for the purpose and correct usage of schema elements. 2.) They improve the application of schema debugging techniques. For instance, after adding the above axioms the knowledge base would become inconsistent if a person has two different birth places due to the functionality axiom. Specifically for the DBpedia² knowledge base, we observed an error in which a person was asserted to be born in Lacrosse, the game, instead of Lacross, the city in the United States. Such errors can be automatically detected when schema information such as the range restriction is present (assuming disjointness of the classes Place and Game). 3.) Additional implicit information can be inferred, e.g. in the above example the birth place of a person can be inferred to be one of the places a person has stayed at. The main benefit of this research is a reduction of the effort of creating and maintaining such schema information.

These enrichment methods were developed in Deliverable 3.3.1 and implemented in the DL-Learner⁴⁵⁶⁸ framework, a general description is given in the next section.

---

¹http://thedatahub.org/
²The approach of not creating schema upfront is sometimes referred to as “grass roots” approach or “after the fact” schema creation.
³For details on Manchester OWL syntax (e.g. used in Protégé, OntoWiki) see http://www.w3.org/TR/owl2-manchester-syntax/.
⁴http://dl-learner.org
There is a large variety of axiom types in OWL, which we support in our enrichment tool. The light-weight learning methods for obtaining enrichment suggestions usually take an entity (a class or property in our case) as input, generate a set of OWL axioms as output and proceed in three phases (see Figure 1):

1. In the first phase, SPARQL queries are used to obtain general information about the knowledge base, in particular we retrieve axioms, which allow to construct the class hierarchy. It can be configured whether to use an OWL reasoner for inferencing over the schema or just taking explicit knowledge into account.\(^5\) Naturally, the schema only needs to be obtained once and can then be re-used by all algorithms and all entities.

2. The second phase consists of obtaining data via SPARQL, which is relevant for learning the considered axiom.

3. In the third phase, the score of axiom candidates is computed and the results returned.

Many of our employed heuristics to suggest axioms are based on counting. For instance, when determining whether a class \(A\) is appropriate as domain of a property \(p\), we count the number of triples using \(p\) in predicate position and the number of subjects in those triples which are instances of \(A\). The latter value is divided by the first value to obtain a score. We illustrate this using a simple example. Let the following triples be given (Turtle syntax):

\[
\text{prefix dbpedia: <http://dbpedia.org/resource/>}.
\text{prefix dbo: <http://dbpedia.org/ontology/>}.
\]

\(^5\)Note that the OWL reasoner only loads the schema of the knowledge base and, therefore, this option usually works even in cases with several hundred thousand classes in our experiments, which used the HermiT reasoner.
In the above example, we would obtain a score of 66.7% (2 out of 3) for the class `dbo:Country` and 100% (3 out of 3) for the class `dbo:PopulatedPlace`\(^6\) as candidates for the range of the property `dbo:currency`.

A disadvantage of using this straightforward method of obtaining a score is that it does not take the support for an axiom in the knowledge base into account. Specifically, there would be no difference between having 100 out of 100 correct observations or 3 out of 3 correct observations.

For this reason, we do not just consider the count, but the average of the 95% confidence interval of the count. This confidence interval can be computed efficiently by using the improved Wald method defined in [1]. Assume we have \( m \) observations out of which \( s \) were successful, then the approximation of the 95% confidence interval is as follows:

\[
\text{max}(0, p' - 1.96 \cdot \sqrt{\frac{p' \cdot (1-p')}{m+4}}) \quad \text{to} \quad \text{min}(1, p' + 1.96 \cdot \sqrt{\frac{p' \cdot (1-p')}{m+4}})
\]

with \( p' = \frac{s+2}{m+4} \)

This formula is easy to compute and has been shown to be accurate in [1].

In the above case, this would change the score to 57.3% (previously 66.7%) for `dbo:Country` and 69.1% (previously 100%) for `dbo:PopulatedPlace`. This indicates that there is not much support for either of those choices in the knowledge base. 100 out of 100 correct observations would score much higher (97.8%). The actual scores for the DBpedia Live[9] as of May 2012 are 99.1% for the class `dbo:PopulatedPlace` and 97.6% for `dbo:Country`.

Note that in this implementation, more general classes in the hierarchy would always score higher. It might be desirable to correct this by slightly penalising very general classes. The drawback of such a penalty could be that the scores would be more difficult to understand for users.

The described method was published in [4] and can be seen as an lightweight complement to the closely related work in the area of statistical schema induction via association rule mining [10], where the whole knowledge base has to be loaded in advance. Association rules are a form of implication patterns and used to discover regularities in a data set. For instance, in [10] an association rule \( A \implies B \) with sufficiently high confidence and support between two classes \( A \) and \( B \) indicates that introducing a subclass relationship \( A \sqsubseteq B \) may be appropriate.

\(^6\)If the reasoning option is turned off, the score would be 33.3%.
3 General Description

In this software release, a user-friendly interface for knowledge engineers, who intent to use the enrichment algorithms described in Deliverable D3.3.1 is provided. It consists of a web user interface, which can be configured for working on a specific knowledge base and a widget, which can be integrated into other tools. In particular, the ORE tool[7] developed in Task 3.4 integrates this widget, but other software like the WebProtege[8] and OntoWiki[3] ontology editors will also be able to use it.

3.1 Availability

The widget is published as open source project and can be downloaded at https://github.com/AKSW/EnrichmentUIWidget.

3.2 Implementation

The Knowledge Base Enrichment User Interface is implemented as jQuery\(^{10}\) widget and also makes use of the JavaScript library jQuery EasyUI\(^{11}\) for some graphical components. The underlying learning algorithms are part of the DL-Learner Framework and called via AJAX requests to a Java Servlet which must be deployed on an accessible server. In order to use the widget in other web projects the following parameters have to be specified:

**SPARQL endpoint URL** The URL of the SPARQL endpoint and optionally the default graph URL on which the enrichment algorithms will be executed.

**Service URL** Declares the URL of the Java Servlet which is called to execute the corresponding learning algorithms.

3.3 Layout

To keep the layout as simple as possible the widget currently consists only of 2 main parts, as it is shown in Figure 2: On the left side there is a configuration panel, where several learning parameters can be defined. The panel on the right side shows the result of the learning process. For each learned axiom type, the axioms and their corresponding accuracy value are displayed in separate tables.

3.4 Configuration Parameters

The enrichment user interfaces allows to configure the following parameters which affect the axiom learning process:

\(^{7}\)http://ore-tool.net/Projects/ORE
\(^{8}\)http://protegewiki.stanford.edu/wiki/WebProtege
\(^{9}\)http://ontowiki.net/Projects/OntoWiki
\(^{10}\)http://jquery.com/
\(^{11}\)http://www.jeasyui.com/
Resource URI is used to specify the resource (property or class) which should be enriched.

Resource Type allows to determine of which type of entity the resource is. This implicitly works as a filter on the allowed and executed learning algorithms.

Inference allows to turn inference on or off. If it is turned on, in a preprocessing step the class hierarchy is computed. Powerful reasoning capabilities may improve the quality of suggestions, in particular for those axioms, which rely on knowing the class hierarchy of the knowledge base, e.g. domain and range axioms.

Max. execution time specifies the maximum execution time in seconds for each algorithm run.

Max. returned axioms specifies the maximum number of returned axioms per axiom type.

Threshold allows to specify a threshold for enrichment suggestions, i.e. suggestions with a lower score will be omitted.

Axiom Types is used to choose for which type of axioms the learning algorithm will be executed.

3.5 Scenario

Suppose, we are working on the knowledge base DBpedia and we want to add more schema information for the property http://dbpedia.org/ontology/league. At first, we enter the URI in the Resource URI field. We don’t know whether it is an object property or a datatype property, so we let the tool determine the Resource Type automatically. We do not want to use inferred knowledge, i.e. we don’t check the Inference.
box. By setting the Max. execution time spinner to 10, the Max. returned axioms spinner to 5 and the Threshold slider to 75, we ensure that all algorithms terminate after 10 seconds and return at most 5 axioms with an accuracy value above 75%. We are interested in super properties and in domain and range of the property. Additionally, we also want to know some characteristics of the property, i.e. whether it is functional, inverse functional, symmetric and transitive. Therefore we select the corresponding Axiom Types. After that, we are done with the configuration and can Start the computation. For each axiom type a request is sent to the Java Servlet in parallel and the returned axioms are shown as tables in the panel on the right side.
# 4 Manual

## 4.1 Obtaining the Widget

The widget is published as open source project and can be downloaded on GitHub at [https://github.com/AKSW/EnrichmentUIWidget](https://github.com/AKSW/EnrichmentUIWidget).

A general documentation can be found at [http://dl-learner.org/wiki/EnrichmentUIWidget](http://dl-learner.org/wiki/EnrichmentUIWidget).

## 4.2 Integrating the Widget

Get the sources from GitHub

```
git clone git://github.com/AKSW/EnrichmentUIWidget.git
```

or download the package from [https://github.com/AKSW/EnrichmentUIWidget/tarball/master](https://github.com/AKSW/EnrichmentUIWidget/tarball/master).

Put the war file `enrichment.war` on a Java Servlet container, e.g. for Tomcat 6 it would be

```
cd EnrichmentUIWidget
cp enrichment.war /PATH/TO/TOMCAT6/webapps/
```

The resulting service URL will then be [http://SERVLET_CONTAINER_URL/enrichment/Enrichment](http://SERVLET_CONTAINER_URL/enrichment/Enrichment)

Link to the necessary JavaScript and CSS files into your project

```
<html>
<head>...
    ...<script type="text/javascript" src="js/jquery.min.js"></script>
    <script type="text/javascript" src="js/jquery-ui.min.js"></script>
    <script type="text/javascript" src="js/jquery.easyui.min.js"></script>
    <script type="text/javascript" src="jquery.ui.enrichment.js"></script>
    <link rel="stylesheet" type="text/css" href="css/ ui-darkness/jquery-ui-1.8.22.custom.css">
    <link rel="stylesheet" type="text/css" href="css/default/easyui.css">
    <link rel="stylesheet" type="text/css" href="css/icon.css">
    <link rel="stylesheet" type="text/css" href="css/enrichment.css">
</head>
...</html>
```

Create a container element with an **ID**, call the jQuery plugin on it using `$("ID").enrichment()` and set the parameters for the SPARQL endpoint URL and the Java Servlet URL, e.g.

```
<div id="enrichment-container"></div>
<script type="text/javascript">
    $("#enrichment-container").enrichment({
        'service_url': 'http://localhost:8080/enrichment/Enrichment',
        'endpoint': {
```
4.3 Reporting Bugs

As it is deployed on GitHub, bugs and issues will be managed on https://github.com/AKSW/EnrichmentUIWidget/issues.
5 Pointers

Enrichment Widget: https://github.com/AKSW/EnrichmentUIWidget
jQuery: http://jquery.com/
jQuery EasyUI: http://www.jeasyui.com/
DL-Learner: http://dl-learner.org/
ORE: http://ore-tool.net/Projects/ORE
References


