

DBpedia Live Extraction

Sebastian Hellmann, Claus Stadler, Jens Lehmann, and Sören Auer

Universität Leipzig, Institute of Computer Science,
Johannisgasse 26, 04103 Leipzig, Germany
{hellmann|lehmann|auer}@informatik.uni-leipzig.de,
mai02jgj@studserv.uni-leipzig.de
<http://aksw.org>

Abstract. The DBpedia project extracts information from Wikipedia, interlinks it with other knowledge bases, and makes this data available as RDF. So far the DBpedia project has succeeded in creating one of the largest knowledge bases on the Data Web, which is used in many applications and research prototypes. However, the heavy-weight extraction process has been a drawback. It requires manual effort to produce a new release and the extracted information is not up-to-date. We extended DBpedia with a live extraction framework, which is capable of processing tens of thousands of changes per day in order to consume the constant stream of Wikipedia updates. This allows direct modifications of the knowledge base and closer interaction of users with DBpedia. We also show how the Wikipedia community itself is now able to take part in the DBpedia ontology engineering process and that an interactive round-trip engineering between Wikipedia and DBpedia is made possible.

1 Introduction

DBpedia is the result of a community effort to extract structured information from Wikipedia, which in turn is the largest online encyclopedia and currently the 7th most visited website according to [alexa.com](http://www.alexa.com). Over the past two years the DBpedia knowledge base has turned into a crystallization point for the emerging Web of Data. Several tools have been built on top of it, e.g. DBpedia Mobile¹, Query Builder², Relation Finder[8], and Navigator³. It is used in a variety of applications, for instance Muddy Boots, Open Calais, Faviki, Zemanta, LODr, and TopBraid Composer (cf. [3]).

Despite this success a disadvantage of DBpedia has been the heavy-weight release process. Producing a release requires manual effort and – since dumps of the Wikipedia database are created on a monthly basis – DBpedia has never reflected the current state of Wikipedia. In this article, we present a live extraction framework, which allows DBpedia to be up-to-date with a minimal delay of only

¹ <http://beckr.org/DBpediaMobile/>

² <http://querybuilder.dbpedia.org>

³ <http://navigator.dbpedia.org>

a few minutes. The main motivation behind this enhancement is that our approach turns DBpedia into a real-time editable knowledge base, while retaining the tight coupling to Wikipedia. It also opens the door for an increased use of DBpedia in different scenarios. For instance, a user may like to add to her movie website a list of highest grossing movies produced in the current year. Due to the live extraction process, this becomes much more appealing, since the contained information will be as up-to-date as Wikipedia instead of being several months delayed. In the future this may also lead to the inclusion of automatically generated tables within Wikipedia itself via (cached) SPARQL queries to the DBpedia SPARQL endpoint. In this sense, the work may be a first step to make Wikipedia more consistent, since information does not have to be duplicated and tediously maintained.

In the framework, we go much further than providing real-time extraction: We also allow Wikipedia editors to maintain the DBpedia ontology, which is the structural backbone of DBpedia (see the following section for details). Particularly, we allow them to map Wikipedia infobox structures to classes and properties in the DBpedia ontology. So far this was done in a manual engineering effort within the DBpedia project and more than 2000 of such mappings have already been created. We now open our interface so that those people who create the data can also control its representation in DBpedia. Furthermore, the ontology itself can now be modified directly within Wikipedia, which opens the way for a collaborative ontology engineering process.

Overall, we make the following contributions:

- extension of DBpedia with a live extraction framework providing up-to-date information,
- provision of a mechanism which allows the Wikipedia community to maintain the DBpedia ontology collaboratively,
- deployment of the framework on a test server

The article is structured as follows: In Section 2, we give an introduction to DBpedia and describe the current status of the project. Section 3 explains how the live extraction framework works. The following Section 4 shows how the Wikipedia community itself can now maintain the DBpedia ontology. Section 5 reviews related work and we discuss our conclusions in Section 6.

2 An Overview of DBpedia

In this section we give a brief introduction into DBpedia from its first steps at the end of 2006 up to its status before our work on the DBpedia live extraction.

The core of DBpedia consists of an infobox extraction process, which was first described in [2]. Infoboxes are templates contained in many Wikipedia articles. They are usually displayed in the top right corner of articles and contain factual information (cf. Figure 1). The infobox extractor processes an infobox as follows: The DBpedia URI, which is created from the Wikipedia article URL,

is used as subject. The predicate URI is created by concatenating the namespace fragment `http://dbpedia.org/property/` and the name of the infobox attribute. Objects are created from the attribute value. Those values are post-processed to obtain suitable value representations. For instance, internal MediaWiki links are converted to DBpedia URI references, lists are detected and represented accordingly, units are detected and converted to standard datatypes etc. Nested templates can also be handled, i.e. the object of an extracted triple can point to another complex structure extracted from a template.

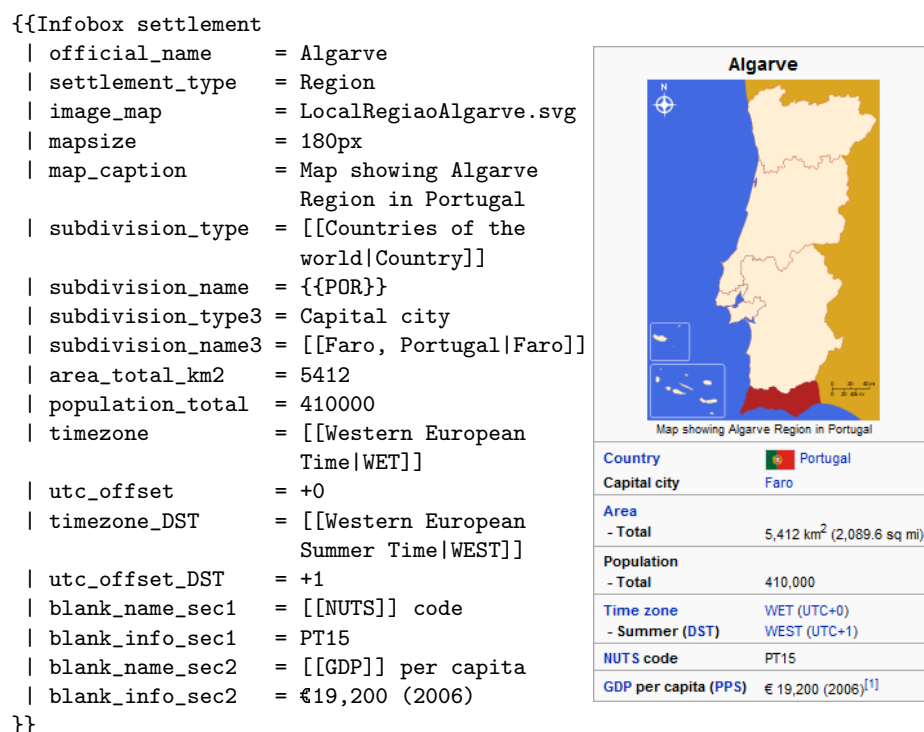


Fig. 1. Mediawiki infobox syntax for Algarve (left) and rendered infobox (right).

Apart from the infobox extraction, the framework has currently 12 extractors which process the following types of Wikipedia content:

- *Labels.* All Wikipedia articles have a title, which is used as an `rdfs:label` for the corresponding DBpedia resource.
- *Abstracts.* We extract a short abstract (first paragraph, represented by using `rdfs:comment`) and a long abstract (text before a table of contents, at most 500 words, using the property `dbpedia:abstract`) from each article.
- *Interlanguage links.* We extract links that connect articles about the same topic in different language editions of Wikipedia and use them for assigning labels and abstracts in different languages to DBpedia resources.

- *Images*. Links pointing at Wikimedia Commons images depicting a resource are extracted and represented by using the `foaf:depiction` property.
- *Redirects*. In order to identify synonymous terms, Wikipedia articles can redirect to other articles. We extract these redirects and use them to resolve references between DBpedia resources.
- *Disambiguation*. Wikipedia disambiguation pages explain the different meanings of homonyms. We extract and represent disambiguation links by using the predicate `dbpedia:disambiguates`.
- *External links*. Articles contain references to external Web resources which we represent by using the DBpedia property `dbpedia:reference`.
- *Pagelinks*. We extract all links between Wikipedia articles and represent them by using the `dbpedia:wikilink` property.
- *Homepages*. This extractor obtains links to the homepages of entities such as companies and organisations by looking for the terms *homepage* or *website* within article links (represented by using `foaf:homepage`).
- *Categories*. Wikipedia articles are arranged in categories, which we represent by using the SKOS vocabulary⁴. Categories become `skos:concepts`; category relations are represented by using `skos:broader`.
- *Geo-coordinates*. The geo-extractor expresses coordinates by using the Basic Geo (WGS84 lat/long) Vocabulary⁵ and the GeorSS Simple encoding of the W3C Geospatial Vocabulary⁶. The former expresses latitude and longitude components as separate facts, which allows for simple areal filtering in SPARQL queries.
- *Metainformation*. A new extractor which introduces organisational properties like direct edit links and oai identifiers for edit and delete operations.

Subsequently, DBpedia has turned into the central knowledge base within the Linking Open Data Initiative (see also [1]). It has been interlinked with other knowledge bases like Geonames, EuroStat, the CIA World Factbook, Freebase, OpenCyc, etc. The collection of this information and its free availability via Linked Data and SPARQL have attracted wide attention within and beyond the Semantic Web community.

While DBpedia was used by an increasing number of developers, a major obstacle was the lack of structure. For instance, there were several spellings of the property “birthPlace” (denoting the place where a person was born) due to the generic nature of the infobox extractor. There was no resolution of synonymous attribute names, which made writing queries against generic infobox data rather cumbersome. As Wikipedia attributes do not have explicitly defined datatypes, a further problem is the relatively high error rate of the heuristics that are used to determine the datatypes of attribute values. Both problems were partially solved by a mapping-based extraction approach (see [3]): A DBpedia ontology was developed and attribute names were mapped to properties within the ontology. The ontology was created by manually arranging the 350 most commonly used

⁴ <http://www.w3.org/2004/02/skos/>

⁵ <http://www.w3.org/2003/01/geo/>

⁶ <http://www.w3.org/2005/Incubator/geo/XGR-geo/>

infobox templates within the English edition of Wikipedia into a subsumption hierarchy consisting of 170 classes and then mapping 2350 attributes from within these templates to 720 ontology properties. The property mappings also define fine-grained rules on how to parse infobox values and define target datatypes, which help the parsers to process attribute values. For instance, if a mapping defines the target datatype to be a list of links, the parser will ignore additional text that might be present in the attribute value. The ontology uses currently 55 different datatypes. Deviant units of measurement are normalized to one of these datatypes. Instance data within the infobox ontology is therefore cleaner and better structured compared to the generic approach. A disadvantage is the lower coverage. Presently, it provides information about roughly 850,000 entities compared to 1,5 million entities covered by the generic approach. Another disadvantage is the manual effort required to maintain and extend those mappings. We will later explain how we intend to solve this problem by a closer collaboration with the Wikipedia community.

3 Live Extraction Framework

In this section we present the design of the DBpedia Live Extraction framework and how it differs from the previous approach, which was based on extracting data from Wikipedia database dumps.

A prerequisite for being able to perform a live extraction is an access to changes made in Wikipedia. The Wikimedia foundation kindly provided us access to their update stream, the *Wikipedia OAI-PMH*⁷ live feed. The protocol allows to pull updates in XML via HTTP. A Java component, serving as a proxy, constantly retrieves new updates and feeds the DBpedia framework. The proxy is necessary to decouple the stream from the framework to simplify maintenance of the software. It also handles other maintenance tasks such as the removal of deleted articles and it processes the new templates, which we will introduce in Section 4. The live extraction workflow uses this update stream to extract new knowledge upon relevant changes in Wikipedia articles.

Figure 2 gives an overview of the DBpedia knowledge extraction framework. The main components of the framework are:

- *PageCollections* which are an abstraction of local or remote sources of Wikipedia articles,
- *Destinations* that store or serialize extracted RDF triples,
- *Extractors* which turn a specific type of wiki markup into triples,
- *Parsers* which support the extractors by determining datatypes, converting values between different units and splitting markups into lists.
- *ExtractionJobs* group a page collection, extractors and a destination into a workflow.

⁷ Open Archives Initiative Protocol for Metadata Harvesting, cf. <http://www.mediawiki.org/wiki/Extension:OAIRepository>

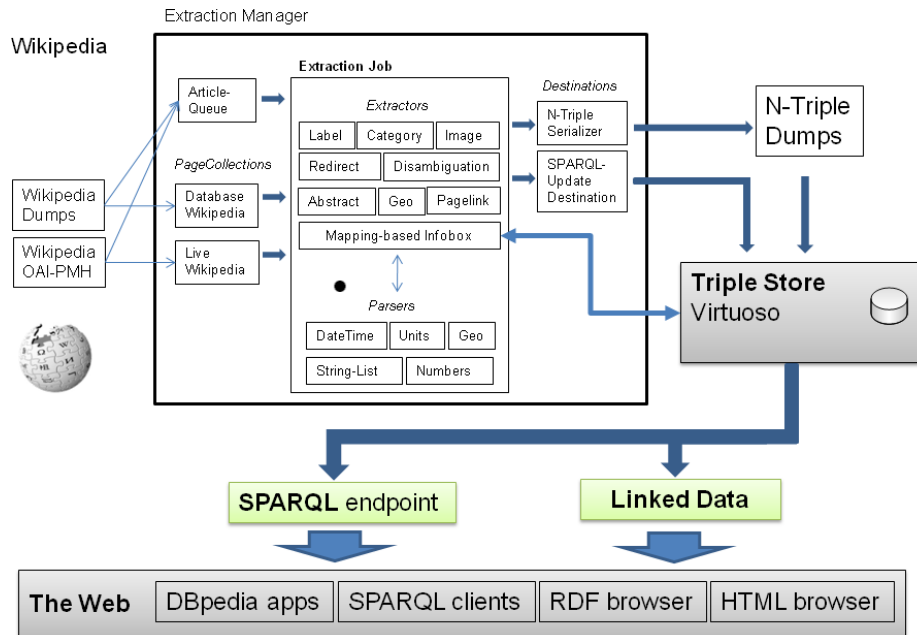


Fig. 2. Overview of DBpedia Live Extraction framework.

- The core of the framework is the *Extraction Manager* which manages the process of passing Wikipedia articles to the extractors and delivers their output to the destination. The Extraction Manager also handles URI management and resolves redirects between articles.

In live extraction mode, article texts are accessed via the *Live Wikipedia page collection*, which obtains the current version of the article, which was preprocessed by the Java proxy from the *OAI-PMH* stream. The content is comprised of the current Wikisource code, language (English only at the moment), an OAI identifier and a page revision id⁸. The *SPARQL-Update Destination* deletes existing triples and inserts new ones into the target triple store. According to our measurements, about 1.4 article pages are updated each second on Wikipedia. This amounts to 120,000 page updates per day and a maximum processing time of 0.71s per page for the live extraction framework. Currently, the framework can handle up to 2.5 pages per second on a 2.4 GHz dual-core machine (this includes consumption from the stream, extraction, diffing and loading the triples into a Virtuoso triple store)⁹. Performance is one of the major engineering hurdles we had to take in order to be able to deploy the framework. The time lag for DBpedia to reflect Wikipedia changes lies between one or two minutes. The

⁸ see here for an example <http://en.wikipedia.org/wiki/Special:Export/Algarve>

⁹ see current statistics at <http://stats.dbpedia.org>

bottleneck here is the update stream, since changes normally need more than one minute to arrive from Wikipedia.

Apart from performance, another important problem is to identify which triples have to be deleted and re-extracted upon an article change. DBpedia contains a “static” part, which is not affected by the live extraction framework. This includes links to other knowledge bases, which are manually updated, the YAGO¹⁰ and Umbel¹¹ class hierarchies and abstracts for other languages, which can not be updated via the English Update Stream. We store the structure of those triples using a SPARQL graph pattern. All objects matching the pattern are then excluded from delete queries. All other parts of DBpedia are maintained by the extraction framework. We redesigned the extractors in such a way that each generates disparate properties and vocabularies. Therefore each extractor can be in one of three states: active, keep, and purge. Depending on the state when a Wikipedia page is changed, the triples extracted by this extractor are either updated (active), not modified (keep), or removed (purge). We use the `origin` annotation of an extracted triple to decide which triples were extracted by an extractor. This annotation allows to directly query for triples from one extractor and can be used to produce new DBpedia releases with SPARQL. As an example, the population of the city of Leipzig is annotated as follows:

Example 1 (OWL 2 Annotations).

```
db:Leipzig    dbp-ont:population    "514492"^^xsd:integer .
db-meta:a2819 rdf:type                owl:Axiom .
db-meta:a2819 owl:annotatedSource  db:Leipzig .
db-meta:a2819 owl:annotatedProperty db-ont:population .
db-meta:a2819 owl:annotatedTarget  "514,492"^^xsd:integer .
db-meta:a2819 db-meta:origin         db-meta:LiveInfoboxExtractor .
db-meta:a2819 db-meta:extractedFromTemplate
                                                    db-meta:CityTemplate .

dbp-meta:a2819 dc:modified
                "2009-06-10T04:00:00-05:00"^^xsd:dateTime .
```

The first line is the triple itself. The next four lines identify the triple by means of the OWL 2 annotation method, which is analogous to RDF reification, but ensures that DBpedia can be processed by standard reasoners. The URL of an axiom is generated by appending a unique ID to <http://dbpedia.org/meta/>. The last four lines contain the actual annotations.

For each triple we store the extractor responsible for the triple and the modification date of the triple. The infobox extractor additionally stores the template used to derive the information. One case where this is used are updates of property mappings, which are explained in the next section. The annotations will also be used to provide regular dataset releases on top of the live extraction.

¹⁰ <http://www.mpi-inf.mpg.de/yago-naga/yago/downloads.html>

¹¹ <http://fgiasson.com/blog/index.php/2008/09/04/exploding-dbpeditas-domain-using-umbel/>

Each data set then simply contains those triples annotated as being created by a particular extractor. The annotations also provide useful metadata apart from their use in the live extraction framework, e.g. a website including Wikipedia content can now query for the last update of a particular piece of information.

However, annotations also increase the size of DBpedia significantly. In order to avoid a decrease in DBpedia query performance, we allow to store them optionally in a separate graph in the used triple store. Currently, this graph has the URI <http://dbpedia.org/meta>. This allows users to ignore them if an application does not want to make use of annotations, but they can also query both graphs, if desired.

We had to make a number of further changes within the DBpedia extraction framework in order to support live extraction. For instance, to parse article abstracts properly, we need to be able to resolve templates. This can only be done if (parts of) the MediaWiki database for the corresponding language edition is (are) available. For this reason we delegate updates from the stream to the MediaWiki database so that the local MediaWiki database remains in sync. In general, all parts of the DBpedia framework, which relied on static databases, files etc., needed to be exchanged so that no user interaction is required. Also, the framework had to be refactored to be language independent to make it compatible to future deployment on language specific DBpedia versions such as the German DBpedia ¹².

4 Template-Based Ontology Engineering

The main goal when providing a live extraction framework, apart from being able to provide up-to-date information, is to enable a tighter integration of Wikipedia and DBpedia. Wikipedia users can get almost instant reward when they modify (semi-)structured information. As mentioned previously, this allows a broader range of DBpedia applications, e.g. semantic content syndication, revealing of inconsistencies in Wikipedia or notification services triggered by new Wikipedia content or events. Even more intriguing is to allow Wikipedia users to control the structure of the extracted knowledge. So far the DBpedia ontology has been created manually. We now handle control over to the Wikipedia community in a non-obtrusive way.

Since the core of DBpedia is the information contained in infoboxes, they are the most interesting target for the extraction of ontological knowledge. As in the mapping-based approach, explained in Section 2, each infobox is mapped to a class in the DBpedia ontology and each attribute in the infobox is mapped to a property. To allow editing those mappings, we introduce three new templates in Wikipedia:

¹² <http://de.dbpedia.org>

Example 2 (DBpedia infobox annotation template).

```
{{DBpedia Template
| relatesToClass = MusicArtist
| mapping =
  {{ DBpedia Attribute | birthplace | birthPlace }}
  {{ DBpedia Attribute | birthdata | birthDate }}
  {{ DBpedia Attribute | DiedIn | deathPlace }}
}}
```

Infoboxes in Wikipedia indicate that the entity described by the Wikipedia page belongs to a certain class. Hence, the DBpedia infobox annotation template maps an infobox template to a class in the DBpedia ontology. For this purpose it contains the attribute `relatesToClass`. The second attribute `mapping` represents a map structure for mapping template attributes to properties in the DBpedia ontology. The value of this attribute represents a list of sub-templates, which map one attribute (given as first parameter) to its corresponding DBpedia ontology property (given as second parameter). We allow parse hints, which can be used as an optional parameter, e.g.:

```
{{ DBpedia Attribute | weighs |weight| parseHint = pound  }}
```

This helps the DBpedia parser to extract correct values for an attribute. It can also be used for converting units, i.e. if the weight of an entity is given in stones, but mapped to a property, which has range `kilogram`, DBpedia then automatically converts the values to the correct unit. In this case, `kilogram` is a custom datatype, which is another feature of OWL incorporated by the DBpedia live extraction.

This template is stored in the `/doc` sub-article of an infobox definition, i.e. for the infobox `Musical Artist`, they are stored in http://en.wikipedia.org/wiki/Template:Infobox_Musical_artist/doc along with other documentation material about the infobox template. This allows the Wikipedia community and us to add these mappings in a non-obstructive way, since the infobox templates themselves are usually write-protected.

Adding the mappings in DBpedia would still allow to maintain the DBpedia ontology externally. However, we decided to integrate the ontology engineering part directly into Wikipedia. For each property and class a Wikipedia page can be created, which contains information about it. Classes and properties are currently stored in <http://meta.wikimedia.org/wiki/DBpedia/ontology/ELEMENT>. We use this namespace, since Wikipedia policy allows to add subpages there. It also ensures that those pages do not appear in search results or have any other negative effect on Wikipedia users. As common in DBpedia, the live extraction uses the URI of the article to generate a DBpedia URI of the form <http://dbpedia.org/ontology/ENTITYNAME>. By convention, classes start with a capital and properties with a lower case letter.

As an example, we give the template for the property birth place (stored at <http://meta.wikimedia.org/wiki/DBpedia/ontology/birthPlace>) and the

class person (stored at <http://meta.wikimedia.org/wiki/DBpedia/ontology/Person>):


Example 3 (object property description template).

```
{{ DBpedia ObjectProperty
| rdfs:label = birthPlace
| rdfs:label@de = Geburtsort
| rdfs:label@fr = lieu de naissance
| rdfs:comment = Relates a living thing to
                 the place where it was born.
| owl:equivalentProperty =
| rdfs:seeAlso = cyc:birthPlace
| rdfs:subPropertyOf =
| rdfs:domain = LivingThing
| rdfs:range = Place
| rdf:type = owl:FunctionalProperty
}}
```

DBpedia Object Property "birthPlace"

[\[edit\]](#)

The information within the table below is used by [DBpedia](#) which is a community project for extracting structured information from Wikipedia and make it freely available on the Web. Please read [DBpedia/ontology](#) for an overview of the DBpedia ontology.

property	value	
rdfs:label	birthplace	
rdfs:label@de	Geburtsort	
rdfs:label@en	birthplace	
rdfs:label@fr	lieu de naissance	
rdfs:comment	Relates a living thing to the place where it was born.	
rdfs:domain	LivingThing	
rdfs:range	Place	
rdfs:subPropertyOf	place	
rdf:type	owl:FunctionalProperty	
owl:equivalentProperty		
rdfs:seeAlso	cyc:birthPlace	

See [DBpedia:birthPlace](#) for all information available about this entity.

Fig. 3. Rendered view of birth place property definition.

The templates have key value pairs and follow a simple, yet powerful principle, which is similar to our infobox extraction method. For each pair, we extract an axiom/triple as follows: The subject is determined by the URL of the Wikipedia page. The predicate is the key (left of the equality sign), where the

namespaces `owl`, `rdfs`, and `rdf` can be used. The object is the value (right of the equality sign). This simple mechanism allows to specify almost arbitrary information about the entity, and the conversion of such a template into triples is straightforward.

Example 4 (class description template).

```
{ { DBpedia Class
  | rdfs:label = person
  | rdfs:label@de = Person
  | rdfs:label@fr = personne
  | rdfs:comment = A person in DBpedia is defined as
                  an individual human being.
  | owl:equivalentClass = foaf:Person,umbel-sc:Person,
                          yago:Person100007846
  | owl:disjointClass =
  | rdfs:seeAlso = opencyc:Person
  | rdfs:subClassOf = LivingThing
} }
```

There are three extensions to this general extraction rule: 1.) For convenience, we allow comma-separated lists as attribute values. The property `relatedTo`, for example, could be annotated with `rdf:type = owl:TransitiveProperty, owl:SymmetricProperty`. 2.) The language tag, e.g. `@en` or `@de`, is appended to the key instead of the value, as would usually be the case, e.g. `rdfs:label@de = person`. 3.) The value can also be a class expression in Manchester OWL syntax [5].

Apart from classes and object properties, we also offer analogous templates for data properties, annotation properties, and custom datatypes. Figure 3 shows a rendered view of the above template for the object property `birthPlace`. Analogous views are created for all supported templates. The view provides a link explaining the DBpedia ontology to Wikipedia users, an overview of the values of the most frequently used properties, e.g. `rdfs:label`, and a pointer to the URI of the extracted entity. Due to the Linked Data principles, this allows users to browse the DBpedia ontology in a simple way.

The live extraction can process updates of the described templates as follows:

- Update and creation of property and class templates are straightforward. Both templates correspond to a well-defined set of OWL axioms extracted by the *DBpedia ontology extractor*, a special purpose extractor we added to the DBpedia framework. If such a template is created, those axioms are written to the triple store by using SPARUL¹³.
- If a class or property template is modified, then the extractor is called and its results are compared to the previous axioms. Axioms, which are no longer present, are deleted and new ones are inserted. This ensures that

¹³ <http://jena.hp1.hp.com/~afs/SPARQL-Update.html>

they have correct modification times and that we can keep track of which Wikipedia user has added which axiom without using the text-based history in Wikipedia. It also minimises the number of triples which have to be deleted and created, i.e. it is more efficient than completely removing all old axioms.

- When a new DBpedia template for an infobox is created, the live extractor acts conservatively, i.e. the mapping is stored internally, but we do not apply it to all infoboxes of the corresponding type. While this would be desirable, it is very expensive to parse all templates. However, we do follow the policy that once information from a template has been extracted, it is always up-to-date with respect to the current mapping (see next point).
- If an attribute mapping is changed from property p to property p' , then we can use the triple annotations (see Example 1) to detect all triples, which were extracted from this template containing p and update them accordingly. Simply renaming all triples p in the store would not be sufficient, since there may be other attributes, which should still map to p .

Overall, we have provided an infobox extraction approach, which has advantages over both infobox extractors mentioned in Section 2: It provides cleaner data than the generic extraction approach and higher flexibility than the previous mapping-based approach, where the mappings are hidden in a database, which is not accessible to the Wikipedia community. The usage of templates has proven a feasible method for common users such as the Wikipedia user community (about half a million people in May 2009¹⁴). We counted about 4.4 million uses of templates, which have been used in the extraction by DBpedia. The actual number of template usage in Wikipedia is even higher. We therefore argue, that the knowledge elicitation process is not hindered by syntactical issues, but rather enhanced by the fast process of reusing existing templates.

5 Related Work

There is a vast body of work related to the semantification of Wikipedia. Comprehensive listings are provided by Michael Bergman¹⁵ and by Wikipedia itself¹⁶. Here we will discuss only two important approaches which – as the DBpedia live extraction – aim at engaging a large community of people in creating a collaboratively edited multi-domain ontology with encyclopedic knowledge.

*Freebase*¹⁷ is a commercial company that builds a huge online database, which users can edit in a similar fashion as they edit Wikipedia articles today. Freebase has employed Wikipedia knowledge as initial content for their database. They synchronise their data with Wikipedia conservatively, i.e. information from Wikipedia is only imported if it cannot override or conflict with knowledge in

¹⁴ source: <http://stats.wikimedia.org/EN/TablesWikipediaEN.htm>

¹⁵ <http://www.mkbergman.com/?p=417>

¹⁶ http://en.wikipedia.org/wiki/Wikipedia:Wikipedia_in_academic_studies

¹⁷ <http://www.freebase.com>

Freebase. It is challenging to keep in sync with Wikipedia, while still encouraging Freebase to be edited by its own users. In our approach this difficulty does not arise, i.e. for the DBpedia live extraction effort to succeed, it is not required to build up a separate community. Instead, we focus on a tight integration into the Wikipedia community. Freebase also builds a schema, but this information can be edited only by its creator to avoid breaking the schema. In contrast, we allow freely editing schema information. Since November 2008 Freebase has published its database as Linked Data and DBpedia as well as Freebase have set RDF links to same entities in the other data source.

The *Semantic MediaWiki* project [7, 10] also aims at enabling the reuse of information within wikis as well as at enhancing search and browsing facilities. Semantic MediaWiki is an elaborate extension of the MediaWiki software, which allows to add structured data into wikis using a specific syntax. Ultimately, DBpedia and Semantic MediaWiki have similar goals. Both want to deliver the benefits of structured information in Wikipedia to its users, but use different approaches to achieve this aim. Semantic MediaWiki requires authors to deal with a new syntax, and covering all structured information within Wikipedia would require converting all information into this syntax. A significant hurdle for Semantic MediaWiki is to be deployed on major Wikipedia language editions. DBpedia differs from Semantic MediaWiki in trying to be as non-obstructive as possible and to use the structure that is already in place in Wikipedia. Therefore, the DBpedia approach does not require changes from Wikipedia authors, and the data extracted within the project can be readily used. Both approaches could be combined synergically by using DBpedia extraction algorithms for existing Wikipedia content, while SMW's typed link functionality could be used to encode and represent additional semantics in wiki texts.

6 Conclusion and Further Work

We want to note that the described live extraction is still an early prototype. We do expect development to continue within the next months until we are finally able to deploy it on the official DBpedia SPARQL endpoint. Furthermore, we have to migrate more than 2500 mappings from the mapping database to corresponding infobox /doc sub-pages. Also, pages for 720 properties and 175 classes have to be added. The current version of the live extraction test store can be queried via SPARQL at <http://dbpedia2.openlinksw.com:8895/sparql/>.

The DBpedia live extraction opens a number of exciting perspectives for future work, which can roughly take the following directions:

Cross-language infobox knowledge fusion. Currently, the live extraction framework is only implemented for the English Wikipedia edition. In the future, we may extend this to other important language editions. Infoboxes within different Wikipedia editions cover different aspects of an entity at varying degrees of completeness. For instance, the Italian Wikipedia contains more knowledge about Italian cities and villages than the English one, while the German Wikipedia contains more structured information about persons than the English edition.

By fusing infobox knowledge across editions, it should be possible to derive at a significantly more detailed multi-domain knowledge base and to increase the quality compared to knowledge bases that are derived from single Wikipedia editions. To do this, we will use interlanguage links in Wikipedia. According to [6], two articles in different languages have the same subject with high probability if there is a language link between the articles in *both* directions. We can use this result to merge content extracted from different language editions by applying conflict resolution and consistency checking strategies within this process.

Multi-language labels and comments: By using the live update stream for several languages and identifying equal articles as described in the paragraph above, we can extract `rdfs:label` and `rdfs:comment` information in various languages. This feature is already supported by the regular DBpedia releases and we intend to incorporate it in the live extraction.

Wikipedia article augmentation. Interlinking DBpedia with other data sources makes it possible to develop a MediaWiki extension that augments Wikipedia articles with additional information as well as media items (pictures, audio) from these sources. For instance, a Wikipedia page about a geographic location like a city or a monument can be augmented with additional pictures from Web data sources such as Flickr or with additional facts from statistical data sources such as Eurostat or the CIA Factbook.

Wikipedia consistency checking: The extraction of different Wikipedia editions and interlinking DBpedia with external Web knowledge sources lays the foundation for checking the consistency of Wikipedia content. For instance, whenever a Wikipedia author edits an infobox within a Wikipedia article, the content of the infobox could be checked against external data sources and the content of infoboxes within different language editions. Inconsistencies could be pointed out along with proposals on how to solve these inconsistencies. In this way, DBpedia and the Web of Data could contribute back to the Wikipedia community and help to improve the overall quality of Wikipedia.

DBpedia consistency checking: The live extraction gives rise to check the consistency of (fragments of) DBpedia by using OWL reasoners. We have investigated techniques for extracting fragments of DBpedia in [4]. Modern OWL reasoners like Pellet [9] are able to pinpoint inconsistencies, i.e. give a set of axioms responsible for the inconsistency. Tools for DBpedia Live could use those techniques to suggest possible edits. In a broader perspective, ensuring the high quality of a large community-maintained knowledge base is a challenge for Semantic Web research and practice.

References

1. Sören Auer, Chris Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a web of open data. In *Proceedings of the 6th International Semantic Web Conference (ISWC)*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer, 2008.
2. Sören Auer and Jens Lehmann. What have innsbruck and leipzig in common? extracting semantics from wiki content. In Enrico Franconi, Michael Kifer, and

- Wolfgang May, editors, *ESWC*, volume 4519 of *Lecture Notes in Computer Science*, pages 503–517. Springer, 2007.
3. Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. To appear.
 4. Sebastian Hellmann, Jens Lehmann, and Sören Auer. Learning of OWL class descriptions on very large knowledge bases. *International Journal On Semantic Web and Information Systems*, 2009. To be published.
 5. Matthew Horridge and Peter F. Patel-Schneider. Manchester syntax for OWL 1.1. *OWLED 2008, 4th international workshop OWL: Experiences and Directions*, 2008.
 6. Daniel Kinzler. Automatischer aufbau eines multilingualen thesaurus durch extraktion semantischer und lexikalischer relationen aus der wikipedia. Master's thesis, Universität Leipzig, 2008. also avialable at <http://lips.informatik.uni-leipzig.de/pub/2008-4>.
 7. Markus Krötzsch, Denny Vrandečić, and Max Völkel. Wikipedia and the Semantic Web - The Missing Links. In *Proceedings of Wikimania*, 2005.
 8. Jens Lehmann, Jörg Schüppel, and Sören Auer. Discovering unknown connections - the DBpedia relationship finder. In *Proceedings of the 1st SABRE Conference on Social Semantic Web (CSSW)*, 2007.
 9. Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *J. Web Sem*, 5(2):51–53, 2007.
 10. Max Völkel, Markus Krötzsch, Denny Vrandečić, Heiko Haller, and Rudi Studer. Semantic wikipedia. In *15th World Wide Web Conference*, pages 585–594, 2006.