DBpedia and the Live Extraction of Structured Data from Wikipedia

Mohamed Morsey, Jens Lehmann, Sören Auer, Claus Stadler and Sebastian Hellmann
Department of Computer Science, University of Leipzig,
Johannisgasse 26, 04103 Leipzig, Germany.
{morsey|lehmann|auer|cstadler|hellmann}@informatik.uni-leipzig.de
`http://aksw.org`

Abstract

Purpose – DBpedia extracts structured information from Wikipedia, interlinks it with other knowledge bases and freely publishes the results on the Web using Linked Data and SPARQL. However, the DBpedia release process is heavy-weight and releases are sometimes based on several months old data. DBpedia-Live solves this problem by providing a live synchronization method based on the update stream of Wikipedia.

Design/methodology/approach – Wikipedia provides DBpedia with a continuous stream of updates, i.e. a stream of recently updated articles. DBpedia-Live processes that stream on the fly to obtain RDF data and stores the extracted data back to DBpedia. DBpedia-Live publishes the newly added/deleted triples in files, in order to enable synchronization between our DBpedia endpoint and other DBpedia mirrors.

Findings – During the realization of DBpedia-Live we learned, that it is crucial to process Wikipedia updates in a priority queue. Recently-updated Wikipedia articles should have the highest priority, over mapping-changes and unmodified pages. An overall finding is that there is a plenty of opportunities arising from the emerging Web of Data for librarians.

Practical implications – DBpedia had and has a great effect on the Web of Data and became a crystallization point for it. Many companies and researchers use DBpedia and its public services to improve their applications and research approaches. The DBpedia-Live framework improves DBpedia further by timely synchronizing it with Wikipedia, which is relevant for many use cases requiring up-to-date information.

Originality/value – The new DBpedia-Live framework adds new features to the old DBpedia-Live framework, e.g. abstract extraction, ontology changes, and changesets publication.

Keywords Knowledge Extraction, RDF, Wikipedia, Triplestore
Paper type Research Paper

1. Introduction

Wikipedia is one of the largest knowledge sources of mankind and the largest encyclopedia on the Web with being the 7^{th} most visited website according to `alexa.com` (in July 2011). Wikipedia is created only by collaborative authoring of its users. Wikipedia is available in more than 280 languages, and the English Wikipedia contains more than 3.5 million articles (in July, 2011)[a]. However, despite its success there are also some issues and untapped potential:

- the search capabilities of Wikipedia are limited to keyword matching,
- inconsistencies may arise due to the duplication of information on different pages and in different Wikipedia language editions.

The main objective of DBpedia is to extract structured information from Wikipedia and to make this information available on the emerging *Web of Data*. Over the past five years the DBpedia knowledge base has turned into a crystallization point for this emerging Web of Data. Several tools using the DBpedia knowledge bases have been built, e.g. DBpedia Mobile[b], Query Builder[c], Relation Finder (Lehmann et al., 2007*a*), and Navigator[d]. It is also used in a variety of commercial applications, for instance Muddy Boots, Open Calais, Faviki, Zemanta, LODr, and TopBraid Composer (cf. (Lehmann et al., 2009)).

Despite this success, a disadvantage of DBpedia has been the heavy-weight release process. Producing a DBpedia dataset release through the traditional dump-based extraction 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. Hence, we extended the DBpedia extraction framework to support a *Live extraction*, which works on a continuous stream of updates from Wikipedia and processes that stream on the fly. It allows DBpedia to be up-to-date with a minimal delay of only a few minutes. The rationale behind this enhancement is that our approach turns DBpedia into a real-time editable knowledge base, while retaining the tight coupling with Wikipedia. It also opens the door for an increased use of DBpedia in different scenarios. For instance, a user may like to add to his/her movie website a list of highest grossing movies produced in the current year. Due to the live extraction process, this scenario becomes much more appealing, since the contained information will be as up-to-date as Wikipedia instead of being several months delayed.

Overall, we make the following contributions:

- realization of the Java-based DBpedia-Live framework, which allows the continuous extraction of up-to-date information from Wikipedia,
- addition of Wikipedia article abstract extraction capability to the extraction framework,
- automatic re-extraction of information from articles which are affected by changes

[a] `http://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia`
[b] `http://beckr.org/DBpediaMobile/`
[c] `http://querybuilder.dbpedia.org`
[d] `http://navigator.dbpedia.org`

in the DBpedia ontology mapping,

- flexible low-priority re-extraction from pages, which have not been modified for a longer period of time – this allows changes in the underlying extraction framework, which potentially affect all Wikipedia pages, while still processing the most recent Wikipedia changes,
- regular publication of extraction deltas for update propagation/synchronization with DBpedia-Live mirrors,
- provisioning of a synchronization tool, which downloads those deltas and updates a DBpedia-Live mirror accordingly, in order to keep it in sync with the main DBpedia-Live instance.

The paper is structured as follows: In Section 2, we give a brief introduction to the Semantic Web technologies. In Section 3 we give an overview on the DBpedia framework and its extractors in general. In Section 4 we describe the system architecture of DBpedia-Live and discuss each component in detail. The ontology engineering process is introduced in Section 5. In Section 6 we present the new features of the new framework and in Section 7 we outline how librarians can benefit from DBpedia. Eventually, related work is presented in Section 8 and we conclude in Section 9 with an outlook on future work.

2. Semantic Web Technologies

There are quite a few different definitions of the concept Semantic Web. Tim Berners-Lee, one of the inventors of the Semantic Web idea, defines it as ”The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation“ (Berners-Lee et al., 2001). According to the World Wide Web consortium (W3C) the concept can be defined as ”Semantic Web is the idea of having data on the Web defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration, and reuse of data across various applications.” (*W3C Semantic Web Activity*, 2009). In order to achieve this vision several technologies were developed and standardized. These include the Resource Description Framework (RDF), Uniform Resource Identifiers (URIs) as well as vocabulary, schema and ontology languages built on top of them such as RDF-Schema, OWL and SKOS. More details about Semantic Web Technologies can be found in (Yu, 2007).

2.1. *Resource Description Framework (RDF)*

RDF is a knowledge representation language for describing arbitrary resources such as Web resources, entities and general concepts. RDF is the main building block of the Semantic Web, and it is for the Semantic Web what HTML is for the conventional Web. RDF has several basic elements, namely resources (also called entities), properties (also predicates or roles), and statements (also triples or facts).

Resources comprise any concrete or abstract entities, e.g. a book or a person. Every resource is identified by a Uniform Resource Identifier (URI) which is guaranteed to be

Subject	Predicate	Object
dbpedia:William_Shakespeare	dbp:birthPlace	"Stratford-upon-Avon, Warwickshire, England"
dbpedia:William_Shakespeare	dbp:dateOfBirth	"April 1564"
dbpedia:William_Shakespeare	dbp:occupation	"Playwright, poet, actor"
dbpedia:William_Shakespeare	dbpo:child	dbpedia:Judith_Quiney
dbpedia:Judith_Quiney	dbp:spouse	"Thomas Quiney"

Table 1: Sample RDF statements.

globally unique, e.g. http://dbpedia.org/resource/William_Shakespeare.

Properties are specific resources which can be used to describe attributes or relations of other resources. For instance, the property http://dbpedia.org/ontology/Person/height identifies the height of a human being.

Statements are used to describe properties of resources in the format subject, predicate, object. According to the RDF data model all three components of a triple can be resources (i.e. URIs), subject and object can also be blank nodes (i.e. unnamed entities) and objects can also be literals consisting of a data value and optionally an associated a datatype or language tag. For example:

```
<http://dbpedia.org/resource/William_Shakespeare>
             <http://dbpedia.org/property/birthPlace>
                   "Stratford-upon-Avon, Warwickshire, England" .
```

This RDF statement simply says ”The subject identified by http://dbpedia.org/resource/William_Shakespeare has the property identified by http://dbpedia.org/property/birthPlace, whose value is equal to "Stratford-upon-Avon, Warwickshire, England"”. Table 1 shows more RDF statements about William Shakespeare. The URIs are abbreviated with namespace prefixes[e].

2.2. *RDF Serialization Formats*

There are several formats for serializing RDF data. A simple text format is the *N-Triples* format (Grant and Beckett, 2004). Each RDF triple is written in a separate line and terminated by a period ".". Typically files with N-Triples have the .nt extension. Figure 1 shows some of the sample triples from Table 1 encoded in N-Triples format. There are also a number of other RDF serialization formats such as the XML serialization *RDF/XML*, the text serialization formats *N3* and *Turtle*, which are very similar to the N-Triples format and at last the *RDFa* serialisation, which allows to integrate RDF into HTML.

2.3. *Ontology*

The W3C defines an ontology as follows: ”An ontology defines the terms used to describe and represent an area of knowledge.” (Heflin, 2004). Figure 2 shows an excerpt of the

[e] http://prefix.cc provides a list of common prefixes

```
<http://dbpedia.org/resource/William_Shakespeare>
   <http://dbpedia.org/property/dateOfBirth> "April 1564" .
<http://dbpedia.org/resource/William_Shakespeare>
   <http://dbpedia.org/ontology/child>
   <http://dbpedia.org/resource/Judith_Quiney> .
```

Figure 1: Sample N-Triples format.

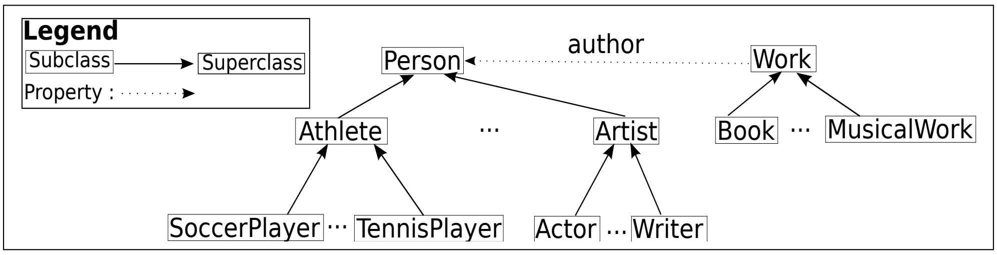


Figure 2: Excerpt of the DBpedia ontology.

DBpedia Ontology. This ontology says that there is a class called ”Writer” which is a subclass of ”Artist”, which is in turn a subclass of ”Person”. William Shakespeare, and Dan Brown are considered instances of the class ”Writer”. Note that there is a property relating an instance of the class ”Work” to an instance of the class ”Person”. For instance, the novel titled ”The Lost Symbol” is an instance of class ”Work” and related via property ”author” to its author ”Dan Brown”.

2.4. *Ontology Languages*

Ontologies can be created using different ontology languages varying in expressiveness. Examples are RDF Schema (RDFS) or the Web Ontology Language (OWL). *RDFS* is a relatively simple language one can use to create a vocabulary for describing classes, subclasses, and properties of RDF resources; it is a W3C recommendation (Brickley and Guha, 2004). Figure 3 shows a part of our ontology from Figure 2 expressed in OWL+N3 syntax. The *Web Ontology Language* (OWL) is also a W3C recommendation (Bechhofer et al., 2004) and builds on RDFS, but additionally allows to express more complex relationships. For example, OWL allows to construct new classes using set operators or property restrictions. Thus implicit information can be represented and revealed later on using an OWL reasoner. There are several OWL dialects balancing differently between expressiveness and reasoning complexity.

2.5. *SPARQL Query Language for RDF*

SPARQL stands for SPARQL Protocol and RDF Query Language. It is used to ask queries against RDF graphs. A SPARQL processor finds sets of triples in the RDF graph that

```
dbo:Writer  rdf:type owl:Class .
dbo:Writer  rdfs:subClassOf  dbo:Artist .
dbo:author  rdf:type owl:ObjectProperty .
```

Figure 3: RDFS representation of a part of our ontology in Turtle format. Turtle is more compact than N-Triples, e.g. by allowing to use prefixes such as dbo, rdf and rdfs in this example (see http://prefix.cc for the full namespaces).

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?spouse WHERE {
       dbpedia:William_Shakespeare  dbo:child  ?child.
       ?child                       dbp:spouse     ?spouse.
}
```

Figure 4: SPARQL query to get the spouse of Shakespeare's child.

match to the required pattern. The results of SPARQL queries can be result sets or RDF graphs (Prud'hommeaux and Seaborne, 2008). For instance, assume that we want to ask the query ”Who is the spouse of William Shakespeare's child?”. Figure 4 shows how this query can be expressed in SPARQL.

2.6. *Triplestore*

A triplestore is a software program capable of storing and indexing RDF data, in order to enable querying this data efficiently. Most triplestores support the SPARQL query language for querying RDF data. Virtuoso (Erling and Mikhailov, 2007), Sesame (Broekstra et al., 2002), and BigOWLIM (Bishop et al., 2011) are typical examples of triplestores. DBpedia is using Virtuoso as the underlying triplestore.

3. Overview on DBpedia

The core of DBpedia consists of an infobox extraction process, which was first described in (Auer and Lehmann, 2007). 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 5). Infoboxes display the most relevant facts of a Wikipedia article in a table often in the form of attribute-value pairs.

Wikipedia's infobox template system has evolved over time with no central coordination. In other words, different communities use different templates to describe the same type of things (e.g. `infobox_city_japan`, `infobox_swiss_town` and `infobox_town_de`). Moreover, different infobox templates use different names to denote the same attribute (e.g.

birthplace and placeofbirth). Attribute values are also expressed using a wide range of different formats and units of measurement (e.g. height = {{height|ft=5|in=7}} and height = {{height|m=1.70}}). We have overcome that obstacle by using two different extraction approaches operating in parallel: A generic approach which aims at wide coverage and a mapping-based approach which aims at better data quality.

3.1. *Generic versus Mapping-based Infobox Extraction*

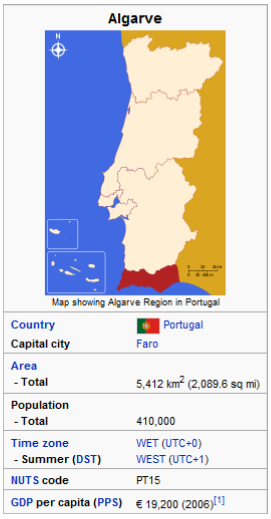
Generic Infobox Extraction. The generic infobox extraction approach processes an infobox as follows: A DBpedia URI is created from the Wikipedia article URL the infobox is contained in. This URI is subsequently used as the subject for all extracted triples. For instance, the URL of the Wikipedia article http://en.wikipedia.org/wiki/William_Shakespeare is used to create the resource http://dbpedia.org/resource/William_Shakespeare, which is subsequently used as the subject. The predicate URI is created by concatenating the namespace http://dbpedia.org/property/ with the name of the infobox attribute. For example, the Wikipedia attribute birthplace results in the property http://dbpedia.org/property/birthplace. Objects are created from the attribute values. Those values are pre-processed and converted into RDF 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. The main advantage of that approach is that it can cover all infobox types along with their attributes. Its drawback is that synonymous, i.e. equivalent attribute names are not resolved. This makes writing queries against generic infobox data rather complex.

Mapping-based Infobox Extraction. In order to overcome the problems of synonymous attribute names and multiple templates being used for the same type of things, we mapped Wikipedia templates to an ontology. The ontology was created manually by arranging the 341 most commonly used infobox templates within the English edition of Wikipedia into a subsumption hierarchy consisting of 314 classes and then mapping 2350 attributes from within these templates to 1425 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 currently uses 376 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 than data that is generated using the generic extraction approach. We will discuss the DBpedia ontology in more detail in Section 5.

3.2. *DBpedia Extractors*

A DBpedia extractor is an executable module, responsible for extracting a specific piece of data from a Wikipedia article. For instance, the abstract extractor extracts the abstract of a Wikipedia article, i.e. the text before the table of contents of that article.

```
{{Infobox settlement
 | official_name      = Algarve
 | settlement_type    = Region
 | image_map          = LocalRegiaoAlgarve.svg
 | mapsize            = 180px
 | map_caption        = Map showing Algarve
                        Region in Portugal
 | subdivision_type   = [[Countries of the
                        world|Country]]
 | subdivision_name   = {{POR}}
 | subdivision_type3  = Capital city
 | subdivision_name3  = [[Faro, Portugal|Faro]]
 | area_total_km2     = 5412
 | population_total   = 410000
 | timezone           = [[Western European
                        Time|WET]]
 | utc_offset         = +0
 | timezone_DST       = [[Western European
                        Summer Time|WEST]]
 | utc_offset_DST     = +1
 | blank_name_sec1    = [[NUTS]] code
 | blank_info_sec1    = PT15
 | blank_name_sec2    = [[GDP]] per capita
 | blank_info_sec2    = €19,200 (2006)
}}
```



Algarve

Map showing Algarve Region in Portugal

Country	Portugal
Capital city	Faro
Area	
- Total	5,412 km^2 (2,089.6 sq mi)
Population	
- Total	410,000
Time zone	WET (UTC+0)
- Summer (DST)	WEST (UTC+1)
NUTS code	PT15
GDP per capita (PPS)	€ 19,200 (2006)[1]

Figure 5: MediaWiki infobox syntax for Algarve (left) and rendered infobox (right).

Currently, the framework has 19 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, 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:wikiPageDisambiguates`.
- *External links.* Articles contain references to external Web resources which we represent by using the DBpedia property `dbpedia:wikiPageExternalLink`.
- *Page links.* We extract all links between Wikipedia articles and represent them by using the `dbpedia:wikiPageWikiLink` property.
- *Wiki page.* Links a Wikipedia article to its corresponding DBpedia resource, e.g.

(<`http://en.wikipedia.org/wiki/Germany`> <`http://xmlns.com/foaf/0.1/primaryTopic`> <`http://dbpedia.org/resource/Germany`>.).

- *Homepages.* This extractor obtains links to the homepages of entities such as companies and organizations by looking for the terms *homepage* or *website* within article links (represented by using `foaf:homepage`).
- *Geo-coordinates.* The geo-extractor expresses coordinates by using the Basic Geo (WGS84 lat/long) Vocabulary[f] and the GeoRSS Simple encoding of the W3C Geospatial Vocabulary[g]. The former expresses latitude and longitude components as separate facts, which allows for simple areal filtering in SPARQL queries.
- *Person data.* It extracts personal information such as surname, and birth date. This information is represented in predicates such as `foaf:surname`, and `dbpedia:birthDate`.
- *PND.* For each person, there is a record containing his name, birth and occupation connected with a unique identifier, which is the PND (Personennamendatei) number. PNDs are published by the German national library[h]. A PND is related to its resource via `dbpedia:individualisedPnd`.
- *SKOS categories.* It extracts information about which concept is a category and how categories are related using the SKOS Vocabulary[i].
- *Page ID.* Each Wikipedia article has a unique ID. This extractor extracts that ID and represents it using `dbpedia:wikiPageID`.
- *Revision ID.* Whenever a Wikipedia article is modified, it gets a new Revision ID. This extractor extracts that ID and represents it using `dbpedia:wikiPageRevisionID`.
- *Category label.* Wikipedia articles are arranged in categories, and this extractor extracts the labels for those categories.
- *Article categories.* Relates each DBpedia resource to its corresponding categories.
- *Infobox.* It extracts all properties from all infoboxes as described. The extracted information is represented using properties in the `http://dbpedia.org/property/` namespace. The names of these properties reflect the names of the attributed of Wikipedia infoboxes without any alterations (unmapped).
- *Mappings.* It extracts structured data based on manually-created mappings of Wikipedia infoboxes to the DBpedia ontology. First, it loads all infobox mappings defined for the required languages, DBpedia-Live supports English language only at the moment, from the mappings wiki. The mappings wiki is available at `http://mappings.dbpedia.org`. It then extracts the value of each Wikipedia property defined for that type of infobox, and generates an appropriate triple for it, based on

[f] `http://www.w3.org/2003/01/geo/`

[g] `http://www.w3.org/2005/Incubator/geo/XGR-geo/`

[h] `http://www.d-nb.de/eng/standardisierung/normdateien/pnd.htm`

[i] `http://www.w3.org/2004/02/skos/`

mappings. We will explain DBpedia mappings, and mappings wiki in Section 5.

Subsequently, DBpedia has turned into the central knowledge base within the Linking Open Data Initiative (see also (Auer et al., 2008)). It has been interlinked with other knowledge bases such as Geonames, EuroStat, the CIA World Factbook, Freebase, and OpenCyc. 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.

3.3. *Dump-Based Extraction*

As mentioned in Section 1, the DBpedia extraction framework works in either dump-based or live mode. The Wikipedia maintainers publish SQL dumps of all Wikipedia editions on a monthly basis. We regularly create new DBpedia versions with the dumps of more than 80 languages. The dump-based extraction uses the Database Wikipedia page collection as the source of article texts and generates N-Triples files as output. The resulting knowledge base is made available as Linked Data, for download[j], and via DBpedia's main SPARQL endpoint[k]. (Lehmann et al., 2009) describes the dump-based release process in detail.

One of the downsides of the dump-based extraction is, that generating new dumps still requires manual efforts and individual steps and processes such as initiating the dump, reviewing the output, updating the website and notifying OpenLink[l] about the new release. Although some parts of this process could potentially be improved by a better automatization, the whole process still takes a lot of time (about 2 days hours to generate the English dump, an additional week to generate abstracts, plus several more days for the reviewing and publishing). An even greater downside is, however, that it is nearly impossible to add fixes and improvements to the dump files once they are published: Even if at one point in time, a set of triples in one of the dump files was changed, these changes would be overridden as soon as a new dump is generated. Furthermore, such a patching system would have to handle difficult co-evolution and syncing scenarios and thus yield further problems, such as how to handle patches that become outdated when the source articles change.

In contrast, the live extraction avoids these problems, as the dataset is always up to date. Improvements and fixes can be made by editing the corresponding articles, mappings, and ontology pages.

4. Live Extraction Framework

In this section, we present the design of the DBpedia-Live extraction framework and the new features added to it.

A prerequisite for being able to perform live extraction is an access to changes made in Wikipedia. The Wikimedia foundation kindly provided us access to their update stream, the

[j] `http://dbpedia.org/downloads`

[k] `http://dbpedia.org/sparql`

[l] OpenLink is hosting the public SPARQL-endpoint for the DBpedia dumps `http://www.openlinksw.com/`

Wikipedia OAI-PMH [m] 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 6. The live extraction workflow uses this update stream to extract new knowledge upon relevant changes in Wikipedia articles.

4.1. *General System Architecture*

The general system architecture of DBpedia-Live is depicted in Figure 6. The main components of DBpedia-Live system are as follows:

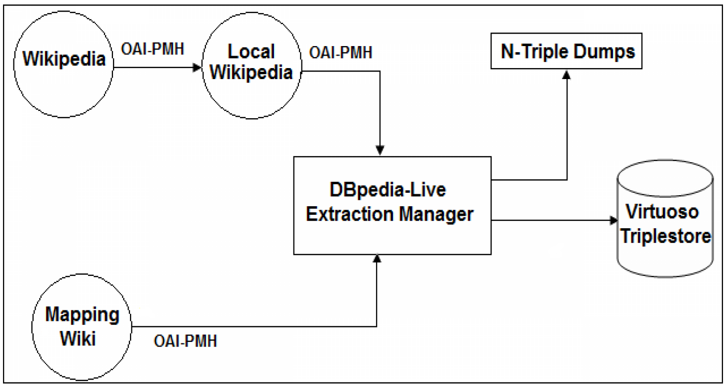


Figure 6: General DBpedia-Live system architecture.

- Local Wikipedia: We have installed a local Wikipedia that will be in synchronization with Wikipedia. The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) (Lagoze et al., 2008) enables an application to get a continuous stream of updates from a wiki. OAI-PMH is also used to feed updates into DBpedia-Live Extraction Manager.
- Mapping Wiki: DBpedia mappings can be found at the MediaWiki instance `http://mappings.dbpedia.org`. We can also use OAI-PMH to get a stream of updates in DBpedia mappings. Basically, a change of mapping affects several Wikipedia pages, which should be reprocessed. We will explain mappings in more detail in Section 5.
- DBpedia-Live Extraction Manager: This component is the actual DBpedia-Live extraction framework. When there is a page that should be processed, the framework

[m] Open Archives Initiative Protocol for Metadata Harvesting, cf. `http://www.mediawiki.org/wiki/Extension:OAIRepository`

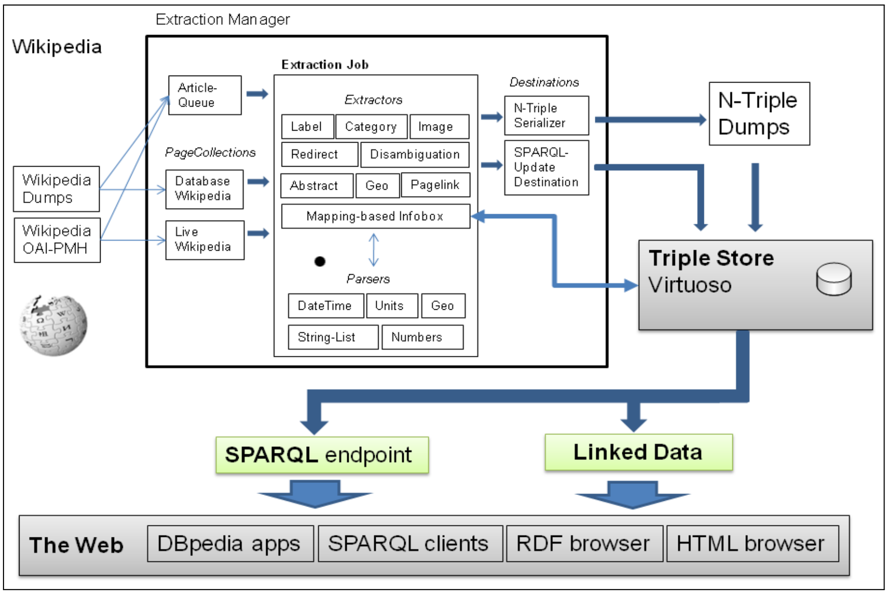


Figure 7: Overview of DBpedia-Live Extraction framework.

applies the extractors on it. After processing a page, the newly extracted triples are inserted into the backend triplestore (Virtuoso), overwriting the old triples. The newly extracted triples are also written as N-Triples file and compressed. Other applications or DBpedia-Live mirrors that should always be in synchronization with our DBpedia-Live can download those files and feed them into its own triplestore. The extraction manager is discussed in more detail below.

4.2. *Extraction Manager*

Figure 7 gives a detailed overview of the DBpedia knowledge extraction framework. The main components of the framework are:

- *PageCollection*s which are an abstraction of local or remote sources of Wikipedia articles,
- *Destination*s that store or serialize extracted RDF triples,
- *Extractor*s which turn a specific type of wiki markup into triples,
- *Parser*s which support the extractors by determining datatypes, converting values between different units and splitting markups into lists.
- *ExtractionJob* groups a page collection, extractors and a destination into a workflow.
- 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 *LiveWikipedia 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 id[n]. The *SPARQL-Update Destination* deletes existing triples and inserts new ones into the target triplestore. 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 1.8 pages per second on a 2.8 GHz machine with 6 core CPUs (this includes consumption from the stream, extraction, diffing and loading the triples into a Virtuoso triplestore, and writing the updates into compressed files)[o]. 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 and two minutes. The 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 as well as the YAGO[p] and Umbel[q] class hierarchies, which can not be updated via the English Update Stream. We store the structure of those triples using a SPARQL graph pattern. Those static triples are stored in a separate graph. All other parts of DBpedia are maintained by the extraction framework. We redesigned the extractors in such a way that each generates triples with disjoint properties. 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).

In order to decide which triples were extracted by an extractor, and also to identify the triples that should be replaced we use an RDB (relational database) assisted method, which is described in more detail in (Stadler et al., 2010). We create an RDB table consisting of 3 fields, namely page_id, resource_uri, and serialized_data. Page_id is the unique ID of the Wikipedia page. Resource_uri is the URI of DBpedia resource representing that Wikipedia article in DBpedia. Serialized_data is the JSON representation of all extracted triples. It is worth noting here that we store the extractor responsible for each group of triples along with those triples in that field. Whenever a Wikipedia page is edited, the extraction method generates a JSON object holding information about each extractor and its generated triples. After serialization of such an object, it will be stored in combination with the corresponding page identifier. In case a record with the same page identifier already exists in this table,

[n] see here for an example `http://en.wikipedia.org/wiki/Special:Export/Algarve`

[o] see current statistics at `http://live.dbpedia.org/livestats`

[p] `http://www.mpi-inf.mpg.de/yago-naga/yago/downloads.html`

[q] `http://fgiasson.com/blog/index.php/2008/09/04/exploding-dbpedias-domain-using-umbel/`

this old JSON object and the new one are compared. The results of this comparison are two disjoint sets of triples which are used on the one hand for adding statements to the DBpedia RDF graph and on the other hand for removing statements from this graph.

We had to make a number of further changes within the DBpedia extraction framework in order to support live extraction.

For instance, parsing article abstracts properly requires the resolution of templates, as described in Section 6.1. This can be done using the MediaWiki API. However, using Wikipedia directly would cause a great load on their servers, which would most likely result in an IP ban. Therefore, we use a local MediaWiki instance, which is a mirror of the English Wikipedia (synchronized via the OAI update stream).

In general, all parts of the DBpedia framework, which relied on static databases, files etc., required 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 Greek or German DBpedia [r].

5. Wiki-Based Ontology Engineering

Since the core of DBpedia is the information contained in infoboxes, they are the most interesting target for the extraction of ontological knowledge. Each infobox is mapped to a class in the DBpedia ontology and each attribute in the infobox is mapped to a property. We keep DBpedia ontology and mappings externally in another wiki, which is the Mappings-Wiki. Details about this can be found in (Hellmann et al., 2009; Lehmann et al., 2009). We provide a brief description in this paper, since the mapping wiki is also used in DBpedia-Live. Through this wiki, a DBpedia user can add more ontology classes, control ontology classes hierarchy. A DBpedia user can also change mappings, i.e. change the relation between an infobox and its corresponding ontology class and/or an infobox property and its corresponding ontology property.

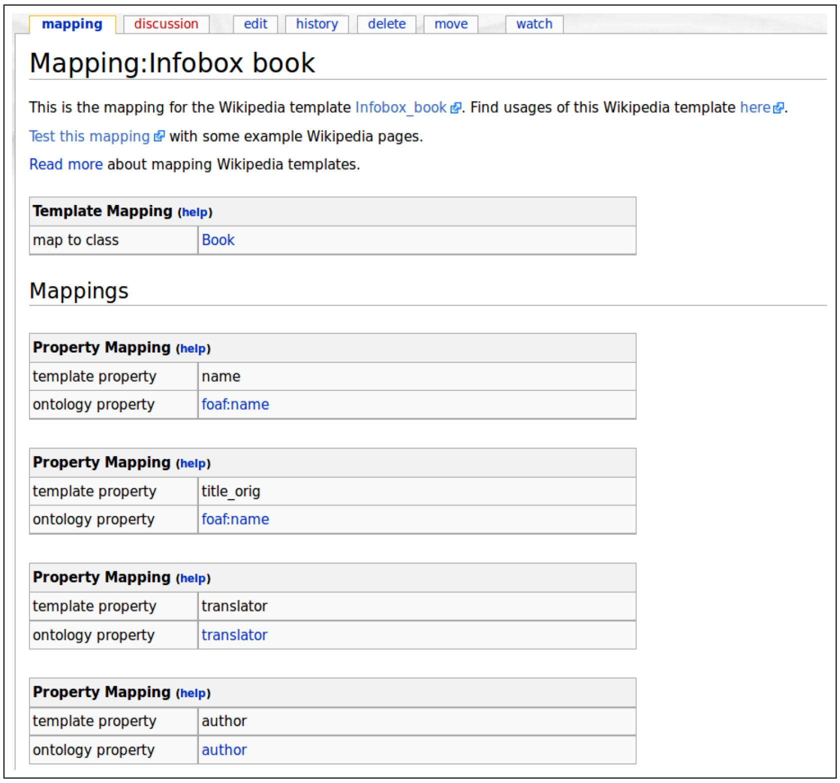
Figure 8 indicates a sample mapping for infobox of a book. This figure indicates only a subset of mappings as there are more properties associated with that type of infobox. As indicated in the figure this infobox is mapped to Book ontology class in DBpedia. Infobox property “name” is mapped to DBpedia property “foaf:name”, whereas infobox property “translator” is mapped to DBpedia property “translator” in DBpedia namespace.

5.1. *DBpedia Ontology*

The DBpedia ontology is based on OWL and forms the structural backbone of DBpedia. It describes classes e.g. writer, musical work, and book. It also describes properties e.g. birth place, running time, and author.

The DBpedia ontology wiki can be found at `http://mappings.dbpedia.org`. In order for users to be able to edit the mappings wiki, they should register themselves first. Afterwards, they should contact DBpedia maintainers to get editor rights on that mappings wiki.

[r] `http://de.dbpedia.org`



Mapping:Infobox book

This is the mapping for the Wikipedia template Infobox_book. Find usages of this Wikipedia template here.

Test this mapping with some example Wikipedia pages.

Read more about mapping Wikipedia templates.

Template Mapping (help)	
map to class	Book

Mappings

Property Mapping (help)	
template property	name
ontology property	foaf:name

Property Mapping (help)	
template property	title_orig
ontology property	foaf:name

Property Mapping (help)	
template property	translator
ontology property	translator

Property Mapping (help)	
template property	author
ontology property	author

Figure 8: Mapping for infobox of a book.

Using that wiki for ontology engineering has several advantages:

- enables any DBpedia user, with little knowledge about wiki scripting, to help DBpedia maintainers in extending and enhancing the ontology.
- enables users to add mapping for other languages, e.g. French, with ease.
- DBpedia-Live can get a stream of updates as it does with Wikipedia itself, which enables detecting and reprocessing pages affected by a mapping change. We will explain that process in more detail in Section 6.2.

5.2. *DBpedia Knowledge Base*

DBpedia knowledge base for the English language, on which DBpedia-Live depends, currently has around 190 million triples. The main advantage of DBpedia knowledge base is that it covers diverse domains, including books, writers, musicians, journalists and many others. Currently DBpedia-Live covers around 3.7 million entities, i.e. 3.7 million articles about different aspects including writers, books, musicians, etc. Most Wikipedia articles contain images, the links to those images are conveyed to DBpedia. DBpedia contains more than 700,000 links to images, and more than 6.7 million links to external web pages.

Ontology Class	No. of Instances	Sample Properties
Person	375916	birthName, birthDate, citizenship
Artist	45814	academyAward, associatedAct, field
Actor	15216	arielAward, geminiAward, nationalFilmAward
Writer	11634	notableWork
MusicalArtist	10797	
Journalist	1809	
Athlete	140282	club, debutTeam, formerTeam
SoccerPlayer	54369	appearancesInLeague, goalsInLeague, youthYears
TennisPlayer	1971	careerPrizeMoney
Work	289367	author, completionDate, license
Film	59802	cinematography, editing, imdbId
MusicalWork	145532	artist, recordDate, recordedIn
Album	104482	compiler, longtype, review
Song	4914	trackNumber
Book	23032	coverArtist, isbn, mediaType
Magazine	2765	editorTitle, issn, previousEditor
Newspaper	4064	associateEditor, chiefEditor, sisterNewspaper
Play	108	characterInPlay, ibdbId, premiereDate
Website	2247	
Organisation	145549	affiliation, chairperson, endowment
EducationalInstitution	37741	alumni, dean, educationSystem
College	78	sisterCollege
Library	445	isil
School	24581	actScore, administrator, ageRange
University	12648	campus, numberOfDoctoralStudents, provost

Table 2: Common DBpedia classes, number of instances of each one, and some properties for it.

Table 2 shows some ontology classes that are interesting to librarians. For instance, DBpedia knowledge base contains data about more than 23,000 books in various disciplines. It also decscribes more than 400 libraries and more than 12,000 universities.

6. New Features of DBpedia-Live

The old php-based framework is deployed on one of OpenLink's servers and currently has a SPARQL endpoint at `http://dbpedia-live.openlinksw.com/sparql`.

In addition to the migration to Scala and Java, the new DBpedia-Live framework has the following new features:

(1) Abstract extraction: The abstract of a Wikipedia article is the first few paragraphs of that article. The new framework has the ability to cleanly extract the abstract of an article.

(2) Mapping-affected pages: Upon a change in mapping, the pages affected by that mapping should be reprocessed and their triples should be updated to reflect that change.

(3) Updating unmodified pages: Sometimes a change in the system occurs, e.g. a change in the implementation of an extractor. This change can affect many pages even if they are not modified. In DBpedia-Live, we use a low-priority queue for such changes, such that the updates will eventually appear in DBpedia-Live, but recent Wikipedia updates are processed first.

(4) Publication of changesets: Upon modifications old triples are replaced with the updated triples. Those added and/or deleted triples are also written as N-Triples files and then

compressed. Any client application or DBpedia-Live mirror can download those files and integrate and, hence, update a local copy of DBpedia. This enables that application to be always in synchronization with our DBpedia-Live.

In the following sections, we will describe each feature in detail.

6.1. *Abstract Extraction*

The abstract extractor extracts two types of abstracts:

(1) Short abstract: is the first paragraph from a Wikipedia article and is represented in DBpedia by `rdfs:comment`.
(2) Long abstract: is the whole text before the table of contents in an article, which is represented by `dbo:abstract`.

The hurdle of abstract extraction is the resolution of templates. A template is a simple sequence of characters that has a special meaning for Wikipedia. Wikipedia renders those templates in a specific way.

The following example indicates a typical Wikipedia template

Example 6.1. `{{convert|1010000|km2|sp=us}}`

This templates tells Wikipedia that the area of some country is 1010000 square kilometers, and when it is rendered, Wikipedia should display its area in both square kilometers, and square miles. So, Wikipedia will render it as “1,010,000 square kilometers (390,000 sq mi)”. DBpedia should behave similarly towards those templates.

In order to resolve those templates used in the abstract of the article, we installed a copy of Wikipedia. The required steps to install a local copy of Wikipedia are:

(1) MySQL: Install MySQL server as back-end relational database for Wikipedia.
(2) SQL dumps: Download the latest SQL dumps for Wikipedia, which are freely available at `http://dumps.wikimedia.org/enwiki/`.
(3) Clean SQL dumps: Those SQL dumps need some adjustment, before you can insert them into MySQL. You can perform this adjustment by running “clean.sh”, which you can download from the website containing the sourcecode, see Section 6.7.
(4) Import SQL dumps: You can now use the script called “import.sh”, which is also available with the sourcecode.
(5) HTTP Server: Apache server should be installed, which will provide a front-end for abstract extraction.

6.2. *Mapping-Affected Pages*

Whenever a mapping change occurs, some pages should be reprocessed. For example, in Figure 8, if the template property called translator, which is mapped to DBpedia property translator, is changed to another property, then all entities belonging to the class Book should be reprocessed. Upon a mapping change, we identify the list of affected DBpedia entities, along with IDs of their corresponding Wikipedia pages.

Basically, the DBpedia-Live framework has a priority-queue which contains all pages waiting for processing. This priority-queue is considered the backbone of our framework as several streams including the live-update stream, mapping-update stream, and unmodified-pages stream, place the IDs of their pages in this queue. DBpedia-live consumes the contents of that queue taking the priority of updates into account.

Specifically, IDs of pages fetched from the live update stream are placed in that queue with the highest priority. The IDs of pages affected by a mapping change are also placed in that queue but with lower priority.

6.3. *Unmodified Pages*

Naturally, there is a large variation of the update frequency of individual articles in Wikipedia. If any change in the DBpedia extraction framework occurs, e.g. a modification of the implementation of an extractor or an addition of a new extractor, this will not be a problem for the frequently updated articles as it is likely that they will be reprocessed soon.

However, less frequently updated articles may not be processed for several months and would, therefore, not reflect the current implementation state of the extraction framework. To overcome this problem, we obtain the IDs of the pages which have not been modified between one and three months ago, and place their IDs in our priority-queue. Those pages have the lowest priority in order not to block or delay live extraction.

Since we use a local synchronized instance of the Wikipedia database, we can query this instance to obtain the list of such articles, which have not been modified between one and three months ago. Directing those queries against Wikipedia itself would place a too high burden on the Wikipedia servers, because the number of unmodified pages can be very large.

6.4. *Changesets*

Whenever a Wikipedia article is processed, we get two disjoint sets of triples. A set for added triples, and another set for deleted triples. We write those 2 sets into N-Triples files, compress them, and publish the compressed files on our server. If another triplestore wants to synchronize with DBpedia-Live, it can just download those files, decompress them and integrate them with its store.

The folder to which we publish the changesets has a specific structure. The folder structure is as follows:

- The parent folder contains a folder for each year while running, e.g. 2010, 2011, 2012, ...
- The folder of a year contains a folder for each month passed while running, e.g. 1, 2, 3, ..., 12.
- The folder of a month contains a folder for each day passed while running, e.g. 1, 2, 3, ..., 28/29/30/31.
- The folder of a day contains a folder for each hour passed while running, e.g. 0, 1, 2, ..., 23.

- Inside the folder of an hour, we write the compressed N-Triples files with added or removed, e.g. 000000.added.nt.gz and 000000.removed.nt.gz. This represents the 2 disjoint sets of added and/or removed triples.

To clarify that structure lets take that example:

Example 6.2. `dbpedia_publish/2011/06/02/15/000000.added.nt.gz`
`and`
`dbpedia_publish/2011/06/02/15/000000.removed.nt.gz`

This indicates that in year 2011, in 6th month of that year, 2nd day of that month, in hour 15, 2 files were written, one for added triples, and one for removed triples.

We also manage to produce a regular dump of all DBpedia triples monthly, i.e. we generate an N-Triples file containing all triples from DBpedia. The benefit of that dump file is that it makes the synchronization process between a local triplestore and our DBpedia-Live endpoint significantly faster. For instance, if we have changeset files starting from the begining of year 2011, and there is another triplestore that should be synchronized with ours, then all changeset files, starting from `dbpedia_publish/2011/01/01/000000` till the moment, must be downloaded and integrated. But, if we generate a dump file in the begining of each month, then the user should only download the latest dump file, load it to his/her Virtuoso triplestore, and start integration from that date. So, for dump file `dbpedia_2011_07_01.nt.bz2`, the user can start integration from `dbpedia_publish/2011/07/01/000000`, which is easier and faster.

6.5. *Synchronization Tool*

The synchronization tool enables a DBpedia-Live mirror to stay in synchronization with our live endpoint. It downloads the changeset files sequentially, decompresses them and integrates them with another DBpedia-Live mirror. As decscribed in Section 6.4, 2 files are written, one for added triples, and the other for deleted ones. That tool, simply downloads both of them, creates the appropriate `INSERT/DELETE` statement and executes it against the triplestore.

An interested user can download this tool and configure it properly, i.e. configure the address of his/her triplestore, login credentials for that triplestore, and so forth. Afterwards, he/she can run it to synchronize that triplestore with ours.

6.6. *Statistics about DBpedia-Live*

According to our measurements, about 84 article pages in average are updated each minute on Wikipedia. This amounts to 120,000 page updates per day. The DBpedia-Live framework is able to process 108 articles in average per minute, which is sufficient to cover the rate of updates in Wikipedia.

Since its official release, an increasing number of requests are sent to our DBpedia-Live endpoint. Figure 9 indicates the number of requests sent to DBpedia-Live endpoint within the last two monthes.

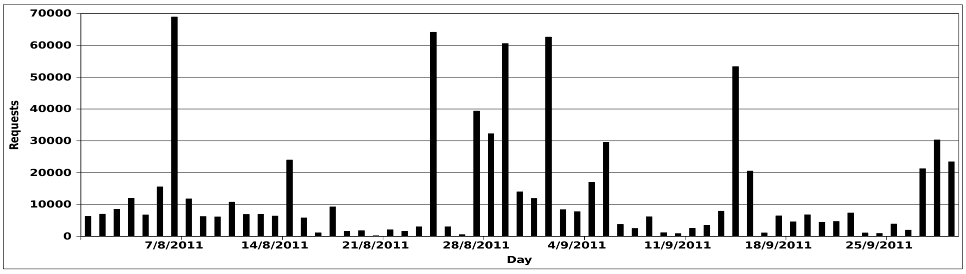


Figure 9: Number of requests sent daily to DBpedia-Live endpoint.

6.7. *Important Pointers*

- SPARQL-endpoint: The DBpedia-Live SPARQL-endpoint can be accessed at `http://live.dbpedia.org/sparql`.
- DBpedia-Live Statistics: Some simple statistics are provided upon extraction on `http://live.dbpedia.org/livestats`.
- Updates: The N-Triples files containing the updates can be found at `http://live.dbpedia.org/liveupdates`.
- DBpedia-Live Sourcecode: `http://dbpedia.hg.sourceforge.net/hgweb/dbpedia/extraction_framework`.
- Synchronization Tool: `http://sourceforge.net/projects/dbpintegrator/files/`.
- Dump files: `http://live.dbpedia.org/dumps/`.

7. How Can Librarians as well as DBpedia and Linked Data benefit from each other?

DBpedia can be very beneficial for librarians as well as other users, since it comprises structured information from Wikipedia and covers a variety of different domains. In the following paragraphs we outline some directions how Librarians as well as DBpedia and Linked Data can mutually benefit from each other.

Providing context information for bibliographic data. One feature of DBpedia is that it provides a broad range of information about a vast variety of domains. In addition to conventional library information systems which give the user only limited information about books, authors and topics, DBpedia can provide more context information. For example, library users could obtain more background information from DBpedia about authors, such as their birth places, nationality or influences.

Curated and stable identifiers. DBpedia and Semantic Web in general depend on URIs, which are guaranteed to be globally unique. This is beneficial as it enables the addition of more facts about a specific entity. For instance, DBpedia contains some facts about the publishing house ”Random House” which is identified by `http://dbpedia.org/`

`resource/Random_House`. A librarian can use this URI (either directly or by establishing an `owl:sameAs` link) to add more facts about that publisher in order to increase the coverage and quality of data available on the Web. Such a (re-)use of stable and curated identifiers facilitates the integration of data and information between different libraries as well as between libraries and other knowledge bases and applications.

Extract and publish structured (meta-)data for library content. Library databases contain very accurate and valuable data about books, publishers, etc. This data can be extracted, converted into RDF, stored in a knowledge base, and interlinked with other knowledge bases, which results in a vast knowledge base about books, their authors and other bibliographic data. One example of such an extracted and published database is the RDF version of the *Personennamendatei* (PND)[s], which is an authoritative source on persons being either authors of books or discussed in books.
It is worth mentioning here that the article abstract is an important piece of metadata of a Wikipedia article. The new DBpedia-Live framework enables the extraction of cleaner abstracts, i.e. the templates contained in the article abstract are properly resolved.

Provide storage facilities for Linked Data. The more researchers use Linked Data and semantic technologies, the more important it is to have providers of Data Web infrastructure. A biomedical researcher who developed an ontology for the classification of diseases could for example outsource the hosting and maintenance of the corresponding Linked Data endpoint to his university library similar as the hosting of Web sites is already outsourced to IT departments or specialized companies. Since in the case of ontologies and Linked Data not only technical knowledge is required, but also expertise with regard to structuring and representing knowledge, libraries could play an important role in this regard.

Support facilities for knowledge based authoring & collaboration. Libraries can provide the collaboration infrastructure for researchers and domain experts. Examples of such a collaboration infrastructure are Semantic Wikis such as OntoWiki (Auer et al., 2007) and Semantic MediaWiki (Krötzsch et al., 2006). In addition to supporting the collaboration Semantic Wikis also serve Linked Data, SPARQL and other Data Web technologies. They can be used to semantically structure information about a particular domain and to help domain experts to collect and integrate data and information. An example of such an application is the Professors Catalog (Riechert et al., 2010), where historians collaboratively created a vast semantic, prosopographical knowledge base about professors working at Universität Leipzig during its 600 years history.

Become linking hubs for the Data Web. When the accurate and comprehensive data available in library databases is converted into knowledge bases and interlinked with other information on the Web, library content can become a hub in the Web of Data. We think that in particular the qualitative and comprehensive structured information available in library

[s] `http://www.gbv.de/wikis/cls/PND_in_RDF`

information system represents a very attractive asset, which could ultimately make libraries 'lighthouses' in the Linked Data 'ocean'.

Authoritative Linked Data for quality assessment. Librarians are experts in indexing and metadata creation of books, authors, etc. Their experiences are of great importance in Linked Data, as they can evaluate the accuracy of data of that domain. The reference data created by libraries can also help to reveal problems that may exist in Linked Data, e.g. incomplete or incorrect data, which thus helps to increase the data quality on the Web.

Hosting & maintenance of Linked Data exploration tools. Several tools are built to enable normal users to explore and benefit from Linked Data available on the Web. An example of such a tool is AutoSPARQL (Lehmann and Bühmann, 2011). AutoSPARQL simplifies the SPARQL query creation process. The user can enter only keywords, which are used to from the target query, e.g. books, Dan Brown. AutoSPARQL uses those keywords to formulate the appropriate SPARQL query and executes it against the endpoint. It can also display the generated SPARQL query to the user. Similar exploration and visualisation tools are Relationship Finder (Lehmann et al., 2007*b*), Sig.ma [t], Sparallax [u], and Exhibit (Huynh et al., 2007). Libraries can become competence centers for maintaining and explaining such visualisation and exploration tools for end users.

Update propagation. The publication of changesets along with the synchronization tool, described in Sections 6.4, and 6.5 respectively, enables the update propagation from our endpoint to other DBpedia-Live mirrors. This feature can be of great importance in case users(including librarians) want to maintain their own endpoints. This helps when they mainly depend on their infrastructure, as it is much faster than depending on external infrastructure.

8. Related Work

There are several projects, which aim at extracting semantic data from Wikipedia.

YAGO2: is an extension of the YAGO knowledge base (Suchanek et al., 2008). YAGO2 uses Wikipedia, Geonames, and WordNet as sources of data. The predecessor of YAGO2, i.e. YAGO just used Wikipedia as its main source of data. It extracts several relations (e.g. subClassOf, and type) mainly from the *category pages* of Wikipedia. Category pages are lists of articles that belong to a specific category. For instance, William Shakespeare is in the category English poets. These lists give candidates for entities (i.e. William Shakespeare), candidates for concepts (IsA(William Shakespeare, Poet)), and candidates for relations (e.g. isCitizenOf(William Shakespeare, England)). YAGO, then links the Wikipedia category hierarchy to the WordNet hierarchy, in order to enhance its classification hierarchy (Suchanek et al., 2008).

[t] http://sig.ma/
[u] http://sparallax.deri.ie

YAGO2 is the new version of the YAGO project. It uses Geonames as an additional source of data. YAGO2 introduces the integration of the spatio-temporal dimension. In contrast to the original YAGO, the methodology of building YAGO2 (and also maintaining it) is systematically designed top-down with the goal of integrating entity-relationship-oriented facts with the spatial and temporal dimensions. Moreover, YAGO represent facts in the form of subject-property-object triples (SPO triples) according to the RDF data model. YAGO2 introduces a new *spatio-temporally* model, which represents facts in the form of SPOTL tuples (i.e. SPO + Time + Location). This new knowledge representation scheme is very beneficial. For example, with the old representation scheme a knowledge base may store that a certain person is the president of a certain country, but presidents of countries change. So, it is crucial to capture the time periods during which facts of this kind actually happened (Hoffart et al., 2010). Both YAGO and YAGO2, however, do not focus on extracting data from Wikipedia infoboxes.

KYLIN: is a project based also on Wikipedia, which aims at creating or completing infoboxes by extracting information from the article text. KYLIN looks for classes of pages with similar infoboxes, determines common attributes, creates training examples, learns the extractors, and runs them on each page. Thus creating new infoboxes or completing existing ones. It uses learning techniques to automatically fill in missing values in incomplete infoboxes. There are several problems which may exist in an infobox, such as incompleteness or inconsistency. For example, some infoboxes contain incomplete data which may, however, exist in the article text, while some other infoboxes may contain data that contradicts with the article text (Wu and Weld, 2007). Although both DBpedia, and KYLIN work on Wikipedia infoboxes both have different objectives. DBpedia aims at extracting data from infoboxes and converting them into semantic data, whereas KYLIN tries to fill the gaps that may exist in some infoboxes.

Freebase: is a large collaborative knowledge base, which uses various sources of data including Wikipedia and MusicBrainz[v]. It was originally developed by the software company Metaweb, which was later acquired by Google. Basically, Freebase also extracted facts from Wikipedia articles as initial content and which the users can later extend and revise (Bollacker et al., 2008). Both DBpedia and Freebase use Wikipedia as a data source, but Freebase uses it only as a starting point in a way that its users can modify the data, whereas DBpedia aims to be closely aligned with Wikipedia.

9. Conclusion and Future Work

Due to the permanent update of Wikipedia articles, we also aim to update DBpedia accordingly. We proposed a framework for instantly retrieving updates from Wikipedia, extracting RDF data from them, and storing this data in a triplestore. Our new revision of the DBpedia-Live extraction framework adds a number of features and particularly solves the following

[v] http://musicbrainz.org/

issues:

- MediaWiki templates in article abstracts are now rendered properly.
- Changes of mappings in the DBpedia mappings wiki are now retrospectively applied to all potentially affected articles.
- Updates can now be propagated easily, i.e. DBpedia-Live mirrors can now get recent updates from our framework, in order to be kept in sync.

Many users can benefit from DBpedia, not only computer scientists. We have pointed out some directions how librarians and libraries can make use of DBpedia and how they can become part of the emerging Web of Data. We see a great potential for libraries to become centers of excellence in knowledge management on the Web of Data. As libraries supported the knowledge exchange through books in previous centuries, they now have the opportunity to extend their scope towards supporting the knowledge exchange through structured data and ontologies on the Web of Data. Due to the wealth and diversity of structured knowledge already available in DBpedia and other datasets on the Web of Data many other scientists, e.g. in life sciences, humanities, or engineering, would benefit a lot from such a development.

There are several directions in which we aim to extend the DBpedia-Live framework:

Support of other languages: Currently, the framework supports only the English Wikipedia edition. We plan to extend our framework to include other languages as well. The main advantage of such a multi-lingual extension is that 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 people than the English edition. This leads to an increase of the quality of extracted data compared to knowledge bases that are derived from single Wikipedia editions. Moreover, this also helps in detecting inconsistencies across different Wikipedia and DBpedia editions.

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 (e.g. pictures and audio) from these sources. For instance, a Wikipedia page about a geographic location such as 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 along with interlinking DBpedia with external Web knowledge builds the base for detecting inconsistencies in Wikipedia content. For instance, whenever a Wikipedia author edits an infobox within a Wikipedia article, the new content of the infobox could be checked against external data sources and information extracted from other language editions. Inconsistencies could be pointed out along with proposals on how to solve these inconsistencies. In this way, DBpedia can provide feedback to Wikipedia maintainers in order to keep Wikipedia data more consistent, which eventually may also lead to an increase of the quality of data in

Wikipedia.

Acknowledgments

We thank the people at Openlink software for their valuable support during the development of DBpedia-Live. This work was supported by a grant from the European Union's 7th Framework Programme provided for the projects LOD2 (GA no. 257943) and LATC (GA no. 256975).

References

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. and Ives, Z. (2008), DBpedia: A nucleus for a web of open data, *in* 'Proceedings of the 6th International Semantic Web Conference (ISWC)', Vol. 4825 of *Lecture Notes in Computer Science*, Springer, pp. 722–735.

Auer, S., Dietzold, S., Lehmann, J. and Riechert, T. (2007), OntoWiki: A tool for social, semantic collaboration, *in* N. F. Noy, H. Alani, G. Stumme, P. Mika, Y. Sure and D. Vrandecic, eds, 'Proceedings of the Workshop on Social and Collaborative Construction of Structured Knowledge (CKC 2007) at the 16th International World Wide Web Conference (WWW2007) Banff, Canada, May 8, 2007', Vol. 273 of *CEUR Workshop Proceedings*, CEUR-WS.org.
URL: *http://ceur-ws.org/Vol-273/paper_91.pdf*

Auer, S. and Lehmann, J. (2007), What have Innsbruck and Leipzig in common? extracting semantics from wiki content, *in* 'Proceedings of the ESWC (2007)', Vol. 4519 of *Lecture Notes in Computer Science*, Springer, Berlin / Heidelberg, pp. 503–517.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F. and Stein, L. A. (2004), OWL Web Ontology Language Reference, Technical report, World Wide Web Consortium.
URL: *http://www.w3.org/TR/owl-ref/*

Berners-Lee, T., Hendler, J. and Lassila, O. (2001), 'The semantic web', *Scientific American* **284**(5), 34–43.
URL: *http://www.scientificamerican.com/article.cfm?id=the-semantic-web*

Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z. and Velkov, R. (2011), 'OWLIM: A family of scalable semantic repositories', *Semantic Web* **2**(1), 33–42.
URL: *http://dx.doi.org/10.3233/SW-2011-0026*

Bollacker, K., Evans, C., Paritosh, P., Sturge, T. and Taylor, J. (2008), Freebase: a collaboratively created graph database for structuring human knowledge, *in* 'SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data', ACM, New York, NY, USA, pp. 1247–1250.
URL: *http://ids.snu.ac.kr/w/images/9/98/SC17.pdf*

Brickley, D. and Guha, R. V. (2004), RDF Vocabulary Description Language 1.0: RDF Schema, Recommendation, World Wide Web Consortium (W3C). http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

Broekstra, J., Kampman, A. and van Harmelen, F. (2002), Sesame: A generic architecture for storing and querying RDF and RDF schema, *in* I. Horrocks and J. Hendler, eds, ‘Proceedings of the First Internation Semantic Web Conference’, number 2342 *in* ‘Lecture Notes in Computer Science’, Springer Verlag, pp. 54–68.

Erling, O. and Mikhailov, I. (2007), RDF support in the virtuoso DBMS, *in* S. Auer, C. Bizer, C. Müller and A. V. Zhdanova, eds, ‘CSSW’, Vol. 113 of *LNI*, GI, pp. 59–68.
URL: *http://aksw.org/cssw07/paper/5_erling.pdf*

Grant, J. and Beckett, D. (2004), RDF test cases, W3C recommendation, World Wide Web Consortium.
URL: `http://www.w3.org/TR/rdf-testcases/`

Heflin, J. (2004), ‘Web ontology language (owl) use cases and requirements’, World Wide Web Consortium, Recommendation REC-webont-req-20040210. http://www.w3.org/TR/2004/REC-webont-req-20040210.

Hellmann, S., Stadler, C., Lehmann, J. and Auer, S. (2009), DBpedia live extraction, *in* ‘Proc. of 8th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE)’, Vol. 5871 of *Lecture Notes in Computer Science*, pp. 1209–1223.
URL:
http://svn.aksw.org/papers/2009/ODBASE_LiveExtraction/dbpedia_live_extraction_public.pdf

Hoffart, J., Suchanek, F. M., Berberich, K. and Weikum, G. (2010), YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia, Research Report MPI-I-2010-5-007, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany.

Huynh, D. F., Karger, D. R. and Miller, R. C. (2007), Exhibit: lightweight structured data publishing, *in* ‘Proceedings of the 16th international conference on World Wide Web’, ACM, Banff, Alberta, Canada, pp. 737–746.
URL: *http://portal.acm.org/citation.cfm?id=1242572.1242672*

Krötzsch, M., Vrandečić, D. and Völkel, M. (2006), Semantic MediaWiki, *in* ‘The Semantic Web - ISWC 2006’, Vol. 4273 of *Lecture Notes in Computer Science*, Springer, Heidelberg, DE, pp. 935–942.
URL: *http://dx.doi.org/10.1007/11926078_68*

Lagoze, C., de Sompel, H. V., Nelson, M. and Warner, S. (2008), ‘The open archives initiative protocol for metadata harvesting’, `http://www.openarchives.org/OAI/openarchivesprotocol.html`.

Lehmann, J., Bizer, C., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R. and Hellmann, S. (2009), ‘DBpedia - a crystallization point for the web of data’, *Journal of Web Semantics* **7**(3), 154–165.
URL: *http://jens-lehmann.org/files/2009/dbpedia_jws.pdf*

Lehmann, J. and Bühmann, L. (2011), Autosparql: Let users query your knowledge base, *in* ‘Proceedings of ESWC 2011’.
URL: *http://jens-lehmann.org/files/2011/autosparql_eswc.pdf*

Lehmann, J., Schüppel, J. and Auer, S. (2007*a*), Discovering unknown connections - the DBpedia relationship finder, *in* ‘Proceedings of the 1st SABRE Conference on Social

Semantic Web (CSSW)'.

Lehmann, J., Schüppel, J. and Auer, S. (2007*b*), Discovering unknown connections - the DBpedia relationship finder, *in* 'Proceedings of 1st Conference on Social Semantic Web. Leipzig (CSSW'07), 24.-28. September', Vol. P-113 of GI-Edition of *Lecture Notes in Informatics (LNI)*, Bonner Köllen Verlag.
URL: *http://www.informatik.uni-leipzig.de/ auer/publication/relfinder.pdf*

Prud'hommeaux, E. and Seaborne, A. (2008), SPARQL query language for RDF, W3C recommendation, W3C.
URL: *http://www.w3.org/TR/rdf-sparql-query/*

Riechert, T., Morgenstern, U., Auer, S., Tramp, S. and Martin, M. (2010), Knowledge engineering for historians on the example of the catalogus professorum lipsiensis, *in* P. F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Z. Pan, I. Horrocks and B. Glimm, eds, 'Proceedings of the 9th International Semantic Web Conference (ISWC2010)', Vol. 6497 of *Lecture Notes in Computer Science*, Springer, Shanghai / China, pp. 225–240.
URL: *http://www.springerlink.com/content/n611284x31l1552p/*

Stadler, C., Martin, M., Lehmann, J. and Hellmann, S. (2010), Update Strategies for DBpedia Live, *in* G. T. W. G. A. Grimnes, ed., 'Proc. of 8th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE)', Vol. 699 of *CEUR Workshop Proceedings ISSN 1613-0073*.
URL: *http://CEUR-WS.org/Vol-699/Paper5.pdf*

Suchanek, F. M., Kasneci, G. and Weikum, G. (2008), 'Yago: A large ontology from wikipedia and wordnet', *Journal of Web Semantics* **6**(3), 203–217.

W3C Semantic Web Activity (2009). http://www.w3.org/2001/sw/.
URL: *http://www.w3.org/2001/sw/*

Wu, F. and Weld, D. S. (2007), Autonomously semantifying wikipedia, *in* 'CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management', ACM, New York, NY, USA, pp. 41–50.
URL: *http://portal.acm.org/citation.cfm?id=1321440.1321449*

Yu, L. (2007), *Introduction to Semantic Web and Semantic Web services*, Chapman & Hall/CRC, Boca Raton, FL.

About the authors

Mohamed Morsey is PhD student at the University of Leipzig. He started his PhD in April 2010 in the "Agile Knowledge Engineering and Semantic Web" (AKSW) research group. He obtained his M.Sc. in Computer Science from Ain Shams University in Cairo, Egypt in 2006. His main interests are Semantic Web, Information Security, and Object Oriented Analysis and Design. He is currently working in DBpedia project. Mohamed Morsey can be contacted at: `morsey@informatik.uni-leipzig.de`

Dr. Jens Lehmann is a postdoctoral researcher at the University of Leipzig and research visitor at the University of Oxford. He is leading the Machine Learning and Ontology Engineering research group within the AKSW center. He obtained a PhD with grade summa cum laude at the University of Leipzig in 2010 and a master degree in Computer Science

from Technical University of Dresden in 2006. His research interests involve Semantic Web, machine learning and knowledge representation. He is founder, leader or contributor of several open source projects, including DL-Learner, DBpedia, LinkedGeoData, ORE, and OntoWiki. He works/worked in several funded projects, e.g. LOD2 (EU IP), LATC (EU STREP) and SoftWiki (BmBF). Dr. Jens Lehmann authored more than 25 articles in international journals and conferences. Jens Lehmann can be contacted at: `lehmann@informatik.uni-leipzig.de`.
Website: http://www.jens-lehmann.org.

Dr. Sören Auer leads the research group Agile Knowledge Engineering and Semantic Web (AKSW) at Universität Leipzig. His research interests are centered around semantic data web technologies. Sören is author of over 70 peer-reviewed scientific publications resulting in a Hirsch index of 17. Sören is leading the large-scale integrated EU-FP7-ICT research project LOD2 and (co-)founder of several high-impact research and community projects such as DBpedia, LinkedGeoData and OntoWiki. He is co-organiser of workshops, programme chair international conferences, area editor of the Semantic Web Journal, serves as an expert for industry, EC, W3C and advisory board member of the Open Knowledge Foundation. Sören Auer can be contacted at: `auer@informatik.uni-leipzig.de`

After obtaining his diploma in Computer Science at the University of Leipzig in 2011, Claus Stadler started his PhD in the "Agile Knowledge Engineering and Semantic Web" (AKSW) research group. His main research interests are Semantic Web technologies, specifically those related to infrastructure, data and information integration, query rewriting and optimization, update propagation, and the modelling of spatial/temporal data. In the course of the LinkedGeoData-project he is currently working with geographical data in the Semantic Web. Claus Stadler can be contacted at: `CStadler@informatik.uni-leipzig.de`

Sebastian Hellmann obtained his Master degree in 2008 from the University of Leipzig, where he is currently researching as a PhD Student in the Agile Knowledge Engineering and Semantic Web (AKSW) research group. He is founder, leader, or contributor of several open source projects, including DL-Learner, DBpedia, and NLP2RDF. Among his research interests are light-weight ontology engineering methods, data integration, and scalability in the Web Of Data. Sebastian is author of over 10 peer-reviewed scientific publications and was chair at the Open Knowledge Conference in 2011. Sebastian Hellmann can be contacted at: `hellmann@informatik.uni-leipzig.de`