LOD2 Deliverable 3.2.2
DBpedia-Live Extraction

Mohamed Morsey, Jens Lehmann

Dissemination Level	Public
Due Date of Deliverable	Month 8, 30/4/2011
Actual Submission Date	7/6/2011
Work Package	WP3, Knowledge Base Creation, Enrichment and Repair
Task	Task T3.2
Type	Report
Approval Status	Approved
Version	1.0
Number of Pages	20
Filename	deliverable-3.2.2.pdf

Abstract: DBpedia is the Semantic Web mirror of Wikipedia. Wikipedia users constantly revise Wikipedia articles almost each second. Hence, data stored in DBpedia triplestore can quickly become outdated, and Wikipedia articles need to be re-extracted. DBpedia-Live, the result of this deliverable, enables such a continuous synchronization between DBpedia and Wikipedia.

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

History

Version	Date	Reason	Revised by
0.1	2011-05-16	Initial version	Jens Lehmann
0.5	2011-05-18	First deliverable version	Mohamed Morsey
1.0	2011-05-23	Deliverable revised and extended	Jens Lehmann

Author list

Organisation	Name	Contact Information
ULEI	Mohamed Morsey	morsey@informatik.uni-leipzig.de
ULEI	Jens Lehmann	lehmann@informatik.uni-leipzig.de

Executive summary

This deliverable contains a brief description of DBpedia Live – the synchronisation framework for aligning Wikipedia and DBpedia. D3.2.2 is a software prototype deliverable. We made all source code openly available in the DBpedia code repository, provide a public SPARQL endpoint and access to all changes performed by DBpedia Live.

Contents

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`. Over the past four 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 Mobile[1], Query Builder[2], Relation Finder[6], and Navigator[3]. It is used in a variety of applications, for instance Muddy Boots, Open Calais, Faviki, Zemanta, LODr, and TopBraid Composer (cf. [5]).

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 deliverable, we present a live extraction framework, which allows DBpedia to be up-to-date with a minimal delay of only 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.

Overall, we make the following contributions:

- migration the previous incomplete DBpedia-Live framework, which was PHP-based, to the new Java-based framework, which also maintains up-to-date information,
- addition of abstract extraction capability,
- re-extraction of mapping-affected pages,
- flexible low-priority re-extraction of 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

[1]`http://beckr.org/DBpediaMobile/`
[2]`http://querybuilder.dbpedia.org`
[3]`http://navigator.dbpedia.org`

- publishing added and deleted triples as compressed N-Triples file.
- building a synchronization tool, which downloads those those compressed N-Triples files, and update another triplestore with them accordingly, in order to keep it in synchronization with ours.
- deployment of the framework on our server.

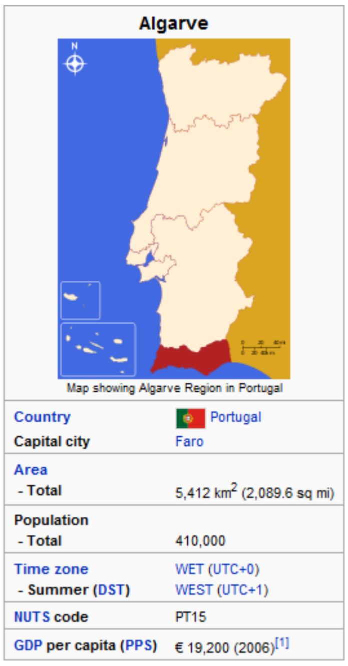
2 Overview

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 2.1). 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 pre-processed and converted to 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 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.

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, 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`.

```
{{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 2.1: Mediawiki infobox syntax for Algarve (left) and rendered infobox (right).

- *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[1] and the GeoRSS Simple encoding of

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

the W3C Geospatial Vocabulary[2]. 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 like `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. 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.
- *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.
- *Mappings.* It extracts structured data based on hand-generated mappings of Wikipedia infoboxes to the DBpedia ontology. First, it loads all infobox mappings defined for the required languages, English only in that case, 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 mappings. We will explain DBpedia mappings, and mappings Wiki in Chapter 4.
- *Infobox.* It extracts all properties from all infoboxes, and the extracted information is represented using properties in the `http://dbpedia.org/property/` namespace. The names of these properties reflect the names of properties of Wikipedia infoboxes, as described before.

Subsequently, DBpedia has turned into the central knowledge base within the Linking Open Data Initiative (see also [1]). It has been interlinked with other

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

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 [5]): 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 infobox templates within the English edition of Wikipedia into a subsumption hierarchy consisting of 305 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.

3 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 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* [1] 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 Chapter 5. The live extraction workflow uses this update stream to extract new knowledge upon relevant changes in Wikipedia articles.

3.1 General System Architecture

The general system architecture of DBpedia-Live is depicted in Figure 3.1. The main components of DBpedia-Live system are as follows:

- 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) [4] 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 `http://mappings.dbpedia.org`. It is also a wiki. We can also use OAI-PMH to get 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 Chapter 4.
- DBpedia-Live Extraction Manager: This component is the actual DBpedia-Live extraction framework. When there is a page that should be processed, the framework applies the extractors on it. After processing a page, the newly extracted triples are inserted into the backend triple store (Virtuoso), overwriting the old triples. The newly extracted triples are also written

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

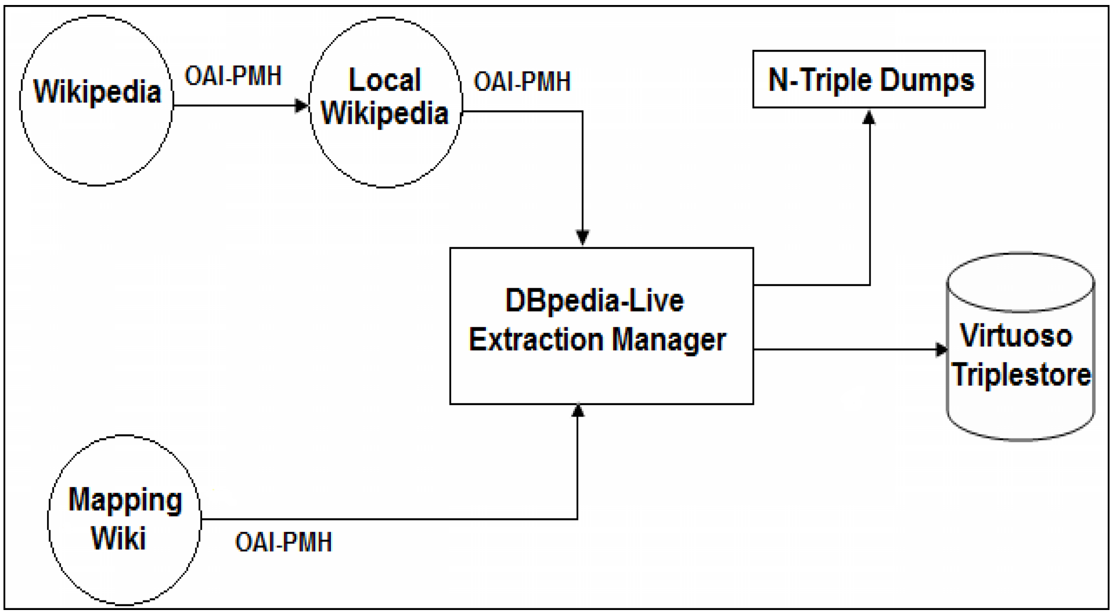


Figure 3.1: General DBpedia-Live system architecture.

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.

3.2 Extraction Manager

Figure 3.2 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.

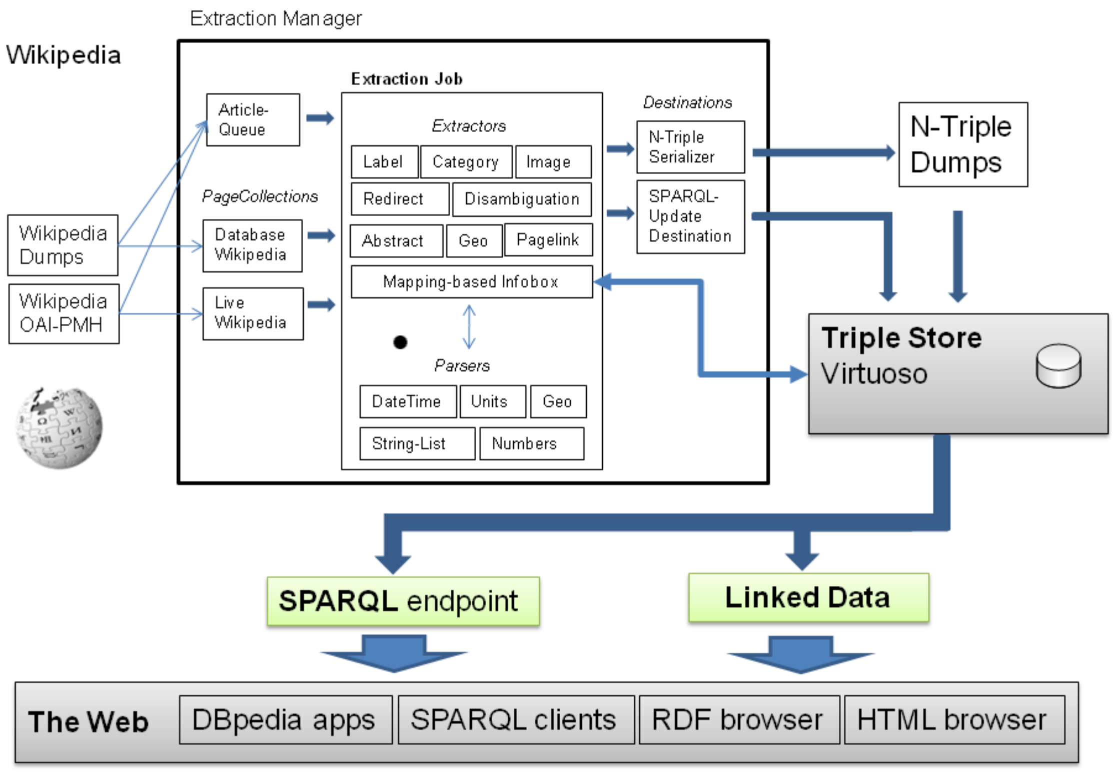


Figure 3.2: Overview of DBpedia Live Extraction framework.

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[2]. 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 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 triple store, and writing the updates into compressed files)[3]. 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

[2]see here for an example `http://en.wikipedia.org/wiki/Special:Export/Algarve`
[3]see current statistics at `http://live.dbpedia.org/livestats`

"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[4] and Umbel[5] 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).

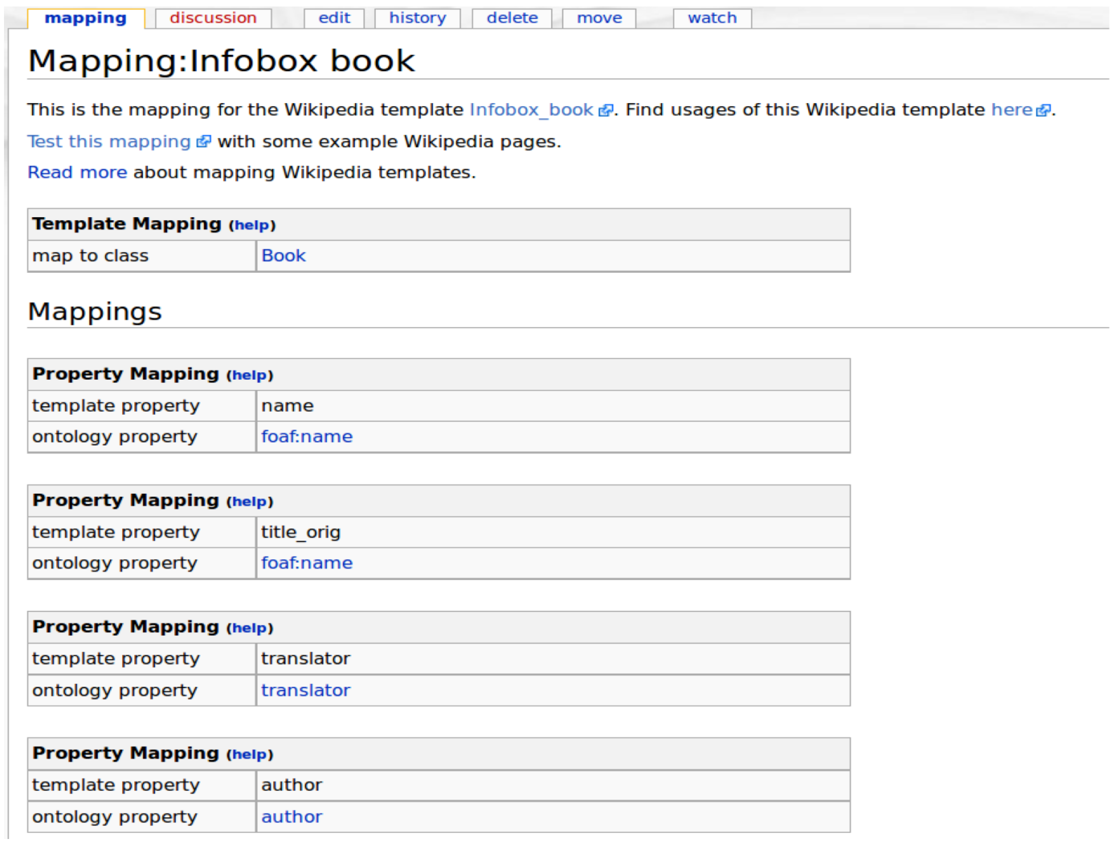
In order to decide which triples were extracted by an extractor, and also to identify the triples that should be replaced we use an RDB (relational database) assisted method, which is described in more detail in [7]. We create an RDB table consisting of 3 fields, namely page_id, resource_uri, and serialized_data. 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, 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, 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 Greek or German DBpedia [6].

[4]`http://www.mpi-inf.mpg.de/yago-naga/yago/downloads.html`
[5]`http://fgiasson.com/blog/index.php/2008/09/04/exploding-dbpedias-domain-using-umbel/`
[6]`http://de.dbpedia.org`

4 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 [5] and [3]. We provide a brief description in this deliverable, 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.



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 4.1: Mapping for infobox of a book.

Figure 4.1 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.

The DBpedia ontology is based on OWL and forms the structural backbone of DBpedia. It describes classes e.g. soccer player, city, and movie. It also describes properties e.g. birth place, area, and running time. The DBpedia ontology wiki can be found at `http://mappings.dbpedia.org`.

In order for a user to be able to edit the mapping wiki, he/she should register him/herself first. Afterwards, the user should contact DBpedia maintainers to get editor rights on that mappings wiki.

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 5.2.

5 New Features

The old php-based framework is deployed on one of OpenLink[1] servers and currently has a SPARQL endpoint at `http://dbpedia-live.openlinksw.com/sparql`.

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

1. Abstract extraction: The abstract of of a Wikipedia article are 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 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 always in synchronization with our DBpedia-Live.

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

5.1 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 table of contents in an article, which is represented by `dbpedia:abstract`.

[1]`http://www.openlinksw.com/`

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

```
{{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 should install 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 5.5.
4. Import SQL dumps: You can now use 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.

5.2 Mapping-Affected Pages

Whenever a mapping change occurs, some pages should be reprocessed. For example, in Figure 4.1, 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.

5.3 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 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.

5.4 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 triples store wants to synchronise 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:

```
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 written, one for added triples, and one for removed triples.

We have also developed a light-weight tool for downloading those files, decompressing them and integrating them with another DBpedia-Live mirror. An interested user can download this tool (see URL below) 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.

5.5 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/`.

Bibliography

[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, *The Semantic Web: Research and Applications, 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria, June 3-7, 2007, Proceedings*, volume 4519 of *Lecture Notes in Computer Science*, pages 503–517. Springer, 2007.

[3] Sebastian Hellmann, Claus Stadler, Jens Lehmann, and Sören Auer. DBpedia live extraction. In *Proc. of 8th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE)*, volume 5871 of *Lecture Notes in Computer Science*, pages 1209–1223, 2009.

[4] Carl Lagoze, Herbert Van de Sompel, Michael Nelson, and Simeon Warner. The open archives initiative protocol for metadata harvesting. `http://www.openarchives.org/OAI/openarchivesprotocol.html`, 2008.

[5] Jens Lehmann, Chris Bizer, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009.

[6] 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.

[7] Claus Stadler, Michael Martin, Jens Lehmann, and Sebastian Hellmann. Update Strategies for DBpedia Live. In Gregory Todd Williams Gunnar Aastrand Grimnes, editor, *Proc. of 8th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE)*, volume 699 of *CEUR Workshop Proceedings ISSN 1613-0073*, February 2010.