The DBpedia Wayback Machine *

Javier D. Fernández
Vienna University of Economics and Business, Vienna, Austria
javier.fernandez@wu.ac.at

Patrik Schneider
Vienna University of Economics and Business, Vienna, Austria
patrik.schneider@wu.ac.at

Jürgen Umbrich
Vienna University of Economics and Business, Vienna, Austria
juergen.umbrich@wu.ac.at

ABSTRACT

DBpedia is one of the biggest and most important focal point of the Linked Open Data movement. However, in spite of its multiple services, it lacks a wayback mechanism to retrieve historical versions of resources at a given timestamp in the past, thus preventing systems to work on the full history of RDF documents. In this paper, we present a framework that serves this mechanism and is publicly offered through a Web UI and a RESTful API, following the Linked Open Data principles.

1. INTRODUCTION

DBpedia is one of the biggest cross-domain datasets in RDF [3] and, not surprisingly, it can be considered as the nucleus for the Linked Open Data cloud [1]. It conforms to the vision of a big semantic dataset, on which a plethora of third-party semantic applications are based. Most third-party tools on top of DBpedia use the official data dumps (and its associated SPARQL endpoint), which are released at least once a year[1]. For applications requiring a live state reflecting the latests Wikipedia changes, DBpedia Live[2] monitors these modifications and converts the Wikipedia articles to RDF statements in real-time and continuously synchronises the DBpedia Live endpoint.

However, the current ecosystem does not offer a "wayback" mechanism, allowing applications to retrieve the RDF version of a Wikipedia article for a certain time in the past. Similar scenarios have been already envisioned for Web archiving, i.e. maintaining snapshots of the Web at different times. For instance, the non-profit project called the *Internet Archive Wayback Machine*[3] periodically crawls the Web and provides these time-preserved website snapshots for open use. The wayback mechanism offers access to the history of a document and enables several new use cases and supports different research areas. For instance, one can extract historical data and capture the "Zeitgeist" of a culture, or use the data to study evolutionary patterns. Previous works on providing time-travel mechanism in DBpedia, such as [6], focus on archiving different snapshots of the dumps and serve these by means of the memento protocol (RFC 7089)[4], an HTTP content negotiation to navigate trough available snapshots. Although thisa perfectly merges the existing time-travel solution for WWW and DBpedia dumps, it is certainly restricted to pre-fetched versions than must be archived beforehand. In turn, storing all versions of DBpedia resources would face traditional scalability problems when archiving dynamic Linked Open Data at Web scale [4].

In this paper we aim at providing the wayback functionality for DBpedia at any selected time based on the revisions of their Wikipedia article, thus allowing Linked Open Data users and third-party systems to work on the full history of resources. To do so, for a given resource and time, we inspect the change history of Wikipedia pages, get the appropriate page version/s and apply the *DBpedia Knowledge Extraction Framework* [1] to convert the Wikipedia version to the DBpedia version in RDF. In addition, we provide the metadata of the retrieved revisions as RDF, so that automatic clients can work on top of our infrastructure. At first sight, it appears to be a simple task accessing the Wikipedia history and converting the exported version to RDF by the DBpedia Information Extraction Framework. At second sight a wayback functionality for DBpedia has to face several challenges:

- **Dynamic mappings and ontologies**: The whole DBpedia ecosystem is community driven and the DBpedia Ontology and the language mappings from Wikipedia tags to the properties and classes are continuously undergoing changes. As such, a Wayback machine has to handle the different versions of the ontology and mappings. Ideally, an agent can request with which mapping and ontology version a Wikipedia article is extracted and transformed into RDF.

- **Evolution of the code base** Similarly to the previous point, the extraction framework itself is consistently improved or new extractors are added (e.g., new extractors for tables in the article content rather than

*Supported by the Austrian Science Fund (FWF): M1720-G11, and the Vienna Science and Technology Fund (WWTF) project ICT12-15

[1]http://wiki.dbpedia.org/services-resources/datasets/change-log

[2]http://live.dbpedia.org/

[3]https://archive.org/web/

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SEMANTiCS '15, September 15 - 17, 2015, Vienna, Austria

© 2015 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ISBN 978-1-4503-3462-4/15/09...$15.00

DOI: http://dx.doi.org/10.1145/2814864.2814889

[4]https://tools.ietf.org/html/rfc7089

only the infoboxes). This could cause inconsistent representation of a particular article at a particular time if we always use the most up-to-date code base. The wayback framework should cater for this and allow to specify the set of extractors (and their version).

- **Modelling revisions** Considering the inherent dynamicity of Wikipedia and the DBpedia extraction framework, the DBpedia Wayback Machine has to provide very detailed provenance information for the extracted RDF such that an agent has full information about the origin and transformation process.

- **Detecting changes in the RDF version** While the Wikipedia revision API gives us full access to all changes for an Wikipedia article, not all of these revisions result in changes in the return of the DBpedia extraction framework (e.g., changes in parts of the article which are not extracted by the current framework). To accurately capture and represent the change history of an DBpedia representation, we need algorithms which allow us to detect if a Wikipedia change would also trigger an RDF change.

In the reminder of this work, we briefly introduce a use case for the DBpedia Wayback Machine (Section 2), and present our framework, the wayback process, design choices and functionalities (Section 3). We conclude and devise future work in Section 4.

2. USE CASE - CITY DATA PIPELINE

Studies like the Green City Index assess and compare the performance of cities regarding their environmental impact and are helpful for public awareness but also for administration and infrastructure providers. They are based on large collections of data sources which should be timely updated easy accessibly by open standards (e.g., RDF, SPARQL), and integrated by a unified vocabulary. The Open City Data Pipeline [2] is a platform which supports such indexes by integrating several data sources like Eurostat Urban Audit and DBpedia. DBpedia is a suitable source for city indicators since it contains data on topics like geography, demography, and weather in the Wikipedia infoboxes. Some of these properties remain static (e.g., city's coordinates) and can be extracted from traditional DBpedia APIs (e.g., the SPARQL Endpoint). However, time-series data that change across time (like the total population or average temperature) and are overwritten in Wikipedia, hence DBpedia only holds the most recent value (at the moment of extraction). The City Data Pipeline uses the DBpedia Wayback Machine to load time-series data of cities by accessing the revision history monthly, filtering selected properties. These are then integrated and are accessible in the City Data Pipeline.

3. DBPEDIA WAYBACK MACHINE

The architecture of the DBpedia Wayback Machine (DWM) framework is depicted in Figure 1 and consists of several modules and services, explained in the following. The framework is developed in Scala[5] which facilitates the integration of the DBpedia components, also developed in Scala.

[5]http://www.scala-lang.org/

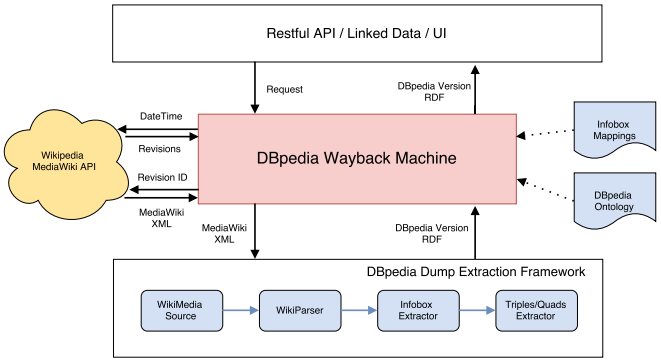

Figure 1: DBpedia Wayback Machine Architecture.

3.1 Operations

We currently offer three operations for a given Wikipedia article (or *resource*), expecting as input the Wikipedia article title (in *English*) and additional parameters.

(a) GET Revisions: This is the initial operation to query the revision history of a resource. The framework takes the resource URI and the latest timestamp(s) as an input and performs the following steps:

1. The *Wikipedia MediaWiki API* is queried to extract the revisions of a single timestamp or a time range;
2. The received revision information is converted into triples using our data model, described in Section 3.2.

The main access point for the entire revision history of a Wikipedia resource is the *MediaWiki* revision API[6]. We query this API and extract the revisions as XML using the query parameters `action=query&prop=revisions&format=xml` to obtain the fields ids, timestamp, userid, etc. We convert the resulting MediaWiki XML into the internal format required by the DBpedia Extraction Framework.

(b) GET Version (by date / revision ID): This operation returns the "historical" RDF version for a given Wikipedia article based on a specified date or revision ID. In case a date is specified, we map its value to the closest revision ID. We perform the following processing pipeline:

1. We extract the MediaWiki version of the article for the given revision ID using the *Wikipedia MediaWiki API*;
2. The DBpedia Information Extraction Framework [1] (DEF) is applied to convert the MediaWiki content into its DBpedia format, represented by RDF quads;
3. The requested return format is determined by using content-negotiation. Our framework supports the typical RDF serialisation formats (such as RDF/XML, turtle or n3). However, we currently distinguish between a triple and quad representation and provide more details in Section 3.2.

The DEF is a collection of several modules (Core, Server, Live, and Wiktionary), the DBpedia ontology, and mappings to convert Wikipedia articles to their DBpedia version. In the Core module are the various extractors (see [1] for details) for different parts of a Wikipedia article such as labels, abstracts, categories, and the infobox. Our current framework uses only the InfoboxExtractor to extract the DBpedia version.

(c) GET changes (between two versions): This oper-

[6]https://en.wikipedia.org/w/api.php

Figure 2: DBpedia Wayback Machine website.

ation computes the difference between two DBpedia revisions. To do so, we apply our GET VERSION function for an article and the two specified revisions and convert the resulting RDF statements into quads. Next, we sort the resulting quads in lexicographically order and perform a line-oriented data comparison (e.g., similar to GNU Diff)[7] based on Myer's diff algorithm [5] to compute the added and deleted statements between the two versions. The algorithm calculates the minimum number of symbol deletions and insertions that transform from one sequence to another.

Resource optimisation.

We observed that converting a Wikipedia article into DBpedia version takes roughly around 1 to 2 seconds. As such, we only process an article for a given version once and cache the results to disk. Optionally, we filter out the resulting RDF statements which are part of the DBpedia ontology, such as property type assertions (e.g., `<dbpedia:population> <rdf:type> <rdf:property>`) and labels (e.g., `<dbpedia:population> <rdfs:label> "Inhabitants"`). This filter is interesting for use cases that require only the time-series values of articles. In such scenarios it is more resource optimal to integrate the full DBpedia ontology together with the compact versions.

Mappings & DBpedia Ontology.

We currently use the most recent DBpedia ontology and mappings for the conversion.[8] An open issue is to use past versions to exactly retrieve what one would see at a given timestamp. Note that mappings are interpreted by extractors, whose past code should be retrieved from the GIT repository of DEF. Thus, we currently focus on providing a fresh view (with the recent and potentially more accurate mappings) of historical data.

3.2 Revision Ontology & Data Model

The revision ontology of the DWM framework is directly derived from the *MediaWiki* representation of the revision data. The *basic* properties are mapped from *MediaWiki* and include `id`, `timestamp`, `user`, `comment`, `sha1`, and `parentid`. The *custom* properties describe linking, provenance, and version comparison information. The linking properties refer to Wikipedia by `sourceWiki`, to DBpedia by `source` and to the extracted version by `data`. The provenance properties include the version of the DBpedia ontology, the extraction framework (i.e., DEF), the extraction date, and used extractors (e.g., the infobox extractor). The version comparison information include the difference to the last version of the MediaWiki (resp. DBpedia) export denoted as `diffwiki` (resp. `diffrdf`). The provenance and version comparison properties are only created when the DBpedia versions are already cached, or the GET CHANGES operation is used.

Depending on the output representation, we either return quads or triples. For quads, the revision ID is encoded in the 4th element resulting in `<DBpediaSubject> <Predicate> <Object> <Revision>`. For triples, we provide two versions. In the first, we use our own dereferenceable URI for the subject by moving the revision ID into it resulting in `<OwnSubject/Revision> <Predicate> <Object>`. In the second, we strip the quads by the context and keep the subjects which results in `<DBpediaSubject> <Predicate> <Object>`. We support two representations for triples, since some use cases demand a single RDF graph with dereferenceable URIs (i.e., reasoners), others need RDF fragments with original URIs (i.e., archiving systems).

The example represents the revision 607920704 of Vienna:

```
@prefix :      <http://data.wu.ac.at/wayback/dbepdia/> .
@prefix wb:   <http://data.wu.ac.at/wayback/ns#> .
:Vienna/revision/id/607920704 a wb:Revision ;
  wb:revid 607920704 ;
  wb:parentid 607920550 ;
  wb:source <http://en.dbpedia.org/resource/Vienna> ;
  wb:data :Vienna/id/607920704
  ...
  wb:extractedTime "2015-06-18T15:13:03Z"^^xsd:dateTime ;
  wb:comment "Example" .
```

The DBpedia version of this revision is by quads:

```
<http://en.dbpedia.org/resource/Vienna>
  <http://en.dbpedia.org/property/population> "1765649"
<http://en.dbpedia.org/resource/Vienna>
  wb:hasRevision :Vienna/revision/id/607920704 .
```

and as triples first with original URIs as follows:

```
<http://en.dbpedia.org/resource/Vienna>
  wb:hasRevision :Vienna/revision/id/607920704 .
<http://en.dbpedia.org/resource/Vienna>
  <http://en.dbpedia.org/property/population> "1765649" .
```

and using the custom URIs in the triples as:

```
:Vienna/id/607920704
  wb:hasRevision :Vienna/revision/id/607920704 .
:Vienna/id/607920704
  <http://en.dbpedia.org/property/population> "1765649"  .
```

3.3 UI & API

The DWM can be publicly accessed by the User Interface (UI) at our website[9] or querying our RESTful API so that third-party applications can be built on top of our proposal. In addition, our URIs are fully dereferenceable and comply with Linked Open Data principles. Figure 2 shows our DWM UI which is similar to typical search UIs. To travel back in time, a user provides the DBpedia resource and the timestamp. Then, it offers the possibility (a) to run the DWM, i.e. to get the DBpedia extraction of that resource at the given timestamp or (b) to retrieve the revision metadata history of the resource from that timestamp.[10] Note that this latter includes information and a link to each revision (as explained in Section 3.2), hence it is interesting for automatically inspecting each revision.

[7] http://www.gnu.org/software/diffutils/
[8] taken from http://wiki.dbpedia.org/Downloads
[9] http://data.wu.ac.at/wayback/
[10] by default 100 revisions from the given timestamp

Operation	Description
[GET] http://data.wu.ac.at/wayback/dbpedia	Base URI of the RESTful API
(A) **/{title}/timestamp/{date}**	Get DBpedia extraction of {title} at timestamp {date}
(B) **/{title}/id/{revID}**	Get DBpedia extraction of {title} at Wikipedia revision {revID}
(C) **/{title}/revision/timestamp/{date}**	Get revision metadata of DBpedia {title} at timestamp {date}
(D) **/{title}/revision/id/{revID}**	Get revision metadata of DBpedia {title} at Wikipedia revision {revID}
(E) **/{title}/diff/timestamp/{date_1}/{date_2}**	Get the number of different triples between the given dates
(F) **/{title}/diff/id/{revID_1}/{revID_2}**	Get the number of different triples between the given Wikipedia revision IDs
?format={format} (or HTTP content negotiation)	Get content in RDF {format} (N-Quads, N-Triples, Turtle, Trig, JSON, XML)
?limit={limit}	Set the upper bound on the number of revisions returned (default is 100) for op. (C).
?originalURIs=true|false	[true] Keep subject URIs of the DBpedia resources (default), or [false] introduce our dereferenceble URIs scheme ({title}/id/{revID})

Table 1: RESTful API of the DBpedia Wayback Machine.

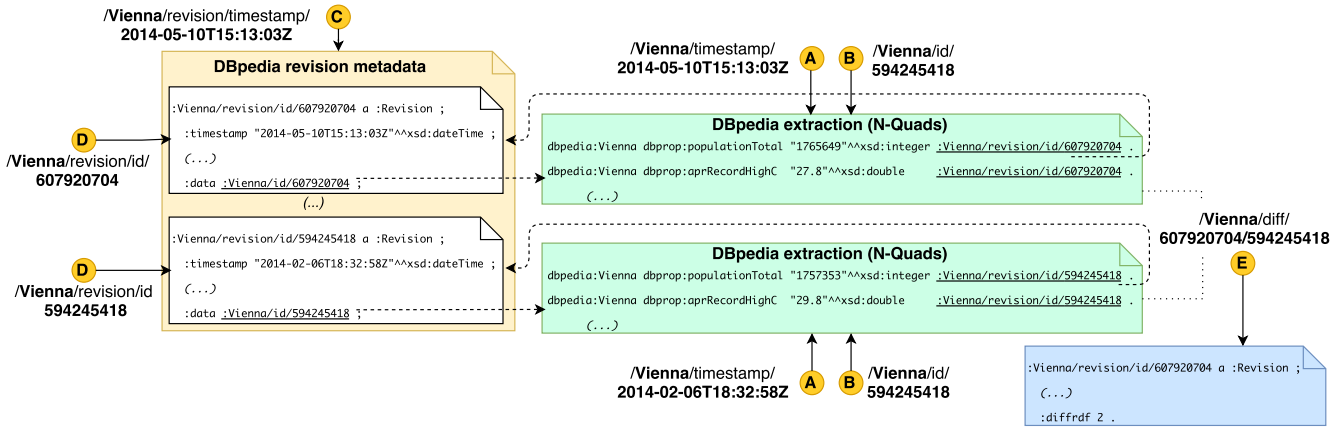

Figure 3: RESTful API examples and document organization of the DBpedia Wayback Machine.

The DWM RESTful API is described in Table 1, whereas Figure 3 depicts all operations in a real example where different revisions and extractions for *Vienna* are requested. As can be seen, the RESTful API serves basic, atomic operations, embracing a light-API/Smart-Client principle emerging in the semantic community recently. As stated, we distinguish three main operations: (i) get the DBpedia extraction (i.e. the RDF version) of a given resource, where the timestamp (operation (A)) or the revision ID (operation (B)) have to be provided, (ii) get revision metadata, providing the appropriate resource and a timestamp (operation (C)) or Wikipedia revision ID (operation (D)), and (iii) calculate the number of different triples between two revisions (operation (E) and (F)). Last rows of Table 1 show the parameters of the RESTful API to set the format of the response (note that this can be provided through HTTP content negotiation in accordance with the Linked Open Data philosophy), and the scheme of the DBpedia subjects (keep the original by default, or replace with our dereferenceable scheme).

4. CONCLUSIONS AND FUTURE WORK

In this paper we present the DBpedia Wayback Machine, which extends the DBpedia services with a wayback mechanism to get the version of a resource at any random time. We also model and offer the metadata of revisions and a diff comparison between version, all within the Linked Open Data paradigm and publicly available in a RESTful API.

Several research and practical challenges remain open. We first plan to enrich our existent Linked Open Data access and implement the memento protocol (RFC 7089), similar to [6], to provide content negotiation in the timestamp dimension. Then, we also want to allow different versions of the DBpedia ontology, community mappings and extractors. Regarding the framework features, we aim to include other extractors (e.g., categories) or further properties such as the list of changed DBpedia properties between versions. Finally, we are interested in extending this on-demand wayback mechanism to other dynamic sources beyond DBpedia.

5. REFERENCES

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *Proc. of ISWC*, 2007.

[2] S. Bischof, C. Martin, A. Polleres, and P. Schneider. Open City Data Pipeline - Collecting, Integrating, and Predicting Open City Data. In *Proc. of Know@LOD*, 2015.

[3] D. Brickley, R. Guha, and (eds.). RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, W3C, 2004.

[4] J. D. Fernández, A. Polleres, and J. Umbrich. Towards Efficient Archiving of Dynamic Linked Open Data. In *Proc. of DIACHRON*, 2015.

[5] E. W. Myers. An O(ND) Difference Algorithm and Its Variations. *Algorithmica*, 1(2):251–266, 1986.

[6] H. Van de Sompel, R. Sanderson, M. L. Nelson, L. L. Balakireva, H. Shankar, and S. Ainsworth. An HTTP-based versioning mechanism for linked data. In *Proc. of LDOW*, 2010.