Where is my URI?

Andre Valdestilhas[1], Tommaso Soru[1], Markus Nentwig[2], Edgard Marx[3],
Muhammad Saleem[1], and Axel-Cyrille Ngonga Ngomo[4]

[1] AKSW Group, University of Leipzig, Germany
{valdestilhas,tsoru,saleem}@informatik.uni-leipzig.de
[2] Database Group, University of Leipzig, Germany
nentwig@informatik.uni-leipzig.de
[3] Leipzig University of Applied Sciences, Germany
edgard.marx@htwk-leipzig.de
[4] Data Science Group, Paderborn University, Germany
axel.ngonga@upb.de

Abstract. One of the Semantic Web foundations is the possibility to dereference URIs to let applications negotiate their semantic content. However, this exploitation is often infeasible as the availability of such information depends on the reliability of networks, services, and human factors. Moreover, it has been shown that around 90% of the information published as Linked Open Data is available as data dumps and more than 60% of endpoints are offline. To this end, we propose a Web service called *Where is my URI?*. Our service aims at indexing URIs and their use in order to let Linked Data consumers find the respective RDF data source, in case such information cannot be retrieved from the URI alone. We rank the corresponding datasets by following the rationale upon which a dataset contributes to the definition of a URI proportionally to the number of literals. We finally describe potential use-cases of applications that can immediately benefit from our simple yet useful service.

Keywords: Link Discovery; Linked Data; Endpoints; URI; Dereferencing

1 Introduction

In the Web of Data, applications such as Link Discovery or Data Integration frameworks need to know where a specific URI is located. However, due to decentralized architecture of the Web of data, reliability and availability of Linked Data services, locating such URIs is not a trivial task. Locating the URIs from a well-known data dump might be easy. For example, it is trivial to know that the URI `http://dbpedia.org/resource/Leipzig` belongs to the DBpedia dataset. However, locating the dataset where the URI `http://citeseer.rkbexplorer.com/id/resource-CS116606` was first defined is a time-consuming task. Consequently, this can greatly affect the scalable and time-efficient deployment of many Semantic Web applications such as link discovery, Linked Data enrichment, and federated query processing. On the other hand, such provenance information about URIs can lead to regenerate and validate the links across datasets.

The availability of the current available services to provide such information is unfortunately one of the key issues in Semantic Web and Linked Data. It has been shown that around 90% of the information published as Linked Open Data is available as data dumps only and more than 60% of endpoints are offline [14]. The availability problem is mostly due to cost associated with storing and providing querying services.

To this end, we propose *Where is my URI?* (WIMU), a low-cost Semantic Web service to determine the RDF data source of URIs along with their use. We also rank the data sources in case a single URI is provided by multiple data sources. The ranking is based-on a scoring function. Currently, our service processed more than 58 billion unique triples from more than 660,000 datasets obtained from LODStats [1] and LOD Laundromat [2]. For each URI, our service provides the corresponding datasets and the number of literals in the datasets having this URI. The service is both available from a web interface as well as can be queried from a client application using the standard HTTP protocol. We believe our service can be used in multiple Linked Data related problems such as devising fast link discovery frameworks, efficient source selection, and distributed query processing.

Our main contributions are as follows:

- We provide a regularly updated[5] database index of more than 660K datasets from LODStats and LOD Laundromat.
- We provide an efficient, low cost and scalable service on the web that shows which dataset most likely defines a URI.
- We provide various statistics of datasets indexed from LODStats and LOD Laundromat.

The service is available from `https://w3id.org/where-is-my-uri/` under GNU Affero public license 3.0 and the source code is available online[6].

The rest of the paper is organized as follows: We first provide a brief overview of the state-of-the-art. We then discuss the proposed approach in detail, including the index creation, the web interface, and the data processing. We finally present our evaluation results and conclude.

2 Related Work

The work presented in [4] shows how to set up a Linked Data repository called *DataTank* and publish data as turtle files or through a SPARQL endpoint. The difference with WIMU is that we provide a RESTful service instead of a setup to configure a Linked Data repository.

The work in [7] is based on an approach developed for the 3*store* RDF triple store and describes a technique for indexing RDF documents allowing the rank and retrieval of their contents. Their index contained 10^7 triples, which was

[5] Updated monthly due to the huge size of data processed.

[6] `https://github.com/dice-group/wimu`

remarkable for the early years of the Semantic Web. Moreover, their system is not available for tests anymore. A similar point here is that the authors claim that for a given URI from an RDF document, the system will retrieve the URLs of documents containing that URI.

In the approach called LOD-A-LOT [5] which is a queryable dump file of the LOD Cloud[7], there are some differences with WIMU. The first, it is not possible to know the provenance of the URI in order to know which dataset the URI was defined. They provide a huge dump file[8] containing all the data from LOD Laundromat[9]. LOD Laundromat itself provides an endpoint to an inverted index of their data[10]. However, finding the original document a URI was defined in is not trivial, as the returned metadata only describe the datasets themselves [2]. Moreover, as the primary aim of LOD Laundromat is to "clean" Linked Data, most dumps are possibly not continuously monitored, once cleaned.

Comparing with all the approaches above, the main advantage of WIMU is that the datasets a URI likely belongs to are ranked using a score. Our index has also a larger coverage, as it includes data from the two largest Linked Data hubs, i.e., LODStats [1] and LOD Laundromat [2], and the most updated SPARQL endpoints. Finally, WIMU is able to process RDF files containing more than one URI at the same time[11].

3 The approach

WIMU uses the number of literals as a heuristic to identify the dataset which most likely defines a URI. The intuition behind this can be explained in two points: (1) Literal values are the raw data that can disambiguate a URI node in the most direct way and (2) The Semantic Web architecture expects that datasets reusing a URI only refer to it without defining more literal values. One more reason for point (1) is that: it is straightforward to understand whether two literal values are different, whereas disambiguating URIs usually requires more effort.

We store the collected data in an Apache Lucene[12] index. Due to runtime performance and complexity reasons, we found that storing the information into Lucene was more convenient than a traditional triple store such as Virtuoso[13]. The rationale behind this choice is that a tuple such as *(URI, Dataset, Score)* would be expressed using at least three triples; for instance:

```
    :R001 :hasURI :URI001
```

[7] http://lod-cloud.net/

[8] A HDT file with more than 500GB which requires more than 16 GB RAM to process.

[9] http://lodlaundromat.org/

[10] http://index.lodlaundromat.org/

[11] For example: https://w3id.org/where-is-my-uri/Find?link=http://www.linklion.org/download/mapping/citeseer.rkbexplorer.com---ibm.rkbexplorer.com.nt.

[12] https://lucene.apache.org/

[13] https://virtuoso.openlinksw.com/

```
:R001 :hasDataset :Dataset001
:R001 :hasScore "20"^^http://www.w3.org/2001/XMLSchema#Integer
```

where `:R001` is an index record URI. Therefore, materializing all records would have substantially increased the space complexity of our index.

3.1 The index creation

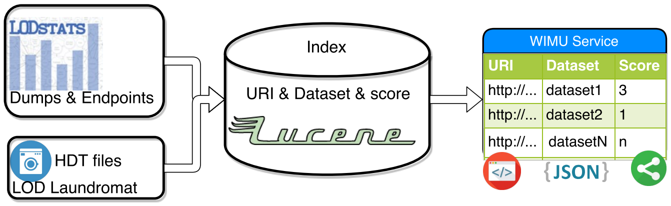


Fig. 1. Creation workflow of the WIMU index.

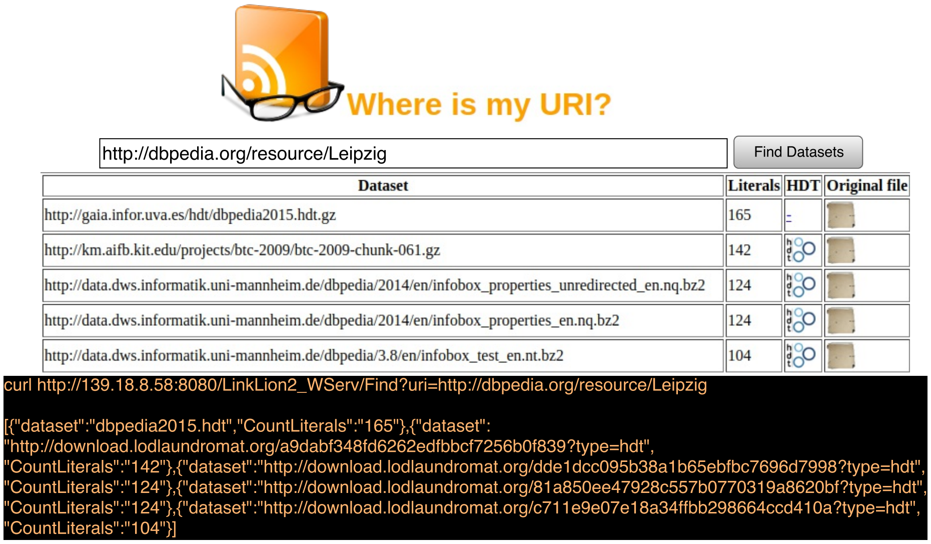
The index creation, the core of our work, is shown in Fig. 1 and consists in the following four steps:

1. Retrieve list of datasets from sources (i.e., LOD Stats and LOD Laundromat).
2. Retrieve data from datasets (i.e., dump files, endpoints, and HDT files).
3. Build three indexes from dump files, endpoints, and HDT files.
4. Merge the indexes into one.
5. Make the index available and browsable via a web application and an API service.

For each processed dataset, we keep its URI as provenance. After we have downloaded and extracted a dump file, we process it by counting the literals as objects for each subject. For endpoints and HDT files, we use a SPARQL query:

```
SELECT ?s (count(?o) as ?c) WHERE {
   ?s [] ?o . FILTER(isliteral(?o))
} GROUP BY ?s
```

We process the data in parallel, distributing the datasets among the CPUs. If a dataset is too large for a CPU, we split it into smaller chunks. To preserve space, dump files are deleted after being processed. The index was generated using an Intel Xeon Core i7 processor with 64 cores, 128 GB RAM on an Ubuntu 14.04.5 LTS with Java SE Development Kit 8.



Dataset	Literals	HDT	Original file
http://gaia.infor.uva.es/hdt/dbpedia2015.hdt.gz	165	-	
http://km.aifb.kit.edu/projects/btc-2009/btc-2009-chunk-061.gz	142	hdt	
http://data.dws.informatik.uni-mannheim.de/dbpedia/2014/en/infobox_properties_unredirected_en.nq.bz2	124	hdt	
http://data.dws.informatik.uni-mannheim.de/dbpedia/2014/en/infobox_properties_en.nq.bz2	124	hdt	
http://data.dws.informatik.uni-mannheim.de/dbpedia/3.8/en/infobox_test_en.nt.bz2	104	hdt	

```
curl http://139.18.8.58:8080/LinkLion2_WServ/Find?uri=http://dbpedia.org/resource/Leipzig

[{"dataset":"dbpedia2015.hdt","CountLiterals":"165"},{"dataset":
"http://download.lodlaundromat.org/a9dabf348fd6262edfbbcf7256b0f839?type=hdt",
"CountLiterals":"142"},{"dataset":"http://download.lodlaundromat.org/dde1dcc095b38a1b65ebfbc7696d7998?type=hdt",
"CountLiterals":"124"},{"dataset":"http://download.lodlaundromat.org/81a850ee47928c557b0770319a8620bf?type=hdt",
"CountLiterals":"124"},{"dataset":"http://download.lodlaundromat.org/c711e9e07e18a34ffbb298664ccd410a?type=hdt",
"CountLiterals":"104"}]
```

Fig. 2. Web interface.

3.2 The web interface and the API service

In order to simplify the access to our service, we create a web interface where it is possible to visualize all the data from the service, as Fig. 2 shows.

The web interface allows the user to query a URI and see the results in a HTML web browser; the API service allows the user to work with an output in JSON format. In Fig. 2, we can see an example of usage of the service, where WIMU is requested for the dataset in which the URI `dbpedia:Leipzig` was defined. Fig. 4 shows the generic usage of WIMU.

4 Use cases

In this section, we present three use-cases to show that our hypothesis works on the proposed tasks.

4.1 Data Quality in Link Repositories

The first use-case is about quality assurance in a link repository by re-applying link discovery algorithms on the stored links. This task concerns important steps of the Linked Data Lifecycle, in particular *Data Interlinking* and *Quality*. Link repositories contain sets of links that connect resources belonging to different datasets. Unfortunately, the subject and the object URIs of a link often do not have metadata available, hence their Concise Bounded Descriptions (CBDs) are hard to obtain. In Section 4.1, $(D_1, ..., D_n|x) : D_n$ represent the datasets and x is the quantity of literals. The **input** for our service in this use-case is S; the **output** is $\{(D_1, 3), (D_2, 1), (D_3, 2)\}$, where $D1$ most likely defines S due to the highest number of literal. In the same way, the dataset that most likely defines T is D_4 with 7 literals. Once we have this information, the entire CBD of the two resources S and T can be extracted and a Link Discovery algorithm can check whether the `owl:sameAs` link among them should subsist.

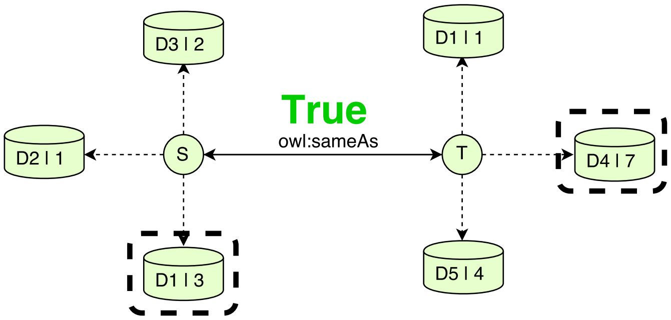


Fig. 3. First use-case.

4.2 Finding class axioms for Link Discovery

A class axiom is needed by the link discovery algorithm to reduce the number of comparisons. Here, the aim is to find two class axioms for each mapping in the link repository.

To this end, we use real data including a mapping[14] from the LinkLion repository [9] between `http://citeseer.rkbexplorer.com/id/resource-CS65161` (S) and `http://citeseer.rkbexplorer.com/id/resource-CS65161` (T). Our service shows that S was defined in four datasets, whereas the dataset with more literals was `http://km.aifb.kit.edu/projects/btc-2009/btc-2009-chunk-039.gz`[15]. Thus, we can deduce where the URI S was most likely defined. Knowing the datasets allows us to extract the axioms of the classes our URIs belong to. The techniques to decrease the complexity vary from choosing the most specific class to using an ontology learning tool such as DL-Learner [8].

4.3 Federated Query Processing

Federated queries, which aim to collect information from more than one datasets is of central importance for many semantic web and linked data applications [12,11].

[14] `http://www.linklion.org/download/mapping/citeseer.rkbexplorer.com---ibm.rkbexplorer.com.nt`

[15] Also available in HDT file from `http://download.lodlaundromat.org/15b06d92ae660ffdcff9690c3d6f5185?type=hdt`

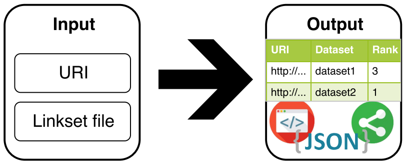


Fig. 4. Usage.

One of the key step in federated query processing is the *source selection*. The goal of the source selection is to find relevant sources (i.e., datasets) for the given user query. In the next step, the federated query processing engine makes use of the source selection information to generate an optimized query execution plan. WIMU can be used by the federated SPARQL engines to find the relevant sources against the individual triple patterns of the given SPARQL query. In particular, our service can be helpful during the source selection and query planning in cost-based SPARQL federation engines such SPLENDID [6], SemaGrow [3], HiBISCuS [13], CostFed [10], etc.

5 Statistics about the Datasets

To the best of our knowledge, LODStats[1] is the only project oriented to monitoring dump files; however, its last update dates back to 2016. Observing Table 1, we are able to say that from LODStats, not all datasets are ready to use. Especially, more than 58% are off-line, 14% are empty datasets, 8% of the triples that have literals as objects are blank nodes and 35% of the online datasets present some error using the Apache Jena parser[16]. A large part of those data was processed and cleaned by LOD Laundromat [2].

Table 1. Datasets.

	LOD Laundromat	LODStats	Total
URIs indexed	4,185,133,445	31,121,342	4,216,254,787
Datasets checked	658,206	9,960	668,166
Triples processed	19,891,702,202	38,606,408,854	58,498,111,056

The algorithm took three days and seven hours to complete the task. Thus, we will create a scheduled job to update our database index once a month. With respect to the information present in the Fig. 5, we can observe that the majority of files from LODStats are in RDF/XML format. Moreover, the endpoints are represented in greater numbers (78.6%), the dominant file format is RDF with 84.1% of the cases, and 56.2% of errors occurred because Apache Jena was not

[16] See https://github.com/dice-group/wimu/blob/master/ErrorJenaParser.tsv.

able to perform SPARQL queries. Among the HDT files from LOD Laundromat, 2.3% of them could not be processed due to parsing errors. Another relevant point is that 99.2% of the URIs indexed with WIMU come from LOD Laundromat, due to 69.8% of datasets from LODstats contain parser errors in which WIMU was not able to process the data.

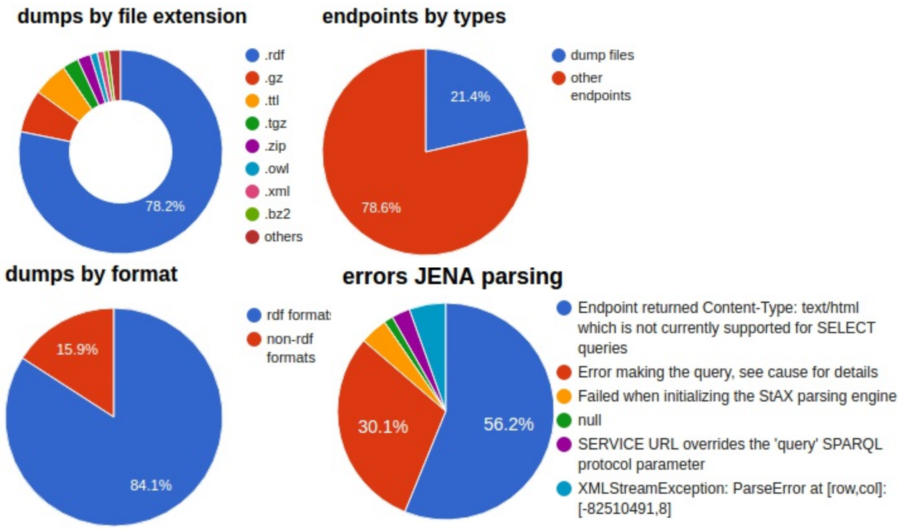


Fig. 5. Dump files and Apache Jena parsing error.

Finally, we validated our heuristic assessing if the URI really belongs to the dataset with more literals. To this end, we took a sample of 100 URIs[17] that belong to at least two datasets, where we assess manually the data in order to check if the results are really correct. As a result, the dataset containing the correct information was found as first result in 90% of the URIs and among the top three in 95% of the URIs.

6 Conclusion and Future work

We provide a database index of URIs and their respective datasets built upon large Linked Data hubs such as LODStats and LOD Laundromat. In order to make this data available and easy to use, we developed a semantic web service. For a given URI, it is possible to know the dataset the URI likely was defined in using a heuristic based on the number of literals. We showed two use-cases and carried out a preliminary evaluation to facilitate the understanding of our work. As future work, we will integrate the service into the version 2.0 of the LinkLion

[17] https://github.com/dice-group/wimu/blob/master/result100.csv

repository, so as to perform linkset quality assurance with the application of link discovery algorithms on the stored links.

References

1. S. Auer, J. Demter, M. Martin, and J. Lehmann. Lodstats–an extensible framework for high-performance dataset analytics. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 353–362. Springer, 2012.
2. W. Beek, L. Rietveld, H. R. Bazoobandi, J. Wielemaker, and S. Schlobach. Lod laundromat: a uniform way of publishing other people's dirty data. In *International Semantic Web Conference*, pages 213–228. Springer, 2014.
3. A. Charalambidis, A. Troumpoukis, and S. Konstantopoulos. Semagrow: Optimizing federated sparql queries. In *Proceedings of the 11th International Conference on Semantic Systems*, SEMANTICS '15, pages 121–128, New York, NY, USA, 2015. ACM.
4. P. Colpaert, R. Verborgh, E. Mannens, and R. Van de Walle. Painless uri dereferencing using the datatank. In *ESWC*, 2014.
5. J. D. Fernández, W. Beek, M. A. Martínez-Prieto, and M. Arias. Lod-a-lot: A queryable dump of the lod cloud. 2017.
6. O. Görlitz and S. Staab. SPLENDID: SPARQL Endpoint Federation Exploiting VoID Descriptions. In *O. Hartig, A. Harth, and J. F. Sequeda, editors, 2nd International Workshop on Consuming Linked Data (COLD 2011) in CEUR Workshop Proceedings*, volume 782, October 2011.
7. S. Harris, N. Gibbins, and T. R. Payne. Semindex: Preliminary results from semantic web indexing. 2004.
8. J. Lehmann. Dl-learner: learning concepts in description logics. *Journal of Machine Learning Research*, 10(Nov):2639–2642, 2009.
9. M. Nentwig, T. Soru, A.-C. N. Ngomo, and E. Rahm. LinkLion: A Link Repository for the Web of Data. In *The Semantic Web: ESWC 2014 Satellite Events - ESWC 2014 Satellite Events*, pages 439–443, 2014.
10. A. Potocki, M. Saleem, T. Soru, O. Hartig, M. Voigt, and A.-C. N. Ngomo. Federated sparql query processing via costfed. 2017.
11. M. Saleem, M. R. Kamdar, A. Iqbal, S. Sampath, H. F. Deus, and A.-C. N. Ngomo. Big linked cancer data: Integrating linked tcga and pubmed. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 27–28:34 – 41, 2014. Semantic Web Challenge 2013.
12. M. Saleem, M. R. Kamdar, A. Iqbal, S. Sampath, H. F. Deus, and A.-C. Ngonga. Fostering serendipity through big linked data. In *Semantic Web Challenge at ISWC*, 2013.
13. M. Saleem and A.-C. Ngonga Ngomo. HiBISCuS: Hypergraph-Based Source Selection for SPARQL Endpoint Federation. In V. Presutti, C. d'Amato, F. Gandon, M. d'Aquin, S. Staab, and A. Tordai, editors, *The Semantic Web: Trends and Challenges*, volume 8465 of *Lecture Notes in Computer Science*, pages 176–191. Springer International Publishing, 2014.
14. P.-Y. Vandenbussche, J. Umbrich, L. Matteis, A. Hogan, and C. Buil-Aranda. Sparqles: Monitoring public sparql endpoints. *Semantic Web*, 8(6):1049–1065, 2017.