

# IDENTIFYING, RELATING AND QUERYING LARGE HETEROGENEOUS RDF SOURCES

Der Fakultät für Mathematik und Informatik  
der Universität Leipzig  
eingereichte

## DISSERTATION

zur Erlangung des akademischen Grades

Doctor rerum naturalium  
(Dr. rer. nat.)

im Fachgebiet Informatik vorgelegt von

**Andre Valdestilhas, Diplom-Informatiker**

geboren am 14.02.1980 in in Sao Bernardo do Campo - Sao  
Paulo, Brazil

Leipzig, den 9.6.2020

## BIBLIOGRAPHIC DATA

AUTHOR:

Andre Valdestilhas

TITLE:

*Identifying, Relating and Querying Large Heterogeneous RDF Sources*

STATISTICAL INFORMATION:

110 pages, 33 figures, 13 tables

INSTITUTION:

Institut für Informatik  
Fakultät für Mathematik und Informatik  
Universität Leipzig

SUPERVISORS:

Prof. Dr.-Ing. habil. Klaus-Peter Fahnrich  
Prof. Dr. rer. nat. Thomas Riechert  
Dr. rer. nat. Muhammad Saleem

TIME FRAME:

November 2014 - March 2020

## ABSTRACT

---

The concept of Linked Data relates to a collection of best practices to publish and connect structured web-based data. However, the number of available datasets has been growing significantly over the last decades. Those datasets now represent the well known as Web of Data, which represent an extensive collection of concise and detailed interlinked data sets from multiple domains with large datasets. Thus, linking entries across heterogeneous data sources such as databases or knowledge bases becomes an increasing challenge. However, connections between datasets play a leading role in significant activities such as cross-ontology question answering, large-scale inferences, and data integration. In Linked Data, the Linksets are well known for executing the task of generating links between datasets. Due to the heterogeneity of the datasets, this uniqueness is reflected in the structure of the dataset, making a hard task to find relations among those datasets, i.e., to identify how similar they are. In this way, we can say that Linked Data involves Datasets and Linksets and those Linksets needs to be maintained.

All those lacks of information directed us to the current drawbacks addressed on this thesis are: Identifying and querying datasets from a huge heterogeneous collection of RDF datasets, to execute this task we need to assure the consistency and to know how the datasets are related and how similar they are.

As results, to deal with the need for identifying LOD Datasets, we created an approach called WIMU, which is a regularly updated database index of more than 660K datasets from LODStats and LOD Laundromat, an efficient, low cost and scalable service on the web that shows which dataset most likely defines a URI and various statistics of datasets indexed from LODStats and LOD Laundromat. To integrate and to query LOD datasets, we provide a hybrid SPARQL query processing engine that can retrieve results from 559 active SPARQL endpoints (with a total of 163.23 billion triples) and 668,166 datasets (with a total of 58.49 billion triples) from LOD Stats and LOD Laundromat. To assure consistency of semantic web Linked repositories where these LOD datasets are located we create an approach for the mitigation of the identifier heterogeneity problem and implement a prototype where the user can evaluate existing links, as well as suggest new links to be rated and a time-efficient algorithm for the detection of erroneous links in large-scale link repositories without computing all closures required by the property axiom. To know how the datasets are related and how similar they are we provide a String similarity algorithm called Most Frequent K Characters, in which is

based in two nested filters, (1) First Frequency Filter and (2) Hash Intersection filter, that allows discarding candidates before calculating the actual similarity value, thus giving a considerable performance gain, allowing to build a LOD Dataset Relation Index, in which provides information about how similar are all the datasets from LOD cloud, including statistics about the current state of those datasets.

The work in this thesis showed that to identify and query LOD datasets, there is a need to be related and to know how those datasets are related, assuring consistency. Our analysis demonstrated that most of the datasets are disconnected from others needing to pass through a consistency and linking process to integrate them, providing a way to query a large number of datasets simultaneously. There is a considerable step towards totally queryable LOD datasets, where the information contained in this thesis is an essential step towards Identifying, Relating, and Querying datasets on the Web of Data.

## PUBLICATIONS

---

This thesis is based on the following publications and proceedings.

### AWARDS AND NOTABLE MENTIONS

- **Where is my URI?** (Honorable mentioned) [114]
- **Triple scoring using a hybrid fact validation approach** (WSDM cup - 3th place) [64]

### CONFERENCES, PEER-REVIEWED

- **DBpediaSameAs: An approach to tackle heterogeneity in DBpedia identifiers.** [113]
- **A High-Performance Approach to String Similarity using Most Frequent K Characters** [116]
- **CEDAL: time-efficient detection of erroneous links in large-scale link repositories** [115]
- **Where is my URI?** [114]
- **More complete resultset retrieval from large RDF datasets** [117]

### AS CO-AUTHOR

- **SPARQL as a Foreign Language** [104]
- **WASOTA What Are The State Of The Art for?** [71]
- **Generating a large dataset for neural question answering over the dbpedia knowledge base** [Hartmann]
- **Neural Machine Translation for Query Construction and Composition** [105]
- **Semantic search user interface patterns: an introduction** [62]
- **Where is Linked Data in Question Answering over Linked Data?** [106]

### PROJECTS BASED ON THIS THESIS

- **ReLOD project.** Participating on the *Next Generation Internet initiative*<sup>1</sup>

---

<sup>1</sup> Next Generation Internet initiative. <https://nlnet.nl/propose/>

## UNDER-REVIEW

- Identifying, Querying, and Relating Large Heterogeneous RDF Sources. Submitted to Semantic Web Journal on March 31, 2020[8].

## ACKNOWLEDGMENTS

---

To my advisor Thomas Riechert for accept me as his PhD student, my friend Edgard Marx for all work and moments that we have done. My friendji Tommaso Soru, all the support, and much more, Diego Mousallem, always helping and being a good friend. Muhammad Saleem, ahhh my chechechan friend. Prof. Dr. Ing. habil. Klaus-Peter Fährnich, ahh good moments. I'll finish it later.

## CONTENTS

1	INTRODUCTION	1
1.1	The need for identifying and querying LOD datasets .	1
1.2	The need for consistency of semantic web Linked repositories . . . . .	2
1.3	The need for Relation and integration of LOD datasets	2
1.4	Methodology and Contributions . . . . .	3
1.5	Chapter overview . . . . .	5
2	BASIC CONCEPTS AND NOTATION	7
3	STATE OF THE ART	9
3.1	Identifying Datasets in Large Heterogeneous RDF Sources	9
3.2	Relating Large amount of RDF datasets . . . . .	10
3.3	Obtaining Similar Resources using String Similarity . .	10
3.4	Heterogeneity in DBpedia Identifiers . . . . .	11
3.4.1	A related problem with sameAs.org . . . . .	12
3.5	Detection of Erroneous Links in Large-Scale RDF Datasets	12
3.6	Querying Large Heterogeneous RDF Datasets . . . . .	14
4	IDENTIFYING DATASETS IN LARGE HETEROGENEOUS RDF SOURCES	17
4.1	Introduction . . . . .	17
4.2	The approach . . . . .	18
4.2.1	The index creation . . . . .	19
4.2.2	The web interface and the API service . . . . .	20
4.3	Use cases . . . . .	20
4.3.1	Data Quality in Link Repositories . . . . .	20
4.3.2	Finding class axioms for Link Discovery . . . . .	21
4.3.3	Federated Query Processing . . . . .	22
4.3.4	Usage examples . . . . .	22
4.4	Statistics about the Datasets . . . . .	23
5	RELATING LARGE AMOUNT OF RDF SOURCES	25
5.1	Introduction . . . . .	25
5.2	The approach . . . . .	27
5.2.1	The index creation . . . . .	27
5.2.2	Identifying duplicated and chunk datasets . . .	29
5.2.3	The file structure . . . . .	31
5.2.4	Querying the index . . . . .	32
5.3	Evaluation . . . . .	33
6	OBTAINING SIMILAR RESOURCES USING STRING SIMILARITY	39
6.1	Introduction . . . . .	39
6.2	Approach . . . . .	40
6.2.1	Improving the Runtime . . . . .	42
6.3	Correctness and Completeness . . . . .	44



6.4	Evaluation . . . . .	46
6.4.1	Parallel implementation . . . . .	47
6.4.2	Runtime Evaluation . . . . .	47
6.4.3	Scalability Evaluation . . . . .	48
6.4.4	Comparison with existing approaches . . . . .	48
7	HETEROGENEITY IN DBPEDIA IDENTIFIERS . . . . .	51
7.1	Introduction . . . . .	51
7.2	Representation of the idea . . . . .	52
7.2.1	The work-flow . . . . .	52
7.2.2	Methodology . . . . .	53
7.3	Evaluation . . . . .	54
7.3.1	Normalization on DBpedia URIs . . . . .	54
7.3.2	Rate the links . . . . .	55
7.3.3	Results . . . . .	55
7.3.4	Discussion . . . . .	56
8	DETECTION OF ERRONEOUS LINKS IN LARGE-SCALE RDF DATASETS . . . . .	57
8.1	Introduction . . . . .	57
8.2	Method . . . . .	59
8.2.1	Error Detection algorithm . . . . .	59
8.3	Error Types and Quality Measure for Linkset Repositories . . . . .	62
8.3.1	Error Types . . . . .	62
8.3.2	Quality Measure . . . . .	63
8.4	Evaluation . . . . .	64
8.4.1	Experimental setup . . . . .	64
8.4.2	Ranking the erroneous candidates . . . . .	65
8.4.3	Runtime experiments . . . . .	66
8.4.4	Consistency by provenance of links . . . . .	67
8.4.5	Comparison with other works . . . . .	68
9	QUERYING LARGE HETEROGENEOUS RDF DATASETS . . . . .	69
9.1	Introduction . . . . .	69
9.2	Definitions . . . . .	71
9.3	The WimuQ . . . . .	71
9.3.1	WimuQ source selection . . . . .	72
9.3.2	WimuQ query execution . . . . .	73
9.3.3	Practical example . . . . .	75
9.4	Evaluation . . . . .	76
9.4.1	Experimental setup . . . . .	76
9.4.2	Results . . . . .	78
10	CONCLUSION . . . . .	81
10.1	Identifying Datasets in Large Heterogeneous RDF Sources . . . . .	81
10.2	Relating Large Amount of RDF Datasets . . . . .	81
10.3	Obtaining Similar Resources Using String Similarity . . . . .	82
10.4	Heterogeneity in DBpedia Identifiers . . . . .	82
10.5	Detection of Erroneous Links in Large-Scale RDF Datasets . . . . .	83
10.6	Querying Large Heterogeneous RDF Datasets . . . . .	83

A	APPENDIX	85
A.1	Detecting Erroneous Link candidates in Educational Link Repositories . . . . .	85
A.1.1	Introduction . . . . .	85
A.1.2	Related works . . . . .	86
A.1.3	Preliminaries . . . . .	87
A.1.4	The Consistency Error Detection Algorithm . .	88
A.1.5	Evaluation . . . . .	89
A.1.6	Conclusion and future work . . . . .	91
B	CV	93
B.1	Abstract . . . . .	93
	SELBSTÄNDIGKEITSERKLÄRUNG	110

## LIST OF FIGURES

Figure 1	Life-cycle of contributions. . . . .	3
Figure 2	Several URIs with the property <code>owl:sameAs</code> from the web site sameAs.org. . . . .	12
Figure 3	Creation workflow of the WIMU index. . . . .	19
Figure 4	Web interface. . . . .	20
Figure 5	First use-case. . . . .	21
Figure 6	Usage. . . . .	22
Figure 7	Dump files and Apache Jena parsing error. . . . .	24
Figure 8	Creating the index. . . . .	27
Figure 9	Querying the index. . . . .	28
Figure 10	String similarity process. . . . .	29
Figure 11	Dataset matching. . . . .	35
Figure 12	Improvements of ReLOD. . . . .	37
Figure 13	Avoided pairs and recall. . . . .	47
Figure 14	Precision, Recall and F-Measure. . . . .	49
Figure 15	Run-time experiments results. . . . .	50
Figure 16	General work-flow. . . . .	53
Figure 17	Relation of the contributions. . . . .	54
Figure 18	Rate a link. Available at <a href="http://dbpsa.aksw.org/SameAsService">http://dbpsa.aksw.org/SameAsService</a> . . . . .	55
Figure 19	Transitive / Redirect links in DBpedia. . . . .	56
Figure 20	Manual detection of erroneous resource candidates. . . . .	58
Figure 21	Error detection. . . . .	60
Figure 22	Detected error types. . . . .	62
Figure 23	Error rank (legends: see table 10) and Runtime results according to the input size, CPU and GPU. . . . .	66
Figure 24	CEDAL CPU/GPU processing . . . . .	67
Figure 25	Pellet vs. ClosureGenerator vs. CEDAL . . . . .	68
Figure 26	Query processing workflow, already including <i>ReLOD</i> [8] . . . . .	72
Figure 27	Average number of results retrieved by the selected approaches across different queries categories of the selected benchmarks. . . . .	78
Figure 28	Average number of datasets discovered and queried by WimUQ across different queries categories of the selected benchmarks. . . . .	79
Figure 29	Average query runtimes of the selected approaches across different queries categories of the selected benchmarks. . . . .	80

Figure 30	Manual detection of erroneous resource candidates. . . . .	86
Figure 31	Detected error types. . . . .	88
Figure 32	Error detection. . . . .	89
Figure 33	Error rank, where Nature—Purl.nt represent the file with links between Nature and Purl.org and Nature—DBpedia.nt represent the links between Nature and DBpedia . . . . .	91

## LIST OF TABLES

Table 1	Parameters . . . . .	22
Table 2	Datasets. . . . .	23
Table 3	Sample datasets from [37] to evaluate our string similarity approach. . . . .	33
Table 4	Similarity table according to Jaccard method applied to a sample data, in which the level of similarity is also represented by the intensity of the color, as more intense color, as more similar are the datasets. . . . .	34
Table 5	A sample of how much a dataset is contained in each other, in which the level of containment is also represented by the intensity of the color, as more intense color, as more contained are the datasets. . . . .	34
Table 6	Top 10 datasets containing exact the same URI and containing the most similar URIs according to our similarity approach, in which #Prop-Class represents the total number of properties and classes from the dataset. . . . .	35
Table 7	Evaluation on the Match algorithm, where <b>DsProp-Match</b> refers to the number of properties/classes the datasets share among each other. . . . .	36
Table 8	Link types . . . . .	64
Table 9	Fictional example results. . . . .	65
Table 10	Legend for the figures 33 and 23(b) . . . . .	65
Table 11	Comparison of results with respect to the provenance of the links. . . . .	68
Table 12	Educational Datasets . . . . .	90
Table 13	Educational Linksets . . . . .	90

## INTRODUCTION

---

The Linked Data concept is well known as a collection of best practices to publish and connect structured web-based data. However, the number of available datasets has been growing significantly over the last decades [18]. Those datasets represent now the well known as Web of Data, which represent a large collection of concise and detailed interlinked data sets from multiple domains with large datasets [94]. Thus, linking entries across heterogeneous data sources such as databases or knowledge bases, becomes an increasingly difficult problem [116; 76; 84]. However, connections between datasets play a leading role in significant activities such as cross-ontology question answering [59], large-scale inferences [111] and data integration [80]. In Linked Data, the Linksets are well known for executing the task of generating links between datasets [76]. Due to the heterogeneity of the datasets, this uniqueness is reflected in the dataset's structure, making a challenging task to find relations among those datasets, i.e., to identify how similar they are. In this way, we can say that Linked Data involves Datasets and Linksets and those Linksets needs to be maintained. There are many ways to maintain Linksets; one of those is to create a semantic web link repository, i.e., LinkLion [69].

The current drawbacks addressed on this thesis are: Identifying and querying datasets from a huge heterogeneous collection of RDF datasets, in order to execute this task we need to know how the datasets are related and how similar they are, also the consistency of those datasets. After a brief explanation about what is involved in semantic web link repositories, we present and discuss the need for improvements on semantic web link repositories related to how the datasets are related and how similar they are, Identifying and querying those datasets, and assure the consistency of those repositories. Finally, a short introduction of each of the chapters is provided.

The following content shows the motivation where the research questions were originated and the related chapters.

### 1.1 THE NEED FOR IDENTIFYING AND QUERYING LOD DATASETS

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, there is a need for a service to **identify** which dataset a URI were defined in order to let Linked Data consumers find the respective RDF data source, in case such information cannot be retrieved from the URI alone.

In order to **query** such amount of LOD datasets various interfaces such as LOD Stats[13], LOD Laudromat[17], SPARQL endpoints provide access to the hundreded of thousands of RDF datasets, representing billions of facts. These datasets are available in different formats such as raw data dumps and HDT files or directly accessible via SPARQL endpoints. Querying such large amount of distributed data is particularly challenging and many of these datasets cannot be directly queried using the SPARQL query language. To deal with such problems we present WIMU[114] and wimuQ[117] covered on the respected chapters 4 and 9.

## 1.2 THE NEED FOR CONSISTENCY OF SEMANTIC WEB LINKED REPOSITORIES

More than 500 million facts on the Linked Data Web are statements across knowledge bases. The **consistency** of these links are of crucial importance for the Linked Data Web as they make a large number of tasks possible, including cross-ontology, question answering and federated queries. However, a large number of these links are erroneous and can thus lead to these applications producing absurd results. We present a time-efficient and complete approach for the detection of erroneous links for properties that are transitive[Valdestilhas et al.; 115], covered with more details on the chapters 7 and 8.

## 1.3 THE NEED FOR RELATION AND INTEGRATION OF LOD DATASETS

Linking and integrating entries across heterogeneous data sources such as databases or knowledge bases, becomes an increasingly difficult problem, in particular w.r.t. the runtime of these tasks. Consequently, it is of utmost importance to provide time-efficient approaches for similarity joins in the Web of Data. While a number of scalable approaches have been developed there is not approach able to deal with such amount of datasets. In the web of data, there are several similar datasets, but there is no place that stores or provide such kind of information about how similar are the datasets, in our case, which attributes the datasets share. To this aim we create a this index[118] to store the relation among them in order to see how similar they are based on their properties and classes<sup>1</sup>, besides statistics about a large number of datasets, in which are covered in more details at chapters 6 and 5.

---

<sup>1</sup> Classes are also know as concepts

## 1.4 METHODOLOGY AND CONTRIBUTIONS

The figure 1 highlights the contributions of this thesis, which addresses problems pertaining to the need for identifying and querying LOD datasets, Consistency of semantic web Linked repositories, how LOD datasets are related and how similar they are, in which can also be visualized in the Linked Data Life-cycle [14] (a simplification in three steps depicted in figure 1). In which the works DBpediaSameAs [Valdestilhas et al.] and CEDAL [115] are related to the **Classification, Evolution/Repair** and **Quality**. The works WIMU [114], wimuQ [117], MFKC [116] and ReLOD [118] are related to **Interlinking, Storage** and **Search/Browsing/Exploration**.

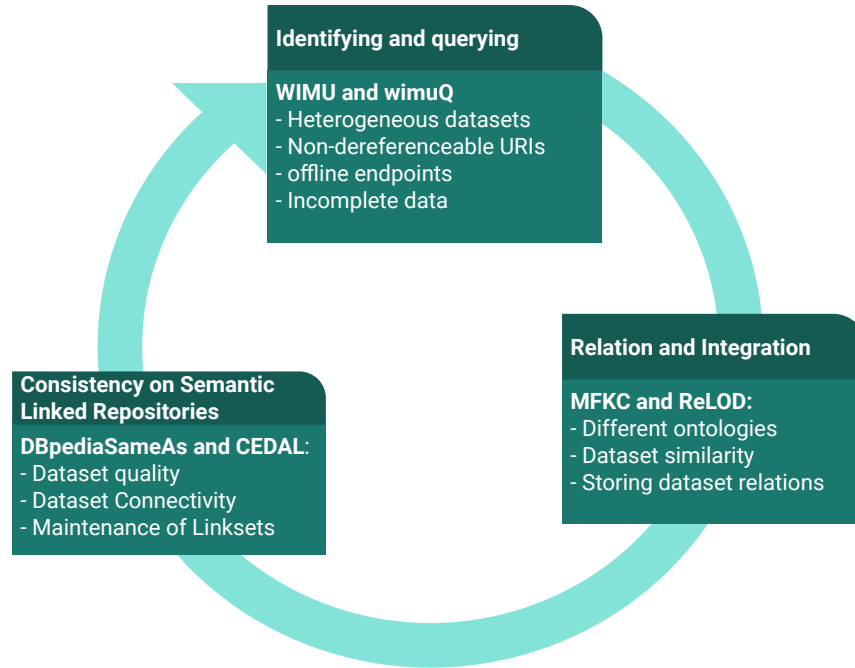


Figure 1: Life-cycle of contributions.

1. With WIMU[114] we provide a regularly updated database index of more than 660K datasets from LODStats and LOD Laundromat, an efficient, low cost and scalable service on the web that shows which dataset most likely defines a URI and various statistics of datasets indexed from LODStats and LOD Laundromat.
2. With wimuQ[117] we present a hybrid SPARQL query processing engine which is able to retrieve results from 559 active SPARQL endpoints (with a total of 163.23 billion triples) and 668,166 datasets (with a total of 58.49 billion triples) from LOD Stats and LOD Laudromat. Thus, to the best of our knowledge, WimuQ is the first federated SPARQL query processing engine that executes SPARQL queries over a net total of 221.7 billion triples. As



part of WimuD, we present a low-cost API dubbed *SPARQL-a-lot* to execute SPARQL queries over LOD-a-lot [34], a 300 GB HDT file of the LOD cloud. For the first time, to the best of our knowledge, we make use of the Concise Bounded Descriptions (CBDs) to execute federate SPARQL queries. We also make use of the WIMU index [114] to intelligently select the capable (also called relevant) [92] data sources pertaining to the given SPARQL query. We evaluated our integrated query engine on two real-data federated SPARQL querying benchmarks *LargeRDFBench* [85], *FedBench* [98] and one non-federated real data SPARQL benchmark selected from *FEASIBLE* [90] benchmarks generation framework.

3. With DBpediaSameAs[113] it was described an approach for the mitigation of the identifier heterogeneity problem and implement a prototype where the user is able to evaluate existing links, as well as suggest new links to be rated. Together with the ability to generate statistics about good and bad links which, brings the possibility to have a quality control for the links to DBpedia. Also we define the DBpedia Unique Identifier (DUI), which instead of several transient `owl:sameAs` DBpedia URIs for the same final address, now is possible to have a unique URI from DBpedia. A DUI goes directly to the final address instead of having to process several possible intermediate results.
4. We present CEDAL[115], a time-efficient algorithm for the detection of erroneous links in large-scale link repositories without computing all closures required by the property axiom. An approach that brings the possibility to track the consistency problems inside link repositories, in which is a scalable algorithm that works well in a parallel and non-parallel mode. Together with a study case applied to a link repository called LinkLion and a new linkset quality measure based on the number of erroneous candidates.
5. We present MKFC[116], in which is based in two nested filters, (1) First Frequency Filter and (2) Hash Intersection filter, that allow to discard candidates before calculating the actual similarity value, thus giving a considerable performance gain, including the *k similarity filter* that allows detecting whether two strings *s* and *t* are similar in a fewer number of steps. We evaluate our approach with respect to its runtime and its scalability with several threshold settings and dataset sizes. Also, we present several parallel implementations of our approach and show that they work well on problems where  $|D_s \times D_t| \geq 10^5$  pairs.
6. We present ReLOD[118], a method to create a repository to store the similarity between a large number of datasets involving the

detection of duplicated datasets and dataset chunks. To this aim, we store those similarities, providing an index to store the relation among them. The similarities processing is based on three properties and classes of each dataset. We give a similarity score to each dataset pair and also provide a search engine to find similar datasets, classes and properties for a given dataset, class or property. We build the index over a total of 668,166 datasets from LOD Stats and LOD Laudromat as well as 559 active SPARQL endpoints. These data sources represent a total of 221.7 billion triples from more than 5 terabytes of information from datasets partially retrieved using the service “Where is My URI” (WIMU). Our evaluation on state-of-the-art real-data shows that more than 90% of datasets from LOD-Laundromat datasets are not using `owl:equivalentProperty` and `owl:equivalentClass` or other way to relate the data, reinforcing the need for an index of relations among LOD datasets.

## 1.5 CHAPTER OVERVIEW

- Chapter 2** introduces the basic concepts and notation that are necessary to understand the rest of this thesis. The notation presented in this chapter is used throughout the thesis.
- Chapter 3** discusses the state-of-the-art research work related to this thesis.
- Chapter 4** is based on [114] and introduces WIMU, a regularly updated database index of more than 660K datasets from LODStats and LOD Laundromat, an efficient, low cost and scalable service on the web that shows which dataset most likely defines a URI.
- Chapter 5** is based on [118] a method to create a repository to store the similarity between a large number of datasets involving the detection of duplicated datasets and dataset chunks.
- Chapter 6** is based on [116] and presents two nested filters, (1) First Frequency Filter and (2) Hash Intersection filter, that allow to discard candidates before calculating the actual similarity value, thus giving a considerable performance gain.
- Chapter 7** is based on [Valdestilhas et al.] presenting an approach which improves the quality of Link repositories with a case applied to DBpedia, in which shows an approach for the mitigation of the identifier heterogeneity problem and implement a prototype where the user is able to evaluate existing links, as well as suggest new links to be rated.
- Chapter 8** is based on [115] present an extension of the previous work [Valdestilhas et al.] discussing the consistency of semantic web Linked repositories in a more wide point of view.

**Chapter 9** is based on [117] and presents a hybrid SPARQL query processing engine which is able to retrieve results from a large number of heterogeneous LOD datasets. The approach is called WimūQ in which is the first federated SPARQL query processing engine that executes SPARQL queries over a net total of 221.7 billion triples.

Finally, **chapter 10** concludes this work and proposes some future work in related areas of research.

## BASIC CONCEPTS AND NOTATION

---

This chapter describes the concepts used along of this work.

**RDF graph.** An RDF graph is a set of RDF triples which has a set of subjects and objects of triples in the graph called nodes. Given an infinite set  $U$  of URIs, an infinite set  $B$  of blank nodes and an infinite set of literals  $L$ , a RDF triple is a triple  $\langle s, p, o \rangle$  where the subject  $s \in (U \cup B)$ , the predicate  $p \in U$  and the object  $o \in (U \cup B \cup L)$ . An RDF triple represents an assertion of a “piece of knowledge”, so if the triple  $\langle s, p, o \rangle$  exists, then, the logical assertion  $p(s, o)$  holds true. An RDF graph is also represented by a collection of RDF triples and it can be seen as a set of statements describing, partially or completely, a certain knowledge.

**Transitive property.** Defined by:  $\forall a, b, c \in X : (p(a, b) \wedge p(b, c)) \implies p(a, c)$ , where  $p$  represents a relation between two elements of a set  $X$ .

**Equivalence.** An equivalence relation is a binary relation that is reflexive, symmetric and transitive. According to OWL semantics, `owl:sameAs` is an equivalence relation.

**Linkset.** According to the W3C,<sup>1</sup> a linkset is a collection of RDF links between two datasets. It is a set of RDF triples in which all subjects are in one dataset and all objects are in another dataset. RDF links often have the `owl:sameAs` predicate, but any other property could occur as the predicate as well. Formally, according to [7], a linkset  $LS$  is a set of RDF triples where for all triples  $t_i = \langle s_i, p_i, o_i \rangle \in LS$ , the subject is in one dataset, i.e. all  $s_i$  are described in  $S$ , and the object is in another dataset, i.e. all  $o_i$  are described in  $T$ , where  $S$  and  $T$  are datasets. We use the word *linkset* for either RDF knowledge base files and dump files from RDF link repositories.

**RDF graph partitioning.** Given a graph  $G = (V, E, lbl, L)$ , a graph partitioning,  $C$ , is a division of  $V$  into  $k$  partitions  $P_1, P_2, \dots, P_k$  such that  $\bigcup_{1 \leq i \leq k} P_i = V$ , and  $P_i \cap P_j = \emptyset$  for any  $i \neq j$ . The edge cut  $E_c$  is the set of edges whose vertices belong to different partitions,  $lbl : E \cup V \rightarrow L$  is a labeling function, and  $L$  is a set of labels.

The definition comes from a recent survey about RDF graph partitioning [110].

---

<sup>1</sup> <https://www.w3.org/TR/void/#linkset>

**Basic Graph Pattern** <sup>2</sup> is a set of Triple Patterns[36].

**RDF Dataset** An RDF dataset is a set:

$$G, (< u_1 >, G_1), (< u_2 >, G_2), \dots (< u_n >, G_n)$$

where  $G$  and each  $G_i$  are graphs, and each  $<u_i>$  is an IRI. Each  $<u_i>$  is distinct.  $G$  is called the default graph.  $(<u_i>, G_i)$  are called named graphs.

**SPARQL** is the language to query RDF datasets, in which the formal definition of a SPARQL Query is: A SPARQL Abstract Query is a tuple  $(E, D, R)$  where  $E$  is a SPARQL algebra expression,  $D$  is an RDF Dataset and  $R$  is a query form.

**Federated Queries** have the aim to collect information from more than one datasets is of central importance for many semantic web and linked data applications [88; 87]. 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.

---

<sup>2</sup> Basic Graph Patterns: <https://www.w3.org/TR/rdf-sparql-query/#BasicGraphPatterns>

## STATE OF THE ART

This chapter is based on [Valdestilhas et al.; 115; 114; 117; 116] and discusses the state-of-the-art research work related to this thesis.

## 3.1 IDENTIFYING DATASETS IN LARGE HETEROGENEOUS RDF SOURCES

The State-of-the-art of Identifying LOD datasets is presented in the following and highlighting the differences with our approach.

The work presented in [22] 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 [42] is based on an approach developed for the 3store 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 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 [34] which is a queryable dump file of the LOD Cloud<sup>1</sup>, 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<sup>2</sup> containing all the data from LOD Laundromat<sup>3</sup>. LOD Laundromat itself provides an endpoint to an inverted index of their data<sup>4</sup>. However, finding the original document a URI was defined in is not trivial, as the returned metadata only describe the datasets themselves [17]. 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 [13] and LOD Laundromat [17], and the most updated SPARQL endpoints. Finally,

<sup>1</sup> <http://lod-cloud.net/>

<sup>2</sup> A HDT file with more than 500GB which requires more than 16 GB RAM to process.

<sup>3</sup> <http://lodlaundromat.org/>

<sup>4</sup> <http://index.lodlaundromat.org/>

WIMU is able to process RDF files containing more than one URI at the same time<sup>5</sup>.

### 3.2 RELATING LARGE AMOUNT OF RDF DATASETS

The work from [10; 11; 12] contains very important statistics about LOD datasets, including a data structure called Equivalence Set Graph (ESG), which allows specifying compact views of large RDF graphs thus easing the accomplishment of statistical observations like the number of concepts defined in a graph. The work helps to show that the LOD datasets are not really linked and the main difference here is that they do not have a goal to compute similarity level among the datasets based on the properties and class similarities.

Loupe [65] has an index of property and classes from some datasets, but still fewer datasets and it is not incremental.

The approach from [30] it is built on the assumption that similar datasets should have a similar structure and include semantically similar resources and relationships based on the combination of Frequent Subgraph Mining (FSM) techniques, used to synthesize the datasets and find similarities among them.

The works from [73] and [15] also provides an approach to identify duplicated data in huge datasets from LODLaundromat and DBpedia. A good point to highlight in those works is the use of bloom filters, which helps to identify duplicated data. The identified different aims compared with our approach are an index of similarity of the datasets based on properties and classes shared among them and the identification of datasets to execute a given SPARQL query.

We should also consider the enforcement from ENTITY RECONCILIATION COMMUNITY GROUP<sup>6</sup>, in which aim to document an existing API, share the experiences and lessons learned from it[26].

Those works cover an important relationship but none of them look into our main drawback here, which is to obtain the similarity level among a huge amount of datasets.

### 3.3 OBTAINING SIMILAR RESOURCES USING STRING SIMILARITY

Our approach can be considered an extension of the state-of-the-art algorithm introduced in [100], which describes a string-based distance function (SDF) based on string hashing [101; 82]. The naive approach of MFKC [100] is a metric for string comparison built on a hash function, which gets a string and outputs the most frequent two characters with their frequencies. This algorithm was used for

<sup>5</sup> For example: <https://wimu.aksw.org/Find?link=http://www.linklion.org/download/mapping/citeseer.rkbexplorer.com---ibm.rkbexplorer.com.nt>.

<sup>6</sup> <https://www.w3.org/community/reconciliation/>

text mining operations. The approach can be divided into two parts: (1) The hashing function is applied to both input strings, where the output is a string that contains the two most frequent characters; the first and third elements keep the characters and second and fourth elements keep the frequency of these characters. (2) The hashes are compared, where will return a real number between 0 and  $\lim$ . By default  $\lim = 10$ , since the probability of having ten occurrences of the two most frequent characters in common between two strings is low. If the output of the function is 10, this case indicates that there is no common character and any value below 10 means there are some common characters shared among the strings.

Our work is similar to the one presented in [124], which features a parallel processing framework for string similarity using filters to avoid unnecessary comparisons.

Among the several types of string similarity, emerging works have been done for measures such as Levenshtein-distance [58], which is a string distance function that calculates the minimum number of edit operations (i.e., delete, insert or update) to transform the first into the second string. The Jaccard Index [51], also called Jaccard coefficient, works on the bitwise operators, where the strings are treated at bit level. REEDED [107] was the first approach for the time-efficient execution of weighted edit distances.

### 3.4 HETEROGENEITY IN DBPEDIA IDENTIFIERS

The state of the art of the work presented in [Valdestilhas et al.] is based on the work [126], in which elaborates a data quality assessment methodology in DBpedia, which comprises of a manual and semi-automatic process. This work drive us to a reinforcement about the concept of data quality used in our work, when in our case will be more a manual process and also we are able to improve the DBpedia data quality.

The work [49], presents a two staged experiment for the creation of gold standards that act as benchmarks for several interlinking algorithms. The similar aspects of this works are: The validation of links and a dubbed manual validation, where the user i.e. validator or evaluator specifies whether a link generated by an interlinked tool is correct or incorrect. The results of the link validation process are used to learn presumably better link specifications and thus achieving high-quality. Also, this work proposes an experiment to investigate the effect of user intervention in dataset interlinking on small knowledge bases.



### 3.4.1 A related problem with sameAs.org

The sameAs.org is a service that leading source of co-reference data on the Semantic Web. For example, when the web site sameAs.org is accessed with a URI from Freebase that should bring information about a country called Brazil.

Was used the URI (<http://rdf.freebase.com/ns/m.015fr>) as parameter to the service, and is received as return more than 1140 URIs as shown in figure 2, but the user can have a doubt about which one is the correct.

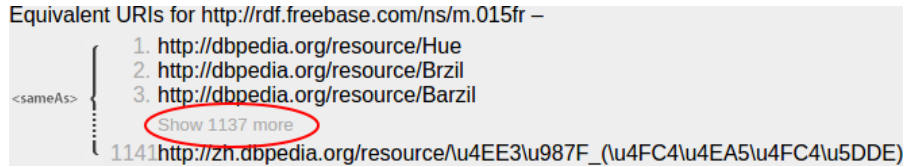


Figure 2: Several URIs with the property `owl:sameAs` from the web site sameAs.org.

Our work is not an alternative to the sameAs web site, but brings possibilities, like, was noticed that the sameAs.org does not provide a way to rate the link, but with this rating, is possible to improve the quality of the data, and bring some facility to the user.

## 3.5 DETECTION OF ERRONEOUS LINKS IN LARGE-SCALE RDF DATASETS

Our work has the aim of detecting erroneous links in large-scale link repositories improving the consistency. The state-of-the-art include the following related works:

- Albertoni et al. [6] focus on the quality dimension of completeness using scoring functions and also introduce a notion of linkset quality, considering only `owl:sameAs`. The work proposes three quality indicators to assess completeness. The extension of this work [4] focuses on `skos:exactMatch` linksets and a multilingual gain.
- The LINK-QA tool [40] uses two network measures designed for Linked Data (i.e., open `sameAs` chains, and description richness) and three classic network measures (i.e., degree, centrality, clustering coefficient) in order to determine whether a set of links can improve the quality of linked data.
- DBpediaSameAs [Valdestilhas et al.] is a work in which Transitive Redirects Links are redundancies at DBpedia that supposes a link to the same place, in other words, they use the `owl:sameAs` property. These links will redirect other links, to provide a transition between the links, hence the name transitive. In this case,

instead of using the transitive links that point to the same final destination URI, the final URI is used directly.

- The work proposed in [123] is a metric-driven approach for interlinking assessment of a single dataset. It introduces the concept of *link-ability*, which shows the potential of a dataset being linked from other datasets, and in general, assesses whether a dataset is good to be interlinked with another dataset using three groups of metrics.
- The approach at [20] proposes strategies in order to reduce the cognitive overhead of creating materialized `owl:sameAs` linksets and to correctly maintain them using two types of components that triple stores should include, which would improve the support for materialized `owl:sameAs` linksets in the creation and maintenance stages.
- [72] evaluates the quantity of links between distributions, datasets, and ontologies categorizing and defining different types of links using probabilistic data structures. The results show valid links, dead links, and a number of namespaces described by distributions and datasets. The analysis was conducted using LOD-Vader [16]. An important point here and in works such as [97; 17; 61] is that they do not mention or use any quality dimension as defined in some important works such as [127; 4]; moreover, they do not consider the axioms related to the properties. The quality is given solely by the number of links between datasets.
- The work described in [77] covers an unsupervised approach for finding erroneous links, in which each link is represented as a feature vector in a higher dimensional vector space, and finds wrong links by means of different multi-dimensional outliers.
- The W3C has a vocabulary to express the quality of data, including a linkset<sup>7</sup>, that is based on the survey by Zaveri et al. [127].
- The work described in [5] discuss results of quality evaluation on linksets created using a framework called LusTRE with two quality measures, the *average linkset reachability* and the *average linkset importing*. Similar to CEDAL, this work realizes that the research on Linked Data quality has been mainly focused on datasets, not on linksets. However, they focus on the SKOS<sup>8</sup> vocabulary, more precisely `skos:exactMatch` linksets, and the experiments from CEDAL were processed in large scale link repositories with more than  $10^7$  triples, not only 31,298 triples.

<sup>7</sup> <https://w3c.github.io/dwbp/vocab-dqg.html#ExpressQualLinkset>

<sup>8</sup> <https://www.w3.org/2004/02/skos/>

- The work described in [24] provides an algorithm with the intention to mitigate the problem of constraints violations in sameAs links automatically. Our approach has some different characteristics, such as CEDAL provides a classification of errors and show that some of them cannot be dealt with automatically in an accurate way, CEDAL use graph partitions in which scales better than this existing approach. This is also shown by the evaluation of CEDAL on larger datasets (19 million vs 3 million), CEDAL preserves the provenance of the links, CEDAL does not remove automatically constraints violations due to the fact that the output results involves semantic accuracy and thus needs human feedback. The similarities include the fact that we also focus on owl:sameAs and we reveal a significant amount of sameAs links that do not adhere to the strict semantics of the OWL standard and hence do not reflect genuine identity.

Common points among these existing works are the improvement and maintenance of link repositories. Aspects include the use of scoring functions, transitive and redirect links, metrics for interlinking datasets, probabilistic data structures, vector space models, and the creation of standards. Our approach provides a high-performance way to dealing with heterogeneous knowledge bases showing the provenance and detecting inconsistent links in large-scale link repositories. Although our work bears some resemblance to existing work on detecting erroneous link candidates in large link repositories, none of the above has considered evaluating the consistency of equivalence relations using a data quality measure.

The novelty of CEDAL with respect to the state-of-the-art can be enumerated in five points. Our approach (1) uses graph partitions and hence scales better than existing approaches using closures, (2) can be applied to larger linksets, with more than 19 million triples, (3) preserves the provenance of the links, (4) shows that some of the error classifications cannot be dealt with automatically in an accurate way and (5) does not remove automatically constraints violations due to the fact that the output results involves semantic accuracy and thus needs human feedback.

### 3.6 QUERYING LARGE HETEROGENEOUS RDF DATASETS

In this section, we provide a brief overview of the state-of-the-art pertaining to querying a huge amount of heterogeneous LOD datasets, in which we start with approaches for federated query processing.

**query federation over multiple sparql endpoints:** DARQ [79], ADERIS [60], FedX [99], Lusail [1], SPLENDID [39], CostFed [96], ANAPSID [2], SemaGrow [21], Odyssey [67], and MULDER [31] are examples of state-of-the-art SPARQL endpoint federation engines.

These approaches can be further divided into 3 categories namely *index-only*, *index-free*, and hybrid (index+ASK) approaches [89].

DARQ and ADERIS are examples of the index-only SPARQL query federation approaches over multiple SPARQL endpoints. DARQ implements a cardinality-based query planner with bind join implementation in nested loops. ADERIS is an adaptive query engine that implements a cost-based query planner. It also makes use of the index-based nested loop join.

FedX and Lusail are examples of the index-free query federation approaches over multiple SPARQL endpoints. FedX only makes use of ASK queries for source selection. It implements a heuristic-based query planner. Comparing to DARQ, the number of endpoints requests is greatly reduced by using bind joins in a block nested loop fashion [99]. A query rewriting algorithm is used to push computation to the local endpoints by relying on information about the underlying RDF datasets. It implements a selectivity-aware query execution plan generation.

SPLENDID, CostFed, ANAPSID, SemaGrow, and Odyssey are examples of the hybrid (index+ASK) SPARQL query federation approaches over multiple SPARQL endpoints. SPLENDID performs cost-based optimization using VOID statistics from datasets. It makes use of bind and hash joins [89]. CostFed also implements a cost-based query planner. The source selection is closely related to HiBISCuS [92]. Both bind and symmetric hash joins are used for data integration. ANAPSID [2] is an adaptive query federation engine that adapts its query execution at runtime according to the data availability and condition of the SPARQL endpoints. ANAPSID implements adaptive group and adaptive dependent joins [89]. SemaGrow adapts source selection approach from SPLENDID. It performs a cost-based query planning based on VOID statistics about datasets. SemaGrow implements bind, hash, and merge joins. Odyssey is also a cost-based federation engine. MULDER describes data sources in terms of RDF molecule templates and utilize these template for source selection, and query decomposition and optimization.

DAW [93] and Fedra [68] are examples of duplicate-aware query federation approaches over multiple SPARQL endpoints.

SaGe[66] is a stateless preemptable SPARQL query engine for public endpoints. The system makes use of preemptable query plans and time-sharing scheduling, SaGe tackles the problem of RDF data availability for complex queries in public endpoints. Consequently, SaGe provides a convenient alternative to the current practice of copying RDF data dumps.

**link traversal based sparql federation:** LDQPS [53], SIHJoin [54], WoDQA [3], and SQUIN [45] are examples of traversal-based federated SPARQL query processing approaches. Both LQPS and SI-

HJoin make use of the index and online discovery via link-traversal to identify the relevant sources pertaining to the given SPARQL query. They implement symmetric hash join. WoDQA performs hybrid (index+ASK) source selection approach. It implements nested loop and bind joins. SQUIN discovers the potentially relevant data during the query execution and thus produce incremental query results. SQUIN uses a heuristic for query execution plan generation, adapted from [43]. As a physical implementation of the logical plans, SQUIN uses a synchronized pipeline of iterators such that the  $i$ -th operator is responsible for the  $i$ -th triple pattern of the given SPARQL query. More recent studies [47] investigated 14 different approaches to rank traversal steps and achieve a variety of traversal strategies. A more exhaustive survey of the traversal-based SPARQ query federation is provided in [46].

Beside the above query federation strategies, low-cost triple pattern fragments (TPF) interfaces [120] can also be used to execute federated SPARQL queries. Comunica [108] is a highly modular meta engine for federated SPARQL query evaluation over support heterogeneous interfaces types, including self-descriptive Linked Data interfaces such as TPF. The system also enables querying over heterogeneous sources, such as SPARQL endpoints and data dumps in RDF serializations. The main drawback here is that the user should know in advance the dataset where the query will be executed.

One of the targets and motivation to build WimuD was to discover which dataset the SPARQL query can be executed, unfortunately, with exception of SQUIN [45], none of the related works provides this feature. Despite the fact that some of them incorporate Nquad support that allows the possibility to know the graph of the triple. The provenance system of WimuD also includes dump files and endpoints with a rank provided by WIMU [114].

## IDENTIFYING DATASETS IN LARGE HETEROGENEOUS RDF SOURCES

---

*This chapter covers one of the Semantic Web foundations, which 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.*

### 4.1 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 [119]. 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 [13] and LOD Laundromat [17]. 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<sup>1</sup> 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://wimu.aksw.org/> under GNU Affero public license 3.0 and the source code is available online<sup>2</sup>.

The rest of the chapter is organized as follows: We discuss the proposed approach in detail, including the index creation, the web interface, and the data processing. We finally present our evaluation results.

## 4.2 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<sup>3</sup> index. Due to runtime performance and complexity reasons, we found that storing

<sup>1</sup> Updated monthly due to the huge size of data processed.

<sup>2</sup> <https://github.com/dice-group/wimu>

<sup>3</sup> <https://lucene.apache.org/>



the information into Lucene was more convenient than a traditional triple store such as Virtuoso<sup>4</sup>. 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
: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.

#### 4.2.1 The index creation

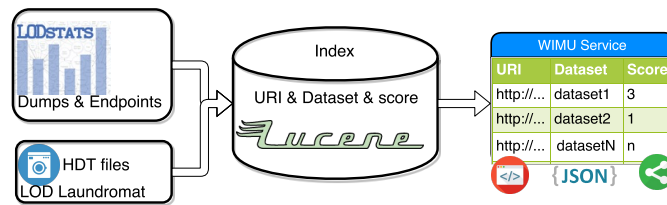


Figure 3: Creation workflow of the WIMU index.

The index creation, the core of our work, is shown in figure 3 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
```

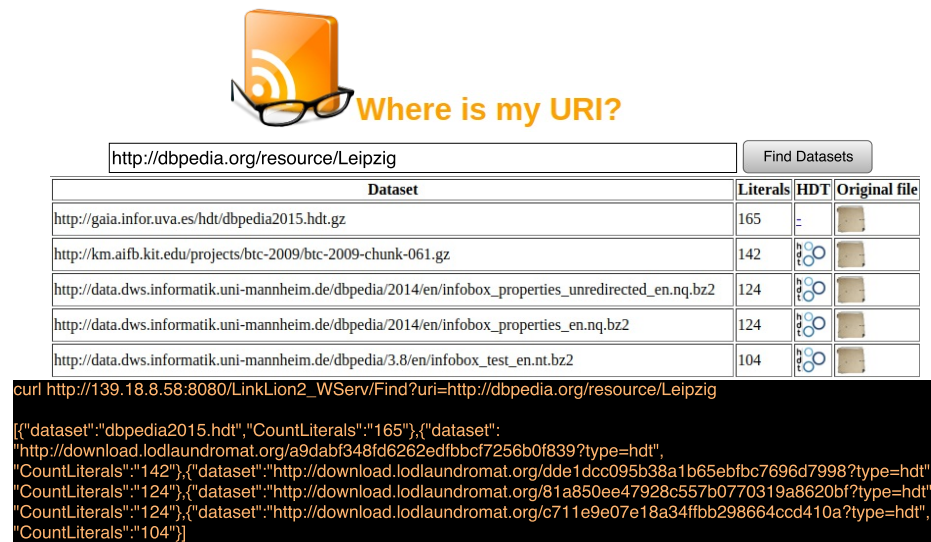
<sup>4</sup> <https://virtuoso.openlinksw.com/>



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.

#### 4.2.2 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 figure 4 shows.



The screenshot shows the WIMU web interface. At the top, there is a logo with an orange square and glasses, and the text "Where is my URI?". Below this is a search bar containing the URI "http://dbpedia.org/resource/Leipzig" and a "Find Datasets" button. The results are displayed in a table with four columns: Dataset, Literals, HDT, and Original file. The table lists five datasets with their respective URI, number of literals, and HDT status. Below the table, there is a terminal window showing the command "curl http://139.18.8.58:8080/LinkLion2\_WServ/Find?uri=http://dbpedia.org/resource/Leipzig" and the resulting JSON output.

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/c711e9e07e18a34ffb298664ccd410a?type=hdt",
"CountLiterals":"104"}}
```

Figure 4: Web interface.

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 figure 4, 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. figure 6 shows the generic usage of WIMU.

### 4.3 USE CASES

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

#### 4.3.1 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 meta-data available, hence their Concise Bounded Descriptions (CBDs) are hard to obtain. In following figure,  $(D_1, \dots, D_n | x) : D_n$  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  $D_1$  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.

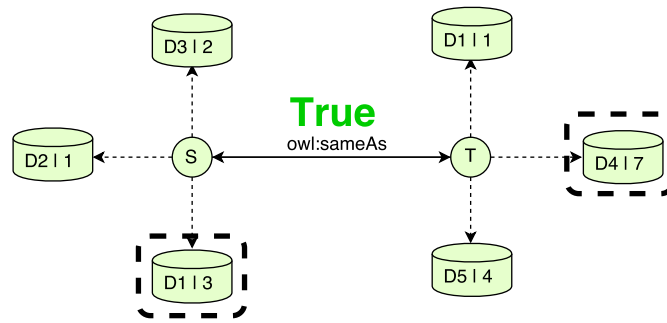


Figure 5: First use-case.

#### 4.3.2 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<sup>5</sup> from the LinkLion repository [69] 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><sup>6</sup>. 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 [55].

<sup>5</sup> <http://www.linklion.org/download/mapping/citeseer.rkbexplorer.com---ibm.rkbexplorer.com.nt>

<sup>6</sup> Also available in HDT file from <http://download.lodlaundromat.org/15b06d92ae660ffdcff9690c3d6f5185?type=hdt>

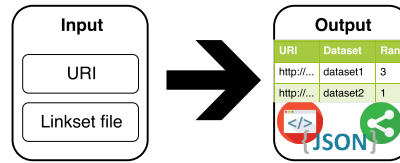


Figure 6: Usage.

Table 1: Parameters

Parameter	Default	Description
top	0	Top occurrences of the datasets where the URI was defined.
uri	-	URI expected to search .

#### 4.3.3 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 [88; 87]. 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 [39], SemaGrow [21], HiBISCuS [92], CostFed [78], etc.

#### 4.3.4 Usage examples

The service API provides a JSON as output, allowing users to use WIMU with some programming language compatible with JSON. Here we give examples, for more details please check the manual<sup>7</sup>.

**Service:** <https://wimu.aksw.org/Find>

Parameters table 1:

Input (Single URI example):

<https://wimu.aksw.org/Find?top=5&uri=http://dbpedia.org/resource/Leipzig>

Output:

```

1 [
2   {
3     "dataset":

```

<sup>7</sup> More examples such as many URIs, linksets, and generation of Concise Bounded Description (CBD) check <https://dice-group.github.io/wimu/>

```

4  "http://downloads.dbpedia.org/2016-10/core-i18n/en/
   infobox_properties_en.ttl.bz2",
5  "CountLiteral": "236"
6  },
7  {
8  "dataset": "http://gaia.infor.uva.es/hdt/dbpedia2015.hdt.gz",
9  "CountLiteral": "165"
10 },
11 {
12 "dataset":
13 "http://download.lodlaundromat.org/
   a9dabf348fd6262edfbbcf7256b0f839?type=hdt",
14 "CountLiteral": "142"
15 },
16 {
17 "dataset":
18 "http://download.lodlaundromat.org/
   dde1dcc095b38a1b65ebfbc7696d7998?type=hdt",
19 "CountLiteral": "124"
20 }
21 ]

```

Java and the API Gson<sup>8</sup>:

```

1 private void exampleJson() throws Exception {
2     String service = "https://wimu.aksw.org/Find?uri=";
3     String uri = "http://dbpedia.org/resource/Leipzig";
4     URL url = new URL(service + uri);
5     InputStreamReader reader = new InputStreamReader(url.openStream());
6     WIMUDataset wData = new Gson().fromJson(reader, WIMUDataset[].class)[0];
7     System.out.println("Dataset:" + wData.getDataset());
8     System.out.println("Dataset(HDT):" + wData.getHdt());
9 }

```

#### 4.4 STATISTICS ABOUT THE DATASETS

To the best of our knowledge, LODStats[13] is the only project oriented to monitoring dump files; however, its last update dates back to 2016. Observing table 2, 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<sup>9</sup>. A large part of those data was processed and cleaned by LOD Laundromat [17].

Table 2: Datasets.

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

<sup>8</sup> <https://github.com/google/gson>

<sup>9</sup> See <https://github.com/dice-group/wimu/blob/master/ErrorJenaParser.tsv>.

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 figure 7, 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 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.

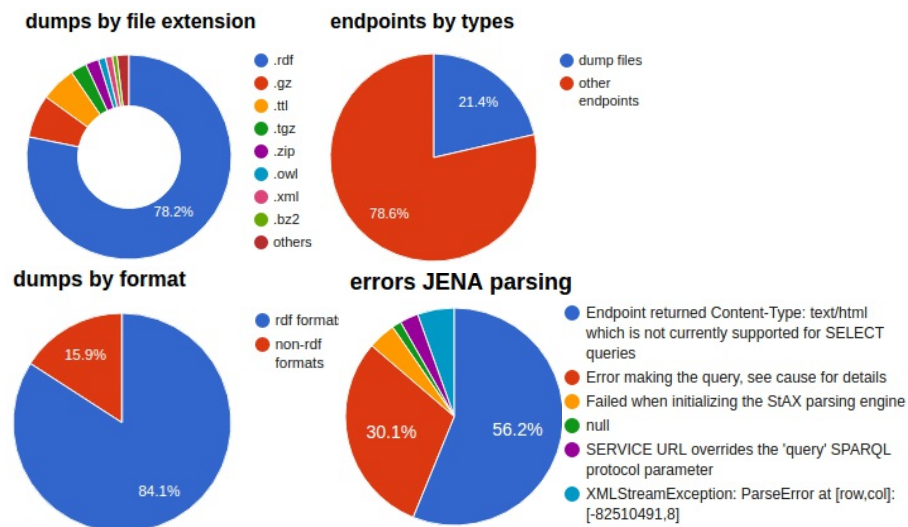


Figure 7: Dump files and Apache Jena parsing error.

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<sup>10</sup> 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.

<sup>10</sup> <https://github.com/dice-group/wimu/blob/master/result100.csv>

*This chapter highlights the challenge where although we have witnessed the growing adoption of Linked Open Data principles for publishing data on the Web, connecting data to third parties remains a difficult and time-consuming task. One question that often arises during the publication process is: “Is there any data set available on the Web we can connect with?”. This simple question unfolds a set of others that hinders data publishers to connect to other data sources. For instance, if there are related data sets, where are they? How many? Do they share concepts and properties? How similar are they? Is there any duplicated data set? To answer these questions, this chapter introduces: (i) a new class of data repositories; (ii) a novel method to store and detect data set similarities (duplicated data set and data set chunks); (iii) an index to store data set relatedness; and, (iv) a search engine to find related data sets. To create the index, we harvested more than 668k data sets from LOD Stats and LOD Laundromat, along with 559 active SPARQL endpoints corresponding to 221.7 billion triples or 5 terabytes of data. Our evaluation on state-of-the-art real-data shows that more than 90% of data sets in the LOD Laundromat do not use the owl:equivalentProperty, owl:equivalentClass nor any other way to relate to one another data, which reaffirms and emphasize the importance of our work.*

## 5.1 INTRODUCTION

People will always describe knowledge in different ways, due to the individuality of the being, different points of view, time and space, in which we assume as a natural phenomenon and standard. Aware of the state of the art from Ontology/Schema/Instance Matching from Linked Data from venues such as VLDB<sup>1</sup>, OEAI<sup>2</sup>, ISWC<sup>3</sup>, ESWC<sup>4</sup>, among others, we observe that the LOD datasets are following this standard.

We notice that there are several similar data sets. Yet, there is no place that stores or provides such kind of information about how similar the data sets are and which attributes the data sets share. This work aims to build an index to store the relation among the data sets to have a place to see how similar the data sets are. We provide a method to index the relations among LOD data sets<sup>5</sup> based on the properties and classes that they share.

<sup>1</sup> <https://www.vldb.org/>

<sup>2</sup> <http://oeai.ontologymatching.org/>

<sup>3</sup> <https://iswc2019.semanticweb.org/>

<sup>4</sup> <https://2019.eswc-conferences.org/>

<sup>5</sup> Data sets from LODStats, LODLaundromat, and endpoints

As a motivation of our work, we describe two distinct scenarios where the proposed index may be useful:

**Scenario 1:** Given the SPARQL query at Listing 1, where the user wants to know if a given data set contains three properties, in which the user already know the query performs well at the SPARQL endpoint from CKAN<sup>6</sup>:

```

1 SELECT * WHERE { ?s ?p ?o.
2 FILTER(?s=<http://purl.org/dc/terms/date> ||
3 ?s=<http://crime.rkbexplorer.com/id/location> ||
4 ?s=<http://purl.org/dc/terms/subject>
5 )}

```

Listing 1: Scenario 1.

The following questions are raised:

- How to know which data sets are able to execute the query?
- Which of the data sets contains the most valuable results?
- Could the results from different data sets complement each other?

In this case an approach is needed to enable the user to know more datasets to extract the required information. An index of dataset relations/similarities could be used to know that the query can also be executed with results at <https://eu.dbpedia.org/sparql> but not at <http://dbpedia.org/sparql>, and we can execute at least in other five<sup>7</sup> data sets to complement the results.

**Scenario 2:** The user need information about cities from all datasets in your repository, more specifically datasets that are compatible with the class <http://dbpedia.org/ontology/City>. Using our previous approach WIMU[114], this URI was found in 11 datasets<sup>8</sup>, but how many URIs in other datasets that also represents a city that were not listed, for instance a city at WikiData<sup>9</sup> is represented by <https://www.wikidata.org/wiki/Property:P131>. We consider here that to execute such task manually on more than 600,000 datasets is not feasible. Thus, having an index of dataset relation will help in this case.

The contributions of this chapter are:

- A method to create an incremental index of similarity among LOD datasets. Thus, to the best of our knowledge, ReLOD is the first dataset relation index over a net total of 221.7 billion triples.

<sup>6</sup> CKAN SPARQL endpoint <https://linked.opendata.cz/sparql>

<sup>7</sup> Datasets available here: <https://tinyurl.com/5dataset>

<sup>8</sup> Datasets available here: <https://tinyurl.com/wimuDbpediaCity>

<sup>9</sup> <https://www.wikidata.org/>



- A creation of a mechanism to search this index by Dataset URI, property, class or SPARQL query. For the first time, to the best of our knowledge, we make a relation index able to query by dataset URI, property, class or SPARQL query.
- A method to identify duplicated datasets and chunks.

The rest of the chapter is organized as follows: First we present our approach in the Section 9.3 then the Section 9.4 shows the evaluation results and discussion.

## 5.2 THE APPROACH

Our observations guide us to a method to create an index of LOD datasets, in which involves a compilation of many other works, starting collecting data from more than 650,000 LOD datasets from LODStats, LODLaundromat and more than 500 endpoints<sup>10</sup>.

### 5.2.1 The index creation

Given a set of Datasets  $\mathbf{D}\{d_1, \dots, d_n\}$ , assuming  $d_n$  represents a RDF dataset<sup>11</sup>, in which each  $d$  contains a set of properties and classes represented by  $\mathcal{P}$ . The next step consists in creating a set  $\mathbf{E}$  containing the identified duplicated datasets. Thus, we can exclude from  $\mathbf{D}$  all datasets contained in  $\mathbf{E}$ .

The goal of algorithm 1<sup>12</sup> is to create an index to store the relations among datasets by comparing the occurrences of classes and properties of each dataset, by String similarity and Instance Matching. The **Input** is a set of Datasets and the **Output** is the index of dataset relations.

figure 8 shows a workflow about how the index is created and the algorithm 1 describe with more details the process of creation of the dataset relation index.

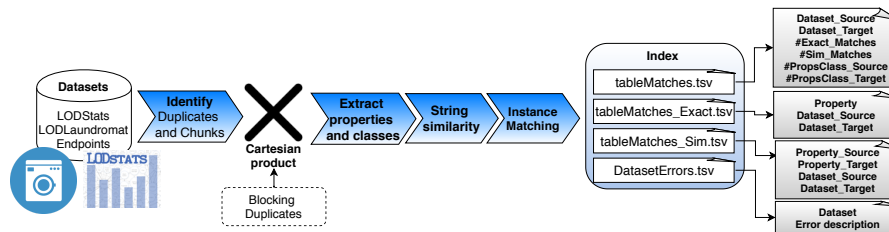


Figure 8: Creating the index.

10 The list of endpoints is available here: [https://github.com/firmao/wimUT/blob/master/Endpoints\\_numtriples\\_lodcloud.csv](https://github.com/firmao/wimUT/blob/master/Endpoints_numtriples_lodcloud.csv)

11 RDF datasets: <https://www.w3.org/TR/rdf11-datasets/>

12 Implementation in Java <https://tinyurl.com/y597qrah>



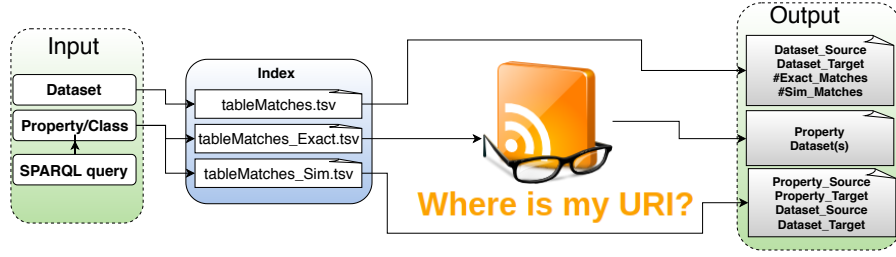


Figure 9: Querying the index.

**Algorithm 1** Creation of the LOD dataset relation index

---

<p><b>Input:</b> <math>D, E</math> <span style="float: right;">▷ Datasets and datasets duplicated</span></p> <p><b>Output:</b> <math>C, X, Y, Z</math> <span style="float: right;">▷ Four tsv files containing the matches(datasets, classes and properties).</span></p> <pre> 1: <math>M_e \{ \}</math> <span style="float: right;">▷</span>    HashMap containing dataset-Target and another hash containing properties and classes exact matched 2: <math>M_s \{ \}</math> <span style="float: right;">▷</span>    HashMap containing dataset-Target and another hash containing properties and classes matched using String similarity 3: <math>M_a \{ \}</math> <span style="float: right;">▷</span> HashMap containing datasets already compared 4: <math>D = E</math> <span style="float: right;">▷</span> Remove duplicates contained in <math>E</math> 5: <b>for</b> <math>d \in D</math> <b>do</b>    <b>end</b>    In parallel <b>for</b> <math>t \in T</math> <b>do</b> </pre>	<p style="text-align: right;"><b>end</b></p> <p style="text-align: right;">(<math>d \neq t</math>) <math>\wedge</math> <b>already-Compared</b>(<math>d, t</math>)</p> <pre> 6: continue <span style="float: right;">▷</span> Skip the current <math>d</math> and <math>t</math> 7: 8: <math>M_e.add(getExactMatches(d, t))</math>    <span style="float: right;">▷</span> Add a set containing the properties/classes that are exact the same in both datasets 9: <math>M_s.add(getSimMatches(d, t, 0.8, M_e))</math> <span style="float: right;">▷</span> Add a set containing the properties that are similar in both datasets, excluding the matches from <math>M_e</math> 10: <math>M_a.add(d, t)</math> <span style="float: right;">▷</span> Add the dataset pair already processed 11: 12: 13: <b>printMaps</b>(<math>M_e, M_s</math>) <span style="float: right;">▷</span> Print the content of <math>M_e</math> and <math>M_s</math> inside the files <math>C, X, Y, Z</math> </pre>
--	--

---

The function **getExactMatches**( $d, t$ ) compares all properties and classes from each dataset  $d$  and  $t$  and return a set of properties and classes that are exact the same, keeping the provenance of the dataset.

The function **getSimMatches**( $d, t, 0.8, M_e$ ) compares all properties and classes from each dataset  $d$  and  $t$  using Jaccard and MFKC[116] String Similarity function with a threshold of 0.8, excluding the exact matches identified previously by the function **getExactMatches** returning a set of properties and classes that has the similarity greater

than the threshold 0.8, keeping the provenance of the dataset, the figure 10 shows our string similarity process.

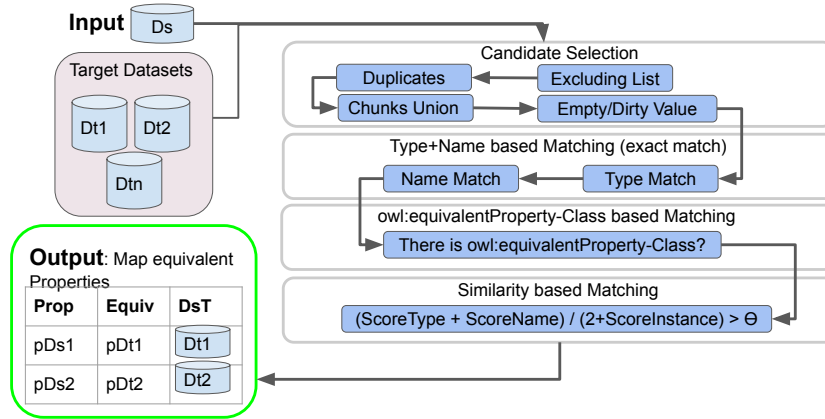


Figure 10: String similarity process.

The function **printMaps**( $M_e, M_s$ ) print the content from the matching functions into the files according the structure described in section 5.2.3.

### 5.2.2 Identifying duplicated and chunk datasets

With the current hardware available (HD 1Tb, Memory 16GB, processor intel core i7). To compare all LOD datasets, in this case more than 600 thousand datasets, implies in more than 360 billion of comparisons( $n * n$ ), assuming that each comparison takes in average 1 millisecond, the whole operation will take more than 10 years.

Thus, we decided to have a blocking method to avoid unnecessary comparisons. To this aim, we create a way to detect duplicated and chunk-dump-files datasets, described on Algorithmus 5. Our blocking method avoid datasets with less than 1 triple, duplicates, less than 5 properties, less than 5 classes.

The function **eliminateDuplicates()**, identify and eliminate duplicated HDT files by comparing the property occurrences of each dataset and the header metadata.

We formalize the problem of Clustering datasets identifying duplicates and chunks as follows:

We consider a set of  $K$  data sources  $S_1, \dots, S_k$  containing property occurrences  $O_1, \dots, O_m$ . Each of property  $P$  is referenced by an URI, e.g., `dbo:City`<sup>13</sup>. Each  $S$  contains a header  $H$ , with meta-data about the data in  $S$ , such that  $H \subset S$ .

The goal is to create clusters of datasets  $C$  in two groups of elements from  $K$ , in which are duplicates  $D$  and chunks  $E$ , such that  $D \subset K : E \subset K : K \subset C$ .

<sup>13</sup> `dbo:City` states for the URI <http://dbpedia.org/ontology/City>

We assume that all datasets are not empty and are RDF compatible with the HDT format.

Firstly we create a dense matrix of property occurrence and dataset. Then it was observed a standard in the dense matrix, that for some datasets the occurrence of the properties was exactly the same. Then, we identify that we can put together those datasets to observe more characteristics. In this second phase, we have a sub-collection of our initial collection of datasets and looking into the header metadata of those files we observe that for some files the meta-data is different. Then we have another sub-collection of files with different headers, and manually reading file by file we realize that they are chunks, in other words, part of a bigger dataset.

Thus we have two phases for clustering dataset (1) Put together datasets that present the same occurrence of properties. (2) Read the header metadata and separate files with a different header. The files with different header are chunks and the rest are duplicates<sup>14</sup>.

---

**Algorithm 2** Identifying duplicates and chunk
 

---

<b>Input: D</b> ▷ Datasets <b>Output: C, X</b> ▷ two files chunks.txt and duplicates.txt. 1: <b>sep = t</b> ▷ Separator for the file, in this case, a tab, but could be another such as “,”, etc.. 2: <b>setHeader = {}</b> 3: <b>cSparql = “Select ?p</b> (count(?p) as ?qtd) where ?s ?p ?o . group by ?p” forall <b>d ∈ D do</b> 4: <b>end</b> <b>H = getPropertyOccur-</b> <b>rence(ds) forall key, value ∈</b> <b>H do</b> <b>end</b> <b>key</b> represent the prop- erty and <b>value</b> represents the number of occurrences. 5: <b>setHeader.add(key)</b>	6: <b>line = line + sep + value</b> 7: 8: <b>printLineDenseMatrix(d +</b> <b>line) if setLines.contains(line)</b> <b>then</b> <b>end</b> Here we identify and put chunks and duplicates to- gether 9: <b>D'.add(d) else</b> 10: <b>end</b> <b>setLines.add(line)</b> 11: 12: 13: <b>printHeaderDenseMatrix(setHeader)</b> 14: <b>C.add(diff(D'))</b> ▷ Here we separate chunks and duplicates 15: <b>X.add(diffD(D'))</b> 16: <b>return C, X</b> ▷ two files chunks.txt and duplicates.txt.
--	---

---

<sup>14</sup> More details, please see the implementation and the documentation on <https://github.com/firmao/wimuT> and the specific implementation java class for this task: <https://github.com/firmao/wimuT/blob/master/src/org/wimu/datasetselection/parallelv1/ClusterKmeans.java>

The function **getPropertyOccurrence(ds)** returns a hash map containing the property and the number of occurrences of each property from a given dataset with the SPARQL query at Listing 2

```
1 Select ?p (count(?p) as ?qtd) where {?s ?p ?o .} group by ?p
```

Listing 2: Property occurrence query.

The function **printLineDenseMatrix(line)** print a line in a file and the function **printHeaderDenseMatrix(setHeader)** print the first line of the dense matrix file containing the header of the matrix, in which refers to the identification of the properties. The function **diff(D')** separates the chunks from duplicates, for this task we look into the header of the files, in this case, only HDT files, where duplicates present exactly the same header metadata information and chunks are different respect to the header.

**Theorem 1:** Let **P** be a collection of property occurrences of datasets **D**, such that each **D** has one **P**, in which **D'** represents the cluster candidates, in which are the datasets identified as chunks and duplicates together, and the function **diff()** return the dump-files that are not duplicated. Thus, the output of Algorithm 1 when applied to **D**, the following hold:

$$|\mathbf{diff}(\mathbf{D}')| = 0$$

All elements from **D'** are duplicated datasets.

To identify the chunks:

$$|\mathbf{diff}(\mathbf{D}')| > 0$$

The result from **diff(D')** are the dump-files identified as chunks.

### 5.2.3 The file structure

Now we describe the file structure that we created to use our index, which consists of 3 TSV<sup>15</sup> files.

- **tableMatches\_Exact.tsv:** Contains the exact match of the properties, with the following fields:
  - **Property:** containing the property URI itself.
  - **Source:** Contains the dataset source where the property was found.
  - **Target:** Contains de dataset matched containing the property.
- **tableMatches\_Sim.tsv:** Represents the similarity of properties from datasets Source and Target, with the following fields:
  - **PropertyS:** The property from the dataset Source.
  - **PropertyT:** The property from the dataset Target.

<sup>15</sup> Tab Separated Value(TSV)

- **Source:** The dataset Source.
- **Target:** The dataset Target.
- **tableMatches.tsv:** Represents the datasets Matched and his respective number of properties exact matched and number of properties matched with String similarity, with the following fields:
  - **Source:** The dataset source.
  - **Target:** The dataset Target.
  - **#ExactMatch:** Number of properties with exact match.
  - **#sim>0.9:** Number of properties with similarity threshold greater than 0.9.

#### 5.2.4 Querying the index

The index provides the following information based on three types of input:

**Dataset:** The index will provide a list of datasets, number of exact matched<sup>16</sup> properties and number of similar<sup>17</sup> properties.

**Set of properties (URIs) separeted by comma:**

- Json file containing the property and the list of datasets where this property were found by exact match.
- A table containing the matches by similarity with the following fields:

**Property Source:** Representing the property found on the dataset source.

**Property Target:** Representing the property found on the dataset target.

**Dataset Source:** The name of the dataset source.

**Dataset Target:** The name of the dataset target.

**SPARQL query:** This type of input extracts the set of properties from the SPARQL query and performs the same operation as **Set of properties (URIs) separeted by comma.**

The index is also incremental, allowing to add more datasets to be processed once a month<sup>18</sup>. The prototype is available online for proof

<sup>16</sup> Exact Match where the URI is exact the same following the principle of uniqueness of the URI

<sup>17</sup> Similarity match occurs when the similarity of the URI is greater than 0.9 if less we perform Instance matching.

<sup>18</sup> Once a month due to the the time to generate that with our hardware takes at least 88 hours

of concept online<sup>19</sup> and the figure 9 shows a workflow about how the index is used.

Table 3: Sample datasets from [37] to evaluate our string similarity approach.

Data set	Source (S)	Target (T)	$ S  \times  T $	Source Property	Target Property
(P3)Abt-Buy	Abt	Buy	$1.20 \times 10^6$	product name, description manufacturer, price	product name, description manufacturer, price
(P4)Amazon-GP	Amazon	Google Products	$4.40 \times 10^6$	product name, description manufacturer, price	product name, description manufacturer, price
(P5)DBLP-ACM	ACM	DBLP	$6.00 \times 10^6$	title, authors venue, year	title, authors venue, year
(P6)DBLP-Scholar	DBLP	Google Scholar	$0.17 \times 10^9$	title, authors venue, year	title, authors venue, year
(P7)MOVIES	DBpedia	LinkedMDB	$0.17 \times 10^9$	dbp:name dbo:director/dbp:name dbo:producer/dbp:name dbp:writer/dbp:name rdfs:label	dc2:title movie:director/movie:director_name movie:producer/movie:producer_name movie:writer/movie:writer_name rdfs:label

### 5.3 EVALUATION

This evaluation aims to answer the following questions: (1) How to identify and quantify similar datasets for a given Dataset?<sup>20</sup> (2) How many datasets are most likely to execute a given SPARQL query? (3) How the detection of duplicated and chunk datasets can help in the process of matching a large amount of datasets? (4) How ReLOD increase the number of datasets identified by wimuQ?

The information contained at table 4 and table 5 are used to see how a sample of datasets are similar, we take 8 famous datasets from *rdfhdt.org*<sup>21</sup> and 12 from [25]. To obtain the results from the table 4. which is to see how similar are the datasets, we use the formula Gleichung (1) and to obtain the results from table 5, to see how much a dataset is contained inside each other we use the formula in Gleichung (2), where A and B represents datasets source and target. We have an alternative way to visualize this sample data as a graph<sup>22</sup>.

$$jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

$$contained(A, B) = \frac{|A \cap B|}{|A|} \quad (2)$$

To answer the question (1) we give to our approach as input a version of the dataset DBpedia<sup>23</sup>, that contains 2339 properties and classes, and as output we can obtain the most similar datasets<sup>24</sup>. Thus

<sup>19</sup> <http://w3id.org/relod/>

<sup>20</sup> To know how many datasets are similar to each other.

<sup>21</sup> <http://www.rdfhdt.org/datasets/>

<sup>22</sup> <https://w3id.org/relod/visual.html>

<sup>23</sup> <http://gaia.infor.uva.es/hdt/DBpedia-3.9-en.hdt.gz>

<sup>24</sup> We compare with the datasets from <http://www.rdfhdt.org/datasets/>

Coefficient	acadonto	acm.rkbexplorer	agrinepaldata	aims	alaska	apertium	arrayexpress	athelia	bfs.270a	biblioteca-nacional	brown	dblp	dbpedia	geonames	linkedgedata	swdf	vivosearch	wiktionary	wordnet	yago
acadonto	1.0	0.02	0.11	0.05	0.00	0.03	0.02	0.06	0.01	0.02	0.02	0.02	0.00	0.02	0.00	0.01	0.01	0.02	0.01	0.02
acm.rkbexplorer	0.02	1.0	0.08	0.22	0.01	0.11	0.01	0.29	0.30	0.07	0.20	0.05	0.00	0.02	0.00	0.02	0.01	0.04	0.02	0.08
agrinepaldata	0.11	0.08	1.0	0.11	0.01	0.16	0.05	0.20	0.05	0.06	0.07	0.04	0.00	0.03	0.00	0.02	0.03	0.05	0.02	0.02
aims	0.05	0.22	0.11	1.0	0.02	0.07	0.08	0.36	0.27	0.08	0.12	0.08	0.00	0.04	0.00	0.06	0.06	0.03	0.03	0.10
alaska	0.00	0.01	0.01	0.02	1.0	0.01	0.00	0.02	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.01	0.02	0.01	0.01	0.07
apertium	0.03	0.11	0.16	0.07	0.01	1.0	0.02	0.05	0.06	0.06	0.07	0.08	0.00	0.05	0.00	0.02	0.02	0.10	0.04	0.20
arrayexpress	0.02	0.01	0.05	0.08	0.00	0.02	1.0	0.06	0.01	0.01	0.02	0.01	0.00	0.01	0.00	0.02	0.04	0.01	0.01	0.40
athelia	0.06	0.29	0.20	0.36	0.02	0.05	0.06	1.0	0.29	0.06	0.20	0.06	0.00	0.03	0.00	0.04	0.05	0.03	0.02	0.03
bfs.270a	0.01	0.30	0.05	0.27	0.01	0.06	0.01	0.29	1.0	0.06	0.15	0.03	0.00	0.01	0.00	0.02	0.02	0.02	0.02	0.02
biblioteca-nacional	0.02	0.07	0.06	0.08	0.01	0.06	0.01	0.06	0.06	1.0	0.08	0.05	0.00	0.02	0.00	0.03	0.01	0.04	0.02	0.05
brown	0.02	0.20	0.07	0.12	0.01	0.07	0.02	0.20	0.15	0.08	1.0	0.09	0.00	0.02	0.00	0.03	0.02	0.03	0.02	0.01
dblp	0.02	0.05	0.04	0.08	0.01	0.08	0.01	0.06	0.03	0.05	0.09	1.0	0.00	0.05	0.00	0.07	0.01	0.06	0.03	0.10
dbpedia	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.0	0.00	0.00	0.01	0.00	0.00	0.00	0.10
geonames	0.02	0.02	0.03	0.04	0.00	0.05	0.01	0.03	0.01	0.02	0.02	0.05	0.00	1.0	0.00	0.02	0.01	0.04	0.01	0.40
linkedgedata	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.0	0.00	0.00	0.00	0.00	0.07
swdf	0.01	0.02	0.02	0.06	0.01	0.02	0.02	0.04	0.02	0.03	0.03	0.07	0.01	0.02	0.00	1.0	0.03	0.01	0.01	0.08
vivosearch	0.01	0.01	0.03	0.06	0.02	0.02	0.04	0.05	0.02	0.01	0.02	0.01	0.00	0.01	0.00	0.03	1.0	0.01	0.01	0.10
wiktionary	0.02	0.04	0.05	0.03	0.01	0.10	0.01	0.03	0.02	0.04	0.03	0.06	0.00	0.04	0.00	0.01	0.01	1.0	0.03	0.06
wordnet	0.01	0.02	0.02	0.03	0.01	0.04	0.01	0.02	0.02	0.02	0.02	0.03	0.00	0.01	0.00	0.01	0.01	0.03	1.0	0.04
yago	0.02	0.08	0.02	0.10	0.07	0.20	0.40	0.03	0.02	0.05	0.02	0.10	0.10	0.40	0.07	0.08	0.10	0.06	0.04	1.0

Table 4: Similarity table according to Jaccard method applied to a sample data, in which the level of similarity is also represented by the intensity of the color, as more intense color, as more similar are the datasets.

Coefficient	acadonto	acm.rkbexplorer	agrinepaldata	aims	alaska	apertium	arrayexpress	athelia	bfs.270a	biblioteca-nacional	brown	dblp	dbpedia	geonames	linkedgedata	swdf	vivosearch	wiktionary	wordnet	yago
acadonto	1.0	0.04	0.16	0.20	0.00	0.04	0.24	0.16	0.04	0.04	0.04	0.04	0.00	0.04	0.04	0.08	0.20	0.04	0.04	0.10
acm.rkbexplorer	0.04	1.0	0.21	0.69	0.01	0.15	0.08	0.62	0.69	0.12	0.38	0.12	0.00	0.04	0.08	0.27	0.19	0.08	0.03	0.08
agrinepaldata	0.16	0.21	1.0	0.64	0.01	0.29	0.86	0.71	0.21	0.14	0.21	0.14	0.00	0.07	0.14	0.43	0.93	0.14	0.03	0.10
aims	0.20	0.69	0.64	1.0	0.02	0.08	0.10	0.43	0.36	0.09	0.16	0.22	0.00	0.05	0.05	0.07	0.32	0.10	0.06	0.10
alaska	0.00	0.01	0.01	0.02	1.0	0.01	0.01	0.02	0.02	0.01	0.01	0.01	0.00	0.00	0.02	0.02	0.05	0.01	0.01	0.01
apertium	0.04	0.15	0.29	0.08	0.01	1.0	0.02	0.20	0.27	0.13	0.20	0.10	0.00	0.13	0.20	0.02	0.40	0.13	0.04	0.10
arrayexpress	0.24	0.08	0.86	0.10	0.01	0.02	1.0	0.06	0.01	0.01	0.02	0.10	0.00	0.01	0.01	0.04	0.11	0.01	0.06	0.15
athelia	0.16	0.62	0.71	0.43	0.02	0.20	0.06	1.0	0.42	0.17	0.38	0.12	0.00	0.07	0.07	0.05	0.42	0.06	0.03	0.13
bfs.270a	0.04	0.69	0.21	0.36	0.02	0.27	0.01	0.42	1.0	0.08	0.21	0.07	0.10	0.04	0.06	0.03	0.13	0.06	0.03	0.14
biblioteca-nacional	0.04	0.12	0.14	0.09	0.01	0.13	0.01	0.17	0.08	1.0	0.12	0.07	0.00	0.04	0.09	0.03	0.22	0.06	0.03	0.21
brown	0.04	0.38	0.21	0.16	0.01	0.20	0.02	0.38	0.21	0.12	1.0	0.15	0.00	0.04	0.06	0.03	0.18	0.06	0.03	0.04
dblp	0.04	0.12	0.14	0.22	0.01	0.10	0.10	0.12	0.07	0.07	0.15	1.0	0.00	0.07	0.07	0.51	0.10	0.10	0.04	0.05
dbpedia	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	1.0	0.00	0.00	0.01	0.00	0.00	0.00	0.19
geonames	0.04	0.04	0.07	0.05	0.00	0.13	0.01	0.07	0.04	0.04	0.04	0.07	0.00	1.0	0.11	0.02	0.07	0.06	0.01	0.01
linkedgedata	0.04	0.08	0.14	0.05	0.02	0.20	0.01	0.07	0.06	0.09	0.06	0.07	0.00	0.11	1.0	0.02	0.01	0.10	0.04	0.20
swdf	0.08	0.27	0.43	0.07	0.02	0.02	0.04	0.05	0.03	0.03	0.03	0.51	0.01	0.02	0.02	1.0	0.06	0.13	0.04	0.13
vivosearch	0.20	0.19	0.93	0.32	0.05	0.40	0.11	0.42	0.13	0.22	0.18	0.10	0.00	0.07	0.01	0.06	1.0	0.10	0.04	0.09
wiktionary	0.04	0.08	0.14	0.10	0.01	0.13	0.01	0.06	0.06	0.06	0.06	0.10	0.00	0.06	0.10	0.13	0.10	1.0	0.04	0.00
wordnet	0.04	0.03	0.03	0.06	0.01	0.04	0.06	0.03	0.03	0.03	0.03	0.04	0.00	0.01	0.04	0.04	0.04	0.04	1.0	0.05
yago	0.04	0.08	0.36	0.17	0.01	0.27	0.05	0.13	0.04	0.13	0.06	0.12	0.00	0.04	0.00	0.02	0.04	0.10	0.04	1.0

Table 5: A sample of how much a dataset is contained in each other, in which the level of containment is also represented by the intensity of the color, as more intense color, as more contained are the datasets.

we sorted those datasets in a way of Top 10 datasets more similar to DBpedia, according to the properties and classes they share. On the table 6 we show the number of properties that they share that contains the exact same URI and cases where they share not the exact URI but very similar<sup>25</sup>. Thus, an example of application could be with a case when the user wants to identify information from other

<sup>25</sup> With similarity level greater than 0.8 according our similarity algorithm.

datasets to complement or enrich information for a given dataset, i.e. facilitating federated queries. The table 7 shows a evaluation with 600 randomly chosen datasets<sup>26</sup> including 3 synthetic manually made datasets.

Dataset	#ExactMatch	#sim > 0.8	#PropClass
swdf	22	64	288
yago	6	65	373546
dblp	6	9	41
linkedgedata	5	617	11799
wiktionary	4	6	31
geonames	4	12	27
wordnet	3	26	69
wikidata	0	5	427
freebase	0	55	17587

Table 6: Top 10 datasets containing exact the same URI and containing the most similar URIs according to our similarity approach, in which #PropClass represents the total number of properties and classes from the dataset.

With our experiments we realize that more than 50% of properties and classes has a match in another dataset. The information can also be observed at figure 11(a), with 600 datasets from LODLaundromat[17], in which highlight the high possibility that one dataset can enrich each other with complementing information from another dataset.

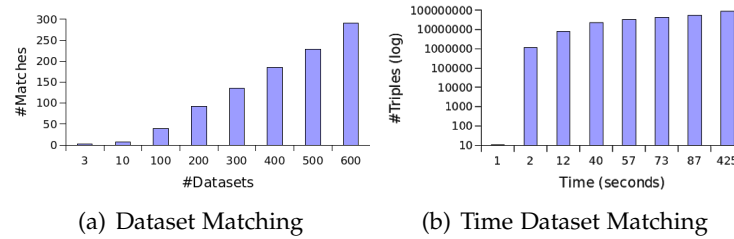


Figure 11: Dataset matching.

The time consumed can be observed on figure 11(b), with 600 datasets from LODLaundromat[17], which shows that 100 million triples were processed in less than 8 minutes, in which is a acceptable time, in which the starts to be time-consuming only after 1 million triples.

As we are using WimuQ[?] to identify datasets for a given SPARQL query together with our matching algorithm, we can answer

<sup>26</sup> European statistics: HDT files from LOD Laundromat: Economic accounts for agriculture (aact) - [http://ec.europa.eu/eurostat/cache/metadata/en/aact\\_esms.htm](http://ec.europa.eu/eurostat/cache/metadata/en/aact_esms.htm)



the question (2) based on famous Sparql queries from two real-data federated SPARQL querying benchmarks *LargeRDFBench* [85], *FedBench* [98] and one non-federated real data SPARQL benchmark selected from *FEASIBLE* [90]. Thus, from the selected datasets we use our approach to select the datasets according to the properties and classes they share among them<sup>27</sup>.

To answer the question (3) we evaluate the dataset duplicated detection algorithm and the detection of dataset chunks. From a 5446 datasets<sup>28</sup> our algorithm detected 2272 duplicates and 1470 chunks in 3 hours.

The figure 12(a) shows a case with 900 datasets chosen randomly, where we identified in 10 cases in which we can see the difference

The figure 12(b) show how many chunks we were able to identify among our sample data, in which lead us to know how segregated is data analysed, giving also the chance to know the complete dataset after the union of all chunks.

The current version of the index prototype has information about 539 SPARQL endpoints from LOD cloud<sup>29</sup> and 915 HDT files from LOD Laundromat<sup>30</sup>. We perform more than 1800000 comparisons in 88 hours.

Where **DsPropMatch** on the table 7 refers to the number of properties/classes the datasets share among each other.

#Datasets	#DsropMatch	time (seconds)	#triples	Synthetic
3	2	1	11	Yes
10	7	2	1198508	no
100	39	12	7996408	no
200	92	40	22982984	no
300	135	57	34792121	no
400	186	73	43864522	no
500	229	87	54227780	no
600	291	425	85041239	no

Table 7: Evaluation on the Match algorithm, where **DsPropMatch** refers to the number of properties/classes the datasets share among each other.

We can observe the quantity of properties/classes that the datasets share related to the number of triples analysed. For instance, from 600 datasets, 291 matches were found, in which the number of triples

<sup>27</sup> In this case, all datasets identified by wimuQ are sharing properties and classes, that is why we are using the graph from [? ].

<sup>28</sup> A subset from those 600 datasets chosen previously

<sup>29</sup> The list of is available here: [https://github.com/firmao/wimuT/blob/master/Endpoints\\_numtriples\\_lodcloud.csv](https://github.com/firmao/wimuT/blob/master/Endpoints_numtriples_lodcloud.csv)

<sup>30</sup> fdsa

was almost the double size of the previous case. Thus, the number of matches in this case cannot be directly related to the number of triples. Due to this fact, the quality of the datasets should be considerate a important phase.

We evaluate the accuracy of our matching algorithm with a small sample, where we can see at figure 12(c), 3 and 12(d) the F-Measure and run-time on six famous pairs of datasets from[37], with those datasets we create a gold standard to compare, in which  $P_1 \dots P_n$  represents the pair of datasets and  $P_1, P_2$  are datasets synthetically generated by the author, i.e.  $P_3$  represents the comparison between the datasets Abt and Buy.

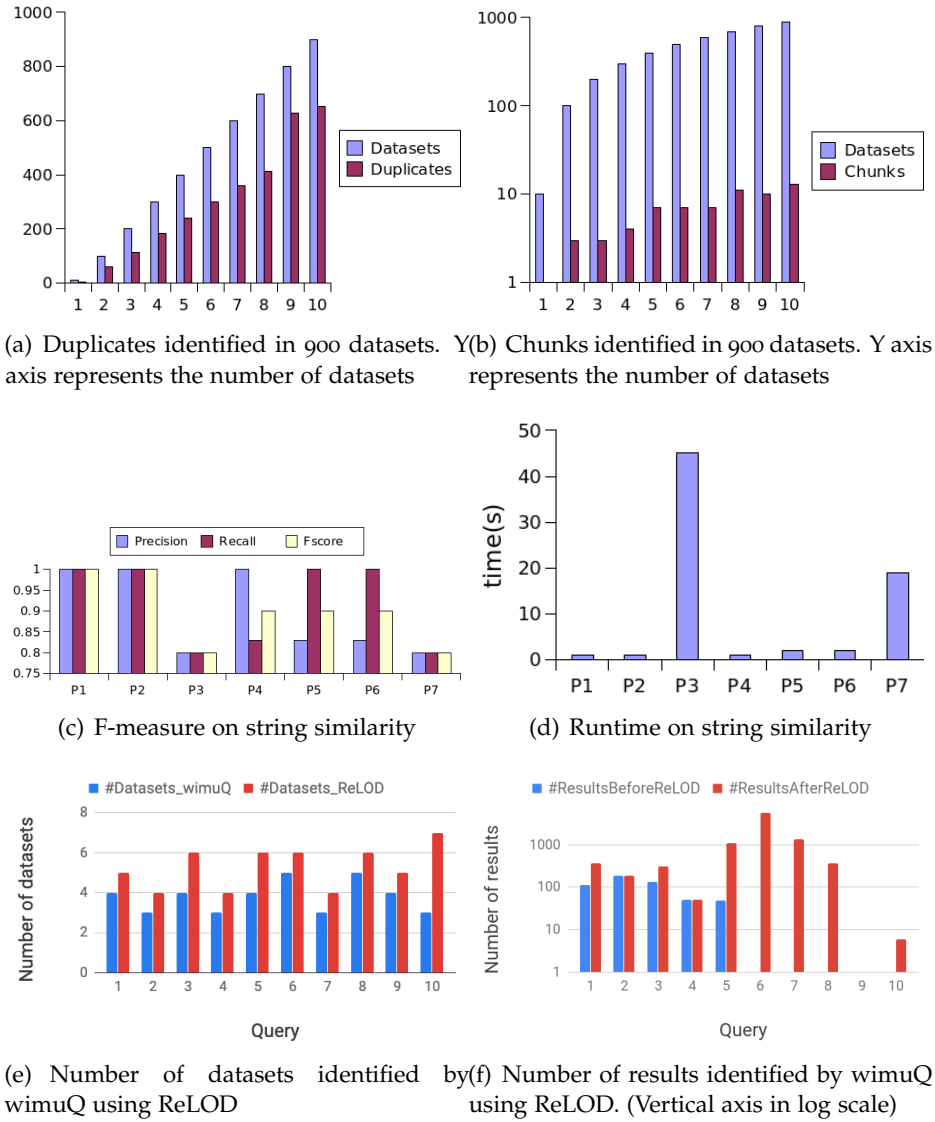


Figure 12: Improvements of ReLOD.

The figure 7 shows 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 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.

To answer the question (4) we selected 10 queries<sup>31</sup> from FedBench[98], where we can observe on the figure 12(e) that thanks to the ReLOD approach we could increase the number of datasets identified by wimuQ, i.e., in query 5, using ReLOD allowed us to find 2 more datasets containing complementary information.

On the other hand, the figure 12(f) reinforces that more datasets do not always imply in more results, i.e., in query 2 and 4, more datasets were identified, but the number of results did not change, in this case, the reason was that the datasets identified by ReLOD were practically the same with different property and class names. The results from queries 6 to 10 were only found thanks to the ReLOD approach<sup>32</sup>.

<sup>31</sup> The queries are available here: [https://github.com/firmao/LDatasetGenerator/blob/master/10\\_Queries\\_fedbench.txt](https://github.com/firmao/LDatasetGenerator/blob/master/10_Queries_fedbench.txt)

<sup>32</sup> The query number 9 obtained only one result with the new approach.

## OBTAINING SIMILAR RESOURCES USING STRING SIMILARITY

---

*To relate the datasets it is necessary to identify links among them. In this chapter, we discuss how linking entries across heterogeneous data sources such as databases or knowledge bases, which becomes an increasingly difficult problem, in particular w.r.t. the runtime of these tasks. Consequently, it is of utmost importance to provide time-efficient approaches for similarity joins in the Web of Data. While a number of scalable approaches have been developed for various measures, the Most Frequent  $k$  Characters (MFKC) measure has not been tackled in previous works. We hence present a sequence of filters that allow discarding comparisons when executing bounded similarity computations without losing recall. Therewith, we can reduce the runtime of bounded similarity computations by approximately 70%. Our experiments with a single-threaded, a parallel and a GPU implementation of our filters suggest that our approach scales well even when dealing with millions of potential comparisons.*

### 6.1 INTRODUCTION

The problem of managing heterogeneity at both the semantic and syntactic levels among various information resources [Valdestilhas et al.; 103] is one of the most difficult problems on the information age. This is substantiated by most of the database research self-assessment reports, which acknowledge that the hard question of semantic heterogeneity, that is of handling variations in meaning or ambiguity in entity interpretation, remains open [103]. In knowledge bases, Ontology Matching (OM) solutions address the semantic heterogeneity problem in two steps: (1) matching entities to determine an alignment, i.e., a set of correspondences, and (2) interpreting an alignment according to application needs, such as data translation or query answering. Record Linkage (RL) and, more recently, Link Discovery<sup>1</sup> (LD) solutions on the other hand aim to determine pairs of entries that abide by a given relation  $R$ . In both cases, string similarities are used to compute candidates for alignments. In addition to being central to RL and LD, these similarities also play a key role in several other tasks such as data translation, ontology merging and navigation on the Web of Data [103; 33].

---

<sup>1</sup> The expression "link discovery" in this paper means the discovery of typed relations that link instances from knowledge bases on the Web of Data. We never use it in the sense of graph theory.

One of the core tasks when developing time-efficient RL and LD solutions hence lies in the development of time-efficient string similarities. In this paper, we study the MFKC similarity function [100] and present an approach for improving the performance of similarity joins. To this end, we develop a series of filters which guarantee that particular pairs of resources do not abide by their respective similarity threshold by virtue of their properties.

The contributions of this paper are as follows:

1. We present two nested filters, (1) First Frequency Filter and (2) Hash Intersection filter, that allow to discard candidates before calculating the actual similarity value, thus giving a considerable performance gain.
2. We present the *k similarity filter* that allows detecting whether two strings  $s$  and  $t$  are similar in a fewer number of steps.
3. We evaluate our approach with respect to its runtime and its scalability with several threshold settings and dataset sizes.
4. We present several parallel implementations of our approach and show that they work well on problems where  $|D_s \times D_t| \geq 10^5$  pairs.

The rest of the chapter is structured as follows: In section 6.2, we present our nested filters that allow reducing the runtime of the MFKC similarity joins. We then evaluate our approach in section A.1.5, where we focus on approaches that aim to improve the time-efficiency of the link discovery task.

## 6.2 APPROACH

Let us call NaiveMFKC the function which computes the MFKC algorithm as described in [100]. Such function works with three parameters, i.e. two strings  $s$  and  $t$  and an integer  $\text{lim}$  and returns the sum of frequencies, where  $f(c_i, s)$  is a function that returns the frequency of the character  $c_i$  in the string  $s$  and  $s \supseteq \{c_1, \dots, c_n\}$ , i.e.  $f(a, \text{"andrea"}) = 2$ , because the character  $a$  has been found twice and the hash functions  $h(s)$  and  $h(t)$  containing the characters and their frequencies. The output of function is always positive, as shown in equation 3.

$$\text{NaiveMFKC}(s, t, \text{lim}) = \text{lim} - \sum_{c_i \in h(s) \cap h(t)}^2 f(c_i, s) + f(c_i, t) \quad (3)$$

Our work aims to reduce the runtime of computation of the MFKC similarity function. Here, we use a sequence of filters, which allow

discarding similarity computations and imply in a reduction of runtime. As input, the algorithm receives datasets  $D_s$  and  $D_t$ , an integer number representing the  $k$  most frequent characters and a threshold  $\theta \in [0, 1]$ . The similarity score of the pair of strings from the Cartesian product from  $D_s$  and  $D_t$  must have a score greater or equal the threshold  $\theta$  to be considered a good pair, i.e. for a given threshold  $\theta$ , if the similarity function has a pair of strings with similarity score less than the threshold,  $\sigma(s, t) < \theta$ , we can discard the computation of the MFKC score for this pair. Our final result is a set which contains the pairs having similarity score greater than or equal to the threshold, i.e.  $\sigma(s, t) \geq \theta$ .

Our work studies the following problem: Given a threshold  $\theta \in [0, 1]$  and two sets of strings  $D_s$  and  $D_t$ , compute the set  $M' = \{(s, t, \sigma(s, t)) \in D_s \times D_t \times \mathbb{R}^+ : \sigma(s, t) \geq \theta\}$ . Two categories of approaches can be considered to improve the runtime of measures: Lossy approaches return a subset  $M''$  of  $M'$  which can be calculated efficiently but for which there are no guarantees that  $M'' = M'$ . Lossless approaches, on the other hand, ensure that their result set  $M''$  is exactly the same as  $M'$ . In this paper, we present a lossless approach that targets the MFKC algorithm. Equation 4 shows our definition for the string similarity function  $\sigma$  for the MFKC.

$$\sigma(s, t) = \frac{\sum_{c_i \in h(s, k) \cap h(t, k)} f(c_i, s) + f(c_i, t)}{|s| + |t|} \quad (4)$$

where  $s$  and  $t$  are strings, such that  $s, t \in \Sigma^*$ ,  $f(c_i, s)$  is a function that returns the frequency of the character  $c_i$  in the string  $s$ , where  $s \supseteq \{c_1, \dots, c_n\}$ ,  $k$  represents the limitation of the elements that belongs to the hashes; set  $h(s, k) \cap h(t, k)$  means the intersection between the keys of hashes  $h(s, k)$  and  $h(t, k)$  (i.e., the most frequent  $K$  characters). We expect two steps to obtain the similarity score:

1. Firstly, we transform the strings  $s$  and  $t$  in two hashes using Most Frequent Character Hashing [100], according to the following example with  $k = 3$ :  
 $s = aabbbcc \rightarrow h(s, k) = \{b = 3, a = 2, c = 2\}$   
 $t = bbccdde \rightarrow h(t, k) = \{b = 2, c = 2, d = 2\}$
2. We calculate the sum of the character frequencies of matching characters on the hashes  $h(s, k)$  and  $h(t, k)$ , then, we normalize dividing by the sum of the length of  $|s|$  and  $|t|$  resulting in a similarity score from 0 to 1 according to the equation 5 and the resulting score should be greater or equals the threshold  $\theta$ .

$$\sigma(s, t, k, \theta) = \frac{\sum_{c_i \in h(s, k) \cap h(t, k)} f(c_i, s) + f(c_i, t)}{|s| + |t|} \geq \theta \quad (5)$$

### 6.2.1 Improving the Runtime

In this section, the runtime of MFKC defined in equation 4 is improved using filters where  $\mathcal{N}$  is the output of first frequency filter,  $\mathcal{L}$  is the output of hash intersection filter and  $\mathcal{A}$  represents the output of the  $k$  similarity filter.

#### First Frequency Filter

As specified in the definition of MFKC [100] this filter assumes that the hashes are already sorted in an descending way according to the frequencies of characters, therefore the first element of each hash has the highest frequency.

Showing that:

$$\sigma(s, t) = \frac{\sum_{c_i \in h(s, k) \cap h(t, k)} f(c_i, s) + f(c_i, t)}{|s| + |t|} \leq \frac{h_1(s, k)k + |t|}{|s| + |t|} \quad (6)$$

implies that  $\sigma(s, t) < \theta$ .

*Theorem 6.2.1.* Let the intersection between hashes  $h(t, k)$  and  $h(s, k)$  be a set of characters from  $c_1$  to  $c_n$ , such that equation 7:

$$h(t, k) \cap h(s, k) = \{c_1, \dots, c_n\} \quad (7)$$

According to the definition of the frequencies  $f((c_i, t))$  we have equation 8:

$$t \supseteq \{c_1, \dots, c_1, \dots, c_n, \dots, c_n\} \quad (8)$$

where each  $c_i$  appears  $f((c_i, t))$  times, therefore:

$$f((c_1, t)) + \dots + f((c_n, t)) \leq |t| \quad (9)$$

Also, as  $n \leq k$ , because  $t \supseteq \{c_1, \dots, c_n\}$ , and  $f((c_i, s)) \leq h_1(s, k) \forall i=1, \dots, n$ , then:

$$f((c_1, s)) + \dots + f((c_n, s)) \leq h_1(s, k) + \dots + h_1(s, k) = n(k) \leq k(h_1(s, k)) \quad (10)$$

Therefore, from equation 9 and equation 10, we obtain the equation 11:

$$\frac{\sum_{c_i \in h(s, k) \cap h(t, k)} f(c_i, s) + f(c_i, t)}{|s| + |t|} = \frac{\sum_{i=1}^n f((c_i, t)) + \sum_{i=1}^n f((c_i, s))}{|s| + |t|} \leq \frac{h_1(s, k)k + |t|}{|s| + |t|} \quad (11)$$

Consequently, the rule which the filter relies on is the following.

$$\langle s, t \rangle \notin \mathcal{N} \Rightarrow \langle s, t \rangle \notin D_s \times D_t \wedge \frac{h_1(s, k)k + |t|}{|s| + |t|} \leq \theta \quad (12)$$

□

### Hash Intersection Filter

In this filter, we check if the intersection between two hashes is an empty set, then the MFKC, represented by  $\sigma$ , will return a similarity score of 0 and we can avoid the computation of similarity in this case. Consequently, the rule which the filter relies on is the following.

$$\langle s, t \rangle \in \mathcal{L} \Rightarrow \langle s, t \rangle \in D_s \times D_t \wedge |h(s) \cap h(t)| > 0 \quad (13)$$

we also can say that the equation 14 represents a valid implication.

$$h(s) \cap h(t) = \emptyset \Rightarrow \sigma(s, t) = 0 \quad (14)$$

The equation 14 means that if the intersection between  $h(s, k)$  and  $h(t, k)$  is a empty set, this implies that the similarity score will be 0. That means there is no character matching, then there is no need to compute the similarity for this pair of strings.

### K Similarity filter

For all the pairs left, the similarity score among them is calculated. After that, the third filter selects the pairs whose similarity score is greater or equal than a threshold  $\theta$ .

$$\langle s, t \rangle \in \mathcal{A} \Leftrightarrow \langle s, t \rangle \in \mathcal{N} \wedge \sigma(s, t) \geq \theta \quad (15)$$

This filter provides a validation and we show that the score of previous  $k$  similarity is always lower than the next  $k$ , according to the equation 16 and in some cases when the similarity score is been reached before compute all elements  $\in h(s, k) \cap h(t, k)$ , thus saving computation in these cases.

Here  $k$  is also used as a index of similarity function  $\sigma_k(s, t)$  in order to get the similarity of all cases of  $k$ , from 1 to  $k$ , also to show the monotonicity.

Therefore we can say that the computation of similarity score occurs until  $\sigma_k(s, t) \geq \theta$ .

We will demonstrate that the similarity score of previous  $k$  similarity is always lower than the next  $k$  similarity, for all  $k \in \mathbb{Z}^* : k \leq |s \cap t|$ .

$$\sigma_{k+1}(s, t) \geq \sigma_k(s, t) \quad (16)$$

We rewrote the equation for the first iteration, according to equation 17

$$\sigma_k(s, t) = \frac{\sum_{c_i \in h(s, k) \cap h(t, k), i=1}^k f(c_i, s) + f(c_i, t)}{|s| + |t|} \quad (17)$$



Let  $k \in \mathbb{Z}_+$  be given and suppose the equation 16 is true for  $k + 1$ .

$$\sum_{c_i \in h(s,k) \cap h(t,k)}^k \left[ f(c_i, s) + f(c_i, t) \right] + f(c_{k+1}, s) + f(c_{k+1}, t) \geq \sum_{c_i \in h(s,k) \cap h(t,k)}^k f(c_i, s) + f(c_i, t) \quad (18)$$

Therefore, we can notice that the sum of frequencies will be always greater or equal 0, according to  $f(c_{k+1}, s) + f(c_{k+1}, t) \geq 0$ . Thus, equation 16 holds true.

#### Filter sequence

The sequence of the filters occurs basically in 4 steps, (1) We starting to make the Cartesian product with the pairs of strings from the datasets  $D_s$  and  $D_t$ , (2) Discarding pairs using the First Frequency Filter( $\mathcal{N}$ ), (3) Discarding pairs where there is no matching characters with the Hash Intersection filter( $\mathcal{L}$ ) and (4) With the Most Frequent Character Filter ( $\mathcal{A}$ ) we will process only similarities greater or equal the threshold  $\theta$ , if the similarity function  $\sigma(s, t) \geq \theta$  in the first  $k$  characters we can stop the computation of the similarity of this pair, saving computation and add to our dataset with resulting pairs  $D_r$ , also shown in the algorithm 3.

### 6.3 CORRECTNESS AND COMPLETENESS

In this section, we prove formally that our MFKC is both correct and complete.

- We say that an approach is correct if the output  $O$  it returns is such that  $O \subseteq R(D_s, D_t, \sigma, \theta)$ .
- Approaches are said to be complete if their output  $O$  is a superset of  $R(D_s, D_t, \sigma, \theta)$ , i.e.,  $O \supseteq R(D_s, D_t, \sigma, \theta)$ .

Our MFKC consists of three nested filters, each of which creates a subset of pairs, i.e.  $\mathcal{A} \subseteq \mathcal{L} \subseteq \mathcal{N} \subseteq D_s \times D_t$ . For the purpose of clearness, we name each filtering rule:

$$R_1 \triangleq \frac{h_1(s, k)k + |t|}{|s| + |t|} < \theta$$

$$R_2 \triangleq |h(s, k) \cap h(t, k)| \neq 0$$

$$R_3 \triangleq \sigma(s, t) \geq \theta$$

Each subset of our MFKC can be redefined as  $\mathcal{N} = \{\langle s, t \rangle \notin D_s \times D_t : R_1, \mathcal{L} = \{\langle s, t \rangle \in D_s \times D_t : R_1 \wedge R_2, \text{ and } \mathcal{A} = \{\langle s, t \rangle \in D_s \times$

**Algorithm 3** MFKC Similarity Joins

---

```

1: GoodPairs = {s1; t1, ..., sn; tn} | (s, t) ∈  $\sum^*$ 
10: if |hs ∩ ht| = 0 then
2:   hs = {e1, e2, ..., en} | ei = <c, f(c, s)>
   Next t ∈ Dt
3:   ht = {e1, e2, ..., en} | ei = <c, f(c, t)>
11:   forall cfreq ∈ hs ∩ ht do
4:     i, freq ∈ ℕ*
   end
   i ≥ k
5:   procedure MFKC(Ds, Dt, θ, k)
12:     Next t ∈ Dt
   forall s ∈ Ds do
13:     freq = freq + cfreq
   end
14:     freq = freq + cfreq
   (in parallel)
15:     σi =  $\frac{\text{freq}}{|s|+|t|}$  if σi ≥ θ then
6:     hs = h(s, k) forall t ∈ Dt
16:     end
   do
   GoodPairs.add(s, t)
   end
17:   Next t ∈ Dt
   (in parallel)
18:   ht = h(t, k)
19:   i = i + 1
20:   if  $\frac{h_{s_1}(k) + |t|}{|s| + |t|} < \theta$  then
21:   end
22:   Next t ∈ Dt
7:   ht = h(t, k)
8:   end
23: return GoodPairs
9:   Next t ∈ Dt
end procedure

```

---

$D_t : R_1 \wedge R_2 \wedge R_3$ . We then introduce  $\mathcal{A}^*$  as the set of pairs whose similarity score is more or equal than the threshold  $\theta$ .

$$\mathcal{A}^* = \{(s, t) \in D_s \times D_t : \sigma(s, t) \geq \theta\} = \{(s, t) \in D_s \times D_t : R_3\} \quad (19)$$

Our MFKC filtering algorithm is correct and complete.

*Theorem 6.3.* Proving theorem 6.3 is equivalent to showing that  $\mathcal{A} = \mathcal{A}^*$ . Let us consider all the pairs in  $\mathcal{A}$ . While our MFKC's correctness follows directly from the definition of  $\mathcal{A}$ , it is complete iff none of pairs discarded by the filters actually belongs to  $\mathcal{A}^*$ . Assuming that the hashes are sorted in a descending way according the frequencies of the characters, therefore the first element of each hash has the highest frequency. Therefore, once we have

$$\frac{h_1(s, k)k + |t|}{|s| + |t|} < \theta,$$

the pair of strings  $s$  and  $t$  can be discarded without calculating the entire similarity. When rule  $R_3$  applies, we have  $\sigma(s, t) < \theta$ , which leads to  $R_3 \Rightarrow R_1$ . Thus, set  $\mathcal{A}$  can be rewritten as:

$$\mathcal{A} = \{(s, t) \in D_s \times D_t : R_2 \wedge R_3\} \quad (20)$$

We are given two strings  $s$  and  $t$  and the respective hashes  $h(s, k)$  and  $h(t, k)$ , the intersection between the characters of these two hashes is a empty set. Therefore, there is no character matching, which implies that  $s$  and  $t$  cannot be considered to have a similarity score greater than or equal to threshold  $\theta$ :

$$h(s, k) \cap h(t, k) = \emptyset \Rightarrow \sigma(s, t) = 0$$

When the rule  $R_3$  applies, we have  $\sigma(s, t) < \theta$ , which leads to  $R_3 \Rightarrow R_2$ . Thus, set  $\mathcal{A}$  can be rewritten as:

$$\mathcal{A} = \{\langle s, t \rangle \in D_s \times D_t : R_3\} \quad (21)$$

which is the definition of  $\mathcal{A}^*$  in equation 19. Therefore,  $\mathcal{A} = \mathcal{A}^*$ .  $\square$

#### *Time complexity*

In order to calculate the time complexity of our MFKC, firstly we considered the most frequent  $K$  characters from a string. The first step is to sort the string lexically. Then, we can reach a linear complexity after this sort, because the input with highest occurrences can be achieved with a linear time complexity. The first string can be sorted in  $O(n \log n)$  and second string in  $O(m \log m)$  times, as some classical sorting algorithms such as merge sort [38] and quick sort [50] that work in  $O(n \log n)$  complexity. Thus, the total complexity is  $O(n \log n) + O(m \log m)$ , resulting in  $O(n \log n)$  as upper bound in the worst case.

## 6.4 EVALUATION

The aim of our evaluation is to show that our work outperforms the naive approach and the parallel implementation has a performance gain in large datasets with size greater than  $10^5$  pairs. A considerable number of pairs reach the threshold  $\theta$  before reaching the last  $k$  most frequent character, that also is a demonstration about how much computation was avoided. Instead of all  $k$ 's we just need  $k - n$  where  $n$  is the  $n^{\text{th}}$  most frequent character necessary to reach the threshold  $\theta$ . An example to show the efficiency of each filter can be found at figure 13, where 10,273,950 comparisons from DBpedia+LinkedGeoData were performed and Performance Gain (PG) =  $\text{Recall}(\mathcal{N}) + \text{Recall}(\mathcal{L})$ . The recall can be seen in figure 14(c). This evaluation has the intention to show results of experiments on data from DBpedia<sup>2</sup> and LinkedGeoData<sup>3</sup>. We considered pairs of labels in order to do the evaluation. We have two motivations to chose these datasets: (1) they have been widely used in experiments pertaining to Link Discovery (2) the distributions of string sizes between these

<sup>2</sup> <http://wiki.dbpedia.org/>

<sup>3</sup> <http://linkedgeodata.org/>

datasets are significantly different [29]. All runtime and scalability experiments were performed on a Intel Core i7 machine with 8GB RAM, a video card NVIDIA NVS4200 and running Ms Windows 10.

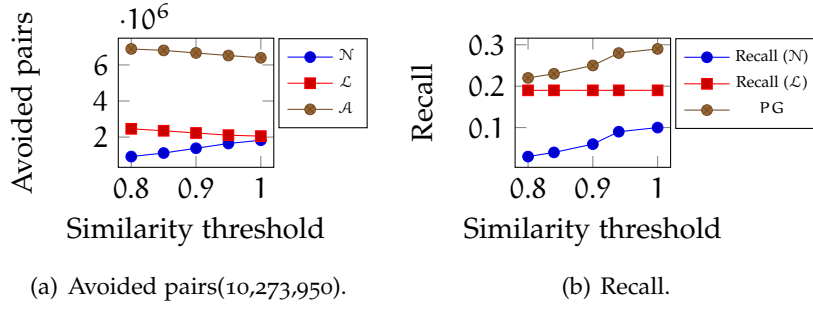


Figure 13: Avoided pairs and recall.

#### 6.4.1 Parallel implementation

Our algorithm contains parallel code snippets with which we perform a load and balance of the data among CPU/GPU cores when available. To illustrate this part of our idea, we can state: Given a two datasets  $S, T$ , that contains all the strings to be compared. Thus, make a Cartesian product of the strings  $S \times T$ , where each pair is the processed separately in threads that are spread among CPU/GPU cores. Thus, we process the each comparison in parallel. The parallel implementation works better in large datasets with size more than  $10^5$ , that was more than one time faster than the approach without parallelism and two times faster than the naive approach as shown in figures 15(a), 15(b) and 15(c).

#### 6.4.2 Runtime Evaluation

The evaluation in figures 15(a) and 15(b) shows that all filter setup outperform the naive approach, and the parallel approach does not suffer significant changes related to the runtime according to the size of the dataset, as show figure 15(c). The experiments related to the variance of  $k$ , also were considered, as show in figure 15(d), the runtime varies according the size of  $k$ , indicating the influence of  $k$  with values from 1 to 120 with 1,001,642 comparisons. The performance (run-time) was improved as shown in figures 15(a), 15(b) and according the recall with a performance gain of 26.07% as shown in figure 13. The time complexity is based on two sort process  $O(n \log n) + O(m \log m)$  resulting in  $O(n \log n)$  as a upper bound in the worst case.

### 6.4.3 Scalability Evaluation

In the experiments (see figures 15(c), 15(e) and 15(f)), we looked at the growth of the runtime of our approach on datasets of growing sizes. The results show that the combination of filters ( $\mathcal{N} + \mathcal{L} + \mathcal{A}$ ) is the best option for datasets of large sizes. This result holds on both DBpedia and LinkedGeoData, so our approach can be used on large datasets and achieves acceptable run-times. We also can realize the quantity of avoided pairs in each combination of filters in figure 13, that consequently brings a performance gain. We looked at experiments with runtime behavior on a large dataset with more than  $10^6$  labels as shown in figure 15(b). The results suggest that the runtime decreases according to the threshold  $\theta$  increment. Thus, one more point showing that our approach is useful on large datasets, where can be used with high threshold values for link discovery area. About our parallel implementation, figure 15(c) shows that our GPU parallel implementation works better on large datasets with size greater than  $10^5$ .

### 6.4.4 Comparison with existing approaches

Our work overcomes the naive approach [100], thus, in order to show some important points we compare our work not only with the state of the art, but with popular algorithms such as Jaccard Index [51]. As shown in figures 15(c), 15(e) and 15(f), our approach outperforms not only the naive approach, but also Jaccard Index. We show that the threshold  $\theta$  and  $k$  have a significant influence related to the runtime. The naive approach present some points to consider, among them, even if the naive approach states that they did experiments with  $k=7$ , the naive algorithm was designed for only  $k=2$ , there are some cases where  $k = 2$  is not enough to get the similarity level expected, i.e.  $s = \text{mystring1}$  and  $t = \text{mystring2}$  limiting  $k = 2$ , we will have  $\sigma_2(s, t) = 0.2$ , showing that the similarity is very low, but when  $k = 8$ , the similarity is  $\sigma_8(s, t) = 0.8$  showing that sometimes we can lose a good similarity case limiting  $k = 2$ . Our work fix all these problems and also has a better runtime, as show figure 15(a), figure 15(b) and figure 15(c).

An experiment with labels from DBpedia and Yago<sup>4</sup> shows that the f-score indicates a significant potential to be used with success as a string similarity comparing with Jaccard Index and Jaro Winkler, as figures 14(a), 14(b) and 14(c) shows. To summarize the key features that makes our approach outperform the naive approach are the following: We use more than two K most frequent characters in our evaluation, our run-time for more than  $10^7$  comparisons is shorter

<sup>4</sup> <http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/>

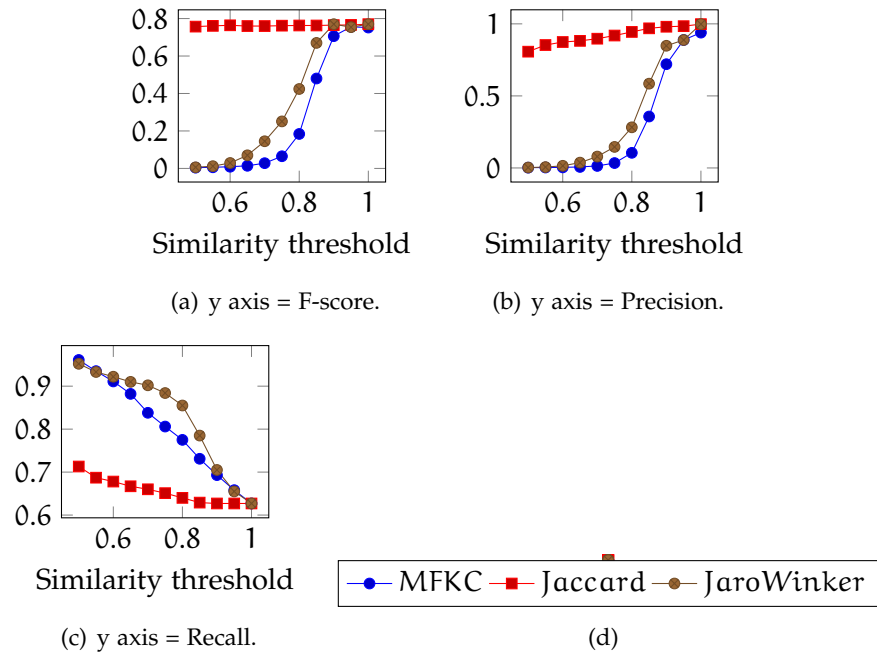
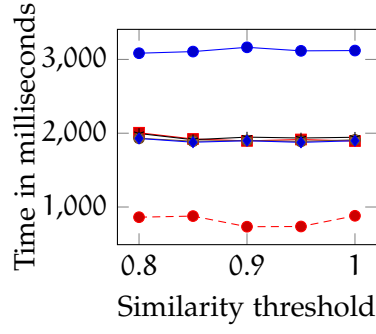
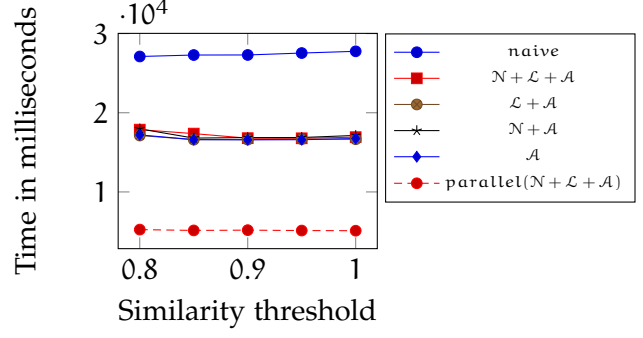


Figure 14: Precision, Recall and F-Measure.

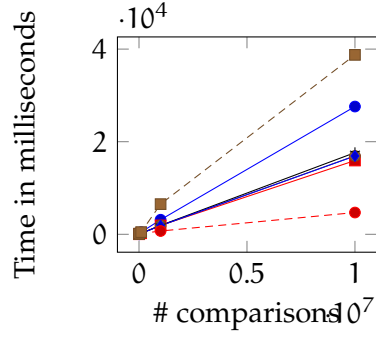
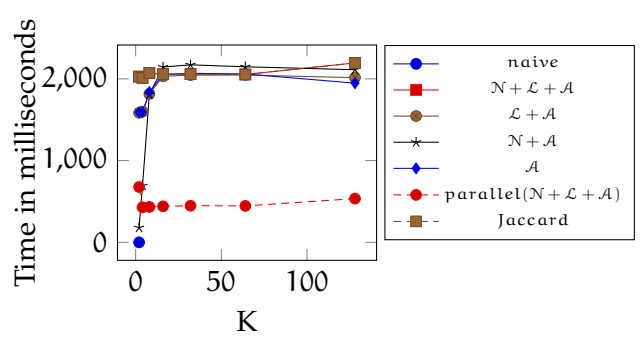
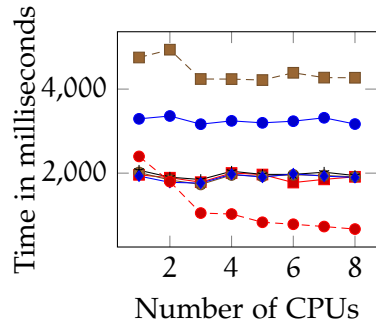
(27,594 against 16,916) milliseconds, we do have a similarity threshold, allowing us to discard comparisons avoiding extra processing, and a parallel implementation, making our approach scalable. Jaccard does not show significant changes varying the threshold, MFKC and Jaro Winkler present a very similar increase of the f-score varying the threshold.



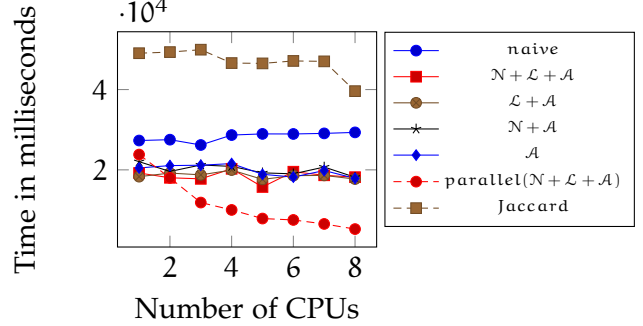
(a) Runtime 1,001,642 comparisons.



(b) Runtime 10,273,950 comparisons.

(c) The parallel approach improves the performance for more than 1 to 120, over 1,001,642 comparisons,  $\theta = 0.95$ .  $10^5$  comparisons.(d) Runtime k most frequent characters, with values of k from the performance for more than 1 to 120, over 1,001,642 comparisons,  $\theta = 0.95$ .  $10^5$  comparisons.

(e) CPU Speedup of algorithm (1,001,642 comparisons).



(f) CPU Speedup of algorithm (10,273,950 comparisons).

Figure 15: Run-time experiments results.

## HETEROGENEITY IN DBPEDIA IDENTIFIERS

---

*In this chapter, we have a practical case with DBpedia improving the quality of the identifiers in order to better relate the datasets. We approach a problem where the DBpedia dataset has multiple URIs within the dataset and from other datasets connected with (transitive) `owl:sameAs` relations and thus referring to the same concepts. With this heterogeneity of identifiers it is complicated for users and agents to find the unique identifier which should be preferably used. We are introducing the concept of DBpedia Unique Identifier (DUI) and a dataset of linksets relating URIs to DUIs. In order to improve the quality of our dataset we developed a mechanism that allows the user to rate and suggest links. As proof of concept an implementation with a graphical web user interface is provided for accessing the linkset and rating the links. The DBpedia sameAs service is available at <http://dbpsa.aks.org/SameAsService>.*

### 7.1 INTRODUCTION

As DBpedia [56] was evolving during the 9 years of its existence, the community extended the linksets to DBpedia resources. Thus, DBpedia has more than one URI that represents the same resource, which leads to the identifier heterogeneity problem. For instance a DBpedia resource can contain `owl:sameAs` links to other data sets such as FreeBase<sup>1</sup>, Wikidata, GeoNames<sup>2</sup> or yago<sup>3</sup>.

Further DBpedia has more than one URI representing the same resource within the dataset, e.g. the `dbpedia:Brassil` has at least the following equivalents within the DBpedia `dbpedia:Republica_Federativa_do_Brasil`, `dbpedia:ISO_3166-1:BR` and `dbpedia:Brazil` which are all redirecting to `dbpedia:Brazil`. Thus a problem to consider is to directly resolve any of the equivalents directly to the final URI e.g. <http://dbpedia.org/resource/Brazil> without any redundancies.

Also, according to Halpin et. al. [41] and Wood et. al. [122], [sameas.org](http://sameas.org) has collected millions of triples with `owl:sameAs` relations. It would be important to promote reciprocal `owl:sameAs` confirmation mechanisms and develop effective trust mechanisms to assure the quality of `owl:sameAs` relations.

To tackle the identifier heterogeneity problem we are making the following contributions:

---

<sup>1</sup> Freebase project webpage: <http://freebase.com>

<sup>2</sup> GeoNames project and exploration webpage: <http://geonames.org>

<sup>3</sup> Yago project webpage: <http://yago-knowledge.org>



- We describe an approach for the mitigation of the identifier heterogeneity problem and implement a prototype where the user is able to evaluate existing links, as well as suggest new links to be rated.
- The ability to generate statistics about good and bad links which, brings the possibility to have a quality control for the links to DBpedia.
- We define the DBpedia Unique Identifier (DUI), which instead of several transient `owl:sameAs` DBpedia URIs for the same final address, now is possible to have a unique URI from DBpedia. A DUI goes directly to the final address instead of having to process several possible intermediate results. For example, with a URI from *Freebase*, 17 redundant URIs from DBpedia where avoided or if one used a service such as *sameAs.org*, 1141 URIs would be avoided.

The rest of the chapter is organized as follows: section 7.2 represents a proposed approach for tackling the identifier heterogeneity problem and section A.1.5 we evaluate our work.

## 7.2 REPRESENTATION OF THE IDEA

This section provides an explanation about our main idea, such as implementation and descriptions.

Before continuing the work, there are some definitions that were adopted.

- **Normalization of the URI:** Is understood by normalizing URIs, the fact of eliminating redundancies.
- **DBpedia unique identifier:** The DBpedia Unique Identifier (DUI) is an unique URI that identifies a resource in the DBpedia repository and also is the result of our normalization.

The idea started with a stand alone service on the web that solves the problem where the user provides a URI as parameter and instead of several transient URIs with `owl:sameAs` property, the user receives a single DUI from our service.

### 7.2.1 The work-flow

The work-flow for requesting the DUI of a given resource is represented in figure 16. Firstly, the user will provide a URI from some address, i.e. FreeBase. Then, instead of possible several results of URIs with the property `owl:sameAs`, our system will return a DUI. Consequently, the user has a possibility to rate, verify, validate, and

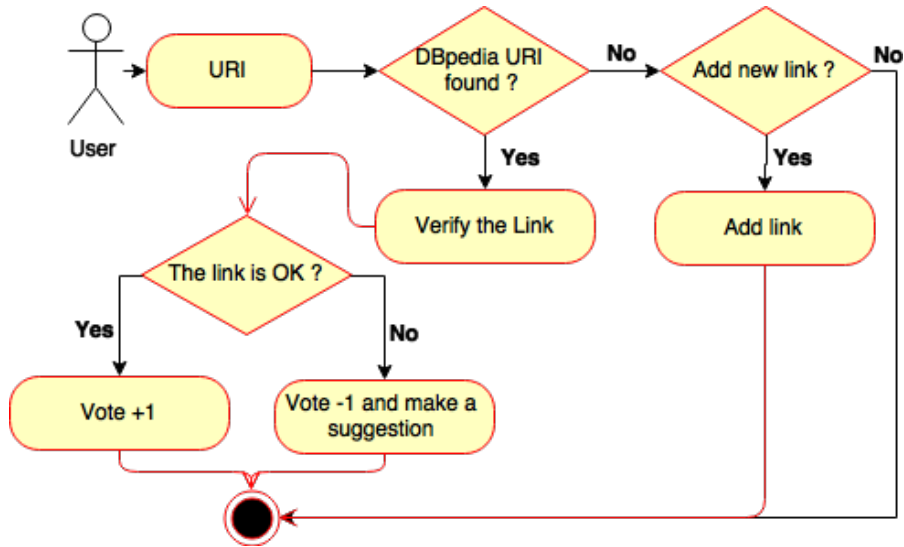


Figure 16: General work-flow.

suggest a different link. Then the rate can give us a chance to have statistics about the quality of the links.

A service, also was implemented, where the user can provide a URI and the API will return the DBpedia identifier like a URI that represents the `owl:sameAs` about the URI provided.

### 7.2.2 Methodology

This section describes in four steps the technique and how the idea was developed, from phase of importing links to a relational database until the development of the service on the web and a GUI.

(1) The files with triples that contains `owl:sameAs` links, were downloaded. (2) All triples were imported in a relational database<sup>4</sup>, because we will use some characteristics of a relational database i.e. comparative with voting system in future works. (3) An implementation of a service on the web was provided, where the user enters the URI and receives a DUI. (4) In order to provide an interface to access this service were created a web system that receive as input a URI, return as output an DBpedia identifier and allow rate and make suggestions about the resulting link.

figure 17 presents the relation of the contribution in a graph form.

Where the DBpedia Link Repository uses the DBpediaSameAs service in order to tackle the heterogeneity and giving the appropriate DUI, that redirects the user to the DBpedia Link Rate interface, thus, providing a feedback to the DBpedia Link Repository, therefore, improving the quality of the DBpedia endpoint.

<sup>4</sup> <http://tinyurl.com/creatdb>

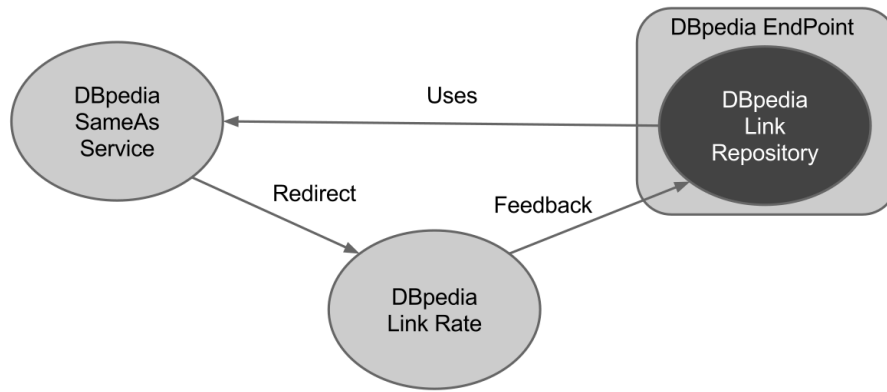


Figure 17: Relation of the contributions.

### 7.3 EVALUATION

The aim of this qualitative<sup>5</sup> evaluation was centered in verifying the behavior of the service DBpediaSameAs, the Graphical User Interface (GUI) that gives the possibility to verify and rate the links.

There are chosen 3 evaluation criteria:

(1) **Normalization on DBpedia URIs:** With this criteria was evaluated if the DBpediaSameAs can provide an normalization on DBpedia URIs. (2) **Rate the Links:** Where was evaluated if the DBpediaSameAs can provide a way to rate the links. (3) **DBpediaSameAs as service:** Was evaluated if DBpediaSameAs can provide a stand alone service on the web that brings the normalization on DBpedia URIs.

#### 7.3.1 Normalization on DBpedia URIs

The criteria used in this evaluation are uniquely to tackle heterogeneity, that was observed during the search of co-references between different data sets with a problem about redundancies.

When was used a URI from freebase in order to obtain a DBpedia URI was observed that at least 3 URIs were returned, that drives to the same final address.

As an example of a real case, executed in our public server, with a URI from Freebase:

```

1 $ curl http://dbpsa.aksw.org/SameAsService/SameAsServlet?uris=http%3A%2F%2Frd.f.
   freebase.com%2Fns%2Fm.015fr
2
3 returns: http://dbpedia.org/resource/Brazil
  
```

Where, in this case, instead of 17 URIs from DBpedia, that goes to the same final address, our approach drives the user directly to the final address.

<sup>5</sup> <http://tinyurl.com/rmethod>

As can be observed on the figure 19 that approach the transitive and redirect URIs, where show that with this approach instead of have several URIs the user can have only one from the DBpediaSameAs. Thus, in this way, providing a normalization on DBpedia URIs.

### 7.3.2 Rate the links

In order to have a link rating, were implemented a GUI that allows the users to give some feedback, suggestions, in this way, improving the quality of the links. The rate is a quite simple process, the GUI just ask the user to rate the link with +1 if the link attends your expectations or -1 if the link is wrong or some type of spam. The GUI was developed using concepts from prefix.cc<sup>6</sup> and work from Zaveri[126] such our system of rate (+1 and -1) and the standard of the web documents. Some improvements and personalization, also was provided, such as the suggestions and the possibility to check the link. The figure 18 shows the moment when the user clicked on the -1 and indicated that the user didn't like the link and was asked to make a suggestion of a new URI.

Your URI	owl:sameAs	DBpedia Unique identifier
< http://rdf.freebase.com/ns/m.015fr >	<a href="http://dbpedia.org/resource/Brazil">http://dbpedia.org/resource/Brazil</a> <a href="http://dbpedia.org/resource/Brazil">http://dbpedia.org/resource/Brazil</a> <a href="http://dbpedia.org/resource/Brazil">http://dbpedia.org/resource/Brazil</a> Total: 17	< http://dbpedia.org/resource/Brazil > +1 -1 <a href="http://dbpedia.org/resource/Br%C3%A9sil">http://dbpedia.org/resource/Br%C3%A9sil</a>

Suggest a new DBpedia URI:

About: Brazil  
Positives (+1): 17 Negatives (-1): 11

Figure 18: Rate a link. Available at <http://dbpsa.aks.org/SameAsService>

The field about a suggestion for a new link will only appear when the user are not satisfied with the current link, then, when clicking on the -1, then the system will ask for an optional suggestion.

### 7.3.3 Results

The results of this work could also be expressed in numbers that was obtained during importing triples to the relational database and with some results from the sameAs.org web site. A total of 62,531,487 triples imported into our database, the time was 2,220 seconds for the whole operation, thus, was noticed that 28,167 triples were imported per second. The source code used to obtain the results is available in our github repository<sup>7</sup>.

<sup>6</sup> <http://prefix.cc>

<sup>7</sup> <https://github.com/firmao/dbpedia-links/blob/master/CreateDB.sh>

### Transitive and Redirect Links

Transitive and Redirect Links are redundancies at DBpedia that supposed has a link to the same place, in other words, they use `owl:sameAs` property, this links will redirect another links, will provide a transition between the links, that's why the name transitive. In this case, instead of using this transitive links that points to the same final destination URI, this final destination URI will be used directly. The figure 19 try to make more clear this explanation.

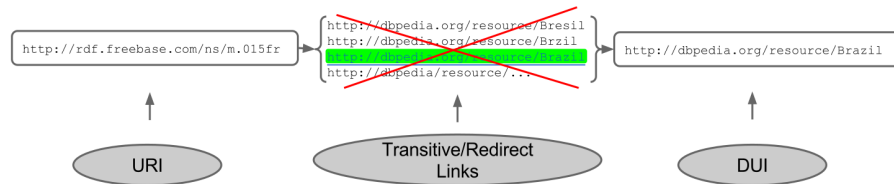


Figure 19: Transitive / Redirect links in DBpedia.

Was discovered and treated 6,473,988 triples with transitive and redirect links from 62,531,487 imported links among 142 domains inside DBpedia. Then, 10.35% of the links can be avoided in some cases.

#### 7.3.4 Discussion

The DBpediaSameAs was evaluated with its normalization of URIs, link rate, and DBpediaSameAs as a stand alone service on the web. As results of the normalization a DUI was obtained in order to tackle the heterogeneity. In other words, instead of several URIs e.g. from sameAs.org one DUI was obtained. The link rate functionality further allows to improve the quality of the dataset.

Despite, the GUI of DBpediaSameAs, also a stand alone service on the web was developed that brings the functionality to get a DUI without a GUI for agents or people which don't need to use the DBpediaSameAs in a Graphical mode, allowing use as an off-the-shelf component.

## DETECTION OF ERRONEOUS LINKS IN LARGE-SCALE RDF DATASETS

*This chapter shows an extension of the previous work[Valdestilhas et al.] in which we discuss the importance of the links for the Linked Data Web as they make a large number of tasks possible, including cross-ontology, question answering and federated queries. However, a large number of these links are erroneous and can thus lead to these applications producing absurd results. We present a time-efficient and complete approach for the detection of erroneous links for properties that are transitive. To this end, we make use of the semantics of URIs on the Data Web and combine it with an efficient graph partitioning algorithm. We then apply our algorithm to the LinkLion repository and show that we can analyze 19,200,114 links in 4.6 minutes. Our results show that at least 13% of the owl:sameAs links we considered are erroneous. In addition, our analysis of the provenance of links allows discovering agents and knowledge bases that commonly display poor linking. Our algorithm can be easily executed in parallel and on a GPU. We show that these implementations are up to two orders of magnitude faster than classical reasoners and a non-parallel implementation.*

### 8.1 INTRODUCTION

Links across knowledge bases play a fundamental role in Linked Data [6] as they allow users to navigate across datasets, integrate Linked Data sources [75], perform federated queries [91] across data sources and perform large-scale inference on the data. Given the importance of links, corresponding repositories such as *sameas.org*<sup>1</sup> and LinkLion [70] (of which *sameas.org* is a subset) have been created. In addition to facilitating the finding of links between resources and knowledge bases, these repositories also allow detecting significant errors across links. For example, according to LinkLion and by virtue of transitivity, the resources *orca:21075* and *orca:1946*<sup>2</sup> stand for the same entity of the real world but have different URIs within the same knowledge base. This clearly goes against the definition of URIs as used in RDF. figure 30 shows a fictional example to help illustrate such problems, which can be classified as **contradiction problems**, according to the quality dimension of consistency [127]. In our example, we can infer that one of the links along the path that led to this inference is wrong or that the knowledge base in itself contains an error. While such errors can be potentially detected by computing

<sup>1</sup> <http://sameas.org/>

<sup>2</sup> orca stands for the namespace <http://orca.cf.ac.uk/id/eprint/>.

the closure of equivalence links using the characteristics of equivalence relations and an inference engine, our experiments with Pellet [Bock et al.] – the fastest inference engine to the best of our knowledge – suggest that inference engines do not scale to the millions of links found on the Web of Data.

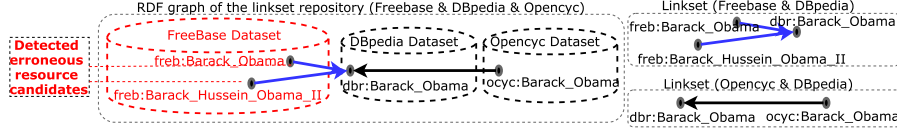


Figure 20: Manual detection of erroneous resource candidates.

The poor performance on closure computations is also known from literature (see, e.g., [9]). For instance, the computation of closures in RDF graphs has several drawbacks. Firstly, it is known that the **size** of the transitive closure of a graph  $G$  is of **quadratic order** in the worst case, making the computation and storage of the closure too expensive for web-scale applications. Secondly, once the transitive closure has been computed, all queries are evaluated over a data source which can be much larger than the original one. This can be particularly **inefficient for queries** that must scan a large part of the input data.

Our intuition is that it is actually not necessary to compute the closures; instead, we can use adjacency lists and create graph partitions based on an algorithm called *Union Find* [109]. Therewith, we obtain a solution with a time complexity that is decreased from  $O(n^2)$  to  $O(m \log n)$ , where  $m$  is the number of operations (either *Union* or *Find*) that are applied to  $n$  elements.

In this paper, we aim to find erroneous links across knowledge bases by reusing the uniqueness of the semantics of URIs within given knowledge bases. We hence present a novel time-efficient algorithm called *Consistency Error Detection Algorithm* (CEDAL), in which the error detection consists of finding distinct resources (i.e., resources with distinct URIs) which share the same dataset, given an RDF graph representing the union of all knowledge bases in a link repository.

With our work, we address the following research questions:

1. Is there a time-efficient algorithm to detect erroneous links in large-scale link repositories?
2. Is there an approach to discover whether a linkset<sup>3</sup> is consistent without computing all closures required by the property axiom?

The contributions of this work are listed below:

<sup>3</sup> We refer to the definition of linkset as defined in the W3C document at <https://www.w3.org/TR/void/>.

- A time-efficient algorithm for the detection of erroneous links in large-scale link repositories without computing all closures required by the property axiom.
- An approach that brings the possibility to track the consistency problems inside link repositories.
- A scalable algorithm that works well in a parallel and non-parallel mode.
- A study case applied to a link repository called LinkLion.
- A new linkset quality measure based on the number of erroneous candidates.

The remainder of this chapter is structured as follows: section 8.2 presents our algorithm for the detection of erroneous links in large-scale link repositories; section 8.3 presents the error types and a quality measure for linksets; section A.1.5 presents the evaluation of our approach.

## 8.2 METHOD

After introducing the terminology and symbolism used in this work, in this section, we present our error detection algorithm.

### 8.2.1 Error Detection algorithm

Our algorithm targets consistency errors in large-scale link repositories. We assume a union of linksets  $\mathcal{L}$  as given input. The aim is to find cases in which equivalent resources (according to the OWL semantics) in  $\mathcal{L}$  share the same dataset. The basic intuition here is equivalent resources (i.e., resources that stand for the same entity from the real world) being in one knowledge base is a clear hint towards (1) an error in the knowledge base itself or (2) an error during the generation of the links that allowed generating this equivalence.

In figure 32, we show how our algorithm works. Given datasets  $D_1, \dots, D_n$ , resources  $R = \{a, b, c, d, e, f\}$ , the idea is to detect two or more resources sharing the same dataset inside the same cluster, following the steps described in the following list and into the algorithm 4.

1. As input, the algorithm receives a set of linksets  $L = \{(r_1, r_2), \dots, (r_n, r_m)\}$ , where  $r_n$  represents the resources.
2. The linksets are merged creating a unique RDF graph  $\mathcal{L} = \bigcup_i L_i$ , where  $L_i = (s, p, o) : s \in D_i^{(s)}, o \in D_i^{(t)}$ ,  $D$  represents source and target datasets of the linksets and  $(s, p, o)$  are subject, predicate and object of an RDF triple.



3. From  $\mathcal{L}$ , create clusters  $\mathcal{C}$ , containing the resources, datasets and the knowledge base.
4. Find cases in which two or more resources  $r_i$  belong to the same dataset  $D$ .
5. Put these resources, related paths, dataset names and knowledge-bases into a list and return this list  $\mathcal{O}$ .
6. The output are all paths that were considered wrong and the original mappings.

Our algorithm works by partitioning a graph inside an adjacency list that contains: (1) An array per vertex for a total of  $V$  arrays, where we are only considering the space for the array pointer, not the contents of the array. (2) Each directed edge is contained once somewhere in the adjacency list, for a total of  $E$  edges, where a bidirectional edge is just 2 directed edges, assuming we are using bi-directional edges.

In this context we are using techniques from an well know algorithm called Union-Find, where the definition in computer science, is based in a disjoint-set data structure, also called a merge-find set, is a data structure that keeps track of a set of elements partitioned into a number of disjoint (non overlapping) subsets. It supports two useful operations:

(1)Find: Determine which subset a particular element is in. Find typically returns an item from this set that serves as its "representative"; by comparing the result of two Find operations, one can determine whether two elements are in the same subset.

(2)Union: Join two subsets into a single subset.

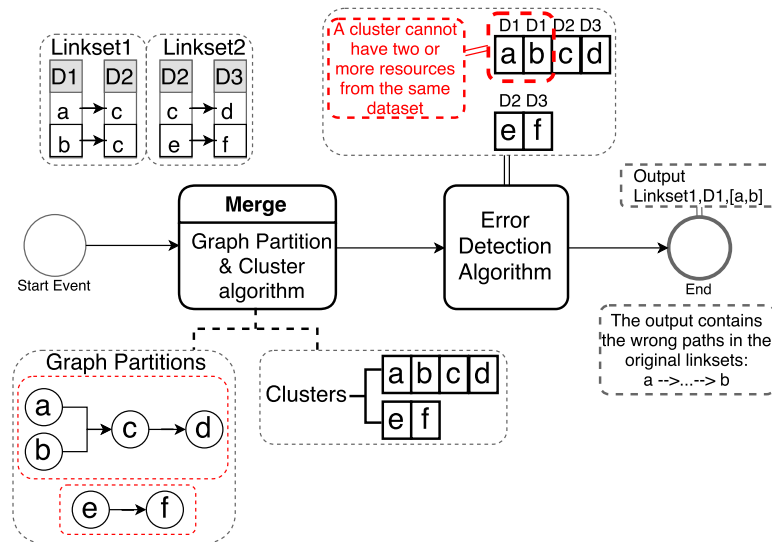


Figure 21: Error detection.

**Algorithm 4** Consistency Error Detection Algorithm (CEDAL)

---

**Input:**  $L : L_i = (s, p, o) : s \in D_i^{(s)}, o \in D_i^{(t)}$  resource.dataset + re-  
**Output:** An error list with erro- source.path  
 neous nodes.

```

4:
5:
6:
7:   return ErrorList
8: end procedure
9: procedure GETPARTITIONS( $G(V, E)$ )
10:   forall  $v \in V$  do
11:     end
12:     push onto MapResourcesDataset
13:      $v, \text{extracDataset}(v)t$ 
14:     push all nodes connected to  $v$ 
15:     onto  $\mathcal{V}$   $\triangleright$  UnionFind
16:     algorithm
17:   end
18:   push  $\mathcal{V}$  onto  $\mathcal{P}$ 
19:   return  $\mathcal{P}$ 
20: end procedure

```

---

*Problem statement.*

We formalize the problem of *detection of erroneous links in large-scale link repositories* as follows.

From this section on, we will refer to the union of all linksets as  $\mathcal{L} = \bigcup_i L_i$ , the set of all datasets as  $\mathcal{D}$ , the clusters (or graph partitions) as  $\mathcal{C}$ , the candidates (i.e., set of resources belonging to the same dataset)  $\mathcal{P}$ . A linkset  $L \in \mathcal{L}$  contains triples (or links)  $(s, p, o)$  such as:

$$(s, p, o) \in L : s \in D_i, o \in D_j, i \neq j \quad (22)$$

Each candidate  $P$  abides by the following restriction:

$$\forall r_i, r_j \in P : r_i \neq r_j \Rightarrow (r_i, x, r_j) \in \mathcal{L}^* \vee (r_j, x, r_i) \in \mathcal{L}^* \quad (23)$$

where  $x$  is a property (e.g., owl:sameAs). In other words, each element in  $P$  is linked to at least another element in the set. Candidates are assigned one of two classes, positive (i.e., candidates with errors) or negative. The positive cases are represented as follows.

$$P \in \mathcal{P}^+ \iff (\exists r_1 \in P \cap D_1, r_2 \in P \cap D_2) \therefore D_1 = D_2 \Rightarrow r_1 \neq r_2 \quad (24)$$

The negative cases are defined in the following equation.

$$P \in \mathcal{P}^- \iff (\forall r_1 \in P \cap D_1, r_2 \in P \cap D_2) \therefore D_1 = D_2 \Rightarrow r_1 = r_2 \quad (25)$$

The target is thus to find the set of erroneous candidates  $\mathcal{P}^+$ . As shown in the next section, we cannot state that a link connecting these resources is wrong, but we can state that the error lies somewhere between the links that connect them and the organization of the dataset they belong to.

### 8.3 ERROR TYPES AND QUALITY MEASURE FOR LINKSET REPOSITORIES

The application of the two measures requires the output from CEDAL, allowing to identify two types of errors among the erroneous candidates from the output.

#### 8.3.1 Error Types

We identified two types of errors, in which can be defined in quality dimensions by [127]. (1) **Semantic accuracy** errors, in which we detect if data values correctly represent the real world facts. (2) **Consistency** and **Conciseness** errors where a knowledge base is free of logical or formal contradictions concerning particular knowledge representation and inference mechanisms and the minimization of redundancy of entities at the schema and the data level. figure 31 shows a fictional example of both error types, in which we represent links between GeoNames<sup>4</sup> and DBpedia.<sup>5</sup>

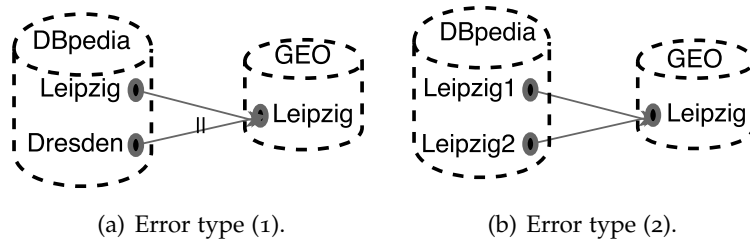


Figure 22: Detected error types.

In this example, figure 31(a) shows an error of type (1), in which an erroneous `owl:sameAs` link between the city of Dresden and the city of Leipzig was detected. The figure 31(b) shows an error of type (2) where the resource about the city of Leipzig is duplicated within the DBpedia dataset.

We manually analyzed a random sample of the errors. Among 100 occurrences, 90% are of type (2). It was not feasible in practice to perform this evaluation in an automatic way, due to the fact that it involves semantic accuracy and thus needs human feedback. Moreover, some URIs are unreachable, resulting many times in *timeout errors*,

<sup>4</sup> <http://www.geonames.org/>

<sup>5</sup> <http://dbpedia.org/>

such as HTTP 404, 500 and 503 errors.<sup>6</sup> In summary, it was not practicable to automatically distinguish these types of errors among the erroneous candidates detected by CEDAL.

### 8.3.2 Quality Measure

Based on the error types from CEDAL, we present three linkset quality measures, evaluating the information accessed by cross-walking the linksets of LinkLion.

The **Semantic Accuracy of linksets** indicates whether the data values from the RDF links represent real world facts. **Example:** Let us assume that we have a linkset from DBpedia and Geonames. A link `<dbr:dresden owl:sameAs geo:leipzig>` would clearly be inaccurate, since Dresden and Leipzig are two different cities.

The **consistency and conciseness of links** inform whether a linkset is free of logical or formal contradictions with respect to particular knowledge representation and inference mechanisms and the minimization of redundancy of resources that belongs to the same dataset inside a linkset repository. **Example:** With a linkset from DBpedia and Geonames, let us assume we found two links represented by `<dbr:leipzig1 owl:sameAs geo:leipzig>` and `<dbr:leipzig2 owl:sameAs geo:leipzig>`. Since `dbr:leipzig1` and `dbr:leipzig2` belong to the same dataset, this characterizes a redundancy and it contradicts the assumption that two URIs in a dataset cannot stand for the same thing from the real world.

In order to evaluate data quality in linksets, on the lines of the works summarized in the Data Quality survey [127], we propose three new metrics:

**M1:** Rate of consistent resources inside linkset repositories.

Let us consider a candidate  $P \in \mathcal{P}$  containing only resources which belong to the same dataset. The rate of consistent candidates is defined as follows:

$$M1 = \frac{\sum_{P \in \mathcal{P}^-} |P|}{\sum_{P \in \mathcal{P}} |P|} \quad (26)$$

where  $\mathcal{P}^-$  is the set of consistent (i.e., non-erroneous) candidates. We call M1 the **consistency index**.

**M2:** Rate of candidates in  $\mathcal{P}$  containing resources whose internal links are real world facts. Let us introduce a function  $f(s, p, o)$  which expresses the verification of a triple  $(s, p, o)$  in the real world, assuming value 1 if the statements holds true and 0 otherwise. This metric addresses errors of type (1).

$$M2 = \frac{|\{P \in \mathcal{P}^+ : \forall r_i, r_j \in P \ r_i \neq r_j \Rightarrow f(r_i, p, r_j) = 1\}| + |\mathcal{P}^-|}{|\mathcal{P}|} \quad (27)$$

<sup>6</sup> <https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>

$M_3$ : Rate of candidates in  $\mathcal{P}$  which are free of redundant resources.  
This metric addresses errors of type (2).

$$M_3 = \frac{|\{P \in \mathcal{P}^+ : \exists r_i, r_j \in \mathcal{P} \ r_i \neq r_j \Rightarrow f(r_i, p, r_j) = 0\}| + |\mathcal{P}^-|}{|\mathcal{P}|} \quad (28)$$

As can be seen,  $M_2$  and  $M_3$  are dependent on each other.

In this paper, we focus on the computation of  $M_1$ . Although  $M_2$  and  $M_3$  are left for future research, we included them to encourage their evaluation and use.

## 8.4 EVALUATION

To verify our hypothesis, in this section we show that CEDAL brings an efficient way to track the erroneous candidates inside large-scale linkset repositories.

### 8.4.1 Experimental setup

As our study case, we use a linkset repository called LinkLion [70] due to some advantages such as provenance, linksets from the most used datasets, i.e. DBpedia, Yago and Opencyc, where the users are empowered to upload links and specify how these were created. Moreover, users and applications can select and download sets of links via dumps or SPARQL queries.

The table 13 shows that 99.9% of links from LinkLion are owl:sameAs links, amounting to 19,200,114 triples. Thus, in our experiments we are using only owl:sameAs links.

Table 8: Link types

Property	Triples
<b>sameAs</b>	19,606,657 (with duplicates)
<b>country</b>	1,309
<b>author</b>	766
<b>spokenIn</b>	624
<b>locatedIn</b>	250
<b>exactMatch</b>	167
<b>near</b>	30
<b>spatial#P</b>	28
<b>seeAlso</b>	14
<b>organism</b>	14
<b>made</b>	4

The experiments were performed using two configurations: (1) a laptop with Intel Core i7, 8 GB RAM, a video card NVIDIA NVS4200, Operational System MS Windows 10 and Java SE Development Kit 8. (2) An Intel Xeon Core i7 processor with 40 cores, 128 GB RAM on an Ubuntu 14.04.5 LTS with Java SE Development Kit 8. The results including the output file for LinkLion are available online. The total number of 19.6million links was processed by our algorithm in 4.6 minutes with the configuration (2). The total amount of errors were 1,352,366 of candidates, where the total amount of domains were 254 and the number of linkset files was 553, where 48.3% of these knowledge base files has less than 10 resources detected as erroneous candidates.

#### 8.4.2 Ranking the erroneous candidates

To evaluate how effective CEDAL is, we create a score in order to rank the erroneous candidates based on the number of detected resources with errors, in which the table 9, show two fictional examples of tuples in the same pattern of the output from CEDAL.

Table 9: Fictional example results.

Knowledge-base	Data-set domain	C	$\mu$
Linkset1.nt	Data-set1	URI <sub>1</sub> ,URI <sub>2</sub>	1
Linkset2.nt	Data-set2	URI <sub>1</sub> ,URI <sub>2</sub> ,URI <sub>3</sub> ,URI <sub>4</sub>	6

The  $\mu$  score is calculated by  $\mu = \frac{|C|(|C|-1)}{2}$ , in which we use the cardinality of C representing the detected erroneous candidates. The figures 33 and 23(b) shows the top 5 erroneous candidates according to the rank score.

Table 10: Legend for the figures 33 and 23(b)

Label	Knowledge Base
K1	dotac.rkbexplorer.com—eprints.rkbexplorer.com.nt
K2	d-nb.info—viaf.org.nt
K3	dblp.rkbexplorer.com—dblp.l3s.de.nt
K4	linkedgeodata.org—sws.geonames.org.nt
K5	citeseer.rkbexplorer.com—kisti.rkbexplorer.com.nt
K6	wiki.rkbexplorer.com—oai.rkbexplorer.com.nt
K7	www4.wiwiss.fu-berlin.de—dbpedia.org.nt
K8	southampton.rkbexplorer.com—nsf.rkbexplorer.com.nt
K9	rae2001.rkbexplorer.com—newcastle.rkbexplorer.com.nt
K10	lod.geospecies.org—bio2rdf.org.nt

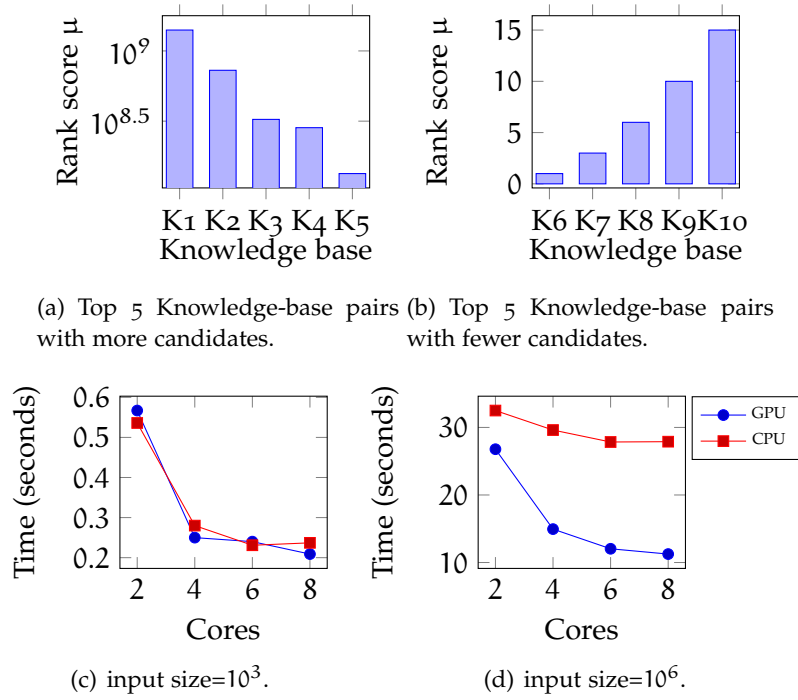


Figure 23: Error rank (legends: see table 10) and Runtime results according to the input size, CPU and GPU.

Considering only linksets between different datasets, the knowledge-base with more errors comes from the links in

`dotac.rkbexplorer.com` - `eprints.rkbexplorer.com`

with 458,324 links per mapping<sup>7</sup>, in which we found 53,074 erroneous resource candidates resulting in a score of 1,408,398,201. The knowledge base with fewer errors comes from the links in

`lod.geospecies.org` - `bio2rdf.org.nt`

with 9,723 links per mapping, in which we found 6 erroneous resource candidates and a score of 15, also 193 datasets with no errors at all.

#### 8.4.3 Runtime experiments

The experiments were performed with the input size varying between  $10^3$  and  $10^6$  RDF triples using the configuration (1), as shown in figures 23(c) and 23(d). Our algorithm processed all 19,200,114 links from LinkLion in 4.6 minutes with the configuration (2). The results indicate that our algorithm scales well to large links repositories and can also be adapted to the hardware on which it is executed. For example, it can be easily implemented to make use of benefits of CPUs and GPUs.

<sup>7</sup> Links per mapping from <http://www.linklion.org/>

### Scalability Evaluation

Our algorithm performs well in parallel and non-parallel environments. The performance of our algorithm improved in accordance to the number of CPUs, showing that our algorithm is scalable, performing well with large linksets with size more than  $10^6$  as shown in figures 23(c) and 23(d).

### Parallel Implementation

Our algorithm implementation contains parallel code snippets in which we perform a load-and-balance of the data among CPU/GPU cores when available. This specific characteristic offers the possibility for utilization when hardware for parallel computing is available, such as CPU/GPU processors.

To illustrate this part of our idea, we can state: Given a graph  $G(V, E)$ , that contains all linksets from the repository  $G(V, E) \subseteq \mathcal{L}$ , this graph has partitions  $P \subset G(V, E)$ . Thus, errors are calculated for each partition, the processes are separated in threads and these threads are spread among CPU/GPU cores. Thus, we process the graph partitions in parallel, as shown in figure 24.

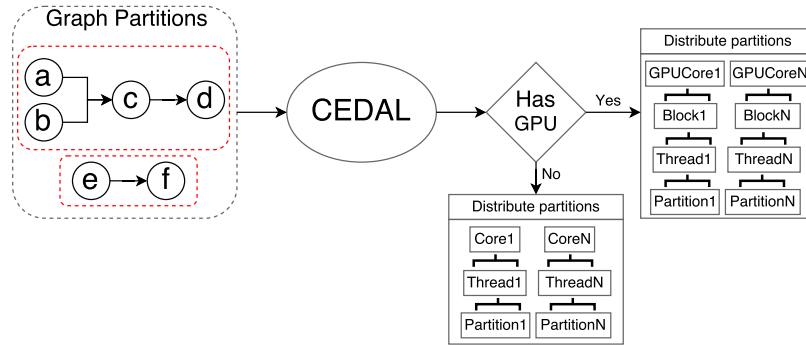


Figure 24: CEDAL CPU/GPU processing

#### 8.4.4 Consistency by provenance of links

Thanks to the information found in LinkLion, we are able to check the provenance of links. This allowed us to analyze them more in details by finding which link discovery framework has created the link. The links were generated by four types of frameworks: *LIMES* [74], *SILK* [121], *DBpedia Extraction Framework* [57] and *sameas.org*<sup>8</sup>. The type of links provided by *sameas.org* were generated into human-curated knowledge bases, such as DBpedia (which is based on Wikipedia), Freebase, and OpenCyc.

We found a 13.5% error rate from *sameas.org* versus a 4.1% error rate of the algorithms such as *LIMES* [74]. They are all below 5% as

<sup>8</sup> <http://sameas.org>



table 11 shows; column Errors represents the rate of all resources belonging to error candidates and column M1 represents the respective quality measure.

Table 11: Comparison of results with respect to the provenance of the links.

Framework	Errors	Resources	Errors (%)	M1
sameas.org	3,792,326	28,130,994	13.5	0.865
LIMES	1,130	27,819	4.1	0.951
Silk	5,933	208,300	2.8	0.972
DBpedia Extraction Framework	12,615	914,180	1.4	0.986
<b>All frameworks</b>	<b>3,812,004</b>	<b>29,281,293</b>	<b>13.0</b>	<b>0.870</b>

According to this data, we can say that algorithms such as LIMES, SILK, and the DBpedia Extraction Framework have a higher consistency index than *sameas.org*. This might be explained by the fact that no mechanism of link validation is present on *sameas.org*.

#### 8.4.5 Comparison with other works

To the best of our knowledge, CEDAL is the first approach that aims to detect the consistency of RDF link repositories. However, the problem that CEDAL addresses can be solved in other ways. One alternative to solving our problem is to use reasoning. However, this approach requires the computation of all closures required by the property axiom. To check whether our approach performs better than a closure-based approach, we compared CEDAL with an algorithm for computing closures – dubbed Closure Generator – without using a reasoner and with Pellet, which is considered the state-of-the-art reasoner [Bock et al.]. figure 25 shows that CEDAL is significantly faster than Pellet, reaching up to three orders of magnitude of speedup when faced with  $10^6$  triples. This result can be partly explained by Pellet also checks the knowledge base for every single coherence and consistency axiom. However, we did not need such an in-depth analysis.

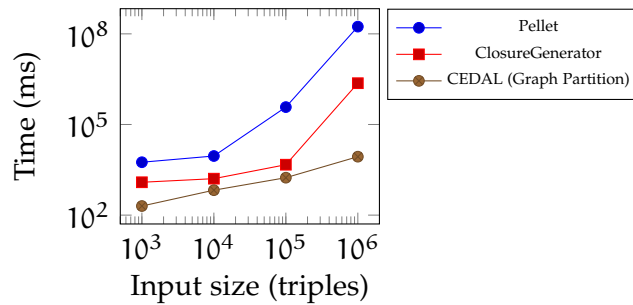


Figure 25: Pellet vs. ClosureGenerator vs. CEDAL

## QUERYING LARGE HETEROGENEOUS RDF DATASETS

---

*Into this chapter, we discuss the fact that over the last years, the Web of Data has grown significantly. Various interfaces such as LOD Stats, LOD Laudromat, SPARQL endpoints provide access to the hundreded of thousands of RDF datasets, representing billions of facts. These datasets are available in different formats such as raw data dumps and HDT files or directly accessible via SPARQL endpoints. Querying such large amount of distributed data is particularly challenging and many of these datasets cannot be directly queried using the SPARQL query language. In order to tackle these problems, we present WimuD, an integrated query engine to execute SPARQL queries and retrieve results from large amount of heterogeneous RDF data sources. Presently, WimuD is able to execute both federated and non-federated SPARQL queries over a total of 668,166 datasets from LOD Stats and LOD Laudromat as well as 559 active SPARQL endpoints. These data sources represent a total of 221.7 billion triples from more than 5 terabytes of information from datasets retrieved using the service “Where is My URI” (WIMU). Our evaluation on state-of-the-art real-data benchmarks shows that WimuD retrieves more complete results for the benchmark queries.*

### 9.1 INTRODUCTION

Currently, the LOD Laudromat along with LOD stats have more than 250 billion facts which are available on 668,166 datasets and 559 active SPARQL endpoints [114]. Querying such large amount of distributed data is particularly challenging in which Federated query processing is one of the key components for accessing this collection of information through these data sources. In general, there are two types of available approaches to execute federated SPARQL queries over distributed RDF data sources [89]. (1) Query federation over multiple SPARQL endpoints (QFME) which collects distributed information from multiple SPARQL endpoints. Commonly, the list of SPARQL endpoints is provided as an input to the federation engine. The federation engine decomposes the original query into multiple sub-queries and execute them accordingly to a specific query execution plan. (2) Link Traversal based SPARQL federation (LTSF) which uses the URI lookups to collect information from distributed RDF data sources. Such approaches do not force the data providers to publish their data as SPARQL endpoints. Instead, the only re-

quirement is that the RDF data sources must follow the Linked Data principles<sup>1</sup> [44].

However, a particular shortcoming of the QFME approaches is that many of the RDF datasets in Linking Open Data (LOD) are not publicly available via SPARQL endpoints. Beek et al. [17] has shown that around 90% of the information published are available only as data dumps. On the other hand, the LTSF approaches are unable to perform lookups for non-dereferenceable URIs. Hence, they may retrieve empty or incomplete results for the latter type of approaches. Note that our previous work [114] shows that around 43% of the URIs of more than 660K RDF datasets from LODStats [32] and LODLaundromat [17] are non-dereferenceable.

In this work, dubbed WimuD, we provide an approach that overcomes the limitations of the state of the art regarding to the coverage of the results. WimuD is a hybrid (endpoints federation + link traversal) federated SPARQL query processing engine which collects distributed information from SPARQL endpoints as well as raw data dumps and HDT files [35]. WimuD integrates exciting state-of-the-art LTSF and QFME approaches into a single query execution framework. Additionally, WimuD overcomes the problem of non-dereferenceable URIs by making use of our previous index of 4.2 billion URIs from more than 660k RDF datasets [114]. Our evaluation on state-of-the-art real-data benchmarks shows that WimuD retrieves more complete results on three different benchmarks for federated SPARQL query engines [98; 90; 86].

The contributions of this paper are as follows:

- We present a hybrid SPARQL query processing engine which is able to retrieve results from 559 active SPARQL endpoints (with a total of 163.23 billion triples) and 668,166 datasets (with a total of 58.49 billion triples) from LOD Stats and LOD Laundromat. Thus, to the best of our knowledge, WimuD is the first federated SPARQL query processing engine that executes SPARQL queries over a net total of 221.7 billion triples
- As part of WimuD, we present a low-cost API dubbed *SPARQL-a-lot* to execute SPARQL queries over LOD-a-lot [34], a 300 GB HDT file of the LOD cloud. For the first time, to the best of our knowledge, we make use of the Concise Bounded Descriptions (CBDs), 4 to execute federate SPARQL queries.
- We make use of the WIMU index [114] to intelligently select the capable (also called relevant) [92] data sources pertaining to the given SPARQL query. We evaluated our integrated query engine on two real-data federated SPARQL querying benchmarks *LargeRDFBench* [85], *FedBench* [98] and one non-federated real

<sup>1</sup> <http://www.w3.org/DesignIssues/LinkedData.html>

data SPARQL benchmark selected from *FEASIBLE* [90] benchmarks generation framework.

WimuQ is open source and available from <https://github.com/firmao/wimuT> along with complete evaluation results. The web version of the WimuD is available from <https://w3id.org/WimuQ/>.

The rest of the chapter is organized as follows: WimuD approach in section 9.3. Section 9.4 shows the evaluation results and discussion.

## 9.2 DEFINITIONS

Let  $Q$  be a SPARQL query and  $\mathcal{D}$  be a finite set of the datasets which can be retrieved using WIMU [114]. For each data source  $D \in \mathcal{D}$ , we write  $G(D)$  to denote the underlying RDF graph exposed by  $D$ . When requesting data source  $D$  to execute a SPARQL query  $Q$ , we expect that the result of  $D$  is the set  $Q_{G(D)}$ . Hence, we define the result set  $S(Q)$  of a SPARQL query over the federation  $\mathcal{D}$  to be a set of solution mappings that is equivalent to  $Q_{G(\mathcal{D})}$  where  $G_{\mathcal{D}} = \bigcup_{D \in \mathcal{D}} G(D)$ .

We formalize the problem in two steps as follows: (1) *find the datasets on which a given query can be executed* and (2) *return the query results from the selected datasets.*<sup>2</sup>

## 9.3 THE WIMUD

In this section, we explain the WimuD approach to the SPARQL query processing. We assume that the reader is familiar with the concepts of RDF and SPARQL, including the notions of an RDF triple, a triple pattern, a basic graph pattern (BGP), and subject, predicate, object of the triple pattern.

Figure 26 shows the workflow of the query processing in our approach, which comprises of 7 main steps: (1) the user issue a SPARQL query to the WimuD interface, which (2) extracts all the URIs used in the given user query. Note the URIs can be used in subject, predicate, or objects of the SPARQL triple patterns. (3) The extracted URIs are then searched in the WIMU index, which gives all the relevant datasets where the extracted URIs can be found. (4) Our Relational index *ReLOD*[8] will provide the most similar datasets, adding more relevant datasets and excluding nonrelevant datasets, (5) the relevant datasets are further filtered based on the source selection algorithm, and (6) the finally-selected relevant datasets are then queried by using the different query processors, depending on the type (HDT, endpoint, datadump, dereferenceable dataset) of the datasets; (7) the results generated by the different query processors are then integrated and sent back to the user. The whole process is like a black box to

<sup>2</sup> Throughout the paper, we refer to RDF dump files or any datasets accessible using a SPARQL endpoint as *datasets*, *data sources* or simply *sources*.

the end user: the user only sends the query and get back the results without knowing the underlying query execution steps.

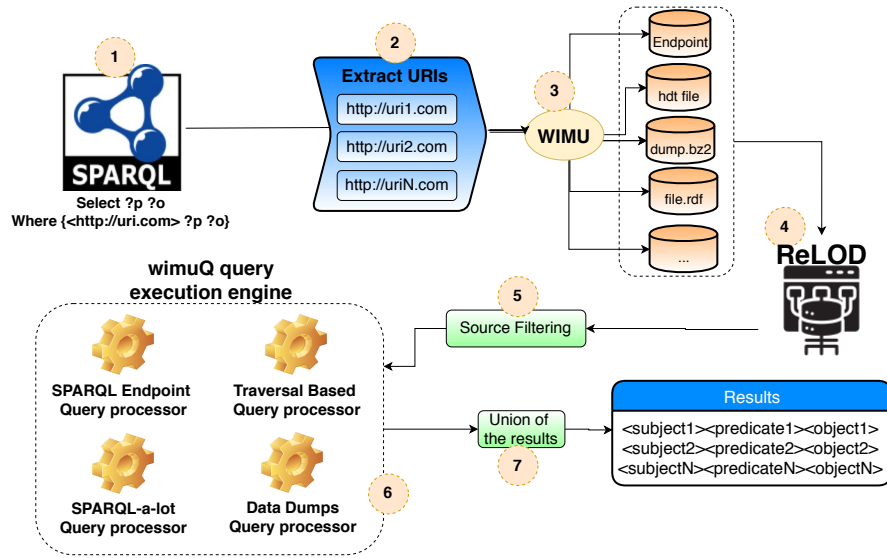


Figure 26: Query processing workflow, already including *ReLOD*[8]

Now we go into the details of these steps, explaining the different components.

### 9.3.1 WimuQ source selection

One of the important optimization steps in federated SPARQL query processing is the *source selection* [78; 92]. The goal of the source selection is to identify the potentially relevant datasets (also called sources) to the given query. We make use of the WIMU service [114] to select the potentially relevant sources. The WIMU service<sup>3</sup> provides URIs lookup facility and identify those datasets which contain the given URI. Currently, this service processed more than 58 billion unique triples and indexed 4.2 billion URIs from more than 660k RDF datasets obtained from the most used Linked Data hubs including LODStats [13] and LOD Laundromat [17]. The identified WIMU datasets can be of four types: (1) SPARQL endpoint, (2) HDT file, (3) dataset with dereferenceable URIs, (4) data dump with non-dereferenceable URIs. The service is both available from a web interface as well as can be queried from a client application using the standard HTTP protocol.

The WimuQ source selection algorithm is given in Algorithm 5 which takes a SPARQL query  $Q$  as input. We provide the set of extracted URIs (from step 2)  $U$  to the WIMU service and retrieve the relevant datasets  $D$  pertaining to the given URIs (Lines 1-4 of Algorithm 5). Now for each relevant dataset  $d \in D$ , if the  $d$  is a SPARQL

<sup>3</sup> WIMU URIs lookup service is available from: <http://wimu.aksw.org/>

endpoint, we extract the individual triple patterns  $t$  of the query and perform a SPARQL ASK of  $t$  in dataset  $d$  (Lines 5-8 of Algorithm 5). We add the dataset  $d$  into the set of relevant SPARQL endpoints  $E$ , if the SPARQL ASK query returns true (Lines 9-13 of Algorithm 5). We directly add  $d$  into sets  $H$  and  $T$  if  $d$  is an HDT file or if the  $d$  is a SPARQL endpoint or dataset with dereferenceable URIs, respectively (Lines 14-19 of Algorithm 5). Finally, if  $d$  is dataset with non-dereferenceable URIs, we apply the RDFSlice [63] technique to only get the relevant RDF slice of the dataset. The RDF slice is then considered as a relevant dataset (Lines 20-23 of Algorithm 5). The purpose of only adding the relevant slice of the complete dataset is to reduce the processing overhead, explained in the next section.

### 9.3.2 WimuD query execution

We make use of the four – SPARQL endpoint, Link Traversal-based, SPARQL-a-alot, Data dumps – query processor to execute federated queries over the aforementioned four types of WIMU datasets. The relevant data sources for each of these query processors are returned by the WimuD source selection discussed in previous section.

In WimuD, we used FedX [99] query processor for SPARQL endpoints query federation and SQUIN for traversal-based query federation. The reason for choosing FedX for SPARQL endpoints federation and SQUIN for traversal-based federation is due the fact that they do not require any pre-computation of dataset statistics and hence are able to retrieve up-to-date results. Thus both are able to run federated queries with zero initial knowledge. In addition, both produce reasonably query runtime performances comparing to state-of-the-art approaches [89; 96; 45].

The list of required endpoints URLs for FedX are returned from the previously discussed source selection Algorithm (ref. E of Algorithm 5). The potentially relevant dereferenceable URIs data sources (ref. T of Algorithm 5) are already identified by the WimuD sources selection algorithm. Thus, we reduced the search space by only considering the dereferenceable URIs data sources.

As mentioned before, FedX can only works with public SPARQL endpoints. SQUIN needs dereferenceable URIs. Both of these engines are unable to execute SPARQL queries over non-dereferenceable URIs datadumps: SPARQL endpoint federation approaches cannot execute queries over such datadumps as they are not exposed as SPARQL endpoints, link traversal-based approaches fail to retrieve results as the URIs are non-dereferenceable. We need to download the dumps first, load it locally, and run some query processing API (e.g., JENA or Sesame) on the loaded datasets. However, we can not simply download the complete dumps and process it locally due to their large amount of data. To solve this problem, we make use of the WimuD

**Algorithm 5** The WimuQ source selection

---

**Input:**  $Q$  ▷ The SPARQL query  
**Output:**  $E, H, T, N$  ▷ Hash sets of relevant sources of SPARQL endpoints, HDT files, dataset with dereferenceable URIs, datadump with non-dereferenceable URIs, respectively

```

1:  $U = \text{extractURIs}(Q)$  ▷ Extract the URIs from the SPARQL query
   forall  $u \in U$  do
2:   end
    $D = D \cup \text{wimuLookup}(u)$  ▷ Datasets from the WIMU
3: forall  $d \in D$  do
   end
   For each dataset if  $d$  is SPARQL endpoint then
   end
    $t \in Q$  ▷ For each Triple pattern in query
4:  $b = \text{ASK}(t, d)$  ▷ SPARQL ASK of  $t$  in  $d$  if  $b = \text{true}$  then
5:   end
    $E.\text{add}(d)$ 
6:
7:
8: if  $d$  is HDT file then
9:   end
    $H.\text{add}(d)$ 
10: if  $d$  is dataset with dereferenceable URIs then
11:   end
    $T.\text{add}(d)$ 
if  $d$  is datadump with non-dereferenceable URIs then
12:   end
    $N.\text{add}(\text{RDFSlice}(d))$  ▷ only add the relevant slice
13:
14:
15: return  $E, H, T, N$  ▷ final relevant sources set

```

---

index to only select those data dumps which are potentially capable to execute the given query. These datadumps are further sliced by using the RDFSlice technique, to only select the required chunks of the datadumps which will provide results to the given SPARQL query. The identified chunks are finally loaded in a local Apache Jena model. The model is then use to execute federated queries.

Finally, we propose a query processor named *SPARQL-a-lot* to execute federated SPARQL queries over HDT files. The SPARQL-a-lot is a CBD-based<sup>4</sup> query execution described in Algorithm 6. The algo-

<sup>4</sup> <https://www.w3.org/Submission/CBD/>



rithm takes a SPARQL query  $Q$  as input and return the resultset  $O$  of the query execution over HDT file. First, the algorithm extracts all of the BGPs from  $Q$  (Line 2 of Algorithm 6). The next step is to extract the subjects, predicates, and objects of all the triple patterns used in  $Q$  (Line 3 of Algorithm 6). The CBDs are then generated from extracted subjects, predicates, and objects (Line 4 of Algorithm 6). The extracted CBDs constitute an RDF dataset of set of triples  $T$  which are then loaded into a local Jena model (Line 5 of Algorithm 6). The query  $Q$  is then finally executed over the Jena model and results are returned (Lines 6-7 of Algorithm 6). The duplicated results from different query processing engines is removed after collecting the final results.

---

**Algorithm 6** Query execution on LOD-a-lot
 

---

**Input:**  $Q$  ▷ The SPARQL query

**Output:**  $O$  ▷ The results of the query

```

1: procedure SPARQL-A-LOT( $Q$ )
2:    $BGP = \text{extractBGP}(Q)$   ▷ Extract the BGP from the SPARQL
   query
3:    $T_{bgp} = \text{extractSPO}(BGPs)$ 
4:    $T = \text{executeAPILOD-a-LOT}(T_{bgp})$   ▷ Generating CBDs from
   s, p, o
5:    $\text{createTDBJena}(T)$ 
6:    $O = \text{execSPARQLTDBJena}(Q)$ 
7:   return  $O$ 
8: end procedure
  
```

---

The results generated by each of WimuD's processors are finally integrated and sent back to the user.

### 9.3.3 Practical example

Given the following SPARQL query (LD1 from FEDBench[98]):

```

1 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2 SELECT * WHERE { ?paper
3 <http://data.semanticweb.org/ns/swc/ontology#isPartOf>
4 <http://data.semanticweb.org/conference/iswc/2008/
5 poster_demo_proceedings> .
6 ?paper <http://swrc.ontoware.org/ontology#author> ?p .
7 ?p rdfs:label ?n .
8 }
  
```

With WimuD we were able to identify 4 HDT datasets available from <https://tinyurl.com/WimuDexData>. All of these datasets are from LOD Laundromat repository [17] containing information about the conference ISWC2008 [102], under the domain <http://data.semanticweb>



.org, originally from SW Dog Food dataset<sup>5</sup>. Thus, *SPARQL-a-lot* is the only query processing engine (because of the selected datasets are HDTs) which is considered for executing the given SPARQL queries. The final results of the query execution is available from <https://tinyurl.com/WimuQExample>.

## 9.4 EVALUATION

In this section, we present the evaluation setup and the corresponding results that validate our hypothesis that we can improve the result-set retrieval if we automatically identify potentially relevant sources from heterogeneous RDF data even if the URIs are not dereferenceable anymore. The goal of this evaluation is to show that by combining different SPARQL query processing approaches, we are able to retrieve more complete results as compared to the results retrieved by the individual approaches.

### 9.4.1 Experimental setup

*Benchmarks used:*

Since WimuD aims to execute SPARQL queries over real-world RDF datasets, we chose three – FedBench [98], LargeRDFBench [85], Feasible [90] – real-world RDF datasets benchmarks in our evaluation:

- **FedBench** is federated SPARQL querying benchmarks. It comprises of a total of 25 queries and 9 real-world interconnected datasets. FedBench queries are further divided into three main categories: (1) 7 queries from Life Sciences (LS) domain, (2) 7 queries from Cross Domain (CD), and (3) 11 queries named Linked Data (LD) for link traversal-based approaches. The detailed statistics of the benchmark's datasets and queries are given in FedBench[98].
- **LargeRDFBench** is also a federated SPARQL querying benchmark. It comprises a total of 40 queries and 13 real-world interconnected datasets. FedBench queries are further divided into four main categories: (1) 14 *Simple* queries, (2) 10 *Complex* queries, (3) 8 *Large Data* queries, and (4) 8 *Complex+High Data Sources* queries. The detailed statistics of the benchmark's datasets and queries are given in [85].
- **FEASIBLE** is a benchmark generation framework which generates customized benchmarks for the queries logs. In our evaluation, we chose exactly the same benchmarks used in [90]: (1) 175 queries benchmark generated from DBpedia queries log and

<sup>5</sup> <https://old.datahub.io/dataset/semantic-web-dog-food>

(2) 175 queries benchmark generated from Semantic Web Dog Food (SWDF) queries log. Further advanced statistics of the used datasets and queries can be found in [90].

To the best of our knowledge, these are the state-of-the-art from the real-data SPARQL benchmarks. All of the 415 queries used in our evaluation is publicly available<sup>6</sup>.

#### *Hardware:*

All the experiments were done on a modest machine with 200 GB of Hard Disk, 8 GB of RAM and a 2.70GHz single core processor. Each of the queries was run 5 times and the average of the results are presented.

#### *SPARQL endpoints:*

As previously stated, the query federation over multiple SPARQL endpoints approaches requires the set of endpoint URLs to be provided as input to the federation engine. We chose a total of 539 active SPARQL endpoints available from LOD cloud<sup>7</sup>. We filtered the endpoints URLs<sup>8</sup> and the total number of triples hosted by each of these endpoints.

#### *Metrics:*

Since WimuDumps aims to retrieve more complete results within the reasonable amount of time, we choose two metrics: (1) coverage in terms of the number of results retrieved from the query executions and (2) the time taken to execute the benchmark queries.

#### *Approaches:*

As mentioned in Section 3, different federation engines available to federate SPARQL queries over endpoints and traversal-based federation. We chose FedX [99] for SPARQL endpoint federation and SQUIN [45] for traversal-based query federation. The reason for choosing these two engines is due the fact they do not require any pre-computation of dataset statistics and hence able to retrieve up-to-date results and able to run federated queries with zero initial knowledge. In addition, both these engines perform reasonably well in terms of query runtime performances w.r.t state-of-the-art approaches [89; 96; 45]. For the sake of completeness, we also compared WimuDumps with SPARQL-a-lot and WimuDumps.

<sup>6</sup> Queries available from [https://github.com/firmao/wimuT/blob/master/queries\\_Location.txt](https://github.com/firmao/wimuT/blob/master/queries_Location.txt)

<sup>7</sup> List of SPARQL endpoints: <https://lod-cloud.net/lod-data.json>

<sup>8</sup> Endpoints URLs with size: <https://goo.gl/H2t5ko>

### 9.4.2 Results

**Coverage of the results:** The main purpose of WimuDumps is to devise a federation engine which is able to retrieve more complete results for the given SPARQL queries. figure 27 shows a comparison of the selected approaches in terms of the average of the number of results retrieved for the different queries categories of the selected benchmarks. The complete results for individual queries can be found on our aforementioned project website. By using different query processing engines, our approach is able to retrieve more results as compared to the results retrieved by only using SPARQL endpoints federation engine (i.e, FedX) link traversal engine (i.e., SQUIN), data dumps, or HDT files. In our evaluation, the average resultset size of WimuDumps is 8481 across the three benchmarks. Out of these, WimuDumps collects about 91% of the results from wimuDumps (avg. resultset size 7651), 7% from SPARQL endpoints (avg. resultset size 556), and 1% from SPARQL-a-lot (avg. resultset size 74).

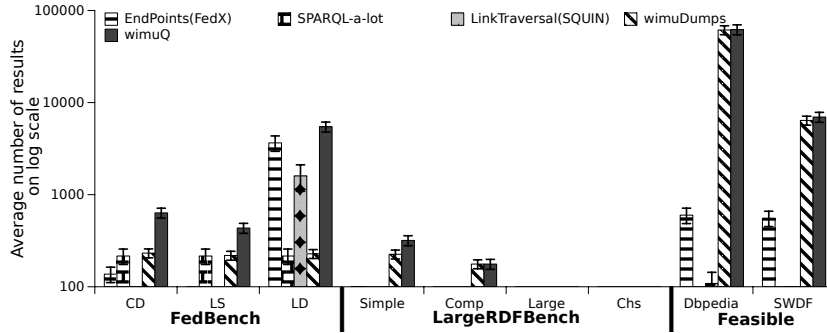


Figure 27: Average number of results retrieved by the selected approaches across different queries categories of the selected benchmarks.

For FedBench, the WimuDumps avg. resultset size is 2,253. Out of these results, about 55% are collected from SPARQL endpoints by using FedX query processing engines (avg. resultset size 1,262). The LinkTraversal (SQUIN) contributed about 25% of the total results (avg. resultset size 549), wimuDumps contributed about 10% of the total results (avg. resultset size 226), and SPARQL-a-lot also contributed about 10% of the results (avg. resultset size 215).

For LargeRDFBench, the WimuDumps avg. resultset size 123. Out of these results, about 81% are collected from wimuDumps (avg. resultset size 100). The SPARQL endpoints contributed about 14% of the results (avg. resultset size 17). The LinkTraversal(SQUIN) contributed 6% of the total results (avg. resultset size 6), and SPARQL-a-lot did not provide results (avg. resultset size 0).

For FEASIBLE, the WimuDumps avg. resultset size 34,537. Out of these results, about 98% are collected from wimuDumps (avg. resultset size 33,893). The SPARQL endpoints contributed about 1.6% (avg. resultset size 577). The LinkTraversal(SQUIN) approach contributed

only about 0.15% (avg. resultset size 54). Finally, SPARQL-a-lot query processing engine only retrieved about 0.03% results (avg. resultset size 11).

In summary, WimūQ is able to retrieve at least one resultset for 76% of the overall 415 queries. The results clearly shows that by combining different query processing engines into a single SPARQL query execution framework lead towards more complete resultset retrieval. An important observation is that the selected approaches are mostly not able to retrieve results for the Large Data and Complex+High (Ch) queries categories of LargeRDFBench. The reason is getting zero results for the Large Data queries is that these queries retrieve results from the LinkedTCGA [95] datasets which were not publicly available via SPARQL endpoints, were not indexed by WIMU, also were not reachable via link traversals. While Ch queries often require higher number of distributed datasets in order to compute the final resultset of the queries. Thus, the approaches were able to find all of the relevant datasets, required to compute the final resultset of the queries. The average number of datasets<sup>9</sup> queries by WimūQ for the selected benchmarks queries categories in given in Figure 28. We can clearly see the highest number of datasets are selected for Ch queries of LargeRDFBench.

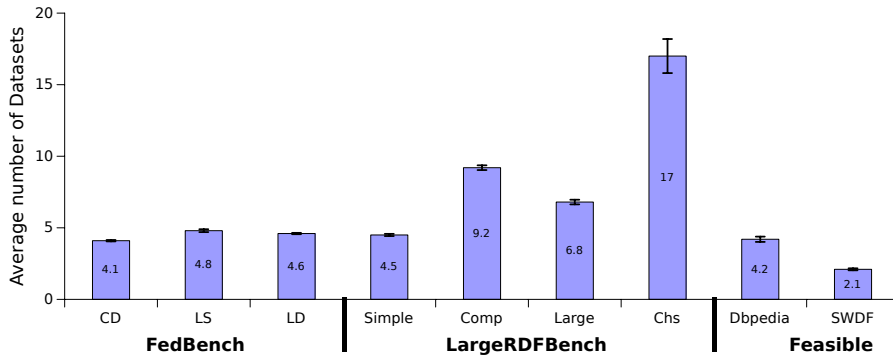


Figure 28: Average number of datasets discovered and queried by WimūQ across different queries categories of the selected benchmarks.

**Query runtime performances:** figure 29 shows a comparison of the selected query processing engines in terms of the average query run times for the different queries categories of the selected benchmarks. The average query runtime of WimūQ is 17 minutes across the three benchmarks. The average query execution to collect results from wimuDumps is about 2 minutes, which in turn is followed by query execution over SPARQL endpoints (avg. query runtime 13 minutes), SPARQL-a-lot (avg. query runtime 58 seconds) and LinkTraversal (SQUIN) (avg. query runtime 36 seconds). Interestingly, WimūQ collects about 91% of the results from wimuDumps yet its average execution time is smaller than query execution over SPARQL endpoints

<sup>9</sup> Here we also point the number of datasets discovered

which provide only 7% of the total results. One possible reason for this could be that in SPARQL endpoint federation, the query processing task split among multiple selected SPARQL endpoints and hence network and the number of intermediate results play an important role in the quer runtime performances.

For FedBench, the WimuQ average query runtime 20 minutes. Out of this, the average avg. query runtime over SPARQL endpoints is 16 minutes, followed by wimuDumps (avg. query runtime 2 minutes), LinkTraversal (SQUIN) (avg. query runtime 49 seconds), and SPARQL-a-lot (avg. query runtime 46 seconds), respectively. For LargeRDFBench, the average query execution of WimuQ is 11 minutes. Out of this query execution over SPARQL endpoints took 8 minuts on average, followed by wimuDumps (avg. query runtime 2 minutes), SPARQL-a-lot (avg. query runtime 35 seconds), and LinkTraversal (SQUIN) (avg. query runtime 25 seconds), respectively. For FEASIBLE, the WimuQ takes on average of 24 minutes per query execution. Out of this, the query federation over SPARQL endpoints took about 18 minutes on average. Which is followed by wimuDumps (avg. query runtime 3 minutes), SPARQL-a-lot (avg. query runtime 1), and LinkTraversal (SQUIN) (avg. query runtime 36 seconds), respectively.

As an overall query runtime evaluation, we can clearly see there is a trade-off between the recall and query runtimes: the highest the recall the highest the query runtimes. Finding a balance between the recall and runtime would be an interesting research question to be considered in the future.

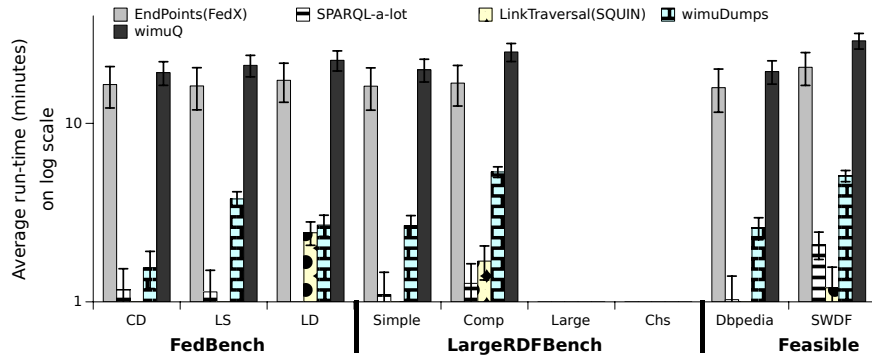


Figure 29: Average query runtimes of the selected approaches across different queries categories of the selected benchmarks.

The results of our evaluation lead us to validate our hypothesis that we can improve the resultset retrieval if we identify potentially relevant sources from heterogeneous RDF data.

## CONCLUSION

---

In this chapter we summarize our research work, highlight our core contributions and give general conclusions and future directions.

### 10.1 IDENTIFYING DATASETS IN LARGE HETEROGENEOUS RDF SOURCES

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

### 10.2 RELATING LARGE AMOUNT OF RDF DATASETS

We present a method to create a repository to store the similarity between a large number of datasets involving the detection of duplicated datasets and dataset chunks.

The method involves the creation of an index over a total of 668,166 datasets from LOD Stats and LOD Laundromat as well as 559 active SPARQL endpoints, representing a total of 221.7 billion triples from more than 5 terabytes of information from datasets partially retrieved using the service “Where is My URI” (WIMU).

For the first time, to the best of our knowledge, we make a relation index that can query by dataset URI, property, class, or SPARQL query.

Our experiments show that more than 90% of datasets from LOD-Laundromat datasets are not using `owl:equivalentProperty` and `owl:equivalentClass` or another way to relate the data, reinforcing the need for an index of relations among LOD datasets.

We realize that the Datasets still not sharing the expected properties, and the ontologies are not aligned, making hard the task to query multiple heterogeneous datasets. We believe that this work can help people to identify similar datasets among a large number of datasets. As future work, the next step is to improve the GUI drawing

a weighted graph that shows the similarity level of each dataset. The source-code is available online<sup>1</sup>.

### 10.3 OBTAINING SIMILAR RESOURCES USING STRING SIMILARITY

We presented an approach to reduce the computation runtime of similarity joins using the Most Frequent k Characters algorithm with a sequence of filters that allow discarding pairs before computing their actual similarity, thus reducing the runtime of computation. We proved that our approach is both correct and complete. The evaluation shows that all filter setup outperform the naive approach. Our parallel implementation works better in larger datasets with size greater than  $10^5$  pairs. It is also the key to developing systems for Record Linkage and Link Discovery in knowledge bases. As future work, we plan to integrate it in link discovery applications for the validation of equivalence links. The source code is free and available online.<sup>2</sup>

### 10.4 HETEROGENEITY IN DBPEDIA IDENTIFIERS

An approach was provided to tackle the heterogeneity working with `owl:sameAs` redundancies that were observed during researching co-references between different data sets and providing a unique DBpedia identifier and give the chance to rate the resulting links and make suggestions.

A proof of concept was implemented as a computer web system in order to present and validate our idea and every concept of this work. The source code is available <sup>3</sup>.

Was noticed in our results that there are benefits when a considerable number of `owl:sameAs` redundancies can be avoided. Rating the links allow users to make link suggestions brings more quality to the repository, and the stand alone service on the web allow you to use the DBpediaSameAs also in a command line textual environment and can be used as an off-the-shelf component.

For the future we plan to: (1) make a study about the results of link rating. This needs a period of usage of the DBpediaSameAs service in order to gather sufficient results for proper analysis. (2) An implementation case with more members of the DBpedia community. A study about how will be the behavior when implement with the DBpedia community.

<sup>1</sup> <https://github.com/firmao/LODDatasetRelationsWeb>

<sup>2</sup> <https://github.com/firmao/StringSimilarity/>

<sup>3</sup> <https://github.com/firmao/DBPediaLinkSameAs.git>

## 10.5 DETECTION OF ERRONEOUS LINKS IN LARGE-SCALE RDF DATASETS

In this paper, we present CEDAL, a new algorithm that allows tracking consistency problems inside linkset repositories. Our approach allows detecting potential causes of errors, for example the linkset, the underlying dataset(s) and the graph path where the problem subsists. We showed that our approach scales well. In particular, we reduced the complexity of obtaining clusters by relying on graph partitions, thus decreasing the time complexity from  $O(n^2)$  to  $O(m \log n)$ . Our results showed that at least 13% of owl:sameAs links we considered are erroneous, and algorithms LIMES, SILK and DBpedia Extraction Framework have a better consistency index than repositories such as *sameas.org*.

In future work, we will carry out a survey on Linkset quality. To the best of our knowledge, the survey [127] is the most complete collection of data quality measures, which, however, misses specific measures for linkset quality, such as ways a linkset can improve dataset quality [6; 123; 20]. Moreover, we will investigate how our graph partition algorithms can improve the performance of SPARQL endpoints by distributing resources among computer clusters, core processors, and GPUs. The CEDAL repository<sup>4</sup> contains all necessary resources to run CEDAL, verify and reproduce our results.

## 10.6 QUERYING LARGE HETEROGENEOUS RDF DATASETS

We presented an approach to execute SPARQL queries over a large amount of heterogeneous RDF data sources available from different interfaces and in different formats. We made use of the Wimur service to identify the potentially relevant sources to the given SPARQL query. We discussed two main types of federated SPARQL query processing approaches namely the endpoints federation and traversal-based federation. The former type of federation only able to execute federated queries over the data available from SPARQL endpoint. While the later, faces problem of URI's dereferenceability. To overcome these issues we proposed a hybrid (endpoints+link-traversal-based) federation engines which integrates four different types of SPARQL query processing engines. Currently, WimurQ able to execute both federated and non-federated SPARQL queries over a total of 668k datasets available from LOD Stats, LOD Laudromat, and LOD cloud active SPARQL endpoints. We evaluated WimurQ by using three state-of-the-art real-data SPARQL benchmarks. We showed that WimurQ is able to successful execute (with some results) majority of the benchmark queries without any prior knowledge of the data

<sup>4</sup> <https://github.com/firmao/CEDAL>



sources. In addition, the WimuD resultset recall is higher with reasonable query execution times.

As future, we will add more URIs into the the WIMU index in order to retrieve more complete and fast results. Furthermore, we will add TPF interfaces and query execution engines such Comunica [108] and SAGE [66] into the WimuD query execution engine. We will continue maintaining<sup>5</sup> and developing, extending WimuD to include the publicly-available triple pattern fragments interfaces as well<sup>6</sup>.

---

<sup>5</sup> Sustainability plan: <https://github.com/firmao/wimuT/wiki/Sustainability-plan>

<sup>6</sup> In case of more information is needed we have an extended version of this paper available at [http://139.18.13.76:8082/KCAP\\_extended.pdf](http://139.18.13.76:8082/KCAP_extended.pdf)

## APPENDIX

*This section complements this work presenting an example of applications of some concepts presented in this thesis.*

### A.1 DETECTING ERRONEOUS LINK CANDIDATES IN EDUCATIONAL LINK REPOSITORIES

*Interlinking educational datasets is a challenging task when the majority of data is not organized in a Linked Data standards. Thus, we face data that is not well interlinked among several different datasets of the educational area. This task usually involves many steps such as Link Discovery, where links across homogeneous datasets are discovered and stored, and quality assessment of these links, where the consistency of these links is verified. However, a large number of these links are erroneous and can thus lead to these applications producing absurd results. The aim of this paper is to provide a way to check the consistency of linksets from Educational area. To this end, we are using an approach called CEDAL which is a time-efficient and complete approach for the detection of erroneous links for properties that are transitive. In order to evaluate our approach we selected 1,049 transitive from linksets from Nature.com, Geonames, educational Programs, among others.*

#### A.1.1 Introduction

Links across knowledge bases play a fundamental role in Linked Data [6] as they allow users to navigate across datasets, integrate Linked Data sources [75], perform federated queries [91] across data sources and perform large-scale inference on the data. Given the importance of links, corresponding repositories such as *nature.com*<sup>1</sup> and LinkLion [70] have been created. In addition to facilitating the finding of links between resources and knowledge bases, these repositories also allow detecting significant errors across links. For example, according to Nature.com and by virtue of transitivity, the resources. <http://purl.org/spar/fabio/ExpressionCollection> and <http://purl.org/ontology/bibo/Collection> stand for the same entity of the real world but have different URIs within the same knowledge base. This clearly goes against the definition of URIs as used in RDF.

figure 30 shows a fictional example to help illustrate such problems, which can be classified as **contradiction problems**, according to the

<sup>1</sup> <http://nature.com/>

quality dimension of consistency [127]. In our example, we can infer that one of the links along the path that led to this inference is wrong or that the knowledge base in itself contains an error.

Educational data needs to be enriched, that is transformed into structured and formal descriptions by linking it to widely established Linked Data vocabularies and datasets on the web also linking these datasets in a efficient way in order to avoid redundancy.

In this context we found linksets from Nature.com, Educational Programs<sup>2</sup> and Geonames, in which such errors can be potentially detected by computing the closure of equivalence links using the characteristics of equivalence relations and an inference engine. In our case we are using CEDAL [?] which is the fastest and complete way to detect consistency errors.

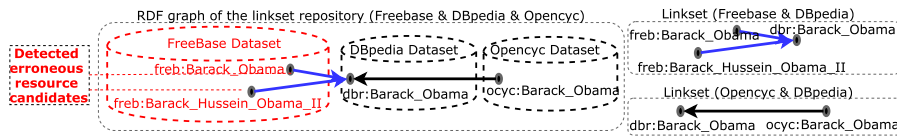


Figure 30: Manual detection of erroneous resource candidates.

### Research questions

The research questions are:

- The link repositories from the educational area are consistent?
- Is there an approach to verify whether the links from educational area are consistent?

### Our contributions

- An analysis of linksets from educational area.
- Applying CEDAL[115] in order to assess the consistency of linksets from educational area.

#### A.1.2 Related works

Works such as [81; 27; 28] present an effort to interlinking educational resources into linked data standards, but without taking care of the problem of the consistency of the linksets.

The work presented in [23] is a research note about a study of available web datasets related to education. They focus in a network of educational web data. The common point is that part of the analysis is about of owl:sameAs links from education area and the novelty

<sup>2</sup> [https://datahub.io/dataset/educationalprograms\\_sisvu](https://datahub.io/dataset/educationalprograms_sisvu)

of our work is about we are taking care about linksets and not only datasets.

There is a study more generic and comprehensive about Linked Data for science in general and education [52], in which makes an introductory analysis of four papers documenting the most recent developments in Linked Data for Science and Education.

We found a work about Information and Communication Technologies (ICT) tools to support learning activities with a Linked Data approach for the discovery of educational ICT tools in the Web of Data [83], in which proposes to collect data from third-party sources, align it to a vocabulary understandable by educators and finally publish it to be consumed by educational applications. The relation here is the effort to turn educational data in linked data and improve the quality.

The work [125] has the aim of introduce the current efforts toward the release and exploitation of The Open University's (OU) Linked Open Data (LOD) referring to production and consumption. The relation here is about bring some improvement to the educational linked data, but without taking care of quality of linksets.

### A.1.3 Preliminaries

**transitive property.** Defined by:  $\forall a, b, c \in X : (aRb \wedge bRc) \implies aRc$ , where  $R$  represents a relation between two elements of a set  $X$ .

**equivalence.** An equivalence relation  $\equiv$  is a binary relation that is reflexive, symmetric and transitive. According to OWL semantics,  $owl:sameAs$  is an equivalence relation.

**linkset.** According to the W3C,<sup>3</sup> a linkset is a collection of RDF links between two datasets. It is a set of RDF triples in which all subjects are in one dataset and all objects are in another dataset. RDF links often have the  $owl:sameAs$  predicate, but any other property could occur as the predicate as well. Formally, according to [7], a linkset  $LS$  is a set of RDF triples where for all triples  $t_i = \langle s_i, p_i, o_i \rangle \in LS$ , the subject is in one dataset, i.e. all  $s_i$  are described in  $S$ , and the object is in another dataset, i.e. all  $o_i$  are described in  $T$ .

We use the word *linkset* for either RDF knowledge base files and dump files from RDF link repositories.

**rdf graph partitioning.** Given a graph  $G = (V, E, lbl, L)$ , a graph partitioning,  $C$ , is a division of  $V$  into  $k$  partitions  $P_1, P_2, \dots, P_k$  such that  $\bigcup_{1 \leq i \leq k} P_i = V$ , and  $P_i \cap P_j = \emptyset$  for any  $i \neq j$ . The edge cut  $E_c$  is the set of edges whose vertices belong to different

<sup>3</sup> <https://www.w3.org/TR/void/#linkset>

partitions,  $\text{lbl} : E \cup V \rightarrow L$  is a labeling function, and  $L$  is a set of labels.

The definition comes from a recent survey about RDF graph partitioning [110].

**error types.** This paper uses two types of errors defined in [? ].

(1) **Semantic accuracy** errors, in which we detect if data values correctly represent the real world facts. (2) **Consistency and Conciseness** errors where a knowledge base is free of logical or formal contradictions concerning particular knowledge representation and inference mechanisms and the minimization of redundancy of entities at the schema and the data level. figure 31 shows a fictional example of both error types, in which we represent links between GeoNames<sup>4</sup> and DBpedia.<sup>5</sup>

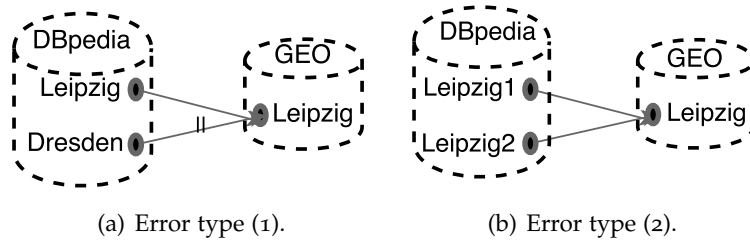


Figure 31: Detected error types.

In this example, figure 31(a) shows an error of type (1), in which an erroneous `owl:sameAs` link between the city of Dresden and the city of Leipzig was detected. The figure 31(b) shows an error of type (2) where the resource about the city of Leipzig is duplicated within the DBpedia dataset.

#### A.1.4 The Consistency Error Detection Algorithm

In order to detect erroneous links among linksets from the educational area, we will use an Consistency Error Detection Algorithm (CEDAL) [? ].

The CEDAL targets consistency errors in large-scale link repositories. Our method assumes a union of linksets  $\mathcal{L}$  as given input. The aim is to find cases in which equivalent resources (according to the OWL semantics) in  $\mathcal{L}$  share the same dataset. The basic intuition here is equivalent resources (i.e., resources that stand for the same entity from the real world) being in one knowledge base is a clear hint towards (1) an error in the knowledge base itself or (2) an error during the generation of the links that allowed generating this equivalence.

<sup>4</sup> <http://www.geonames.org/>

<sup>5</sup> <http://dbpedia.org/>

In figure 32, we show how the algorithm works. Given datasets  $D_1, \dots, D_n$ , resources  $R = \{a, b, c, d, e, f\}$ , the idea is to detect inside clusters two or more resources sharing the same dataset.

The algorithm works by partitioning a graph inside an adjacency list that contains: (1) An array per vertex for a total of  $V$  arrays, where we are only considering the space for the array pointer, not the contents of the array. (2) Each directed edge is contained once somewhere in the adjacency list, for a total of  $E$  edges, where a bidirectional edge is just 2 directed edges, assuming we are using bi-directional edges.

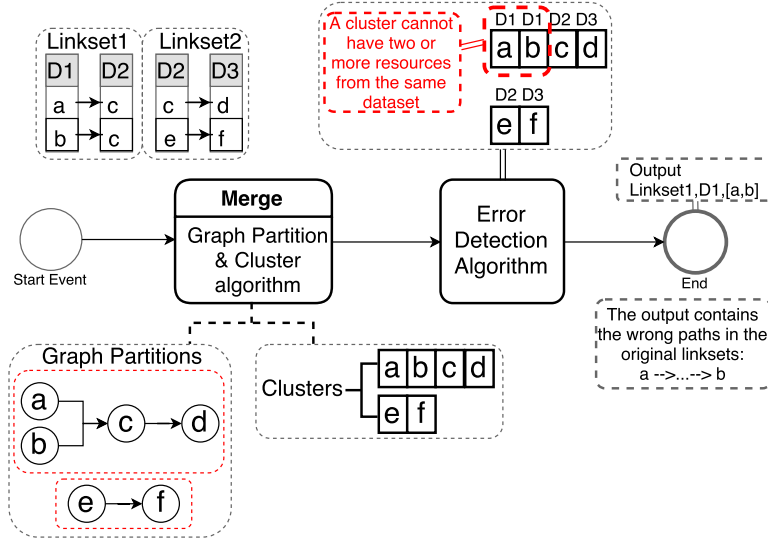


Figure 32: Error detection.

#### A.1.5 Evaluation

The evaluation shows the detected erroneous link candidates and fixing the linksets with problems.

##### Experimental setup

We evaluated our work using the datasets from Linked Education<sup>6</sup> listed in table 12, in which the main datasets are from *educational Programs*<sup>7</sup> and dump-files<sup>8</sup> from Nature<sup>9</sup>, where we used a sample of 1049 links.

The CEDAL can be applied only to transitive properties, that are respective *skos:exactMatch*, *owl:sameAs* and *skos:equivalentClass*, in which according to the table 13 are a total of 1,046 links. All linkset files from were copied locally to run the experiments.

<sup>6</sup> <https://linkededucation.wordpress.com/data-models/data-sets/>

<sup>7</sup> [https://datahub.io/dataset/educationalprograms\\_sisvu](https://datahub.io/dataset/educationalprograms_sisvu)

<sup>8</sup> <http://data.nature.com/downloads/latest/linksets/nq/>

<sup>9</sup> [nature.org](http://nature.org)

Table 12: Educational Datasets

Dataset	State	number of transitive links
<a href="http://data.cnr.it/sparql">http://data.cnr.it/sparql</a>	on-line	3907
<a href="http://meducator.open.ac.uk/resourcesrestapi/rest/meducator/sparql">http://meducator.open.ac.uk/resourcesrestapi/rest/meducator/sparql</a>	off-line	
<a href="http://data.open.ac.uk/query">http://data.open.ac.uk/query</a>	on-line	20956
<a href="http://asn.jesandco.org:8890/sparql">http://asn.jesandco.org:8890/sparql</a>	off-line	
<a href="http://services.data.gov.uk/education/sparql">http://services.data.gov.uk/education/sparql</a>	off-line	
<a href="http://knowone.csc.kth.se/sparql/ariadne-big">http://knowone.csc.kth.se/sparql/ariadne-big</a>	off-line	
<a href="http://data.nature.com/sparql/">http://data.nature.com/sparql/</a>	off-line	
<a href="http://seek.rkbexplorer.com/sparql/">http://seek.rkbexplorer.com/sparql/</a>	off-line	

Table 13: Educational Linksets

Linkset (Source x Target)			
Source	Target	Links (triples)	Type
educationalPrograms	geonames	792	owl:sameAs
nature.com	dbpedia.org/resource	94	skos:exactMatch
nature.com	wikidata.org	92	skos:exactMatch
nature.com	schema.org	13	owl:equivalentClass
nature.com	purl.org	26	owl:equivalentClass
nature.com	cidoc-crm.org	10	owl:equivalentClass
nature.com	dbpedia.org/ontology	9	owl:equivalentClass
nature.com	xmlns.com	5	owl:equivalentClass
nature.com	vivoweb.org	5	owl:equivalentClass

The experiments were performed using a laptop with Intel Core i7, 8 GB RAM, a video card NVIDIA NVS4200, Operational System Linux Ubuntu 14.4 and Java SE Development Kit 8. The results including the output file are available on CEDAL online repository<sup>10</sup>. All links were processed by our algorithm in 190 milliseconds. The total amount of errors were 20 of candidates, with 2 different domains and 2 knowledge base files.

#### *Ranking the erroneous candidates*

To evaluate how effective CEDAL detect the erroneous links in educational linksets, CEDAL has a score in order to rank the erroneous candidates based on the number of detected resources with errors.

The score  $\mu$  is calculated by  $\mu = \frac{|C|(|C|-1)}{2}$ , in which we use the cardinality of  $C$  representing the detected erroneous candidates.

The figure 33 shows the the number of erroneous candidates according the rank score.

<sup>10</sup> <https://github.com/firmao/CEDAL/tree/master/CEDALEducation>

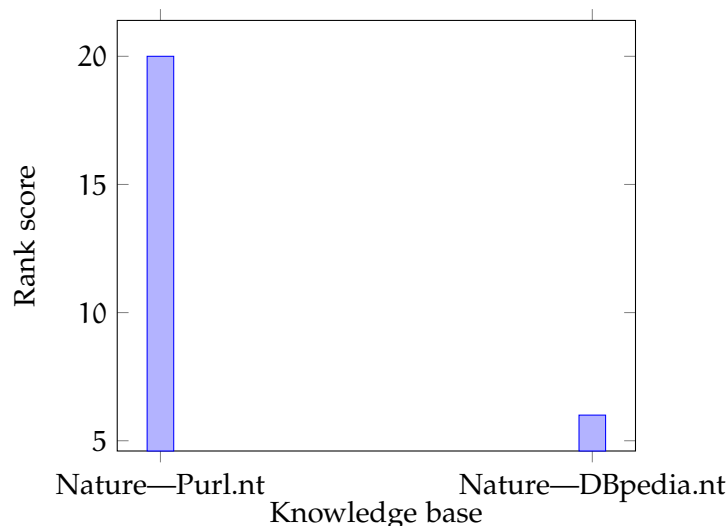


Figure 33: Error rank, where Nature—Purl.nt represent the file with links between Nature and Purl.org and Nature—DBpedia.nt represent the links between Nature and DBpedia

According to our rank, the knowledge-base with more errors comes from `npg – core – any – linkset.nq.nt – – – purl.org.nt` with 25 links per mapping, in which we found 20 erroneous resource candidates. The knowledge base with fewer errors was `npg – journals – dbpedia – linkset.nq.nt – – – dbpedia.org.nt` with 94 links per mapping, in which we found 6 erroneous resource candidates and a sum of score of 3.

#### *Analyzing and Fixing the problems*

We use an example from CEDAL output file<sup>11</sup> to show how to fix a Linkset. Among the errors detected we choose the linkset file `npg – core – any – linkset.nq.nt`, in which has a case where 2 error candidates that are transitive and are in the same dataset. In this case, we have an error type 2, where the following resource [http://dbpedia.org/resource/Cell\\_Death\\_and\\_Differentiation](http://dbpedia.org/resource/Cell_Death_and_Differentiation) is duplicated within the DBpedia dataset.

#### *A.1.6 Conclusion and future work*

In this paper, we use an approach dubbed CEDAL in order to assess consistency problems inside linkset repositories from the educational area. CEDAL allows detecting potential causes of errors, for example, the linkset, the underlying dataset(s) and the graph path where the problem subsists. To evaluate the approach, we selected 1049 transitive links using datasets from Nature.com and Educational Programs. Our results showed that 20 links we considered are erroneous.

<sup>11</sup> <https://github.com/firmao/CEDAL/blob/master/CEDALEducation/output.csv>



We show a scenario where is possible to see how fragile is the educational linked data and a way to improve it detecting consistency problems. As future work we will apply this approach to more educational linked data, extend CEDAL to work with more link types and provide a usability study.

## CV

## B.1 ABSTRACT

In 1998 I started working in companies in the computer science area and academic research since 2004. I studied Computer Science, from 1999-2003, at the University of Grande ABC in São Paulo/Brazil, where I obtained a bachelor's degree. During my studies, I was always a curious and research-active student because I knew that to pursue a scientific career, one must begin early. I did research in various areas, one of which was Digital Interactive Television. This allowed me to participate and contribute to LINCOM (Laboratory of Interactivity Computation and Multimedia) of Technological Institute of Aeronautics, where we focused on the development of a Framework for hyperprograms for Interactive Television and I obtained my Master's degree. In October of 2014, I join the AKSW research group as a Ph.D. student, start working with Linked Data, Data Integration and Large Collection of RDF datasets. I've worked for a while on research projects for academy and companies.

## PROGRAM COMMITTEE MEMBER / REVIEWER

- CSCUBS (<http://cscubs.cs.uni-bonn.de/>)
- INNOV (<https://www.iaria.org/conferences/INNOV.html>)
- FutureTech (<http://www.futuretech-conference.org/>)
- QuWeDa 2019 (<https://sites.google.com/site/quweda2019/home>)
- The Semantic Web Journal (<http://www.semantic-web-journal.net/>)

## MAIN PROJECTS

- **Project Name: Platona - Shacl GUI Editor**  
**Company:** Instituts für Angewandte Informatik (InfAI) an der Universität Leipzig  
**Role:** Software Engineer  
**Start date :** 03/2020.  
**Description :** Full-stack software development applied to the development of a Graphical User Interface to use SHACL (<https://www.w3.org/TR/shacl/>) shapes RDF constraints.

- **Project Name: Telepark**

**Company:** Instituts für Angewandte Informatik (InfAI) an der Universität Leipzig

**Role:** Software Engineer

**Start date :** 10/2019

**End date 0.5in :** 03/2020

**Description :** Full-stack software development applied to medical systems, including the application of FHIR standard, recognition of barcodes type DataMatrix. Technologies: Java, Kotlin, JavaScript, React, SpringBoot, PostgreSQL, Docker.

- **Project Name: Automatic Dependent Surveillance System (SVDA)**

**Client:** Foundation Technologies Critical Applications (ATECH)

**Role:** Researcher

**Start date :** 08/2011.

**End date 0.5in :** 01/2013

**Description :** Researching technologies for Air Traffic Control to increase airspace utilization, operational efficiency, flexibility and mainly its safety. Applied to multilateralism and hyperbolic positioning algorithms on Systems of Automatic Dependence Surveillance. Technologies: Java, C++, Matlab.

- **Project Name: Harpia project-Risk Analysis and Applied Artificial Intelligence**

**Client:** Brazil's Federal Revenue and Technology Institute of Aeronautics

**Role:** Programmer

**Environment:** RAD 7, Rational RequisitePro, RSA 7, Java, Hibernate and Spring.

**Start date :** 02/11/06 **end date:** 02/15/08

**Description :** Implementation of project of artificial intelligence

**Responsibilities:** J2EE development, which involves the implementation of security systems and artificial intelligence algorithms applied to systems for the Federal Revenue of Brazil. We use Oracle as our relational database system, Java and Hibernate to deal with the data and business rules and very strict quality assurance, involving the quality of code and security as well.

- **Work Title: System Analyst.**

**Company:** Municipal Government of Santo André

**Role:** Systems analyst.

**Environment:** Oracle and C# .

**Start date:** 03/2011

**End date** : 09/2014

**Description** : System development using C# , Java, and Oracle as database system, in which I worked specifically on data integration of Geographical Information Systems (GIS) and financial systems.

- **Work Title: System Analyst.**

**Company:** CPMBraxis / Itaú Bank

**Role:** Outsourced work within Itaú Bank.

**Environment:** Windows, Java, DB2, Eclipse, Rational Rose, Clear Case, Visual Basic, Visual Studio .NET, C# , VB.Net.

**Start date:** 02/24/08.

**End date** : 01/03/10

**Description** : CPMBraxis provides services in IT Consulting.

**Responsibilities:** Develop and specify computer systems for financial institutions such as banks. The majority of the systems developed were basically to provide communication between Mainframe systems and web systems, such as cobol/db2 to java/oracle.

- **Work Title: Programmer.**

**Company:** CredSystem Administrator of credit cards.

**Role:** Outsourced work within CredSystem.

**Environment:** Java, Firebird DB, Linux, Windows, CVS.

**Start date:** 06/24/05

**End date** : 01/03/06

**Description** : Development of system for data transmission and control.

**Responsibilities:** Specification and implementation of computer systems for Credit card systems, in which were developed drivers for communication with card machines and management of data from database systems such as firebird.

## EDUCATION

- **Bachelor:** Monography title (CSM Language - A new programming language for learning). Bachelor in Computer Science at Universidade do Grande ABC. 2003.
- **Masters:** Thesis title(CossackIDE: An Environment for the development and execution of Hyperprograms. Master in Science area of Computer Engineer at Technology Institute of Aeronautics. 2009.
- **Ph.D.:** Thesis title(Identifying, Querying and Relating Large Heterogeneous RDF Sources). Ph.D. in Computer Science at Universität Leipzig.

TECHNICAL TRAINING

- SUN CERTIFIED PROGRAMMER FOR JAVA[tm] 2 PLATFORM.
- SUN CERTIFIED WEB COMPONENT DEVELOPER FOR J2EE[tm] PLATFORM.

## BIBLIOGRAPHY

---

- [1] Abdelaziz, I., Mansour, E., Ouzzani, M., Aboulmaga, A., and Kalnis, P. (2017). Lusail: a system for querying linked data at scale. *Proceedings of the VLDB Endowment*, 11(4):485–498. (Cited on page 14.)
- [2] Acosta, M., Vidal, M.-E., Lampo, T., Castillo, J., and Ruckhaus, E. (2011). ANAPSID: An Adaptive Query Processing Engine for SPARQL Endpoints. In Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., and Blomqvist, E., editors, *The Semantic Web – ISWC 2011*, volume 7031 of *Lecture Notes in Computer Science*, pages 18–34. Springer Berlin Heidelberg. (Cited on pages 14 and 15.)
- [3] Akar, Z., Halaç, T. G., Ekinci, E. E., and Dikenelli, O. (2012). Querying the Web of Interlinked Datasets using VoID Descriptions. In C.Bizer et al., editors, *Linked Data on the Web (LDOW2012) in CEUR Workshop Proceedings*, volume 937. (Cited on page 15.)
- [4] Albertoni, R., De Martino, M., and Podesta, P. (2015). A linkset quality metric measuring multilingual gain in skos thesauri. In *Proceedings of the 2nd Workshop on Linked Data Quality (LDQ2015), Portorož, Slovenia*. (Cited on pages 12 and 13.)
- [5] Albertoni, R., De Martino, M., and Podestà, P. (2016). Linkset quality assessment for the thesaurus framework lustre. In *International Conference, MTSR 2016, Göttingen, Germany, November 22-25, 2016, Proceedings*. Springer. (Cited on page 13.)
- [6] Albertoni, R. and Pérez, A. G. (2013). Assessing linkset quality for complementing third-party datasets. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, EDBT '13, New York, NY, USA. ACM. (Cited on pages 12, 57, 83, and 85.)
- [7] Alexander, K., Cyganiak, R., Hausenblas, M., and Zhao, J. (2009). Describing linked datasets. In *LDOW*. (Cited on pages 7 and 87.)
- [8] Andre Valdestilhas, Tommaso Soru, M. S. E. M. W. B. C. S. B. P. N. K. H. T. R. (2020). Identifying, querying, and relating large heterogeneous rdf sources. *Semantic Web Journal (Under review)*. (Cited on pages VI, XI, 71, and 72.)
- [9] Arenas, M., Gutierrez, C., and Pérez, J. (2008). An extension of sparql for rdfs. In *Semantic Web, Ontologies and Databases*. Springer. (Cited on page 58.)

- [10] Asprino, L., Beek, W., Ciancarini, P., van Harmelen, F., and Presutti, V. (2019a). The linked open data cloud is more abstract, flatter and less linked than you may think! *arXiv preprint arXiv:1906.08097*. (Cited on page 10.)
- [11] Asprino, L., Beek, W., Ciancarini, P., van Harmelen, F., and Presutti, V. (2019b). Observing lod using equivalent set graphs: it is mostly flat and sparsely linked. In *International Semantic Web Conference*, pages 57–74. Springer. (Cited on page 10.)
- [12] Asprino, L., Beek, W., Ciancarini, P., van Harmelen, F., and Presutti, V. (2019c). Triplifying equivalence set graphs. In *2019 ISWC Satellite Tracks (Posters and Demonstrations, Industry, and Outrageous Ideas), ISWC 2019-Satellites*, pages 253–256. CEUR-WS. (Cited on page 10.)
- [13] Auer, S., Demter, J., Martin, M., and Lehmann, J. (2012). Lodstats—an extensible framework for high-performance dataset analytics. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 353–362. Springer. (Cited on pages 2, 9, 18, 23, and 72.)
- [14] Auer, S., Lehmann, J., and Ngomo, A.-C. N. (2011). Introduction to linked data and its lifecycle on the web. In *Reasoning Web*, pages 1–75. (Cited on page 3.)
- [15] Baron Neto, C., Kontokostas, D., Kirschenbaum, A., Publio, G., Esteves, D., and Hellmann, S. (2017). Idol: Comprehensive & complete lod insights. In *Proceedings of the 13th International Conference on Semantic Systems (SEMANTiCS 2017)*. (Cited on page 10.)
- [16] Baron Neto, C., Müller, K., Brümmer, M., Kontokostas, D., and Hellmann, S. (2016). Lodvader: An interface to lod visualization, analytics and discovery in real-time. In *WWW Companion volume*. (Cited on page 13.)
- [17] Beek, W., Rietveld, L., Bazoobandi, H. R., Wielemaker, J., and Schlobach, S. (2014). Lod laundromat: a uniform way of publishing other people’s dirty data. In *International Semantic Web Conference*, pages 213–228. Springer. (Cited on pages 2, 9, 13, 18, 23, 35, 70, 72, and 75.)
- [18] Bizer, C., Heath, T., and Berners-Lee, T. (2011). Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227. IGI Global. (Cited on page 1.)
- [Bock et al.] Bock, J., Haase, P., Ji, Q., and Volz, R. Benchmarking owl reasoners. (Cited on pages 58 and 68.)

- [20] Casanova, M. A., Vidal, V. M., Lopes, G. R., Leme, L. A. P. P., and Ruback, L. (2014). On materialized sameas linksets. In *International Conference on Database and Expert Systems Applications*. Springer. (Cited on pages 13 and 83.)
- [21] Charalambidis, A., Troumpoukis, A., and Konstantopoulos, S. (2015). Semagrow: Optimizing federated sparql queries. In *Proceedings of the 11th International Conference on Semantic Systems, SEMANTICS '15*, pages 121–128, New York, NY, USA. ACM. (Cited on pages 14 and 22.)
- [22] Colpaert, P., Verborgh, R., Mannens, E., and Van de Walle, R. (2014). Painless uri dereferencing using the datatank. In *ESWC*. (Cited on page 9.)
- [23] d’Aquin, M., Adamou, A., and Dietze, S. (2013). Assessing the educational linked data landscape. In *Proceedings of the 5th Annual ACM Web Science Conference*, pages 43–46. ACM. (Cited on page 86.)
- [24] de Melo, G. (2013). Not quite the same: Identity constraints for the web of linked data. In *AAAI*. (Cited on page 14.)
- [25] Debattista, J., Attard, J., Brennan, R., and O’Sullivan, D. (2019). Is the lod cloud at risk of becoming a museum for datasets? looking ahead towards a fully collaborative and sustainable lod cloud. In *Companion Proceedings of The 2019 World Wide Web Conference, WWW '19*, page 850–858, New York, NY, USA. Association for Computing Machinery. (Cited on page 33.)
- [26] Delpeuch, A. (2019). A survey of openrefine reconciliation services. *CoRR*, abs/1906.08092. (Cited on page 10.)
- [27] Dietze, S., Sanchez-Alonso, S., Ebner, H., Qing Yu, H., Giordano, D., Marenzi, I., and Pereira Nunes, B. (2013). Interlinking educational resources and the web of data: A survey of challenges and approaches. *Program*, 47(1):60–91. (Cited on page 86.)
- [28] Dietze, S., Yu, H. Q., Giordano, D., Kaldoudi, E., Dovrolis, N., and Taibi, D. (2012). Linked education: Interlinking educational resources and the web of data. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing, SAC '12*, pages 366–371, New York, NY, USA. ACM. (Cited on page 86.)
- [29] Dreßler, K. and Ngomo, A.-C. N. (2014). On the efficient execution of bounded jaro-winkler distances. (Cited on page 47.)
- [30] Emaldi, M., Corcho, O., and López-de Ipina, D. (2015). Detection of related semantic datasets based on frequent subgraph mining. *IESD@ ISWC*, 5(6):7. (Cited on page 10.)



- [31] Endris, K. M., Galkin, M., Lytra, I., Mami, M. N., Vidal, M.-E., and Auer, S. (2018). Querying interlinked data by bridging rdf molecule templates. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXIX*, pages 1–42. Springer. (Cited on page 14.)
- [32] Ermilov, I., Lehmann, J., Martin, M., and Auer, S. (2016). LOD-Stats: The data web census dataset. In *International Semantic Web Conference*, pages 38–46. Springer. (Cited on page 70.)
- [33] Euzenat, J., Shvaiko, P., et al. (2007). *Ontology matching*, volume 333. Springer. (Cited on page 39.)
- [34] Fernández, J. D., Beek, W., Martínez-Prieto, M. A., and Arias, M. (2017). Lod-a-lot: A queryable dump of the lod cloud. (Cited on pages 4, 9, and 70.)
- [35] Fernández, J. D., Martínez-Prieto, M. A., Gutiérrez, C., Polleres, A., and Arias, M. (2013). Binary rdf representation for publication and exchange (hdt). *Web Semantics: Science, Services and Agents on the World Wide Web*, 19(o). (Cited on page 70.)
- [36] Fletcher, G. H. (2008). An algebra for basic graph patterns. In *workshop on Logic in Databases, Rome, Italy*. (Cited on page 8.)
- [37] Georgala, K., Obraczka, D., and Ngomo, A.-C. N. (2018). Dynamic planning for link discovery. In *European Semantic Web Conference*, pages 240–255. Springer. (Cited on pages XIII, 33, and 37.)
- [38] Goldstine, H. H. and von Neumann, J. (1948). *Planning and coding of problems for an electronic computing instrument*. Institute for Advanced Study. (Cited on page 46.)
- [39] Görlitz, O. and Staab, S. (2011). 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. (Cited on pages 14 and 22.)
- [40] Guéret, C., Groth, P. T., Stadler, C., and Lehmann, J. (2012). Assessing linked data mappings using network measures. In *ESWS*. (Cited on page 12.)
- [41] Halpin, H., Hayes, P. J., McCusker, J. P., McGuinness, D. L., and Thompson, H. S. (2010). When owl: sameas isn’t the same: An analysis of identity in linked data. In *International Semantic Web Conference (1)*, pages 305–320. (Cited on page 51.)
- [42] Harris, S., Gibbins, N., and Payne, T. R. (2004). Semindex: Preliminary results from semantic web indexing. (Cited on page 9.)

- [43] Hartig, O. (2011). Zero-knowledge query planning for an iterator implementation of link traversal based query execution. In Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., and Pan, J., editors, *The Semantic Web: Research and Applications*, pages 154–169, Berlin, Heidelberg. Springer Berlin Heidelberg. (Cited on page 16.)
- [44] Hartig, O. (2013a). An Overview on Execution Strategies for Linked Data Queries. In *Datenbank-Spektrum*, volume 13, pages 89–99. Springer. (Cited on page 70.)
- [45] Hartig, O. (2013b). Squin: a traversal based query execution system for the web of linked data. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 1081–1084. ACM. (Cited on pages 15, 16, 73, and 77.)
- [46] Hartig, O., Bizer, C., and Freytag, J.-C. (2009). Executing sparql queries over the web of linked data. In *International Semantic Web Conference*, pages 293–309. Springer. (Cited on page 16.)
- [47] Hartig, O. and Özsu, M. T. (2016). Walking without a map: Ranking-based traversal for querying linked data. In *International Semantic Web Conference*, pages 305–324. Springer. (Cited on page 16.)
- [Hartmann] Hartmann, A.-K. Generating a large dataset for neural question answering over the dbpedia knowledge base. (Cited on page V.)
- [49] Hassan, M., Lehmann, J., and Ngomo, A.-C. N. (2015). Inter-linking: Performance assessment of user evaluation vs. supervised learning approaches. In *24th International World Wide Web Conference (WWW 2015): workshop: Linked Data on the Web (LDOW2015), Florence, Italy, May 18 to 22, 2015, Proceedings*. (Cited on page 11.)
- [50] Hoare, C. A. (1962). Quicksort. *The Computer Journal*, 5(1):10–16. (Cited on page 46.)
- [51] Jaccard, P. (1912). The distribution of the flora in the alpine zone. *New phytologist*, 11(2):37–50. (Cited on pages 11 and 48.)
- [52] Keßler, C., d’Aquin, M., and Dietze, S. (2013). Linked data for science and education. *Semantic Web*, 4(1):1–2. (Cited on page 87.)
- [53] Ladwig, G. and Tran, T. (2010). Linked Data Query Processing Strategies. In Patel-Schneider, P., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J., Horrocks, I., and Glimm, B., editors, *The Semantic Web – ISWC 2010*, volume 6496 of *Lecture Notes in Computer Science*, pages 453–469. Springer Berlin Heidelberg. (Cited on page 15.)

- [54] Ladwig, G. and Tran, T. (2011). SIHJoin: Querying Remote and Local Linked Data. In Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., and Pan, J., editors, *The Semantic Web: Research and Applications*, volume 6643 of *Lecture Notes in Computer Science*, pages 139–153. Springer Berlin Heidelberg. (Cited on page 15.)
- [55] Lehmann, J. (2009). DL-learner: learning concepts in description logics. *Journal of Machine Learning Research*, 10(Nov):2639–2642. (Cited on page 21.)
- [56] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., Kleef, P., Auer, S., and Bizer, C. (2014). DBpedia - a Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Sem. Web Journal*. (Cited on page 51.)
- [57] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., et al. (2015). DBpedia—a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195. (Cited on page 67.)
- [58] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. (Cited on page 11.)
- [59] Lopez, V., Uren, V., Sabou, M. R., and Motta, E. (2009). Cross ontology query answering on the semantic web: an initial evaluation. In *Proceedings of the fifth international conference on Knowledge capture*, pages 17–24. ACM. (Cited on page 1.)
- [60] Lynden, S., Kojima, I., Matono, A., and Tanimura, Y. (2011). ADERIS: An Adaptive Query Processor for Joining Federated SPARQL Endpoints. In R. Meersman, T. Dillon, P. Herrero, A. Kumar, M. Reichert, L. Qing, B.-C. Ooi, E. Damiani, D.C. Schmidt, J. White, M. Hauswirth, P. Hitzler, M. Mohania, editors, *On the Move to Meaningful Internet Systems (OTM2011), Part II. LNCS*, volume 7045, pages 808–817. Springer Heidelberg. (Cited on page 14.)
- [61] Mäkelä, E. (2014). Aether—generating and viewing extended void statistical descriptions of rdf datasets. In *European Semantic Web Conference*. Springer. (Cited on page 13.)
- [62] Marx, E., Khalili, A., and Valdestilhas, A. (2017a). Semantic search user interface patterns: an introduction. (Cited on page V.)
- [63] Marx, E., Shekarpour, S., Soru, T., Braşoveanu, A. M., Saleem, M., Baron, C., Weichselbraun, A., Lehmann, J., Ngomo, A.-C. N., and Auer, S. (2017b). Torpedo: Improving the state-of-the-art rdf

- dataset slicing. In *Semantic Computing (ICSC), 2017 IEEE 11th International Conference On*, pages 149–156. IEEE. (Cited on page 73.)
- [64] Marx, E., Soru, T., and Valdestilhas, A. (2017c). Triple scoring using a hybrid fact validation approach. *WSDM Cup*. (Cited on page V.)
- [65] Mihindukulasooriya, N., Rizzo, G., Troncy, R., Corcho, O., and García-Castro, R. (2016). A two-fold quality assurance approach for dynamic knowledge bases: The 3cixty use case. In *(KNOW@LOD/CoDeS)@ ESWC*. (Cited on page 10.)
- [66] Minier, T., Skaf-Molli, H., and Molli, P. (2018). Sage: Preemptive query execution for high data availability on the web. *CoRR*, abs/1806.00227. (Cited on pages 15 and 84.)
- [67] Montoya, G., Skaf-Molli, H., and Hose, K. (2017). The odyssey approach for optimizing federated sparql queries. In *International Semantic Web Conference*, pages 471–489. Springer. (Cited on page 14.)
- [68] Montoya, G., Skaf-Molli, H., Molli, P., and Vidal, M.-E. (2015). *ISWC*, chapter Federated SPARQL Queries Processing with Replicated Fragments. (Cited on page 15.)
- [69] Nentwig, M., Soru, T., Ngomo, A.-C. N., and Rahm, E. (2014a). LinkLion: A Link Repository for the Web of Data. In *The Semantic Web: ESWC 2014 Satellite Events - ESWC 2014 Satellite Events*, pages 439–443. (Cited on pages 1 and 21.)
- [70] Nentwig, M., Soru, T., Ngomo, A.-C. N., and Rahm, E. (2014b). Linklion: A link repository for the web of data. In *The Semantic Web: ESWC 2014 Satellite Events*, pages 439–443. Springer. (Cited on pages 57, 64, and 85.)
- [71] Neto, C. B., Esteves, D., Soru, T., Moussallem, D., Valdestilhas, A., and Marx, E. Wasota: What are the states of the art? (Cited on page V.)
- [72] Neto, C. B., Kontokostas, D., Hellmann, S., Müller, K., and Brümmer, M. Assessing quantity and quality of links between linked data datasets. (Cited on page 13.)
- [73] Neto, C. B., Kontokostas, D., Hellmann, S., Müller, K., and Brümmer, M. (2016). Assessing quantity and quality of links between linked data datasets. In *Proceedings of the Workshop on Linked Data on the Web co-located with the 25th International World Wide Web Conference (WWW 2016)*. (Cited on page 10.)
- [74] Ngomo, A.-C. N. and Auer, S. (2011). Limes—A time-efficient approach for large-scale link discovery on the web of data. *integration*, 15:3. (Cited on page 67.)

- [75] Ngomo, A. N., Sherif, M. A., and Lyko, K. (2014). Unsupervised link discovery through knowledge base repair. In *ESWC*, pages 380–394. (Cited on pages 57 and 85.)
- [76] Ngonga Ngomo, A.-C. and Auer, S. (2011). Limes - a time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of IJCAI*. (Cited on page 1.)
- [77] Paulheim, H. (2014). Identifying wrong links between datasets by multi-dimensional outlier detection. In *WoDOOM*. (Cited on page 13.)
- [78] Potocki, A., Saleem, M., Soru, T., Hartig, O., Voigt, M., and Ngomo, A.-C. N. (2017). Federated sparql query processing via costfed. (Cited on pages 22 and 72.)
- [79] Quilitz, B. and Leser, U. (2008). Querying Distributed RDF Data Sources with SPARQL. In Bechhofer, S., Hauswirth, M., Hoffmann, J., and Koubarakis, M., editors, *The Semantic Web: Research and Applications*, volume 5021 of *Lecture Notes in Computer Science*, pages 524–538. Springer Berlin Heidelberg. (Cited on page 14.)
- [80] Rahm, E. (2016). The case for holistic data integration. In *East European Conference on Advances in Databases and Information Systems*, pages 11–27. Springer. (Cited on page 1.)
- [81] Rajabi, E., Sicilia, M.-A., and Sanchez-Alonso, S. (2015). Inter-linking educational resources to web of data through iee lom. *Computer Science and Information Systems*, 12(1):233–255. (Cited on page 86.)
- [82] Rivest, R. (1992). The MD5 message-digest algorithm. (Cited on page 10.)
- [83] Ruiz-Calleja, A., Vega-Gorgojo, G., Asensio-Pérez, J. I., Bote-Lorenzo, M. L., Gómez-Sánchez, E., and Alario-Hoyos, C. (2012). A linked data approach for the discovery of educational ict tools in the web of data. *Computers & Education*, 59(3):952–962. (Cited on page 87.)
- [84] Saeedi, A., Nentwig, M., Peukert, E., and Rahm, E. (2018). Scalable matching and clustering of entities with famer. *Complex Systems Informatics and Modeling Quarterly*, (16):61–83. (Cited on page 1.)
- [85] Saleem, M., Hasnain, A., and Ngomo, A.-C. N. (2016). Largerdf-bench: A billion triples benchmark for sparql endpoint federation. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, o(o). (Cited on pages 4, 36, 70, and 76.)

- [86] Saleem, M., Hasnain, A., and Ngomo, A.-C. N. (2018a). Largerdf-bench: a billion triples benchmark for sparql endpoint federation. *Journal of Web Semantics*, 48:85–125. (Cited on page 70.)
- [87] Saleem, M., Kamdar, M. R., Iqbal, A., Sampath, S., Deus, H. F., and Ngomo, A.-C. N. (2014). 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. Semantic Web Challenge 2013. (Cited on pages 8 and 22.)
- [88] Saleem, M., Kamdar, M. R., Iqbal, A., Sampath, S., Deus, H. F., and Ngonga, A.-C. (2013a). Fostering serendipity through big linked data. In *Semantic Web Challenge at ISWC*. (Cited on pages 8 and 22.)
- [89] Saleem, M., Khan, Y., Hasnain, A., Ermilov, I., and Ngonga Ngomo, A.-C. (2015a). A fine-grained evaluation of sparql endpoint federation systems. *Semantic Web Journal*, pages 1–26. (Cited on pages 15, 69, 73, and 77.)
- [90] Saleem, M., Mehmood, Q., and Ngomo, A.-C. N. (2015b). Feasible: A feature-based sparql benchmark generation framework. In *The Semantic Web-ISWC 2015*, pages 52–69. Springer. (Cited on pages 4, 36, 70, 71, 76, and 77.)
- [91] Saleem, M., Ngomo, A.-C. N., Parreira, J. X., Deus, H. F., and Hauswirth, M. (2013b). Daw: Duplicate-aware federated query processing over the web of data. In *ISWC*. Springer. (Cited on pages 57 and 85.)
- [92] Saleem, M. and Ngonga Ngomo, A.-C. (2014). HiBISCuS: Hypergraph-Based Source Selection for SPARQL Endpoint Federation. In Presutti, V., d’Amato, C., Gandon, F., d’Aquin, M., Staab, S., and Tordai, A., editors, *The Semantic Web: Trends and Challenges*, volume 8465 of *Lecture Notes in Computer Science*, pages 176–191. Springer International Publishing. (Cited on pages 4, 15, 22, 70, and 72.)
- [93] Saleem, M., Ngonga Ngomo, A.-C., Xavier Parreira, J., Deus, H., and Hauswirth, M. (2013c). DAW: Duplicate-Aware Federated Query Processing over the Web of Data. In Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J., Aroyo, L., Noy, N., Welty, C., and Janowicz, K., editors, *The Semantic Web – ISWC 2013*, volume 8218 of *Lecture Notes in Computer Science*, pages 574–590. Springer Berlin Heidelberg. (Cited on page 15.)
- [94] Saleem, M., Padmanabhuni, S. S., Ngomo, A.-C. N., Almeida, J. S., Decker, S., and Deus, H. F. (2013d). Linked cancer genome atlas database. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 129–134. ACM. (Cited on page 1.)



- [95] Saleem, M., Padmanabhuni, S. S., Ngomo, A.-C. N., Almeida, J. S., Decker, S., and Deus, H. F. (2013e). Linked Cancer Genome Atlas Database. In M. Sabou, E. Blomqvist, T. Di Noia, H. Sack, T. Pellegrini, editors, *Proceedings of the 9th International Conference on Semantic Systems*, pages 129–134, New York, NY, USA. ACM. (Cited on page 79.)
- [96] Saleem, M., Potocki, A., Soru, T., Hartig, O., and Ngomo, A.-C. N. (2018b). Costfed: Cost-based query optimization for sparql endpoint federation. *Semantics*, 137:163–174. (Cited on pages 14, 73, and 77.)
- [97] Scharffe, F., Liu, Y., and Zhou, C. (2009). Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In *Proc. IJCAI 2009 workshop on Identity, reference, and knowledge representation (IR-KR), Pasadena (CA US)*. (Cited on page 13.)
- [98] Schmidt, M., Görlitz, O., Haase, P., Ladwig, G., Schwarte, A., and Tran, T. (2011). FedBench: A Benchmark Suite for Federated Semantic Data Query Processing. In Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., and Blomqvist, E., editors, *The Semantic Web – ISWC 2011*, volume 7031 of *Lecture Notes in Computer Science*, pages 585–600. Springer Berlin Heidelberg. (Cited on pages 4, 36, 38, 70, 75, and 76.)
- [99] Schwarte, A., Haase, P., Hose, K., Schenkel, R., and Schmidt, M. (2011). FedX: Optimization Techniques for Federated Query Processing on Linked Data. In Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., and Blomqvist, E., editors, *The Semantic Web – ISWC 2011*, volume 7031 of *Lecture Notes in Computer Science*, pages 601–616. Springer Berlin Heidelberg. (Cited on pages 14, 15, 73, and 77.)
- [100] Seker, S. E., Altun, O., Ayan, U., and Mert, C. (2014). A novel string distance function based on most frequent k characters. *International Journal of Machine Learning and Computing*, 4(2):177–182. (Cited on pages 10, 40, 41, 42, and 48.)
- [101] Seker, S. E. and Mert, C. (2013). A novel feature hashing for text mining. *Journal of Technical Science And Technologies*, 2(1):37–40. (Cited on page 10.)
- [102] Sheth, A. P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., and Thirunarayan, K. (2008). The semantic web-iswc 2008. (Cited on page 75.)
- [103] Shvaiko, P. and Euzenat, J. (2013). Ontology matching: State of the art and future challenges. *Knowledge and Data Engineering, IEEE Transactions on*, 25(1):158–176. (Cited on page 39.)

- [104] Soru, T., Marx, E., Moussallem, D., Publio, G., Valdestilhas, A., Esteves, D., and Neto, C. B. (2017). Sparql as a foreign language. *arXiv preprint arXiv:1708.07624*. (Cited on page [V](#).)
- [105] Soru, T., Marx, E., Valdestilhas, A., Esteves, D., Moussallem, D., and Publio, G. (2018). Neural machine translation for query construction and composition. *arXiv preprint arXiv:1806.10478*. (Cited on page [V](#).)
- [106] Soru, T., Marx, E., Valdestilhas, A., Moussallem, D., Publio, G., and Saleem, M. (2020). Where is linked data in question answering over linked data? *arXiv preprint arXiv:2005.03640*. (Cited on page [V](#).)
- [107] Soru, T. and Ngonga Ngomo, A.-C. (2013). Rapid execution of weighted edit distances. In *Proceedings of the Ontology Matching Workshop*. (Cited on page [11](#).)
- [108] Taelman, R., Van Herwegen, J., Vander Sande, M., and Verborgh, R. (2018). Comunica: a modular sparql query engine for the web. In *International Semantic Web Conference*, pages 239–255. Springer. (Cited on pages [16](#) and [84](#).)
- [109] Tarjan, R. E. (1975). Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2). (Cited on page [58](#).)
- [110] Tomaszuk, D., Skonieczny, Ł., and Wood, D. (2015). Rdf graph partitions: A brief survey. In *BDAS*. Springer. (Cited on pages [7](#) and [88](#).)
- [111] Urbani, J., Kotoulas, S., Maassen, J., Van Harmelen, F., and Bal, H. (2010). Owl reasoning with webpie: calculating the closure of 100 billion triples. In *Extended Semantic Web Conference*, pages 213–227. Springer. (Cited on page [1](#).)
- [Valdestilhas et al.] Valdestilhas, A., Arndt, N., and Kontokostas, D. DBpediaSameAs: An approach to tackle heterogeneity in dbpedia identifiers. (Cited on pages [2](#), [3](#), [5](#), [9](#), [11](#), [12](#), [39](#), and [57](#).)
- [113] Valdestilhas, A., Arndt, N., and Kontokostas, D. (2015). Dbpediasameas: An approach to tackle heterogeneity in dbpedia identifiers. In *SEMANTiCS (Posters & Demos)*, pages 1–4. (Cited on pages [V](#) and [4](#).)
- [114] Valdestilhas, A., Soru, T., Nentwig, M., Marx, E., Saleem, M., and Ngomo, A.-C. N. (2018). Where is my uri? In *European Semantic Web Conference*, pages 671–681. Springer. (Cited on pages [V](#), [2](#), [3](#), [4](#), [5](#), [9](#), [16](#), [26](#), [69](#), [70](#), [71](#), and [72](#).)



- [115] Valdestilhas, A., Soru, T., and Ngomo, A.-C. N. (2017a). Cedal: time-efficient detection of erroneous links in large-scale link repositories. In *Proceedings of the International Conference on Web Intelligence*, pages 106–113. ACM. (Cited on pages [V](#), [2](#), [3](#), [4](#), [5](#), [9](#), and [86](#).)
- [116] Valdestilhas, A., Soru, T., and Ngomo, A.-C. N. (2017b). A high-performance approach to string similarity using most frequent k characters. In *OM@ ISWC*, pages 1–12. (Cited on pages [V](#), [1](#), [3](#), [4](#), [5](#), [9](#), and [28](#).)
- [117] Valdestilhas, A., Soru, T., and Saleem, M. (2019). More complete resultset retrieval from large heterogeneous rdf sources. In *Proceedings of the 10th International Conference on Knowledge Capture, K-CAP '19*, pages 223–230, New York, NY, USA. ACM. (Cited on pages [V](#), [2](#), [3](#), [6](#), and [9](#).)
- [118] Valdestilhas, A., Soru, T., and Saleem, M. (2020). Relod: The incremental lod dataset relation index. *Submitted to the Semantic Web Journal*. (Cited on pages [2](#), [3](#), [4](#), and [5](#).)
- [119] Vandenbussche, P.-Y., Umbrich, J., Matteis, L., Hogan, A., and Buil-Aranda, C. (2017). Sparqls: Monitoring public sparql endpoints. *Semantic Web*, 8(6):1049–1065. (Cited on page [17](#).)
- [120] Verborgh, R., Vander Sande, M., Hartig, O., Van Herwegen, J., De Vocht, L., De Meester, B., Haesendonck, G., and Colpaert, P. (2016). Triple pattern fragments: a low-cost knowledge graph interface for the web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 37:184–206. (Cited on page [16](#).)
- [121] Volz, J., Bizer, C., Gaedke, M., and Kobilarov, G. (2009). Silk—A link discovery framework for the web of data. *LDOW*, 538. (Cited on page [67](#).)
- [122] Wood, D., Zaidman, M., Ruth, L., and Hausenblas, M., editors (2014). *Linked Data: Structured data on the Web*. Manning. (Cited on page [51](#).)
- [123] Yaghouti, N., Kahani, M., and Behkamal, B. (2015). A metric-driven approach for interlinking assessment of rdf graphs. In *Computer Science and Software Engineering (CSSE), 2015 International Symposium on*. IEEE. (Cited on pages [13](#) and [83](#).)
- [124] Yan, C., Zhao, X., Zhang, Q., and Huang, Y. (2017). Efficient string similarity join in multi-core and distributed systems. *PLOS ONE*, 12(3):1–16. (Cited on page [11](#).)
- [125] Zablith, F., Fernandez, M., and Rowe, M. (2011). The ou linked open data: production and consumption. In *Extended Semantic Web Conference*, pages 35–49. Springer. (Cited on page [87](#).)

- [126] Zaveri, A., Kontokostas, D., Sherif, M. A., Böhmann, L., Morsey, M., Auer, S., and Lehmann, J. (2013). User-driven quality evaluation of dbpedia. In *Proceedings of the 9th International Conference on Semantic Systems, I-SEMANTICS '13*, pages 97–104, New York, NY, USA. ACM. (Cited on pages [11](#) and [55](#).)
- [127] Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., and Auer, S. (2015). Quality assessment for linked data: A survey. *Semantic Web*, 7(1). (Cited on pages [13](#), [57](#), [62](#), [63](#), [83](#), and [86](#).)

## SELBSTÄNDIGKEITSERKLÄRUNG

---

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, den 9.6.2020

.....  
Andre Valdestilhas