Identifying, Querying, and Relating Large Heterogeneous RDF Sources

Andre Valdestilhas [a] Muhammad Saleem [a] Edgard Marx [b] Bernardo Pereira Nunes [d] Tommaso Soru [a] Wouter Beek [c] Claus Stadler [a] Konrad Höffner [e] Thomas Riechert [b]

[a] *Computer Science Institute, University of Leipzig, {valdestilhas,tsoru,saleem,cstadler}@informatik.uni-leipzig.de*
[b] *Leipzig University of Applied Science (HTWK), {edgard.marx, thomas.riechert}@htwk-leipzig.de*
[c] *VU University Amsterdam (VUA) w.g.j.beek@vu.nl*
[d] *Australian National University (ANU), bernardo.nunes@anu.edu.au*
[e] *Institute for Medical Informatics, Statistics and Epidemiology (IMISE), konrad.hoeffner@imise.uni-leipzig.de*

Abstract. The Linked Open Data (LOD) principles have been widely adopted due to its undeniable advantages; however, publishing and connecting data to third parties remains a difficult and time-consuming task. A question often raised during the publication process is whether there is a dataset available on the Web with which we can connect. Although it seems a trivial question, it unfolds in quite complex issues, such as, where the related datasets are, how many there are, how similar they are and how to query these datasets in a heterogeneous environment. This paper tackle the aforementioned questions introducing (i) a novel method to detect datasets similarities including duplicated chunks of data among RDF datasets; (ii) a publicly queryable index called ReLOD responsible for identifying datasets sharing properties and classes; and, (iii) a SPARQL query processing engine called wimuQ able to execute both federated and non-federated SPARQL queries over a large amount of RDF data. To create the ReLOD index and execute SPARQL queries over the Web of Data, we harvested more than 668 k datasets from LOD Stats and LOD Laundromat, along with 559 active SPARQL endpoints corresponding to 221.7 billion triples or 5 TB of data. We presented an evaluation of the accuracy of ReLOD and the query execution performance of the wimuQ over such massive amount of data.

Keywords: Indexing, Querying the Web of Data, Federated Query, Source selection, Big Data techniques, Integrating different sources, Identifying datasets

1. Introduction

The concept of Linked Data relates to a set of best practices[1] to publish and connect structured web-based data. The number of datasets published according to these principles has grown significantly over the last few decades Bizer et al. [10]. These datasets form the *Web of Data* which represents a large collection of interlinked datasets from multiple domains Saleem et al. [48, 50]. The interlinked property of the Web of Data is of central importance for tasks like cross-ontology question answering Lopez et al. [26], federated querying and data integration [41, 52], large-scale distributed inferences Urbani et al. [60], and distributed property paths evaluations Mehmood et al. [29]. However, interlinking RDF datasets is still an issue due to heterogeneous and autonomous nature of the Web of Data Ngonga Ngomo and Auer [36], Saeedi et al. [45], Valdestilhas et al. [61]. Consequently, identifying relevant datasets and executing SPARQL queries over large corpus of the Web of Data becomes a difficult task [2, 11, 33, 38, 51, 66]. This is because various interfaces, such as LOD Stats [7], LOD Laundromat [9], and SPARQL endpoints provide access to hundreds of thousands of RDF datasets, representing billions of facts. These datasets are available in different formats such as raw data dumps, HDT files, or directly accessible via SPARQL endpoints. Querying such a large amount of distributed data is

[1] Linked Data rules: https://www.w3.org/DesignIssues/LinkedData.html

1570-0844/20/$35.00 © 2020 – IOS Press and the authors. All rights reserved

particularly challenging, and many of these datasets cannot be directly queried using the SPARQL query language.

The current challenges addressed in this paper are identifying, querying, and relating datasets from a large heterogeneous collection of RDF datasets. To perform these tasks we need to first identify the set of existing RDF datasets in the Linking Open Data (LOD) cloud. We then need to know meta data about the contents of these datasets, e.g., how they are related to each other, and how similar they are to each other. To this end, we developed Wimu [62], a regularly updated database index of more than 660 k datasets from LODStats and LOD Laundromat, storing all the URIs used in these datasets. Wimu is publicly available as a web service[2] and can be used to retrieve a set of datasets that contains a given URI. Later, in Valdestilhas et al. [63], we introduced wimuQ, 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. The wimuQ engine partly makes use of the Wimu index for source selection in distributed query processing. In this paper, we present another index called ReLOD, which stores information about how LOD cloud datasets are related to each other and how similar they are. We integrated ReLOD index with the wimuQ engine to further increase the recall (in terms of total results retrieved) of the wimuQ engine. The contributions of this work are as follow:

- A method to create an incremental index of similarity among LOD datasets. To the best of our knowledge, ReLOD is the first dataset relation index over a net total of 221,7 billion triples.
- Creation of a mechanism to query this index by Dataset URI, property, class or SPARQL query. For the first time, to the best of our knowledge, a relation index can support querying by dataset URI, property, class, and SPARQL query.
- A method to identify duplicated datasets and chunks over large LOD datasets.
- We integrated ReLOD index with the wimuQ query engine. This resulted in improved source selection in terms of the number of relevant sources identified by the wimuQ for the given SPARQL query. Consequently, the recall and the query execution time of the wimuQ is improved.

[2]Wimu Web service: http://wimu.aksw.org/l

The rest of the paper is organized as follows: We first motivate the ReLOD index by presenting some scenarios where this index is particularly useful. We then present related work in Section 3, followed by our approach in Section 4. Section 5 contains the evaluation results and discussion. Finally, Section 6 concludes the paper along with future directions.

2. Motivation

The ReLOD index can be particularly useful in some practical scenarios, for example:

- **Source selection in SPARQL federation.** The goal of the source selection in federated SPARQL query processing is to identify the set of relevant also called capable sources (datasets) for the given SPARQL query. Our index is particularly helpful to assist the source selection step performed by the federated SPARQL query processing engines. In this paper, we will show how the ReLOD index is utilized by the WimuQ federation engine to retrieve more complete results.
- **Duplicate-aware SPARQL federation.** Duplicate-aware federated SPARQL query processing remains a hot topic in Semantic Web and Linked Data [32, 47]. The goal is to identify the duplicated chunks of data during the source selection and skip retrieving duplicated information. Since ReLOD contains information about duplicated chunks among LOD cloud datasets, we believe our index could be helpful in devising efficient duplicate-aware federated SPARQL query processing engines.
- **Linking RDF datasets.** According to the fourth rule of the Linked data design principles, a new dataset needs to be linked with other datasets by creating links between the source and target datasets. Link Discovery frameworks like LIMES [35] require to specify both source and target datasets for creating possible links. With the significant growth of the LOD datasets, it is often difficult to know what dataset could potentially have links to the given dataset. We believe ReLOD index could be helpful in identifying the relevant datasets that could be considered for linking with new dataset.

3. State of the art

This section is divided into two parts, first we discuss various approaches used to find RDF datasets similarities. We then categorize various approaches used for distributed SPARQL query processing.

3.1. Datasets Similarity

Roder et al. [43] present a probabilistic topic modelling approach to determine related datasets. The approach requires metadata about the datasets to determine datasets relatedness. The works from Asprino et al. [4, 5, 6] present several statistics about LOD datasets including a data structure called Equivalence Set Graph (ESG) that allows specifying compact views of large RDF graphs. The work from Ellefi et al. [14] introduces a dataset recommendation approach to identify linking candidates based on the presence of schema overlap between datasets. They use the concept of dataset profiles where a dataset is characterized through a set of schema concept labels that best describe it and can be potentially enriched by retrieving their textual descriptions. Mihindukulasooriya et al. [30] created an index of property and classes used on a limited number of datasets.

Rouces et al. [44] presents integration rules between arbitrary datasets and a mediated schema as a method for linking datasets. The work is particularly useful for establishing complex mappings between linked open datasets. Emaldi et al. [15] present a work built on the assumption that similar datasets should have a similar structure. The similarity between datasets is established by using Frequent Subgraph Mining (FSM) techniques. The works from Neto et al. [34] and Baron Neto et al. [8] made use of the bloom filters to identify duplicates. To et al. [59] presents an approach to determine the degrees of equivalences between relations (properties) defined by different LOD vocabularies. They approach models the datasets properties as nodes and the level of relatedness as label of the link connecting two property nodes.

Papadakis et al. [37] introduces Data Blocking, an approach for attribute clustering and comparison by identifying pairs of matching entities among two large, heterogeneous, and duplicate free but overlapping collections of entities. The approach from Hassanzadeh et al. [23] replaces traditional schema matching techniques showing that even attributes with different meaning can sometimes be useful in aligning data. Rekatsinas et al. [42] provides a new data source management system that automatically assesses the quality of data sources based on a collection of data quality metrics and enables the automated and interactive discovery of valuable sources for user applications. Xie et al. [67] introduces a method for property matching that makes use of the *owl:sameAs* links. The method is organized into two steps: (1) Extract sameAs and transform it into tables. (2) Apply table matching techniques and propose a matching criterion to find relationships between properties. They use similarity functions, including string similarity and numeric similarity. A possible shortcoming of this approach is that many of the RDF datasets in HDT format do not contain *owl:sameAs* links [63]. The *Entity Reconciliation Community Group*[3], is doing valuable work, which aims to match entities across data sources using different identifiers and formats [13]. A survey about related topics is presented in [64].

Table 1 summarizes the works with aim of relating Large amount of RDF datasets.

Table 1
A table summarizing the state of the art overview regarding Relating a Large amount of RDF datasets with the following characteristics: Compute similarity level among the datasets based on the properties and class similarities (Sim), Greater than 1 Billion triples (1B triples), Identify duplicated data (IDP)

Approach	Sim	1B triples	IDP
ESG [4]	-	✓	-
Loupe [30]	-	-	-
FSM [15]	✓	-	-
BloomFilters [34]	-	-	✓
MFKC [61]	✓	-	-
ReLOD	✓	✓	✓

3.2. Large Scale SPARQL Query Processing

In this section, we provide a brief overview of the state of the art pertaining to querying a huge amount of heterogeneous LOD datasets. The state-of-the-art federated querying approaches can be divided based on the underlying linked data interface namely SPARQL endpoints, Triple patterns fragments, datadumps, and HDT. Table 2 summarizes the main points.

[3]https://www.w3.org/community/reconciliation/

Table 2
State of the art overview regarding the Querying of Large Heterogeneous RDF Datasets, showing which type of SPARQL query interfaces are supported by current engines including the following characteristics: SPARQL query federation on SPARQL endpoints (SQF), Link Traversal-based SPARQL federation (LTF), Linked Data Fragments (LDF), HDT interface (HDT)

Approach	SQF	LTF	LDF	HDT
DARQ [40]	✓	-	-	-
ADERIS [27]	✓	-	-	-
FedEx [57]	✓	-	-	-
Lusail [1]	✓	-	-	-
SPLENDID [18]	✓	-	-	-
CostFed [55]	✓	-	-	-
ANAPSID [2]	✓	-	-	-
SEMAGrow [11]	✓	-	-	-
Odissey [33]	✓	-	-	-
Mulder [16]	✓	-	-	-
DAW [47]	✓	-	-	-
Fedra [32]	✓	-	-	-
SaGe [31]	✓	-	-	-
LDQPS [24]	✓	✓	-	-
SIHJoin [25]	✓	✓	-	-
WoDQA [3]	✓	✓	-	-
SQUIN [20]	✓	✓	-	-
Comunica [58]	✓	✓	✓	✓
wimuQ [63]	✓	✓	✓	✓

Query federation over multiple SPARQL endpoints (SQF): The approaches in this category retrieve results from multiple SPARQL endpoints, each hosting different RDF datasets. Due to the SPARQL endpoints, these approaches are relatively fast. However, a downside of this kind of approaches is that they require the underlying data to be exposed via SPARQL endpoints. DARQ Quilitz and Leser [40], ADERIS Lynden et al. [27], FedX Schwarte et al. [57], Lusail Abdelaziz et al. [1], SPLENDID Görlitz and Staab [18], CostFed Saleem et al. [55], ANAPSID Acosta et al. [2], SemaGrow Charalambidis et al. [11], Odyssey Montoya et al. [33], and MULDER Endris et al. [16] 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 Saleem et al. [52].

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. In comparison to DARQ, the number of endpoint requests is greatly reduced by using bind joins in a block-nested loop fashion Schwarte et al. [57]. 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 Saleem et al. [52]. CostFed also implements a cost-based query planner. The source selection is closely related to HiBISCuS Saleem and Ngonga Ngomo [46]. Both bind and symmetric hash joins are used for data integration. ANAPSID Acosta et al. [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 Saleem et al. [52]. SemaGrow adapts the source selection approach from SPLENDID. It performs 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 utilizes these templates for source selection, and query decomposition and optimization. DAW Saleem et al. [47] and Fedra Montoya et al. [32] are examples of duplicate-aware query federation approaches over multiple SPARQL endpoints. SaGe Minier et al. [31] is a stateless preemptable SPARQL query engine for public endpoints. The system makes use of preemptable query plans and time-sharing scheduling, tackling the problem of RDF data availability for complex queries in public endpoints. A more exhaustive overview and comparison of SPARQL endpoint federation engines is provided in [39, 52].

Link Traversal-based SPARQL federation (LTF): These approaches do not require the underlying data to be exposed via SPARQL endpoints. The only re-

quirement is that the RDF should follow the aforementioned Linked Data Principles. These approaches make use of the link traversal and URIs dereferenceability to executed distributed SPARQL querying over multiple RDF datasets. LDQPS Ladwig and Tran [24], SIHJoin Ladwig and Tran [25], WoDQA Akar et al. [3], and SQUIN Hartig [20] are examples of traversal-based federated SPARQL query processing approaches. Both LQPS and SIHJoin make use of index and online discovery via link-traversal to identify relevant sources pertaining to a given SPARQL query. They implement a symmetric hash join. WoDQA performs a hybrid (index+ASK) source selection approach. It implements nested loop and bind joins. SQUIN discovers the potentially relevant data during the query execution and produces incremental query results. SQUIN uses a heuristic for query execution plan generation, adapted from Hartig [19]. 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 Hartig and Özsu [21] investigated 14 different approaches to rank traversal steps and achieve a variety of traversal strategies. A more exhaustive survey of the traversal-based SPARQL query federation is provided in Hartig et al. [22].

Link Data Fragments-based SPARQL federation (LDF): Besides the above query federation strategies, low-cost Triple Pattern Fragments(TPF) interfaces [65] can also be used to execute federated SPARQL queries. Comunica [58], a highly modular meta-engine for federated SPARQL query evaluation over heterogeneous interface 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.

SPARQL Federation over HDT (HDT): A large amount of LOD datasets is available via HDT files [63]. The SPARQL federation engines in this category execute queries over large HDT files available from different sources.

In this work, we aim to execute both federated and non-federated queries over RDF data sources available from different interfaces such as SPARQL endpoints, HDT files, and RDF Data dumps. Currently, we are executing SPARQL queries over a net total of 221.7 billion triples from these interfaces.

4. The proposed approach

In this section, we discuss the proposed approach in details. We assume that the reader is familiar with the concepts of RDF and SPARQL (e.g. notions of an RDF triple, a triple pattern and basic graph pattern (BGP)).

Figure 1 shows the workflow of the query processing in our approach, which comprises 7 main steps: (1) the user issues a SPARQL query to the wimuQ interface, which (2) extracts all the URIs used in the subject, predicate or object of the SPARQL query 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) The ReLOD[4] helps in identifying more relevant datasets which were not included in the previous step. (5) The relevant datasets are further filtered by using the source selection algorithm (discussed in the next section). (6) The finally-selected relevant datasets are then queried by using the different query processors, depending on their respective serialization format (HDT, RDF from a SPARQL endpoint, RDF datadump, dereferenceable RDF dataset). (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 the end user. The user only sends the query and receives the results without knowing the underlying query execution steps. In the following section, we explain the ReLoD index introduced in this paper.

4.1. ReLOD—*The incremental LOD Dataset relation index*

The ReLOD index shows the similarities among datasets in terms of the number of properties they share. Currently, it contains information about more than 650 000 LOD datasets from LODStats, LOD-Laundromat, and more than 500 endpoints.[5]. In addition, the index also identifies the duplicated chunks among the datasets, allowing to answer two main questions: (1) For a given dataset, how to identify similar datasets based on the number of properties they share? (2) For a given set of properties, how do we identify datasets containing these properties?

[4] discussed in Section 4.1

[5] The list of endpoints is available at https://github.com/firmao/wimuT/blob/master/Endpoints_numtriples_lodcloud.csv

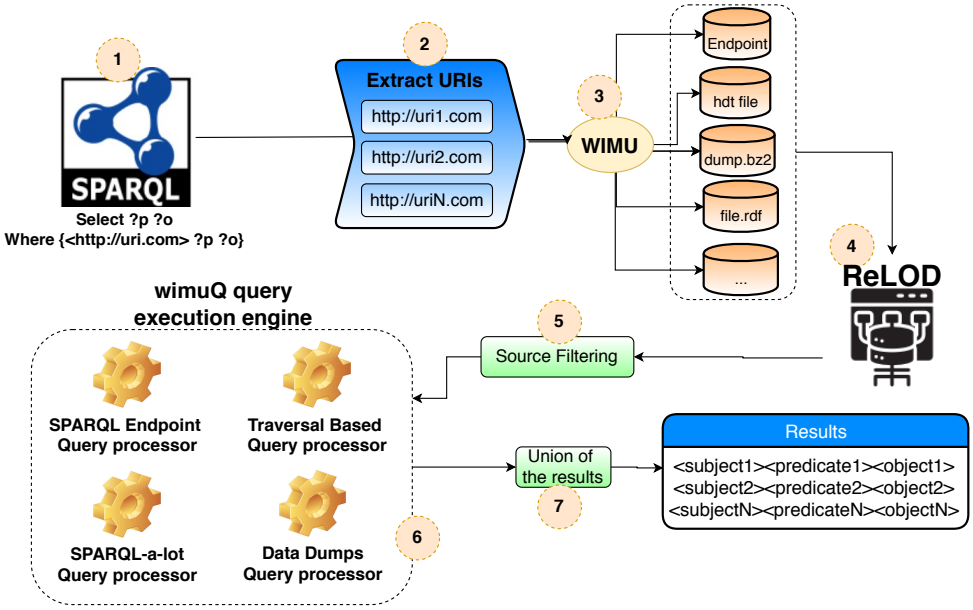

Fig. 1. Query processing workflow.

4.1.1. *The index creation*

The ReLOD index creation process is shown in Figure 2, which comprises of three main steps:

1. **Identifying duplicated datasets.** The first step is to identify duplicated datasets. For this, we store the number of properties of each dataset in a hash map. For a given dataset, we compare the number of properties (i.e, predicates) with all other datasets and select the candidate datasets which has exactly the same number of properties. In the second step, the strings of the predicates are compared. If two datasets have exactly the same predicates, then we compare their triples for the final duplicate detection. We just consider a single duplicated dataset in the remaining steps.
2. **Exact similarity.** In the second step, we identify the predicates which are exactly similar, i.e, their strings are the same. We store all such predicates, along with the corresponding datasets, in a table called "tableMatches_Exact.tsv".
3. **Partial similarity.** In the final step, we identify the predicates which are not exactly similar but a large portion of the string is similar, thus they can be potentially related to each other. To identify such predicates among datasets, we compare the strings of the a.) predicate URI, b.) the type (i.e, class of the predicate if available) of the predicate and c.) the property label (if available from the dataset schema) using the Most Frequent K Characters (MFKC) string similarity function introduced in [61]. The MFKC gives a similarity value between `0` and `1`, where 0 means no similarity and `1` means exact similarity among two strings. The final partial similarity then calculated using the following formula.

$$partialSim = \frac{\begin{array}{c}2MFKC(a)\\ +MFKC(b)\\ +MFKC(c)\end{array}}{4} \quad (1)$$

In this equation, the value of the $partialSim$ is between `0` and leass than `1`, where `0` means no similarity and higher values means high similarity. The similarity of the URIs of the predicates are weighted double as compared to their classes and property labels. We set a threshold of $partialSim > 0$ for storing them in "tableMatches_Sim.tsv". We also store the number of exact matches and the number of similar matches between different datasets in a separate table called "tableMatches.tsv". Finally, the datasets which contain RDF parse errors are stored in "DatasetErrors.tsv".

4.1.2. *Querying the index*

The ReLOD index can be queried by using three different types of inputs (ref. Figure 3) :

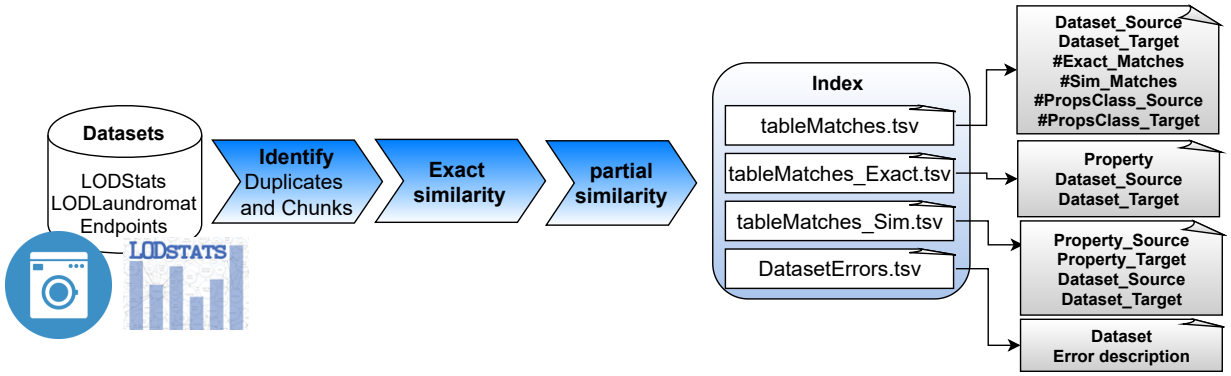

Fig. 2. Creating the index.

Dataset: For a given input datasets, the index will retrieve all the datasets which have exact or partial matches along with the number of predicates that are exactly or partially duplicated.

Set of properties (URIs): The user can provide a set of URIs (separated by comma) and the index will retrieve all the exact or partially matched predicates along with their source datasets.

SPARQL query: The user can provide a SPARQL query and the index will retrieve all the datasets which contain the predicates used in the given SPARQL query. This is the typical source selection process in the federated SPARQL querying [38, 46].

It tooks about 88 hours to generated the current ReLOD index using a standard hardware. We each every month for new datasets available from our sources. The prototype is available online from http://w3id.org/relod/ as a proof-of-concept. An example dataset search is shown in Fig. 4.

Next section details the source selection and query execution mechanisms of the proposed engine.

4.2. The source selection

The goal of the source selection is to identify the potentially relevant datasets (also called sources) to the given SPARQL query. To this end, we first extract all the URIs used in the given SPARQL query. We then make use of both WIMU [62] and ReLOD services to select potentially relevant sources. The WIMU service[6] provides a URI lookup facility and identifies those datasets containing the given URI. Currently, this service has processed more than 58 billion unique triples and indexed 4.2 billion URIs from more than 660 k RDF datasets obtained from LODStats [7] and LOD Laundromat Beek et al. [9]. The identified WIMU+ReLOD datasets can be of four types: (1) SPARQL endpoint, (2) HDT file, (3) dataset with dereferenceable URIs and (4) data dump with non-dereferenceable URIs. The source service is available from a web interface and can also be queried from a client application using the standard HTTP protocol.

[6]WIMU URIs lookup service is available from: http://wimu.aksw.org/

4.3. The query execution

We make use of the four query processors – SPARQL endpoint, Link Traversal-based, SPARQL-a-alot, Data dumps – 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 wimuQ source selection discussed in the previous section.

We used the FedX [57] query processor for *SPARQL endpoints query federation* and SQUIN for traversal-based query federation. The reason for choosing FedX for SPARQL endpoint federation and SQUIN for traversal-based federation is because 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 reasonable query runtime performances in comparison to state-of-the-art approaches [20, 52, 55].

As mentioned before, FedX can only work with public SPARQL endpoints. SQUIN needs dereferenceable URIs. Both of these engines are unable to execute SPARQL queries over non-dereferenceable URI

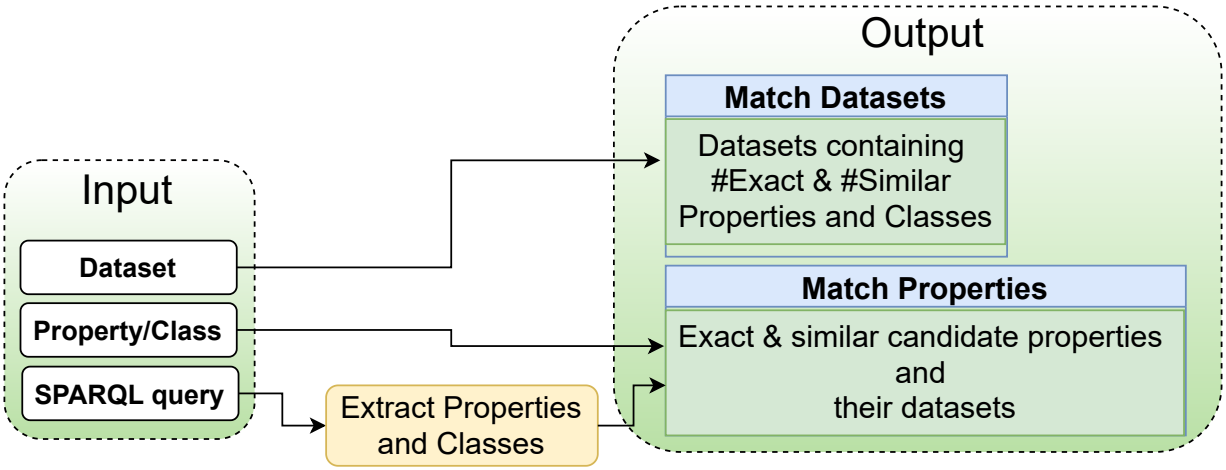

Fig. 3. Querying the index.

Input:

http://query.wikidata.org/sparql

Visualize dataset relations

Output:

Dataset	#Exact Match	#Similarity > 0.9
7682278f9dd608f09c4c073a915e58a1.hdt	1853	28824
449ffaddb6023ae27686a0b416447677.hdt	160	28824
62f1f8087724003a0d5cec3bfd00f5c5.hdt	110	28824
7682278f9dd608f09c4c073a915e58a1.hdt	16	24
http://pt.dbpedia.org/sparql	13	10000
http://es.dbpedia.org/sparql	7	28824
0fb5cc2fb77fcd6c187ef3b4856f7813.hdt	6	2938
http://eu.dbpedia.org/sparql	4	4097
0fb5cc2fb77fcd6c187ef3b4856f7813.hdt	4	28824
449ffaddb6023ae27686a0b416447677.hdt	3	30

Fig. 4. Example of the first ten results from 1,738 similar datasets.

datadumps. SPARQL endpoint federation approaches cannot execute queries over such datadumps as they are not exposed as SPARQL endpoints. Link traversal-based approaches on the other hand fail to retrieve results as the URIs are non-dereferenceable. There is a need to download the dumps first, load them locally, and use a query processing API such as Jena or Sesame on the loaded datasets. However, we cannot simply download the complete dumps and process them locally due to their large amount of data. To solve this problem, we make use of the Wimu index to select only those data dumps that are potentially capable of executing the given query. These datadumps are further sliced by using an RDF slice [28] technique, to select only required chunks of the datadumps that 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.

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

5. Evaluation

In this section, we present the evaluation setup and the corresponding results that validate our hypothesis that we can improve the resultset retrieval if we automatically identify potentially relevant sources from heterogeneous RDF data even if the URIs are not dereferenceable anymore with our new approach ReLOD together with wimuQ[63]. 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.

As a justification for the chosen benchmarks we state that one of the goals of wimuQ is to execute SPARQL queries over real-world RDF datasets, we chose three – FedBench Schmidt et al. [56], LargeRDFBench Saleem et al. [54], Feasible Saleem et al. [53] – real-world RDF datasets benchmarks in our

evaluation. 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 are publicly available.

Regarding Precision, Recall, and F-measure. wimuQ is based on existing federated query engines. Its Precision, Recall, and F-measure is entirely based on these engines. There are already extensive works that perform this evaluation, which can be assessed by FedBench [56], LargeRDFBench [54] and Feasible [53] in the related work section. Notice, however, that, by coherence, the F-measure, Precision, and Recall of these engines will remain the same independent of the target datasets.

Further, a Cherry-Picking would not be fair since we know in advance which datasets are indexed. Instead, we describe the limitations of our approach and the state-of-the-art.

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

Instead, we asked another question: Given a federated query engine, is it possible to improve its resultset retrieval by devising a better source selection approach?

We show that wimuQ can successfully execute (with results) the majority of the benchmark queries. Besides, wimuQ resultset recall is higher with reasonable query execution times.

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.

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

We showed that wimuQ successfully executes (with results) the majority of the benchmark queries without any prior knowledge of the data sources. Besides, the wimuQ resultset recall is higher with reasonable query execution times.

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.

5.1. Experimental setup

5.1.1. Benchmarks used:

Since wimuQ aims to execute SPARQL queries over real-world RDF datasets, we chose three – FedBench [56], LargeRDFBench [54], Feasible [53] – 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 Schmidt et al. [56].
- **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 Saleem et al. [54].
- **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 Saleem et al. [53]: (1) 175 queries benchmark generated from DBpedia queries log and (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 Saleem et al. [53].

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 are publicly available.[7]

5.1.2. Hardware:

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

[7] Queries available from https://github.com/firmao/wimuT/blob/master/queriesLocation.txt

5.1.3. 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.[8] We filtered the endpoints URLs[9] and the total number of triples hosted by each of these endpoints.

5.1.4. Metrics:

Since wimuQ aims to retrieve more complete results within the reasonable amount of time, we chose 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.

5.1.5. Approaches:

As aforementioned in Section 3, different federation engines available to federate SPARQL queries over endpoints and traversal-based federation. We chose FedX Schwarte et al. [57] for SPARQL endpoint federation and SQUIN Hartig [20] for traversal-based query federation. The reason for choosing these two engines 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 and able to run federated queries with zero initial knowledge. In addition, both of these engines perform reasonably well in terms of query runtime performances w.r.t state-of-the-art approaches Hartig [20], Saleem et al. [52, 55]. For the sake of completeness, we also compared wimuQ with SPARQL-a-lot, WimuDumps**, and using ReLOD approach, we obtain datasets that sharing properties**.

5.1.6. Results

Coverage of the results: The main purpose of wimuQ is to devise a federation engine which is able to retrieve more complete results for the given SPARQL queries. Figure 5 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 wimuQ is 8 481 across the three benchmarks. Out of these, wimuQ collects about 91 % of the results from wimuDumps (avg. resultset size 7 651), 7 % from SPARQL endpoints (avg. resultset size 556), and 1 % from SPARQL-a-lot (avg. resultset size 74).

For FedBench, the wimuQ 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 wimuQ 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 wimuQ 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, the SPARQL-a-lot query processing engine only retrieved about 0.03 % results (avg. resultset size 11).

Figure 5 shows the effect on the average SPARQL result set sizes when combining wimuQ with *ReLOD* (bars labelled with wimuQ+ReLOD) and contrasts it with other approaches. Combining wimuQ with ReLOD consistently yields an higher amount of results due to inclusion of related datasets. The concrete improvements of our approach are quantified as follows: For FedBench there is an increase by 36 095 results split among CD(2 183), LS(5 549) and LD(27 609). For LargeRDFBench there are 51 675 further results divided among Simple(32 421), Comp(19 245), LD(7), and Chs(2). With Feasible there are 88 254 additional results divided among DBpedia(79 380) and SWDF(8 874). This increase of results in due to ReLOD identifying similar properties among the datasets. For example, the property dbo:city in DBpedia corresponds to wd:Q515[10] in Wiki-

[8] List of SPARQL endpoints: https://lod-cloud.net/lod-data.json

[9] Endpoints URLs with size: https://goo.gl/H2t5ko

[10] https://www.wikidata.org/wiki/Q515

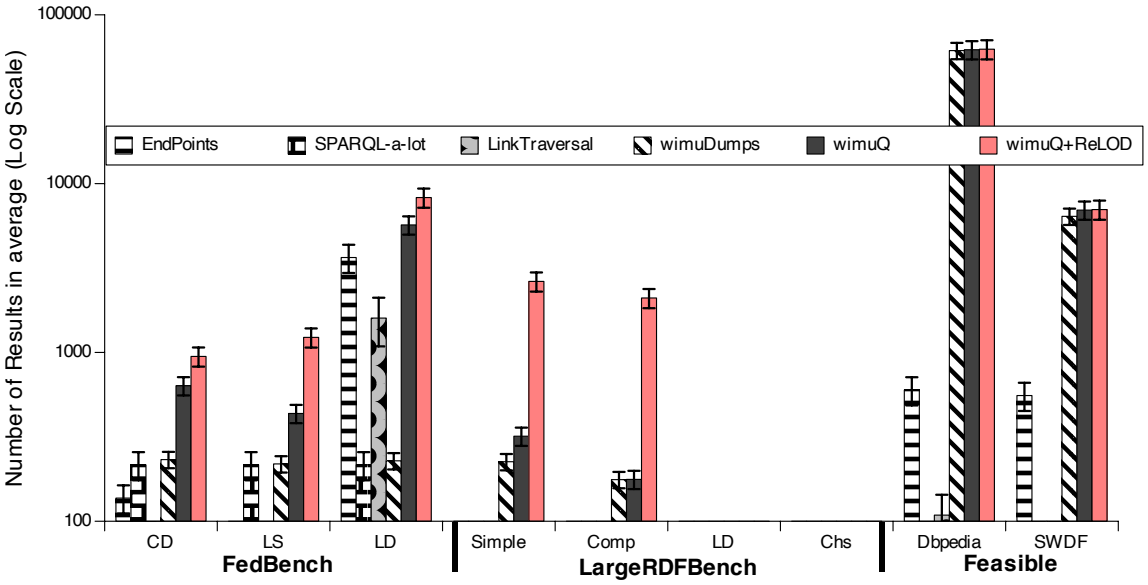

Fig. 5. Average SPARQL result set sizes (number of bindings) retrieved by the selected approaches across different queries categories of the selected benchmarks.

data. Furthermore, ReLOD reconized that the property http://www.geonames.org/ontology#name is mentioned in two datasets, namely 7dea9bd54a6de11bb1f65d9dcf2e3683.hdt and e3860989e46dfd87c8e09d324f603ef4.hdt. In the specific case of query 7 from LargeRDFBench (Simple) ReLOD was able to identify 16 related datasets and compute results across them whereas the other approaches just used the single given benchmark dataset.

In summary, wimuQ+ReLOD is able to retrieve at least one resultset for 87 % of the overall 415 queries, which 11% more results thanks to the ReLOD approach. The results clearly shows that combining different query processing engines into a single SPARQL query execution framework lead towards more complete resultset retrieval.

We observed a limitation that the selected approaches are mostly unable to retrieve results for the Large Data and Complex+High (Ch) queries categories of LargeRDFBench. The reason behind the "no results" for the Large Data queries is that these queries retrieve results from the LinkedTCGA Saleem et al. [49] datasets, which were not publicly available via SPARQL endpoints, were not indexed by WIMU, and were also not reachable via link traversals. While Ch queries often require a higher number of distributed datasets to compute the queries' final resultset. Thus, the approaches could not find all of the relevant datasets needed to calculate the queries' final resultset. **Part of this limitation was solved using the ReLOD index, providing alternative datasets that share properties.** The average number of datasets[11] queries by wimuQ for the selected benchmarks queries categories is given in Figure 6. We can see the highest number of datasets are selected for Ch queries of LargeRDFBench. **With wimuQ+ReLOD we enable the expansion of the set of SPARQL datasets queries are evaluated on due to the ReLOD index's discovery of additional datasets based on the properties they share. For the FedBench query groups ReLod leads to 2 additional datasets for CD, 3 datasets for LS, and 2 datasets for LD. For LargeRDFBench the numbers are: 2 datasets for Simple, 2 datasets for Comp, 3 datasets for LD, and 2 datasets for Chs. For Feasible there are 3 datasets for DBpedia and 4 datasets for SWDF. At present, given an input SPARQL query, wimuQ derives a set of queries for on each possible substitution of the input query's predicates with related ones obtained from ReLod. In the presented work the maximum number of substitutions was capped at 4. We aim to mitigate this limitation in the future with an im-**

[11] Here we also point the number of datasets discovered

proved SPARQL query rewriting approach that expands predicates of the original query with all candidates obtained via ReLod.

Query runtime performances: Figure 7 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 wimuQ 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, wimuQ collects about 91 % of the results from wimuDumps yet its average execution time is smaller than query execution over SPARQL endpoints which provides only 7 % of the total results. One possible reason for this could be that in SPARQL endpoint federation, the query processing task is split among multiple selected SPARQL endpoints and hence network and the number of intermediate results play an important role in the query runtime performances.

For FedBench, the wimuQ average query runtime is 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 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.

We also add the cost to use our newest approach ReLOD, where it was expected to expend more time, as we need to add the time to query the reLOD index. For FedBench, the wimuQ+ReLOD average query runtime was increased in 3 minutes, (CD) increased 11.39%, (LS) 15.55%, and (LD) 24.02%. For LargeRDFBench, the average query execution of wimuQ+ReLOD was increased in 5 minutes, (Simple) 25.24%, (Comp) 30.78%, as (LD) and (Chs) didn't get results and no time was calculated. For FEASIBLE, the average query execution of wimuQ+ReLOD was increased in 4 minutes, (DBpedia) increased 33.72%, and (SWDF) by 11.51%.

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

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.

5.2. Evaluation on Relating LOD datasets

This part of the evaluation aims to answer the following questions: (1) How can one identify and quantify similar datasets for a given Dataset?[12] (2) How many datasets are most likely to execute a given SPARQL query? (3) How can the detection of duplicated and chunk datasets help in the process of matching a large amount of datasets? (4) How can ReLOD increase the number results and 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*[13] and 12 from Debattista et al. [12]. To obtain the results from the Table 4, which is to see how similar the datasets are, we use the formula Eq. (2) and to obtain the results from Table 5, to see how much a dataset is contained inside each other we use the formula in Eq. (3), thus, normalizing the similarity score between 0 and 1, where A and B represent source and target datasets. We also observed data integration cases when a dataset contains other datasets, e.g., part of the Yago dataset in DBpedia. We offer an alternative way to visualize this sample data as a graph[14].

$$jaccard(A,B) = \frac{|A \bigcap B|}{|A \bigcup B|} \quad (2)$$

[12] To know how many datasets are similar to each other.
[13] http://www.rdfhdt.org/datasets/
[14] https://w3id.org/relod/visual.html

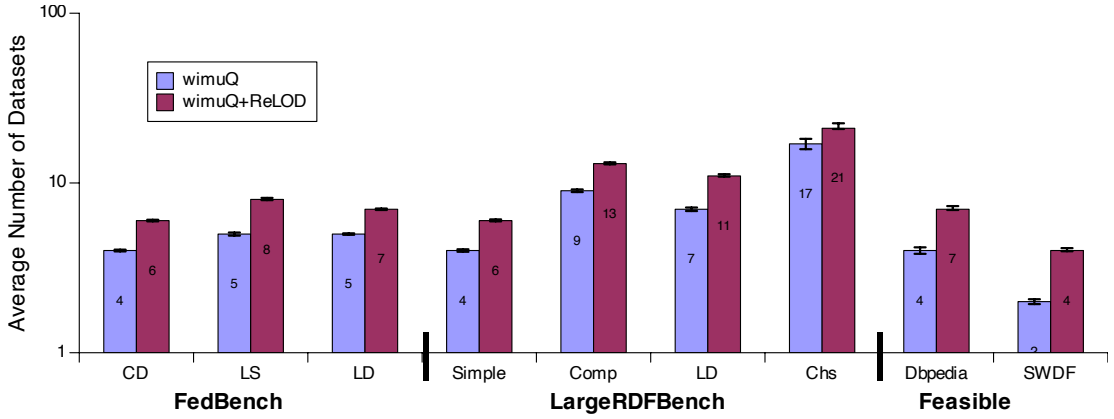

Fig. 6. Average number of datasets discovered and queried by wimuQ across different queries categories of the selected benchmarks.

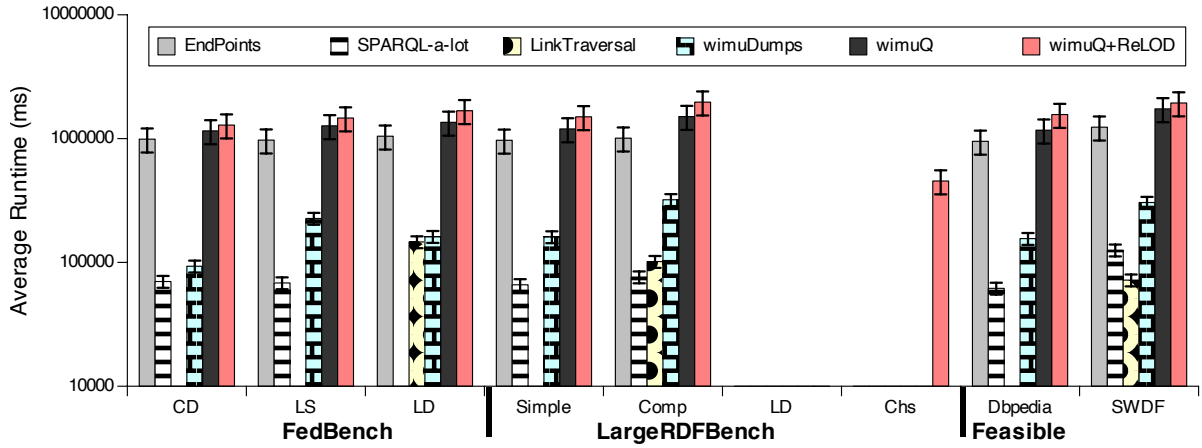

Fig. 7. Average query runtimes of the selected approaches across different queries categories of the selected benchmarks.

$$contained(A,B) = \frac{|A \bigcap B|}{|A|} \quad (3)$$

To answer question 1, we give to our approach as input a version of the dataset DBpedia[15], which contains 2 339 properties and classes, and as output we can obtain the most similar datasets[16]. Thus, the datasets were sorted into the 10 datasets that were the most similar to DBpedia, according to the properties and classes they share. On Table 6, we show the number of properties that they share that contain the exact same URI and cases where they share URI that are not the exact same but very similar.[17] Thus, an example of an application would be when the user wants to identify information from other datasets to complement or enrich knowledge for a given dataset, e.g, facilitating federated queries. Table 7 shows an evaluation with 600 randonly chosen datasets[18] including 3 synthetic manually made datasets.

The low level of similarity on Table 6 is because we want to find datasets not so similar to identify the possibility of integration. Also, the high num-

[15]http://gaia.infor.uva.es/hdt/DBPedia-3.9-en.hdt.gz

[16]We compare with the datasets from http://www.rdfhdt.org/datasets/

[17]With similarity level greater than 0.8 according our similarity algorithm.

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

Dataset File	Size	Repository
acadonto.nt.gz.hdt	708K	http://s001.adaptcentre.ie/lod/hdtdumps/
acm.rkbexplorer.com_id_.nt.gz.hdt	11K	http://s001.adaptcentre.ie/lod/hdtdumps/
agrinepaldata.com_.nt.gz.hdt	18K	http://s001.adaptcentre.ie/lod/hdtdumps/
aims.fao.org_aos_jita_.nt.gz.hdt	117K	http://s001.adaptcentre.ie/lod/hdtdumps/
alaska.eagle-i.net_i_.nt.gz.hdt	222K	http://s001.adaptcentre.ie/lod/hdtdumps/
apertium-rdf-ca-it.nt.gz.hdt	1.0M	http://s001.adaptcentre.ie/lod/hdtdumps/
arrayexpress-e-afmx-1.nt.gz.hdt	9.1M	http://s001.adaptcentre.ie/lod/hdtdumps/
athelia.com.nt.gz.hdt	9.5K	http://s001.adaptcentre.ie/lod/hdtdumps/
beta.vivosearch.org_institution_ponce-school-medicine.nt.gz.hdt	1.0M	http://s001.adaptcentre.ie/lod/hdtdumps/
bfs.270a.info_dataset_.nt.gz.hdt	26K	http://s001.adaptcentre.ie/lod/hdtdumps/
biblioteca-nacional-escolar-bnescolar.nt.gz.hdt	14K	http://s001.adaptcentre.ie/lod/hdtdumps/
brown-corpus-in-rdf-nif.nt.gz.hdt	9.6K	http://s001.adaptcentre.ie/lod/hdtdumps/
dblp-20170124.hdt	1GB	http://www.rdfhdt.org/datasets/
DBPedia-3.9-en.hdt	34GB	http://www.rdfhdt.org/datasets/
geonames-11-11-2012.hdt	344MB	http://www.rdfhdt.org/datasets/
linkedgeodata-2012-09-10.hdt	461MB	http://www.rdfhdt.org/datasets/
swdf-2012-11-28.hdt	2.3MB	http://www.rdfhdt.org/datasets/
wiktionary_en_2012-07-21.hdt	212MB	http://www.rdfhdt.org/datasets/
wordnet31.hdt	23MB	http://www.rdfhdt.org/datasets/
yago2s-2013-05-08.hdt	903MB	http://www.rdfhdt.org/datasets/

Table 3
Dataset description used on Table 4, Table 5 and Table 6

ber of URIs is non-dereferenceable, which only influences similarity, not exact match cases.

With our experiments we realize that more than 50 % of the properties and classes has a match in another dataset. The information can also be observed at Table 6 and Table 7, which highlights the possibility that one dataset can enrich another with complementing information from another dataset.

The time consumed can be observed on Table 7, which shows that 100 million triples were processed in less than 8 minutes, but the process becomes impractically time-consuming after only about 1 million triples.

As we are using wimuQ [63] to identify datasets for a given SPARQL query together with our matching algorithm, we can answer question (2) based on famous SPARQL queries from two real-data federated SPARQL querying benchmarks *LargeRDFBench* [54],*FedBench* [56] and one non-federated real data SPARQL benchmark selected from *FEASIBLE* [53] benchmarks generation framework is given in Figure 6. Thus, from the selected datasets, we use our approach to select the datasets according to the properties and classes they share among them.[19]

To answer question (3) we evaluate the dataset duplicated detection algorithm and the detection of dataset chunks. From a 5 446 datasets[20] our algorithm detected 2 272 duplicates and 1 470 chunks within 3 hours.

We did experiments with 900 datasets randomly chosen in which we identified 10 cases showing an expressive amount of duplicated datasets determined by our method. We also observed how segregated the datasets are in identifying the chunks.

The current version of the index prototype has information about 539 SPARQL endpoints from LOD cloud[21] and 915 HDT files from LOD Laundromat.[22] We performed more than 1 800 000 comparisons in 88 hours.

[19] In this case, all datasets identified by wimuQ are sharing properties and classes, that is why we are using the graph from Valdestilhas et al. [63].

[20] A subset from those 600 datasets chosen previously

[21] The list of is available here: https://github.com/firmao/wimuT/blob/master/Endpoints_numtriples_lodcloud.csv

[22] fdsa

Coeficient	acadonto	acm.rkbexplorer	agrinepaldata	aims	alaska	apertium	arrayexpress	athelia	bfs.270a	biblioteca-nacional	brown	dblp	dbpedia	geonames	linkedgeodata	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
linkedgeodata	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. The level of similarity is also represented by the intensity of the color: the more intense color, the more similar the datasets are.

#DsPropMatch in Table 7 refers to the number of properties/classes, the datasets shared.

We can observe the quantity of properties/classes that the datasets share related to the number of triples analysed. For example, from 600 datasets analyzed, we found 291 that were considered matches according to their properties' similarity. The number of triples was almost double the size of its matches. Thus, the number of matches, in this case, cannot be directly related to the number of triples.

We evaluate the accuracy of our matching algorithm with a small sample, where we can see at Fig. 8, Table 8 and Fig. 9 the F-Measure and runtime on six famous pairs of datasets from Georgala et al. [17]. With these datasets we create a gold standard consisting of tuples $(P_1, \ldots, P_n)$, where P_1 and P_2 represent a pair of datasets synthetically generated by the author, i.e., P_3 represents the comparison between the datasets *Abt* and *Buy*.

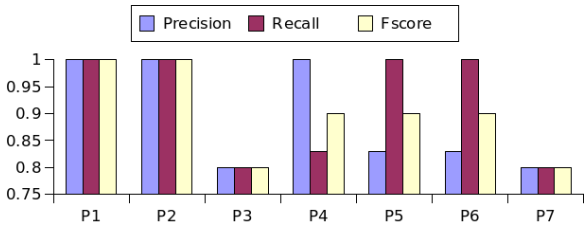

Fig. 8. F-measure of our string similarity.

According to Valdestilhas et al. [62], 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 unable 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 Laundro-

Coeficient	acadonto	acm.rkbexplorer	agrinepaldata	aims	alaska	apertium	arrayexpress	athelia	bfs.270a	biblioteca-nacional	brown	dblp	dbpedia	geonames	linkedgeodata	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
linkedgeodata	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 the portion of the datasets that are contained in each other. The level of containment is also represented by the intensity of the color.

Dataset	#ExactMatch	#sim > 0.8	#PropClass
swdf	22	64	288
yago	6	65	373 546
dblp	6	9	41
linkedgeodata	5	617	11 799
wiktionary	4	6	31
geonames	4	12	27
wordnet	3	26	69
wikidata	0	5	427
freebase	0	55	17 587

Table 6

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

mat, and 69.8 % of datasets from LODstats contain parser errors in which WIMU was not able to process the data.

#Datasets	#DsPropMatch	Avg.time (s)	#Avg.triples
3	1.66	0.33	3.66
10	4.03	0.2	119850.8
100	33.72	0.12	79964.08
200	67.12	0.2	114914.92
300	100.45	0.19	115973.73
400	133.79	0.18	109661.30
500	167.12	0.17	108455.56
600	200.48	0.70	141735.39

Table 7

The results show the average of the number of triples and executing time to get the property matches and the average number of property matches. **DsPropMatch** refers to the number of properties/classes the datasets share among each other

On the Section 5.1.6 we show results that can answer the question (4), specifically depicted on Fig. 5 and Fig. 6.

On the other hand, reinforces that more datasets do not always imply more results, i.e., in query 2 and 4, more datasets were identified, but the number of re-

Table 8
Sample datasets from Georgala et al. [17] to evaluate our string similarity approach.

Dataset	Source (S)	Target (T)	$\|S\| \times \|T\|$	Source Property	Target Property
(P3) Abt-Buy	Abt	Buy	1.20×10^6	product name, description manufacturer, price	product name, description manufacturer, price
(P4) Amazon-GP	Amazon	Google Products	4.40×10^6	product name, description manufacturer, price	product name, description manufacturer, price
(P5) DBLP-ACM	ACM	DBLP	6.00×10^6	title, authors venue, year	title, authors venue, year
(P6) DBLP-Scholar	DBLP	Google Scholar	0.17×10^9	title, authors venue, year	title, authors venue, year
(P7) MOVIES	DBpedia	LinkedMDB	0.17×10^9	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

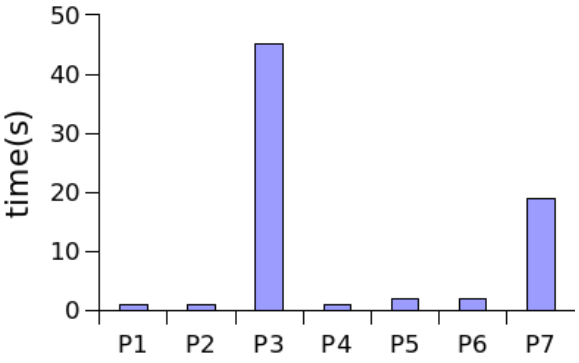

Fig. 9. Runtime of our string similarity.

sults 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.[23]

5.3. Usability

Fig. 10 **shows our usability study, where we conduct the study with seven PhD students from our research group who were interested in identifying similar data sources among a large number of RDF sources. The score of each questions varies from 1 (Strongly disagree) to 5 (Strongly agree).**

The results of all 10 questions[24] gives us a score of 32.57 (on a scale of 0 - 40) which is a SUS score of 32.57 $\times$ 2.5 $\approx$ 81. This indicates a high level of usability.

Besides the fact that this was a very early version of this prototype and the users expected more, the resulting scores from the usability study was better than we had expected and we obtained more feedback to improve our work.

6. Conclusion

We present ReLOD,[25], 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 TB of information from datasets par-

[23] The query number 9 obtained only one result with the new approach.

[24] Questions and usability methodology available here: http://www.measuringu.com/sus.php

[25] https://w3id.org/relod

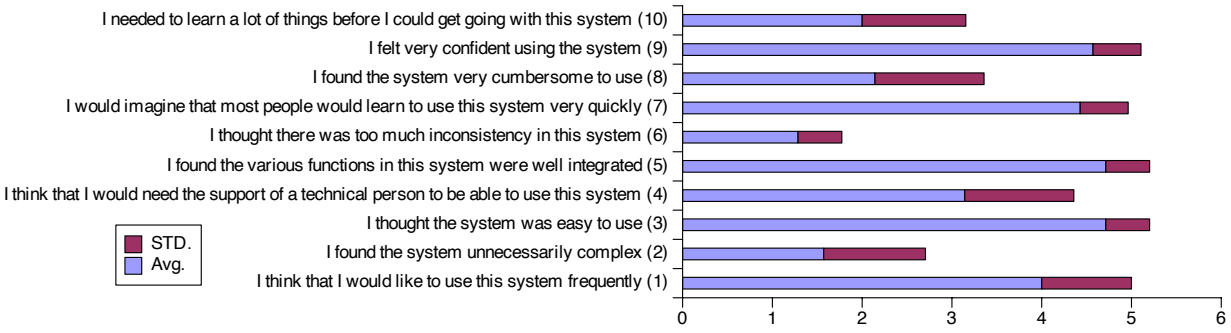

Fig. 10. Result of usability evaluation using SUS questionnaire.

tially retrieved using the service "Where is My URI" (WIMU)Valdestilhas et al. [62], in which the query engine wimuQ Valdestilhas et al. [63] uses the ReLOD to identify the most similar datasets increasing the number of results.

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

With wimuQ+ReLOD we retrieve at least one resultset for 87 % of the overall 415 queries, which 11% more results thanks to the ReLOD approach. The results clearly show that combining different query processing engines into a single SPARQL query execution framework leads to more complete resultset retrieval.

Our experiments show that more than 90 % of datasets from LODLaundromat datasets are not using `owl:equivalentProperty` and `owl:equivalentClass` or another way to relate 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, hindering the task of querying 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 majority of dataset files from LOD Laundromat are encoded with names,[26] which is not informative in a human point of view. Thus, we will provide a service and an index that includes more informative names for the datasets as future work. We will develop a method to convert the MD5 code from the HDT to URLs with more informative namespaces. Instead of the MDA code, we have the dataset's namespace giving us more information about the dataset avoiding the need to open the HDT file to obtain this information.

We will provide a more informative interface about the dataset relations, e.g., including a preview sample of the datasets and inform which properties they share, not only the number of properties.

As future work, we will explore the benefits of supporting non-dereferenceable URIs, regenerating dead data sources, including the re-creation of linksets and other relations among data sources, in this manner, working in a new way to conceive a Wayback machine for data sources.

We will make a better assessment, including more techniques and methods to detect duplicated datasets and String similarity algorithms deduplication as future work.

We cannot finish this paper without reminding the reader of an aporia, which says - *Different people describe things differently, and this anguish will always lie with us*—That means, at least, this study needs a continuation.

The source-code is available online.[27]

References

[1] Ibrahim Abdelaziz, Essam Mansour, Mourad Ouzzani, Ashraf Aboulnaga, and Panos Kalnis. Lusail: A system for querying linked data at scale. *Proceedings of the VLDB Endowment*, 11 (4):485–498, 2017.

[2] Maribel Acosta, Maria-Esther Vidal, Tomas Lampo, Julio Castillo, and Edna Ruckhaus. ANAPSID: An Adaptive Query Processing Engine for SPARQL Endpoints. In Lora Aroyo,

[26] For example: "b63446ad7d9f8960762c50a5a3492120.hdt"

[27] https://github.com/firmao/LODDatasetRelationsWeb

Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Noy, and Eva Blomqvist, editors, *The Semantic Web – ISWC 2011*, volume 7031 of *Lecture Notes in Computer Science*, pages 18–34. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-25072-9. . URL http://dx.doi.org/10.1007/978-3-642-25073-6_2.

[3] Ziya Akar, Tayfun Gökmen Halaç, Erdem Eser Ekinci, and Oguz Dikenelli. 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, 2012.

[4] Luigi Asprino, Wouter Beek, Paolo Ciancarini, Frank van Harmelen, and Valentina Presutti. The linked open data cloud is more abstract, flatter and less linked than you may think! *arXiv preprint arXiv:1906.08097*, 2019.

[5] Luigi Asprino, Wouter Beek, Paolo Ciancarini, Frank van Harmelen, and Valentina Presutti. Observing lod using equivalent set graphs: It is mostly flat and sparsely linked. In *International Semantic Web Conference*, pages 57–74. Springer, 2019.

[6] Luigi Asprino, Wouter Beek, Paolo Ciancarini, Frank van Harmelen, and Valentina Presutti. Triplifying equivalence set graphs. In *2019 ISWC Satellite Tracks (Posters and Demonstrations, Industry, and Outrageous Ideas), ISWC 2019-Satellites*, pages 253–256. CEUR-WS, 2019.

[7] Sören Auer, Jan Demter, Michael Martin, and Jens Lehmann. Lodstats–an extensible framework for high-performance dataset analytics. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 353–362. Springer, 2012.

[8] Ciro Baron Neto, Dimitris Kontokostas, Amit Kirschenbaum, Gustavo Publio, Diego Esteves, and Sebastian Hellmann. Idol: Comprehensive & complete lod insights. In *Proceedings of the 13th International Conference on Semantic Systems (SEMANTiCS 2017)*, 2017. URL https://svn.aksw.org/papers/2017/SEMANTiCS_IDOL/public.pdf.

[9] Wouter Beek, Laurens Rietveld, Hamid R Bazoobandi, Jan Wielemaker, and Stefan Schlobach. Lod laundromat: A uniform way of publishing other people's dirty data. In *International Semantic Web Conference*, pages 213–228. Springer, 2014.

[10] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227. IGI Global, 2011.

[11] Angelos Charalambidis, Antonis Troumpoukis, and Stasinos Konstantopoulos. Semagrow: Optimizing federated sparql queries. In *Proceedings of the 11th International Conference on Semantic Systems*, SEMANTICS '15, pages 121–128, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3462-4. . URL http://doi.acm.org/10.1145/2814864.2814886.

[12] Jeremy Debattista, Judie Attard, Rob Brennan, and Declan O'Sullivan. 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, 2019. Association for Computing Machinery. ISBN 9781450366755. . URL https://doi.org/10.1145/3308560.3317075.

[13] Antonin Delpeuch. A survey of openrefine reconciliation services. *CoRR*, abs/1906.08092, 2019. URL http://arxiv.org/abs/1906.08092.

[14] Mohamed Ben Ellefi, Zohra Bellahsene, Stefan Dietze, and Konstantin Todorov. Dataset recommendation for data linking: An intensional approach. In *European Semantic Web Conference*, pages 36–51. Springer, 2016.

[15] Mikel Emaldi, Oscar Corcho, and Diego López-de Ipina. Detection of related semantic datasets based on frequent subgraph mining. *IESD@ ISWC*, 5(6):7, 2015.

[16] Kemele M Endris, Mikhail Galkin, Ioanna Lytra, Mohamed Nadjib Mami, Maria-Esther Vidal, and Sören Auer. Querying interlinked data by bridging rdf molecule templates. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXIX*, pages 1–42. Springer, 2018.

[17] Kleanthi Georgala, Daniel Obraczka, and Axel-Cyrille Ngonga Ngomo. Dynamic planning for link discovery. In *European Semantic Web Conference*, pages 240–255. Springer, 2018.

[18] Olaf Görlitz and Steffen Staab. SPLENDID: SPARQL Endpoint Federation Exploiting VoID Descriptions. In *O. Hartig, A. Harth, and J. F. Sequeda, editors, 2nd International Workshop on Consuming Linked Data (COLD 2011) in CEUR Workshop Proceedings*, volume 782, October 2011.

[19] Olaf Hartig. Zero-knowledge query planning for an iterator implementation of link traversal based query execution. In Grigoris Antoniou, Marko Grobelnik, Elena Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Pan, editors, *The Semantic Web: Research and Applications*, pages 154–169, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-21034-1.

[20] Olaf Hartig. 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, 2013.

[21] Olaf Hartig and M Tamer Özsu. Walking without a map: Ranking-based traversal for querying linked data. In *International Semantic Web Conference*, pages 305–324. Springer, 2016.

[22] Olaf Hartig, Christian Bizer, and Johann-Christoph Freytag. Executing sparql queries over the web of linked data. In *International Semantic Web Conference*, pages 293–309. Springer, 2009.

[23] Oktie Hassanzadeh, Ken Q Pu, Soheil Hassas Yeganeh, Renée J Miller, Lucian Popa, Mauricio A Hernández, and Howard Ho. Discovering linkage points over web data. *Proceedings of the VLDB Endowment*, 6(6):445–456, 2013.

[24] Günter Ladwig and Thanh Tran. Linked Data Query Processing Strategies. In PeterF. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, JeffZ. Pan, Ian Horrocks, and Birte Glimm, editors, *The Semantic Web – ISWC 2010*, volume 6496 of *Lecture Notes in Computer Science*, pages 453–469. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-17745-3. . URL http://dx.doi.org/10.1007/978-3-642-17746-0_29.

[25] Günter Ladwig and Thanh Tran. SIHJoin: Querying Remote and Local Linked Data. In Grigoris Antoniou, Marko Grobelnik, Elena Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Pan, editors, *The Semantic Web: Research and Applications*, volume 6643 of *Lecture Notes in Computer Science*, pages 139–153. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-21033-4. . URL http://dx.doi.org/10.1007/978-3-642-21034-1_10.

[26] Vanessa Lopez, Victoria Uren, Marta Reka Sabou, and Enrico Motta. 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, 2009.

[27] Steven Lynden, Isao Kojima, Akiyoshi Matono, and Yusuke Tanimura. 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, 2011.

[28] Edgard Marx, Saeedeh Shekarpour, Tommaso Soru, Adrian MP Braşoveanu, Muhammad Saleem, Ciro Baron, Albert Weichselbraun, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, and Sören Auer. Torpedo: Improving the state-of-the-art rdf dataset slicing. In *Semantic Computing (ICSC), 2017 IEEE 11th International Conference On*, pages 149–156. IEEE, 2017.

[29] Qaiser Mehmood, Muhammad Saleem, Ratnesh Sahay, Axel-Cyrille Ngonga Ngomo, and Mathieu D'Aquin. Qppds: Querying property paths over distributed rdf datasets. *IEEE Access*, 7:101031–101045, 2019.

[30] Nandana Mihindukulasooriya, Giuseppe Rizzo, Raphaël Troncy, Oscar Corcho, and Raúl García-Castro. A two-fold quality assurance approach for dynamic knowledge bases: The 3cixty use case. In *(KNOW@ LOD/CoDeS)@ ESWC*, 2016.

[31] Thomas Minier, Hala Skaf-Molli, and Pascal Molli. Sage: Preemptive query execution for high data availability on the web. *CoRR*, abs/1806.00227, 2018. URL http://arxiv.org/abs/1806.00227.

[32] Gabriela Montoya, Hala Skaf-Molli, Pascal Molli, and Maria-Esther Vidal. *ISWC*, chapter Federated SPARQL Queries Processing with Replicated Fragments. 2015.

[33] Gabriela Montoya, Hala Skaf-Molli, and Katja Hose. The odyssey approach for optimizing federated sparql queries. In *International Semantic Web Conference*, pages 471–489. Springer, 2017.

[34] Ciro Baron Neto, Dimitris Kontokostas, Sebastian Hellmann, Kay Müller, and Martin Brümmer. 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)*, April 2016. URL http://ceur-ws.org/Vol-1593/article-07.pdf.

[35] Axel-Cyrille Ngonga Ngomo and Sören Auer. Limes—A time-efficient approach for large-scale link discovery on the web of data. *integration*, 15:3, 2011.

[36] Axel-Cyrille Ngonga Ngomo and Sören Auer. Limes—A time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of IJCAI*, 2011.

[37] George Papadakis, Ekaterini Ioannou, Themis Palpanas, Claudia Niederee, and Wolfgang Nejdl. A blocking framework for entity resolution in highly heterogeneous information spaces. *IEEE Transactions on Knowledge and Data Engineering*, 25 (12):2665–2682, 2012.

[38] Alexander Potocki, Muhammad Saleem, Tommaso Soru, Olaf Hartig, Martin Voigt, and Axel-Cyrille Ngonga Ngomo. Federated sparql query processing via costfed. 2017.

[39] Umair Qudus, Muhammad Saleem, Axel-Cyrille Ngonga Ngomo, and Young-koo Lee. An empirical evaluation of cost-based federated sparql query processing engines. *Semantic Web Journal*, 2020.

[40] Bastian Quilitz and Ulf Leser. Querying Distributed RDF Data Sources with SPARQL. In Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, *The Semantic Web: Research and Applications*, volume 5021 of *Lecture Notes in Computer Science*, pages 524–538. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-68233-2. . URL http://dx.doi.org/10.1007/978-3-540-68234-9_39.

[41] Erhard Rahm. The case for holistic data integration. In *East European Conference on Advances in Databases and Information Systems*, pages 11–27. Springer, 2016.

[42] Theodoros Rekatsinas, Xin Luna Dong, Lise Getoor, and Divesh Srivastava. Finding quality in quantity: The challenge of discovering valuable sources for integration. In *CIDR*, 2015.

[43] Michael Röder, Axel-Cyrille Ngonga Ngomo, Ivan Ermilov, and Andreas Both. Detecting similar linked datasets using topic modelling. In *European Semantic Web Conference*, pages 3–19. Springer, 2016.

[44] Jacobo Rouces, Gerard de Melo, and Katja Hose. Complex schema mapping and linking data: Beyond binary predicates. 2016.

[45] Alieh Saeedi, Markus Nentwig, Eric Peukert, and Erhard Rahm. Scalable matching and clustering of entities with famer. *Complex Systems Informatics and Modeling Quarterly*, (16): 61–83, 2018.

[46] Muhammad Saleem and Axel-Cyrille Ngonga Ngomo. HiBISCuS: Hypergraph-Based Source Selection for SPARQL Endpoint Federation. In Valentina Presutti, Claudia d'Amato, Fabien Gandon, Mathieu d'Aquin, Steffen Staab, and Anna Tordai, editors, *The Semantic Web: Trends and Challenges*, volume 8465 of *Lecture Notes in Computer Science*, pages 176–191. Springer International Publishing, 2014. ISBN 978-3-319-07442-9. . URL http://dx.doi.org/10.1007/978-3-319-07443-6_13.

[47] Muhammad Saleem, Axel-Cyrille Ngonga Ngomo, Josiane Xavier Parreira, HelenaF. Deus, and Manfred Hauswirth. DAW: Duplicate-AWare Federated Query Processing over the Web of Data. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, JosianeXavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web – ISWC 2013*, volume 8218 of *Lecture Notes in Computer Science*, pages 574–590. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-41334-6. . URL http://dx.doi.org/10.1007/978-3-642-41335-3_36.

[48] Muhammad Saleem, Shanmukha S Padmanabhuni, Axel-Cyrille Ngonga Ngomo, Jonas S Almeida, Stefan Decker, and Helena F Deus. Linked cancer genome atlas database. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 129–134. ACM, 2013.

[49] Muhammad Saleem, Shanmukha S. Padmanabhuni, Axel-Cyrille Ngonga Ngomo, Jonas S. Almeida, Stefan Decker, and Helena F. Deus. 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, 2013. ACM. ISBN 978-1-4503-1972-0. . URL http://doi.acm.org/10.1145/2506182.2506200.

[50] Muhammad Saleem, Muhammad Intizar Ali, Aidan Hogan, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. Lsq: The linked sparql queries dataset. In *The Semantic Web-ISWC*

2015, pages 261–269. Springer, 2015. URL https://svn.aksw.org/papers/2015/ISWC_LSQ/public.pdf.

[51] Muhammad Saleem, Muhammad Intizar Ali, Ruben Verborgh, and A-C Ngonga Ngomo. Federated query processing over linked data. In *Tutorial at International Semantic Web Conference*, 2015. URL https://www.slideshare.net/muhammad_saleem/federated-sparql-query-processing-iswc2015-tutorial.

[52] Muhammad Saleem, Yasar Khan, Ali Hasnain, Ivan Ermilov, and Axel-Cyrille Ngonga Ngomo. A fine-grained evaluation of sparql endpoint federation systems. *Semantic Web Journal*, pages 1–26, 2015. URL https://content.iospress.com/articles/semantic-web/sw186.

[53] Muhammad Saleem, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. Feasible: A feature-based sparql benchmark generation framework. In *The Semantic Web-ISWC 2015*, pages 52–69. Springer, 2015. URL https://svn.aksw.org/papers/2015/ISWC_FEASIBLE/public.pdf.

[54] Muhammad Saleem, Ali Hasnain, and Axel-Cyrille Ngonga Ngomo. Largerdfbench: A billion triples benchmark for sparql endpoint federation. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 0(0), 2016. ISSN 1570-8268. URL http://www.websemanticsjournal.org/index.php/ps/article/view/513.

[55] Muhammad Saleem, Alexander Potocki, Tommaso Soru, Olaf Hartig, and Axel-Cyrille Ngonga Ngomo. Costfed: Cost-based query optimization for sparql endpoint federation. *Semantics*, 137:163–174, 2018.

[56] Michael Schmidt, Olaf Görlitz, Peter Haase, Günter Ladwig, Andreas Schwarte, and Thanh Tran. FedBench: A Benchmark Suite for Federated Semantic Data Query Processing. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Noy, and Eva Blomqvist, editors, *The Semantic Web – ISWC 2011*, volume 7031 of *Lecture Notes in Computer Science*, pages 585–600. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-25072-9. . URL http://dx.doi.org/10.1007/978-3-642-25073-6_37.

[57] Andreas Schwarte, Peter Haase, Katja Hose, Ralf Schenkel, and Michael Schmidt. FedX: Optimization Techniques for Federated Query Processing on Linked Data. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Noy, and Eva Blomqvist, editors, *The Semantic Web – ISWC 2011*, volume 7031 of *Lecture Notes in Computer Science*, pages 601–616. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-25072-9. . URL http://dx.doi.org/10.1007/978-3-642-25073-6_38.

[58] Ruben Taelman, Joachim Van Herwegen, Miel Vander Sande, and Ruben Verborgh. Comunica: A modular sparql query engine for the web. In *International Semantic Web Conference*, pages 239–255. Springer, 2018.

[59] Nhuan D To, Marek Z Reformat, and Ronald R Yager. Linked open data: Uncertainty in equivalence of properties. In *Advances in Fuzzy Logic and Technology 2017*, pages 418–429. Springer, 2017.

[60] Jacopo Urbani, Spyros Kotoulas, Jason Maassen, Frank Van Harmelen, and Henri Bal. Owl reasoning with webpie: calculating the closure of 100 billion triples. In *Extended Semantic Web Conference*, pages 213–227. Springer, 2010.

[61] Andre Valdestilhas, Tommaso Soru, and Axel-Cyrille Ngonga Ngomo. A high-performance approach to string similarity using most frequent k characters. In *OM@ ISWC*, pages 1–12, 2017.

[62] Andre Valdestilhas, Tommaso Soru, Markus Nentwig, Edgard Marx, Muhammad Saleem, and Axel-Cyrille Ngonga Ngomo. Where is my uri? In *European Semantic Web Conference*, pages 671–681. Springer, 2018.

[63] André Valdestilhas, Tommaso Soru, and Muhammad Saleem. More complete resultset retrieval from large heterogeneous rdf sources. In *Proceedings of the 10th International Conference on Knowledge Capture*, pages 223–230, 2019.

[64] Ruben Verborgh and Miel Vander Sande. The Semantic Web identity crisis: In search of the trivialities that never were. *Semantic Web Journal*, 11(1):19–27, January 2020. . URL https://ruben.verborgh.org/articles/the-semantic-web-identity-crisis/.

[65] Ruben Verborgh, Miel Vander Sande, Olaf Hartig, Joachim Van Herwegen, Laurens De Vocht, Ben De Meester, Gerald Haesendonck, and Pieter Colpaert. 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, 2016.

[66] Xin Wang, Thanassis Tiropanis, and Hugh C. Davis. LHD: Optimising Linked Data Query Processing Using Parallelisation. In *C. Bizer, T. Heath, T. Berners-Lee, M. Hausenblas, S. Auer, editors, Proceedings of the WWW2013 Workshop on Linked Data on the Web in CEUR Workshop Proceedings*, volume 996, May 2013. URL http://eprints.soton.ac.uk/350719/.

[67] Cheng Xie, Ying Lin, and Hongming Cai. Instance-driven property alignment in linked open data cloud. In *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*, pages 420–425. IEEE, 2018.