LODChain: Strengthen the Connectivity of Your RDF Dataset to the Rest LOD Cloud

Michalis Mountantonakis[1,2(✉)] and Yannis Tzitzikas[1,2]

[1] Institute of Computer Science - FORTH-ICS, Heraklion, Greece
{mountant,tzitzik}@ics.forth.gr
[2] Computer Science Department, University of Crete, Heraklion, Greece

Abstract. It is not an easy task for a data owner to publish a dataset as Linked Data with connections to existing datasets since there are too many datasets, thus it is hard to find the related ones, to download them and to check their content (let alone to apply entity matching over them). However, the connections with other datasets are important for discoverability, browsing, and querying in general. To alleviate this problem in this paper we introduce `LODChain`, a service that can help a provider to strengthen the connections between his/her dataset and the rest of datasets. `LODChain` finds the common entities, schema elements and triples among the dataset at hand and hundreds of LOD Datasets and through equivalence reasoning it suggests to the user various inferred connections, as well as related datasets. In addition, it detects erroneous mappings, and offers various content-based dataset discovery services, for enabling the enrichment of datasets' content. The key difference with the existing approaches is that they are metadata-based, while what we propose is data-based. We present an implementation of `LODChain`, and we report various experimental results over real and synthetic datasets.

Keywords: Linked data · Connectivity · Dataset search · Data discovery

Resource Type: Software Service.
Resource URL: `https://demos.isl.ics.forth.gr/LODChain`.
Permanent URL: `https://doi.org/10.5281/zenodo.6467419`.

1 Introduction

An increasing number of datasets are published through Linked Open Data (LOD) principles, i.e., over 10,000 datasets [24]. For making a new RDF dataset more discoverable and reusable, for improving its trustworthiness and for enriching its content, several tasks should be executed before its actual publishing to the web. Indeed, it is a prerequisite to discover existing datasets, to create connections with them, through *equivalence relationships* such as `owl:sameAs`, to check the quality of such relationships and to create rich metadata. These tasks can be assisted at large scale through existing approaches; for discovering and exporting relevant datasets through metadata-based services like http://lod-cloud.net [19] or Google Dataset Search [6], for transforming and querying data [11,34], for creating schema and instance mappings [8,30], for quality assessment [28,48], for finding all the URIs of a given entity [22,39] and others.

However, the huge volume of LOD datasets makes it difficult to discover every possible relevant dataset, especially given that a) existing approaches for publishing RDF datasets do not favor their discoverability and reusability; e.g., [10] states that "up till now, data consumers could, painfully, crawl or search the LOD cloud diagram for a potential dataset", and b) Dataset Search engines rely on metadata and ignore the actual content of datasets [7]. Also, even if one has discovered and fetched the content of all the datasets, it would be very costly and time consuming to find commonalities with these datasets at scale, to create mappings and to check their quality. Such Data Integration tasks usually require manual work, thus huge human effort, if applied at scale, as well as high storage and computational capacity, which can be prohibitively expensive [24].

Due to these limitations, the major target of LOD, i.e., linking and integration [5], has not been yet reached. Indeed, LOD Cloud is sparsely linked; publishers tend to connect their datasets with few and popular datasets and ontologies [2,22], and "LOD Cloud is at risk of becoming a museum for datasets" [10]. Hence, there is a high need for services that can strengthen the connectivity of a dataset to the rest of LOD Cloud. To alleviate this problem, we introduce the research prototype `LODChain`, which receives a dataset, e.g., before its actual publishing, computes the transitive and symmetric closure of its `owl:sameAs` (for entities), `owl:equivalentProperty` (for properties) and `owl:equivalentClass` (for classes) relationships with hundreds of LOD datasets, i.e., those indexed by LODsyndesis KB [25], and offers connectivity and enrichment services. For a new dataset, say D_{new}, `LODChain` a) spots errors in equivalence mappings, b) infers new mappings, connections and all the common elements, e.g., entities and triples, between D_{new} and LOD datasets, c) discovers its K most relevant datasets, and d) enriches D_{new}, e.g., for offering advanced query capabilities.

Comparing to large-scale services, like LODsyndesis [22,25] or LODlaundromat [34], to the best of our knowledge `LODChain` is the first service for strengthening the connectivity of an RDF dataset to the rest of LOD Cloud, at any time, even before its actual publishing. In [26], we described a preliminary version of `LODChain` providing analytics only for the entities of Cultural Heritage datasets. In this paper, we introduce the current version of `LODChain`, which also leverages the schema and the triples of a dataset of any domain and offers more connectivity analytics and services. We present use cases showcasing its impact for the discoverability, reusability and trustworthiness of a dataset, and the process of `LODChain`, including methods for computing the `owl:sameAs` closure between a new dataset and the precomputed inference (of 45 million owl:sameAs mappings) from hundreds of LOD datasets. Finally, we provide comparative results for the effectiveness and the efficiency by using 5 real and 2 synthetic datasets, indicatively, we obtained at least 450% increase to the connections of real datasets.

The rest of the paper is organized as follows. Section 2 introduces the related work, whereas Sect. 3 presents several use cases by using the real dataset *WW1LOD* [18] containing historical data about World War I. Section 4 presents the process of `LODChain`, while Sect. 5 provides comparative results for its efficiency and effectiveness. Finally, Sect. 6 concludes the paper and identifies directions for future research.

2 Related Work

Several approaches have been proposed for aiding the creation, integration and publication of an RDF dataset, see a recent survey [24] for more details. Here, we focus on a) large scale services for hundreds or even thousands of RDF datasets, b) dataset search and discovery approaches and c) data enrichment approaches, which can be exploited for strengthening the connectivity of a dataset.

Large Scale Services for Multiple RDF Datasets. First, LODLaundromat [34] offers data cleaning and transformation for thousands of RDF documents, whereas LOD-a-lot [11] provides a single integrated file containing the documents of LODLaundromat, for enabling their reusability and for offering more advanced query capabilities. Moreover, there are services for finding all the datasets of a given URI, e.g., WIMU [39] and LODsyndesis [23,25], all the equivalent URIs of a given URI, such as MetaLink [3] and LODsyndesis, i.e., by computing the transitive closure of millions of `owl:sameAs` mappings, and approaches for detecting erroneous `owl:sameAs` links [31,40]. Such services can be exploited from the publisher of a new dataset, for enriching the connectivity of their dataset, e.g., by adding inferred links, and for checking its quality.

RDF Dataset Search and Discovery. Existing services, such as LOD Cloud (http://lod-cloud.net), Google Dataset Search [7], LODatio [12] and LODAtlas [32], provide metadata-based search for discovering relevant RDF datasets, whereas [44] evaluates different snippet generation algorithms that can be used for performing RDF Dataset Search for thousands of datasets. Furthermore, [29] introduces a framework for content-based similarity dataset discovery, by using external knowledge bases (e.g., Wikidata), whereas LODsyndesis offers content-based dataset discovery for finding the most relevant datasets to a given one.

RDF Dataset Enrichment. There are several approaches introducing methods (and their importance) for enriching the content of datasets for several domains, e.g., for tourism [35,46], for Open City Data [4] or for marine domain [37]. Additionally, one can enrich the data for a given entity, by visiting dereferencable equivalent URIs, e.g., through services like Metalink or LODsyndesis, by using instance matching tools, e.g., Silk [42], by exploiting link-traversal queries [38], or through SPARQL queries to the endpoints of relevant datasets. A catalog of SPARQL endpoints is available from SPARQLES [41] and SpEnD [47].

Comparison with Existing Approaches. Concerning the mentioned works, a) the process of large-scale services (e.g., [25,34]) is done periodically for a high number of open datasets, i.e., it is infeasible to exploit the offered services for unpublished data or for closed linked data, as big organizations/companies maintain (e.g., [14]), b) the dataset search services (e.g., [7,32]) are metadata-based, and c) the data enrichment approaches are either domain specific (e.g., [35,37]), or require additional effort for discovering enriched data (e.g., [3]). On the contrary, to the best of our knowledge, `LODChain` is the first service that can be used by a publisher at any time, for strengthening the connectivity of a dataset (of any domain) by providing content-based connectivity analytics and enrichment to the rest of LOD datasets. Thereby, it can be used for unpublished or "closed" data, without needing to download any software or to fetch any RDF dataset.

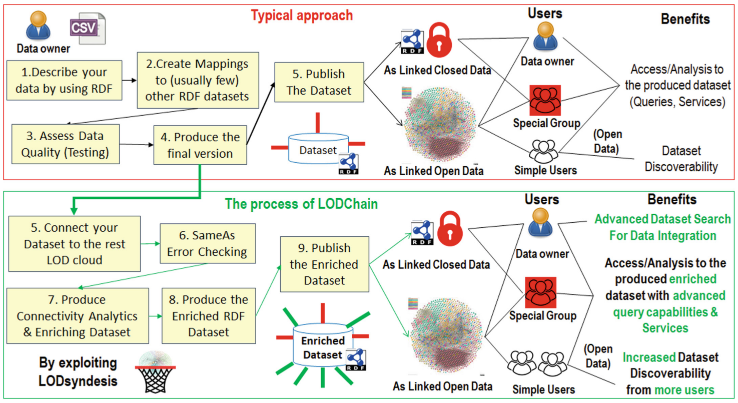
Typical approach
Data owner
CSV
1.Describe your data by using RDF
2.Create Mappings to (usually few) other RDF datasets
3. Assess Data Quality (Testing)
4. Produce the final version
5. Publish The Dataset
Dataset
As Linked Closed Data
As Linked Open Data
Users
Data owner
Special Group
Simple Users
(Open Data)
Benefits
Access/Analysis to the produced dataset (Queries, Services)
Dataset Discoverability
The process of LODChain
5. Connect your Dataset to the rest LOD cloud
6. SameAs Error Checking
7. Produce Connectivity Analytics & Enriching Dataset
8. Produce the Enriched RDF Dataset
9. Publish the Enriched Dataset
Enriched Dataset
By exploiting LODsyndesis
As Linked Closed Data
As Linked Open Data
Users
Data owner
Special Group
Simple Users
(Open Data)
Benefits
Advanced Dataset Search For Data Integration
Access/Analysis to the produced enriched dataset with advanced query capabilities & Services
Increased Dataset Discoverability from more users

Fig. 1. Typical approach versus the `LODChain` approach for publishing an RDF dataset

3 Data Publishing - Typical Approach Versus `LODChain`

We define the user categories that can have access to a given RDF dataset: i) data owners (publishers), i.e., people, organizations, universities, etc., that are the creators and owners of a given dataset, ii) a special group of interest, i.e., certain individuals, e.g., people, services, within an organization, a research project, etc., iii) users (or services), i.e., any user on the web like publishers of other datasets, or services that have access or/and reuse the content of a dataset. This category is a superset of the previous two, i.e., publishers and special group of interest. Figure 1 shows the typical approach of publishing an RDF dataset. Indeed, a data owner describes the desired data in RDF format, creates mappings with usually few RDF datasets and after assessing its quality, e.g., through competency queries [45], he/she produces the final version. For achieving this target, more steps are usually required, e.g., data conversion, transformation, etc. [24]. Afterwards, the final version is published, either to an open domain (e.g., to LOD Cloud) which can be publicly accessible, discoverable and reusable from any user/service on the web (for performing an analysis, for creating an application, etc.), or to a closed domain, i.e., these benefits are provided to a special group of interest.

Concerning the process of LODChain (lower part of Fig. 1), it receives an RDF dataset, e.g., before its actual publishing, for checking and improving its connectivity to the LOD Cloud and for enriching its content (by exploiting LODsyndesis). The target is to increase the benefits of publishing a dataset for any category of users. Below, we present how LODChain can contribute to these benefits (lower right side of Fig. 1), by showing a scenario with a real dataset.

Benefits of LODChain for Data Publishing (Use Cases). We describe use cases by following the steps of the lower side of Fig. 1. For the introduced scenario, we use the small real dataset *WW1LOD* [18]. It contains data about World War I and includes 47,616 triples and 547 sameAs mappings to 5 RDF datasets, including popular ones, such as DBpedia [16] and GeoNames (more statistics are given in §5). For checking more cases, we use a synthetic version of *WW1LOD*, say *WW1LOD*$_{synt}$, where we have added 50 erroneous `owl:sameAs` mappings. The scenario starts when the data owner decides to use LODChain for improving the connectivity of his dataset, before its actual publishing. We suppose that the first version of the dataset is *WW1LOD*$_{synt}$ (upper left side of Fig. 2). We show how the lifecycle of *WW1LOD*$_{synt}$ can be changed by using LODChain, and for each *Use Case* (UC) we indicate its potential impact. The use cases are also presented in an online video (https://youtu.be/Kh9751p32tM).

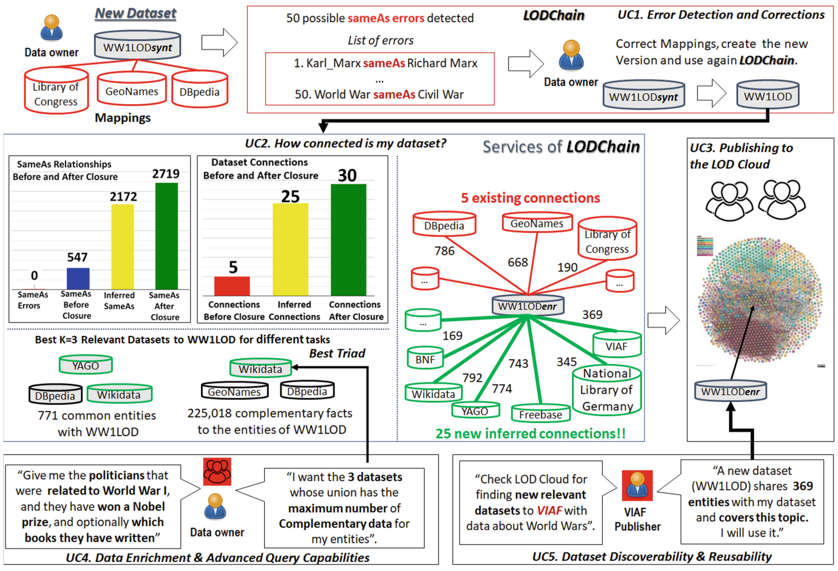
New Dataset
Data owner
WW1LODsynt
Library of Congress
GeoNames
DBpedia
Mappings
50 possible sameAs errors detected
List of errors
1. Karl_Marx sameAs Richard Marx
...
50. World War sameAs Civil War
LODChain
UC1. Error Detection and Corrections
Correct Mappings, create the new Version and use again LODChain.
Data owner
WW1LODsynt
WW1LOD
UC2. How connected is my dataset?
Services of LODChain
SameAs Relationships Before and After Closure
0
547
2172
2719
SameAs Errors
SameAs Before Closure
Inferred SameAs
SameAs After Closure
Dataset Connections Before and After Closure
5
25
30
Connections Before Closure
Inferred Connections
Connections After Closure
Best K=3 Relevant Datasets to WW1LOD for different tasks
Best Triad
YAGO
DBpedia
Wikidata
771 common entities with WW1LOD
Wikidata
GeoNames
DBpedia
225,018 complementary facts to the entities of WW1LOD
5 existing connections
DBpedia
GeoNames
Library of Congress
786
668
190
...
WW1LODenr
369
169
VIAF
BNF
743
345
792
774
National Library of Germany
Wikidata
YAGO
Freebase
25 new inferred connections!!
UC3. Publishing to the LOD Cloud
WW1LODenr
"Give me the politicians that were related to World War I, and they have won a Nobel prize, and optionally which books they have written"
Data owner
"I want the 3 datasets whose union has the maximum number of Complementary data for my entities".
UC4. Data Enrichment & Advanced Query Capabilities
"Check LOD Cloud for finding new relevant datasets to VIAF with data about World Wars".
VIAF Publisher
"A new dataset (WW1LOD) shares 369 entities with my dataset and covers this topic. I will use it."
UC5. Dataset Discoverability & Reusability

Fig. 2. A scenario with 5 use cases by uploading a new dataset to LODChain

UC1. Error Detection and Corrections. The publisher uploads the version *WW1LOD*$_{synt}$ to LODChain, which informs the publisher that there are 50 possible erroneous mappings, and provides a list containing these errors (upper side of Fig. 2). The publisher downloads the list for correcting the mappings and for creating the version *WW1LOD*. Without checking the quality of equivalence relationships, which is quite difficult at scale [24], it can result in erroneous relationships, which can negatively affect the trustworthiness of a publisher.

UC2. How Connected is My Dataset? The scenario continues by uploading the version *WW1LOD*. `LODChain` analyzes the dataset, informs the publisher that there are no errors, and it offers several connectivity analytics. Some of these real results are shown in UC2 of Fig. 2, i.e., we inferred 2,172 `owl:sameAs` relationships (397% increase), which resulted to 25 new connections, i.e., 500% increase, since the initial version had only 5 connections. In Fig. 2 (and also in Fig. 1) the inferred connections are depicted as edges/nodes with green color and the old connections with red color, whereas the label of each edge indicates the number of their common entities. Except for finding new connections with popular datasets (e.g., Wikidata [43], YAGO [33]), which is expected due to transitivity (they share millions of entities with DBpedia), it is feasible to discover connections with not so popular datasets that were unknown to the publisher. As regards dataset discovery and selection, indicatively Fig. 2 shows the best triad of datasets offering a) the most common entities and b) the most complementary triples for *WW1LOD* entities. We suppose that the publisher exports all the inferred data and analytics for reusing them (i.e., version $WW1LOD_{enr}$), although one can decide to use any subset of these enriched data.

UC3. Publishing the Enriched Dataset to LOD Cloud. Suppose that the publisher decides to upload the enriched version $WW1LOD_{enr}$, including all the inferred equivalence mappings and connectivity analytics, to LOD Cloud (see UC3 in Fig. 2). Below, we explain the possible impact for any users' category.

UC4. Advanced Query Capabilities, Enrichment and Verification. We mention the benefits of using `LODChain` for the data owner or/and for a special group of interest. By enriching *WW1LOD* through multiple datasets, more complex queries can be answered, such as the following: "Give me the politicians that were related to first World War, they have won a Nobel prize and optionally information for the books that they have written" (UC4 in Fig. 2). That query requires data from several datasets, e.g., for "Theodore Roosevelt", which is a possible answer, the first part can be answered from *WW1LOD* (http://ldf.fi/ww1lod/96403a6a), the second (nobel prizes) from Wikidata (https://www.wikidata.org/wiki/Q33866), and the third (books) from the National Library of Germany (http://d-nb.info/gnd/118749633). Although *WW1LOD* was not connected to Wikidata and the National Library of Germany (Fig. 2), `LODChain` inferred and added these connections in the enriched version.

However, it is not always feasible to export and use all the relevant datasets due to huge data volume, thereby, `LODChain` also offers services for selecting the K most relevant datasets for a desired task. For example, two possible tasks are to find the combination of K = 3 datasets providing i) "the most complementary triples for the entities of *WW1LOD*" (UC4 in Fig. 2), i.e., for data enrichment, or ii) "the most common entities with *WW1LOD*", i.e., for data verification. From the 30 connected datasets, there exists 4,060 combinations of 3 datasets, thereby, it is quite expensive to check any possible combination. However, by exploiting the precomputed results of `LODChain`, one can find very fast the K = 3 most relevant datasets for a task (e.g., UC4 in Fig. 2). These examples indicate the impact of `LODChain` for the dataset selection process, i.e., the K most relevant datasets differ according to the desired needs, even for the same dataset.

UC5. Dataset Search, Discoverability and Reusability. Here, we mention the benefits for any user/service (in case of open data). Suppose that publishers of other datasets periodically check the LOD Cloud for discovering relevant datasets to their dataset, that were recently published, or an automated service informs them when a new dataset is connected with their dataset. E.g., in UC5 of Fig. 2, the publisher of VIAF desires to find datasets having a) common entities with VIAF and b) information about World Wars, i.e., for enriching or for verifying their content. They observe that *WW1LOD* not only covers this topic, but also has 369 common entities with VIAF, thereby they decide to use it. Without `LODChain`, it would be extremely difficult for most publishers (25 out of 30) to discover that *WW1LOD* is relevant to their dataset (and to reuse it), e.g., the first version of *WW1LOD* did not have links to VIAF.

4 `LODChain`: Connecting Your Dataset to the LOD Cloud

First, we describe LODsyndesis (which is used from `LODChain`), and then the steps of `LODChain`, by showing a running example of how to strengthen the connectivity of a new dataset, i.e., D_{new} (upper left part of Fig. 3). For finding new connections for D_{new}, it is prerequisite D_{new} to contain links to at least one dataset, e.g., in Fig. 3, D_{new} is connected with 2 datasets: DBpedia and Wikidata. Indeed, we do not perform instance and schema matching, but we infer new connections by computing the closure of equivalence mappings among D_{new} and the rest of LOD datasets. Finally, the data of D_{new} are saved temporarily in LODsyndesis indexes for a user session for producing the desired output.

4.1 LODsyndesis Aggregated Knowledge Graph

`LODChain` is based on LODsyndesis [25], which is an *Aggregated Knowledge Graph* derived by *aggregating* the content of datasets, computing the transitive and symmetric closure of 45 million *equivalence relationships*, and offering *semantics-aware indexes and services* for over 412 million entities and 2 billion triples from 400 LOD datasets. Concerning the quality of the mentioned closure, it has been evaluated in a semi-manual way in our past work [20]. Afterwards, LODsyndesis keeps a unique representation for each real world entity, property and class, while also storing their provenance. The lower right side of Fig. 3 shows a graph representation of the LODsyndesis data about the Greek composer "Mikis Theodorakis". LODsyndesis has precomputed the `owl:sameAs` closure for "Mikis Theodorakis" URIs, has stored their provenance, and has replaced all these URIs by a single internal URI (see the single node for M. Theodorakis). The same process has been done for all the entities (e.g., "Paris"), properties (e.g., "bornYear") and so forth. Regarding the triples (i.e., facts), it stores at the same place in the index, all the triples of an entity occurring either as a subject or as an object, and their provenance. Thereby, triples having entities as objects are stored twice in the index. The lower right side of Fig. 3 shows the triples for M. Theodorakis and their provenance (see the bold text under each node).

4.2 The Steps of `LODChain`

Step A. Input. `LODChain` supports many formats: NTriples, NQuads, RDF/XML and Turtle (by using the RDF4J library https://rdf4j.org/). The publishers just give a link of their dataset in one of these formats. They can optionally type the title and domain of their dataset and can select to perform the process only for a subset of their dataset (e.g., 10,000 triples) for having a very fast overview.

Step B. Computation of Equivalence Relationships Closure and Provenance. The objective is to detect which real world objects are a) common in D_{new} and LODsyndesis, b) unique in D_{new}, and to detect c) errors in the equivalence relationships of D_{new}. `LODChain` reads the triples of D_{new}, collects all the `owl:sameAs`, `owl:equivalent Property` and `owl:equivalentClass` relationships and partitions the URIs in three sets: entities, properties and classes.

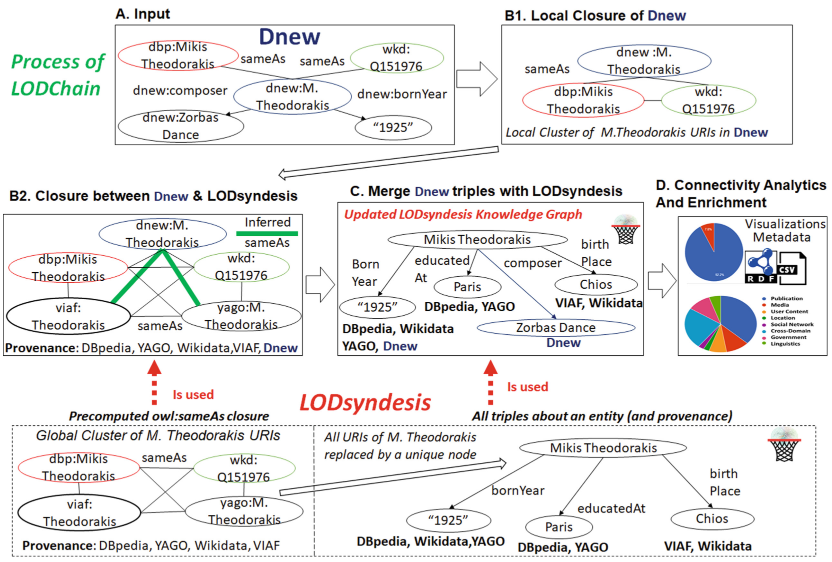

Fig. 3. Running example. The steps of `LODChain` for a new dataset D_{new}

B1. Local Computation of Closure in D_{new}. For each type of these URIs and *equivalence relationships*, we use the signature-based algorithm proposed in [22], for computing the transitive and symmetric closure of the equivalence relationships of D_{new}. Then we store all the URIs of D_{new} referring to the same real world object in the same local "cluster" (i.e., class of equivalence). At the end, the set $C(D_{new})$ is created, which includes all the "local" clusters of D_{new} (which are pairwise disjoint). For an entity e in D_{new} we shall use $loc(e) \in C(D_{new})$ to denote the local cluster including the URIs of e, e.g. the local cluster of "M. Theodorakis" (Step B1 of Fig. 3) contains 3 URIs, $loc(e) = \langle$dnew:M. Theodorakis, dbp:Mikis Theodorakis, wkd:Q151976$\rangle$. Finally, for the URIs that are not part of any equivalence relationship, their cluster contains a single URI, e.g., in Fig. 3 for the entity "Zorbas Dance", $loc(e) = \{dnew : Zorbas\_Dance\}$.

B2. Computation of Closure Between D_{new} and LODsyndesis. Here, we merge the local "clusters" of D_{new}, with the results of the precomputed closure of LODsyndesis. We shall use $C(LOD)$ to denote the set of all the clusters of LODsyndesis (which are pairwise disjoint), and $glob(e) \in C(LOD)$ to denote the cluster of entity e in LODsyndesis, e.g., for "Mikis Theodorakis" $glob(e) = \{$dbp:Mikis Theodorakis, wkd:Q151976, viaf:Theodorakis, yago:M. Theodorakis$\}$ (lower left side of Fig. 3). Moreover, $prov(e)$ denotes the provenance of e in LODsyndesis, e.g., for the mentioned entity $prov(e) = \{$DBpedia,YAGO,Wikidata, VIAF$\}$. Each $loc(e) \in C(D_{new})$ belongs to exactly one of the three below rules:

Rule 1. No Match Between Clusters: New Entities. Here, $loc(e)$ does not match with any global cluster of LODsyndesis, thereby the corresponding entity exists only in D_{new}, i.e., for a given $loc(e) \in C(D_{new})$, $\nexists\ glob(e') \in C(LOD)\ s.t.\ loc(e) \cap glob(e') \neq \emptyset$. In such a case, we add $loc(e)$ to $C(LOD)$.

```
Algorithm 1: Computation of Closure between $D_{new}$ and LODsyndesis
Input: Local Closure $C(D_{new})$ and global closure $C(LOD)$
Output: The common and unique entities, and errors in owl:sameAs mappings
1  $uniqEnt \leftarrow \emptyset$, $cmnEnt \leftarrow \emptyset$, $errors \leftarrow \emptyset$
2  forall $loc(e) \in C(D_{new})$ do                        // for each local cluster
3      $glob_{new}(e) \leftarrow \emptyset$                 // init. the new global cluster of $e$
4      forall $u \in loc(e)$ do                     // For each URI $u$ of local cluster
5          if $u \in glob(e'), glob(e') \in C(LOD)$ then          // If $u$ in LODsyndesis
6              if $glob_{new}(e) \equiv \emptyset$ then             // $1^{st}$ global cluster for $e$
7                  $glob_{new}(e) \leftarrow glob(e')$            // Store the global cluster
8              else if $glob_{new}(e) \neq glob(e')$ then   // $2^{nd}$ global cluster for $e$
9                  delete $glob_{new}(e)$    // Delete $e$, matches 2 glob. clusters
10                 $errors \leftarrow errors \cup \{loc(e)\}$            // Add $loc(e)$ to errors
11                 break and go to line 2    // Continue with the next $loc(e)$
       // After finishing with all the URIs of $loc(e)$
12
13     if $glob_{new}(e) \equiv \emptyset$ then                        // No global cluster found
14         $glob_{new}(e) \leftarrow loc(e)$           // $loc(e)$ is the global cluster of $e$
15         $uniqEnt \leftarrow uniqEnt \cup \{e\}$                // Add $e$ to unique entities
16         $prov(e) \leftarrow D_{new}$                                // Store its provenance
17     else if $glob_{new}(e) \neq \emptyset$ then          // A single global cluster found
18         $glob_{new}(e) \leftarrow glob_{new}(e) \cup loc(e)$        // Update global cluster of $e$
19         $cmnEnt \leftarrow cmnEnt \cup \{e\}$                 // Add $e$ to common entities
20         $prov(e) \leftarrow prov(e) \cup \{D_{new}\}$                  // Update its provenance
21 return $cmnEnt$, $uniqEnt$, $errors$
```

Rule 2. Single Match Between Clusters: Inferring New Relationships. If a given $loc(e) \in C(D_{new})$ matches with exactly one $glob(e')$, i.e. if $\exists!\ glob(e') \in C(LOD)\ s.t.\ glob(e') \cap loc(e) \neq \emptyset$, then we assume that $e \equiv e'$, and we perform the following operation: $loc(e) \cup glob(e')$. In step B1 of Fig. 3, two URIs of the local cluster, i.e., dbp:Mikis Theodorakis and wkd:Q151976, belong also to the same global cluster in LODsyndesis (see the lower side of Fig. 3). By merging these clusters (step B2 of Fig. 3), we inferred two new `owl:sameAs` mappings for the URI "dnew:M. Theodorakis", we updated its provenance, and we managed to discover two new connections for D_{new}, i.e., VIAF and YAGO.

Rule 3. Cluster Conflicts: Detecting Possible Errors. If a $loc(e) \in C(D_{new})$ matches with two or more clusters of LODsyndesis, i.e., if $\exists\ glob(e_1), glob(e_2) \ \in C(LOD).\ s.t.\ glob(e_1)\ \cap\ loc(e) \neq \emptyset\ ,\ glob(e_2) \cap loc(e) \neq \emptyset\ , glob(e_1) \neq glob(e_2)$, then this is an indication of error. For instance, suppose that we have added the following erroneous mapping in D_{new}: ⟨dnew:M. Theodorakis, `owl:sameAs`, dbp:Theodore Roosevelt⟩. Due to closure, in Step B1 the result would be $loc(e)$={dnew:M. Theodorakis, dbp:Mikis Theodorakis, wkd:Q151976, dbp:Theodore Roosevelt}. However, by proceeding to Step B2, $loc(e)$ would match with two clusters of LODsyndesis, i.e., the URIs dbp:Mikis Theodorakis and dbp:Theodore Roosevelt refer to different entities and belong to different global clusters. `LODChain` identifies such cases and informs the user.

Algorithm for Step B2. Algorithm 1 detects if there is zero, one or more global clusters, that match with the URIs of $loc(e)$, for finding common and unique entities, and errors in the `owl:sameAs` mappings. Algorithm 1 reads each $loc(e)$ separately and iterates over all its URIs (lines 2–11). For each URI u, it performs a binary search in LODsyndesis index (see line 5), for checking if it occurs in a global cluster (i.e., if it exists in LODsyndesis). Concerning the rules, for the local clusters belonging to Rule 1, the lines 6–11 will never be executed, since there is not a global cluster containing at least one URI of $loc(e)$. On the contrary, the lines 13–16 will be executed and the entity e will be stored as unique. Regarding Rule 2, the first time that we find a URI of $loc(e)$ that belongs to a global cluster, we retrieve and store the corresponding global cluster (lines 5–7). In case of finding another URI(s) of $loc(e)$ belonging to the same global cluster, we just continue with the next URI of $loc(e)$. At the end, lines 17–20 are executed for updating the global cluster of e (i.e., $glob_{new}(e)$) and its provenance, and for storing the entity as a common one. Concerning Rule 3, in case of detecting a second different global cluster that matches $loc(e)$, lines 8–11 are executed exactly one time, $loc(e)$ is stored as an error and we continue with the next $loc(e)$. Finally, Algorithm 1 returns the common and unique entities, and the `sameAs` errors.

This algorithm reads each URI u of D_{new} once (i.e., each URI of D_{new} exists in exactly one local cluster), and then it performs a binary search for u in LODsyndesis index. Therefore, its time complexity is $\mathcal{O}(|U_{D_{new}}| * log(|U_{LOD}|))$, where $U_{D_{new}}$ are all the URIs of D_{new}, U_{LOD} are all the URIs in LODsyndesis, and $log(|U_{LOD}|)$ is the cost of the binary search in LODsyndesis. On the other hand, it keeps in memory all the updated global clusters, containing the entities of D_{new}, i.e., its space complexity is $\mathcal{O}(\sum_{\forall loc(e) \in C(D_{new})} |glob_{new}(e)|)$.

How to Reduce the Number of Index Reads? A limitation is that $|U_{LOD}|$ can be huge, i.e., LODsyndesis contains more than 412 million URIs. Although we use a binary search (which is logarithmic in scale), we desire to further decrease the cost of searching to the index of LODsyndesis. For this reason, we exploit the prefixes of URIs, i.e., they usually indicate the company or university that publishes the dataset (data owner). For instance, the prefix of the URI "http://dbpedia.org/resource/Mikis_Theodorakis" is "http://dbpedia.org/". We use the prefix index of LODsyndesis [22], which contains all the prefixes for the URIs that are indexed in LODsyndesis, for checking very fast if a URI occurs in at least one existing dataset. Indeed, we know that if the prefix of a URI does not exist in the prefix index, the URI cannot be found in LODsyndesis [22].

How to Use the Prefix Index: Since the size of the prefix index is quite small compared to the index of LODsyndesis, i.e., it contains less than 1 million prefixes, before executing the line 5 of Algorithm 1, we can search if the prefix of URI u occurs in the prefix index (of LODsyndesis), and only if it is true, we can perform a binary search for u in the index of LODsyndesis. Otherwise, we just continue with the next URI. For further reducing the cost, even for searching in the prefix index, and since prefixes are highly repeated in a given dataset (e.g., in our experiments, each dataset has on average only 23.6 prefixes), we keep in memory the prefixes that we have already seen. As it is shown in §5, it can highly reduce the execution time, i.e., even more than 5× for real datasets.

Algorithm 2: Merging the triples (facts) of D_{new} with LODsyndesis

```
Input: The common entities $cmnEnt$ and their triples in $D_{new}$
Output: Common and unique triples of $D_{new}$ entities to LODsyndesis datasets
1  $uniqTriples \leftarrow \emptyset$, $cmnTriples \leftarrow \emptyset$
2  forall $e \in cmnEnt$ do                                   // Read each common entity $e$
3      forall $\langle s,p,o \rangle \in T(D_{new}, e)$ do             // Read each triple of $e$ in $D_{new}$
4          if $\langle s,p,o \rangle \in T(LOD, e)$ then         // If triple exists in LODsyndesis
5              $prov(\langle s,p,o \rangle) \leftarrow prov(\langle s,p,o \rangle) \cup \{D_{new}\}$      // Update provenance
6              $cmnTriples \leftarrow cmnTriples \cup \{\langle s,p,o \rangle\}$       // Add to $cmnTriples$
7          else                                  // Triple is offered only from $D_{new}$
8              $prov(\langle s,p,o \rangle) \leftarrow \{D_{new}\}$                 // Store its provenance
9              $uniqTriples \leftarrow uniqTriples \cup \{\langle s,p,o \rangle\}$      // Add to $uniqTriples$
10             $T(LOD, e) \leftarrow T(LOD, e) \cup \{\langle s,p,o \rangle\}$         // Add to LODsyndesis
11 return $uniqTriples, cmnTriples$
```

Step C. Merging Triples of D_{new} with LODsyndesis. Here, we merge the triples for the common entities of D_{new} to LODsyndesis, for finding common and unique facts, and possibly complementary facts from other LOD datasets (data enrichment). This is performed only for the common entities (Rule 2), since for the entities belonging only to D_{new} (Rule 1), we know that all their triples are unique. We denote all triples having an entity e either as a subject or as an object, in LODsyndesis as $T(LOD, e)$ and in D_{new} as $T(D_{new}, e)$. We desire to find for each common entity e, a) the common triples in LODsyndesis, i.e., $T(LOD, e) \cap T(D_{new}, e)$, b) the unique triples of D_{new}, i.e., $T(D_{new}, e) \setminus T(LOD, e)$, and optionally c) the complementary triples to D_{new}, i.e., $T(LOD, e) \setminus T(D_{new}, e)$.

Algorithm for Step C. Algorithm 2 shows how to compute the unique and common triples. It receives as input the common entities (*cmnEnt*), and $\forall e \in cmnEnt$, it accesses its entry in LODsyndesis index through a random access file mechanism. The pointer of the entry of each entity (in the index) has been obtained from the binary search of the previous step (i.e., line 5 of Algorithm 1). Then, for each triple in $T(D_{new}, e)$, it checks if it occurs in LODsyndesis (lines 3–10). If it is true, we update the provenance and we store the triple as common (lines 4–6), otherwise, we add the unique triple to $T(LOD, e)$ (lines 7–10). In the worst case, i.e., all the entities of D_{new} are part of LODsyndesis, we iterate and keep in memory all the entities and triples of D_{new}, i.e., time and space complexity is $\mathcal{O}(|cmnEnt| + |T(D_{new})|)$. For finding complementary facts, we can extend Algorithm 2 by also iterating over $T(LOD, e)$. However, since it can be expensive, for producing connectivity analytics we can use pre-computed posting lists containing the provenance of each triple of e [25]. In step C of Fig. 3, we updated the triples and we found common, complementary and unique facts for D_{new}, e.g., the fact "M. Theodorakis, bornYear, 1925" is verified from 3 other datasets, the fact "M. Theodorakis, educatedAt, Paris" is complement to D_{new} and the fact "M. Theodorakis, composer, Zorbas Dance" is offered only by D_{new}.

Step D. Connectivity Analytics and Enrichment Services. By using the updated (temporal) LODsyndesis indexes and specialized lattice-based algorithms (presented in [22,25]), we provide both connectivity and dataset discovery content-based measurements (e.g., see Step D of Fig. 3). The algorithms exploit the posting lists of an index (i.e., containing information about the provenance of entities, triples, etc.) for computing content-based metrics among any combination of datasets, by solving the corresponding maximization problems [22,25]. `LODChain` offers connectivity analytics and data discovery services for the input dataset, through several visualizations and HTML tables (e.g., UC2 of Fig. 2).

First, `LODChain` provides a list of possible errors in case of detecting erroneous `owl:sameAs` mappings (Rule 3 of step B2), that can be used for correcting the mappings. Second, a plenty of connectivity metrics are computed and visualized, i.e., a) the inferred equivalence mappings, b) common, unique and complementary elements, i.e., how many entities, schema elements and triples of D_{new} exist in ≥ 1 other datasets, how many only in D_{new}, and how many complementary triples exist for the entities of D_{new}, c) connections of D_{new} due to closure, i.e., the datasets having common entities with D_{new} before and after the computation of closure, d) top-10 connections of D_{new} (separately for entities, schema and triples), and many others. Regarding *Dataset Discovery*, `LODChain` finds the K most relevant datasets to D_{new} with common or complementary elements. Third, it offers an enriched version of D_{new} in RDF format having:

all the common elements and their provenance, the inferred `owl:sameAs` mappings, complementary triples, and rich metadata, created through VoID [1] and VoIDWH [21], that describe through triples the results of connectivity analytics.

Accessing LODChain and Sustainability Plan. LODChain is available in https://demos.isl.ics.forth.gr/LODChain and is running on a common machine with 8 cores, 8 GB memory and 60 GB disk space in *okeanos* service [13]. There is no need to download any software for using it, and sample datasets are offered in the webpage. Regarding its sustainability plan, LODChain can be used with any updated version of LODsyndesis' indexes, thereby, one major plan is to investigate ways for aiding the update of LODsyndesis at least periodically.

5 Experimental Evaluation

The objective is to measure the efficiency of LODChain and to provide connectivity analytics. We use a common machine with 8 cores and 8 GB memory, which contains the indexes of LODsyndesis. These indexes include over 2 billion triples and 412 million URIs from 400 RDF datasets, i.e., statistics are given in [24,25]. Concerning the datasets, we use the 5 real and 2 synthetic RDF datasets of Table 1. The real datasets include from 3K to 1.5M of triples, they contain links to few other datasets, i.e., see more statistics in Sect. 5.2, and only a few number of unique prefixes, i.e., on average 23.6 prefixes. Indeed, most of their URIs contain a prefix that cannot be found in LODsyndesis. In the worst case for real datasets, only 26.4% of URIs contain a prefix that occurs in LODsyndesis. We also use two synthetic datasets having the same number of triples, URIs and `owl:sameAs` mappings, for evaluating two cases. In the first one (HiConn), most URIs are part of LODsyndesis (highly connected), whereas in the second (LowConn), most URIs and their prefix are new (almost disconnected from LODsyndesis).

Table 1. The 7 evaluation RDF datasets (5 Real and 2 Synthetic)

ID	Dataset (abbreviation)	Domain	# of Triples	# of sameAs	# of URIs	# of URIs with prefix in LODsyndesis
R1	GReek Children Art Museum (**GRC**) [15]	User content	2,212	0	452	58 (12.8%)
R2	Geological TimeScale (**GTS**) [9]	Geography	13,271	173	1,206	181 (15.0%)
R3	World War 1 LOD (**WW1LOD**) [18]	Publication	47,616	547	11,690	2,191 (18.7%)
R4	MuziekWeb (**MW**) [17]	Media	506,582	10,563	153,538	105,548 (6.8%)
R5	Persons of National Library of Netherlands (**PNLN**) [36]	Publication	1,500,000	268,861	636,230	168,444 (26.4%)
S1	Synthetic 1 (**HiConn**)	–	1,000,000	181,105	425,500	400,697 (94.1%)
S2	Synthetic 2 (**LowConn**)	–	1,000,000	181,105	425,500	1,020 (2.3%)

Table 2. Total execution time for each dataset, with and without prefix index

Dataset	w/o Prefix index	With prefix index	Achieved speedup
GRC	7.47 s	5.26 s	1.41×
GTS	10.37 s	7.11 s	1.45×
WW1LOD	79.74 s	39.04 s	2.04×
MW	779.10 s	143.11 s	5.44×
PNLN	3,881.02 s	2,110.12 s	1.83×
LowConn	1,960.08 s	62.80 s	31.11×
HiConn	2,531.00 s	2,446.41 s	1.03×

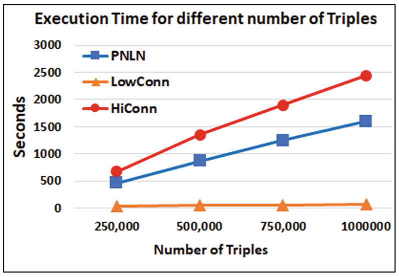

Fig. 4. Execution time with different number of triples for the same dataset

5.1 Efficiency

We measure the execution time i) for the whole process, ii) for the different steps of `LODChain`, and iii) by using a different number of triples for the same dataset.

Execution Time - The Gain of Prefix Index. Concerning the real datasets (first five rows of Table 2), as the size of the dataset grows, the execution time increases. However, by exploiting the prefix index, the execution time highly decreases for all the real datasets, i.e., we achieved a speedup from 1.41× to 5.44×. Since these datasets contain a high percentage of URIs with new prefixes (that are not part of LODsyndesis) we managed to reduce the reads to LODsyndesis indexes (and thus the execution time). By using that approach, we needed for the first four real datasets from 5 s to 2.5 min, whereas for the largest dataset, i.e. PNLN, we needed 35 min. For the synthetic datasets (last two rows of Table 2), the execution time is extremely different in case of using the prefix index, although they contain the same number of triples, URIs and `owl:sameAs` mappings. For the LowConn dataset, we achieved a 31.11× speedup by exploiting the prefix index, while for the whole process 1 min was needed, although the dataset contains 1 million triples. On the contrary, for the HiConn, which consists of the same number of triples, we needed 40 min to complete the process, even by using the prefix index, which is rational, since most URIs of HiConn contain a prefix that is common in LODsyndesis.

Execution Time for Different Numbers of Triples/URIs. Figure 4, shows the execution time for different numbers of triples (and URIs), by using the three largest datasets (including the two synthetic datasets). We can see that as the number of triples (and URIs) grows, the execution time linearly increases, which is expected, since the time complexity of the most time consuming tasks (results are presented in the next paragraph), i.e., computation of closure and merging of triples with LODsyndesis are linearithmic and linear, respectively (see Sect. 4).

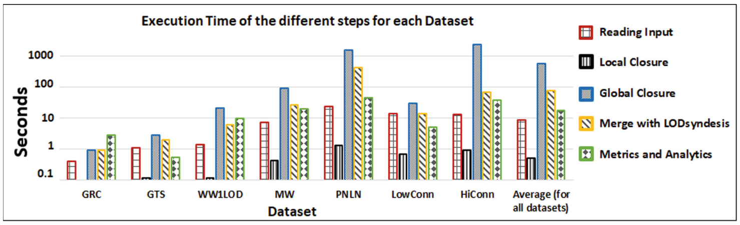

Fig. 5. Execution time for the different steps for each dataset (log scale with base 10)

Table 3. Connectivity Analytics for each real dataset to 400 other LOD datasets

ID	Measurement	GRC	GTS	WW1LOD	MW	PNLN
1	# of `owl:sameAs`	–	173	547	10,563	268,861
2	# of inferred `owl:sameAs`	–	548	2,172	26,113	309,134
3	Increase % of `owl:sameAs`	–	316%	397%	247%	114%
4	# of detected errors in `owl:sameAs`	–	0	0	1	20
5	# of unique entities	411	902	10,271	54,089	297,091
6	# of common entities	39	130	825	3,066	75,181
7	common entities percentage	8.6%	12.5%	7.4%	5.3%	20.1%
8	# of connections before `LODChain`	2	2	5	5	3
9	# of inferred connections	17	9	25	26	33
10	# of connections after `LODChain`	19	11	30	31	36
11	Increase % of connections	850%	450%	500%	520%	1100%
12	# of unique properties	19	21	65	19	4
13	# of common properties	23	58	81	8	14
14	common properties percentage	54.7%	73.4%	55.4%	29.6%	77.7%
15	# of unique classes	1	20	52	8	0
16	# of common classes	1	8	23	2	1
17	common classes percentage	50.0%	28.5%	30.6%	20.0%	100%
18	# of unique facts	2,897	16,908	53,505	611,750	1,231,972
19	# of common facts	0	25	368	7,935	193,248
20	common facts percentage	0%	0.1%	0.6%	1.2%	13.5%
21	# of new facts for D_{new} entities	26,658	17,632	362,339	597,475	3,632,412
22	% of facts enrichment for D_{new} entities	920%	104%	672%	96%	254%
23	Dataset with most common entities	Wikidata	DBpedia	Wikidata	Wikidata	VIAF
24	Dataset with most common facts	–	Opencyc	YAGO	Wikidata	VIAF
25	Dataset with most complementary facts	Wikidata	DBpedia	GeoNames	Freebase	VIAF

Execution Time of Different Steps. Figure 5 depicts in log scale (with base 10) the execution time of the different steps of `LODChain` for each single dataset and also the average time (of all the datasets) by using the approach with the prefix index. The most time-consuming task is the computation of closure among the URIs of a new dataset and LODsyndesis (i.e., global closure), e.g., for PNLN dataset, it requires the 76.4% of the time, whereas the process of merging the triples can require enough time in case of having a high number of common entities, e.g., for PNLN it requires 20.2% of the total time. On the contrary, the rest of steps are quite fast in any case, i.e., less than 50 s even in the worst case. In general, datasets with high connectivity require more time to be processed, i.e., for datasets having a lot of common entities with other datasets (such as PNLN), we need to access more times the indexes of LODsyndesis.

5.2 Connectivity Analytics Over Real Datasets

Table 3 provides connectivity analytics for the 5 real datasets. We can see (IDs 1–3) the high increase (even 397%) in the number of `owl:sameAs` mappings. On the contrary, we detected only a very few number of `sameAs` errors (ID 4), e.g., only 0.007% for PNLN dataset. Concerning the entities, each dataset shares several common entities with existing datasets, i.e., from 5.3% to 20.1% of their total entities (IDs 5–7). Due to the inference, we obtain a high increase in the number of connections after using `LODChain`, i.e., from 450% to 1100% (IDs 8–11). Indeed, the initial versions of the datasets included mappings to few LOD datasets (from 2 to 5), whereas `LODChain` discovered many inferred connections (from 9 to 33). Concerning the schema elements (IDs 12–17), most datasets use existing ontologies, and have several common properties (even 77.7%). In some cases we obtained a high number of common facts (IDs 18–20) with other datasets, i.e., 13.5% for PNLN, which can be used for data verification. Also, we found a high number of complementary facts (IDs 21–22) for the entities of each dataset, which can aid data enrichment, e.g., 254% more facts were discovered for PNLN entities. Also, popular datasets like Wikidata and DBpedia offer several common entities, common and complementary facts for the input datasets (IDs 23–25). All the datasets, and the results of even more analytics are online in [27].

6 Conclusion and Future Work

Since the current way of publishing an RDF dataset does not favor its connectivity, and thus its discoverability, reusability and content enrichment, we proposed a novel service, called `LODChain`, for strengthening the connections of any RDF dataset (even before its actual publishing) to the rest of LOD cloud. For showcasing its potential impact, we described use cases involving different categories of users, and we detailed its process, which includes methods for computing the equivalence reasoning among the input dataset and hundreds of LOD datasets. For evaluating its impact and efficiency we used 5 real and 2 synthetic datasets; `LODChain` produced connectivity analytics for datasets with thousands of triples even in less than 1 min, whereas the connections of real datasets increased from 450% to 1100%. As a future research, work and long-term plan, we want to a) investigate ways to parallelize `LODChain` (by extending the techniques of [23]), b) improve the GUI and perform a usability evaluation with dataset owners, c) support entity matching for finding connections for totally disconnected RDF datasets, and d) create an evaluation benchmark for such connection services.

Resource Availability Statement: The source code and URL of LODChain, the datasets and the experimental results are available in [27]. A tutorial video presenting LODChain is available in https://youtu.be/Kh9751p32tM.

Acknowledgments. This work has received funding from the European Union's Horizon 2020 coordination and support action 4CH (Grant agreement No 101004468).

References

1. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets with the VoID vocabulary (2011)
2. Asprino, L., Beek, W., Ciancarini, P., Harmelen, F.V., Presutti, V.: Observing LOD using equivalent set graphs: it is mostly flat and sparsely linked. In: International Semantic Web Conference, pp. 57–74. Springer (2019). https://doi.org/10.1007/978-3-030-30793-6_4
3. Beek, W., Raad, J., Acar, E., van Harmelen, F.: MetaLink: a travel guide to the LOD cloud. In: Harth, A., et al. (eds.) ESWC 2020. LNCS, vol. 12123, pp. 481–496. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49461-2_28
4. Bischof, S., Harth, A., Kämpgen, B., Polleres, A., Schneider, P.: Enriching integrated statistical open city data by combining equational knowledge and missing value imputation. J. Web Semant. **48**, 22–47 (2018)
5. Bizer, C., Heath, T., Berners-Lee, T.: Linked data: the story so far. In: Semantic Services, Interoperability and Web Applications: Emerging Concepts, pp. 205–227. IGI global (2011)
6. Brickley, D., Burgess, M., Noy, N.: Google dataset search: building a search engine for datasets in an open web ecosystem. In: The World Wide Web Conference, pp. 1365–1375 (2019)
7. Chapman, A., et al.: Dataset search: a survey. VLDB J. **29**(1), 251–272 (2019). https://doi.org/10.1007/s00778-019-00564-x
8. Christophides, V., Efthymiou, V., Palpanas, T., Papadakis, G., Stefanidis, K.: An overview of end-to-end entity resolution for big data. ACM Comput. Surv. (CSUR) **53**(6), 1–42 (2020)
9. Cox, S.J.D., Richard, S.M.: A geologic timescale ontology and service. Earth Sci. Inf. **8**(1), 5–19 (2014). https://doi.org/10.1007/s12145-014-0170-6
10. Debattista, J., Attard, J., Brennan, R., O'Sullivan, D.: Is the LOD cloud at risk of becoming a museum for datasets? Looking ahead towards a fully collaborative and sustainable LOD cloud. In: Proceedings of WWW Conference, pp. 850–858 (2019)
11. Fernández, J.D., Beek, W., Martínez-Prieto, M.A., Arias, M.: LOD-a-lot. In: International Semantic Web Conference, pp. 75–83. Springer (2017). https://doi.org/10.1007/978-3-319-68204-4_7
12. Gottron, T., Scherp, A., Krayer, B., Peters, A.: LODatio: a schema-based retrieval system for linked open data at web-scale. In: Extended Semantic Web Conference, pp. 142–146. Springer (2013). https://doi.org/10.1007/978-3-642-41242-4_13
13. GRNET: Okeanos cloud computing service. https://okeanos.grnet.gr. Accessed 25 July 2022
14. Hubauer, T., Lamparter, S., Haase, P., Herzig, D.M.: Use cases of the industrial knowledge graph at siemens. In: International Semantic Web Conference (P&D/Industry/BlueSky) (2018)

15. Kotis, K., Angelis, S., Chondrogianni, M., Marini, E.: Children's art museum collections as linked open data. Int. J. Metadata Semant. Ontol. **15**(1), 60–70 (2021)
16. Lehmann, J., et al.: Dpedia-a large-scale, multilingual knowledge base extracted from Wikipedia. Semant. Web **6**(2), 167–195 (2015)
17. Weigl, D.M., et al.: Interweaving and enriching digital music collections for scholarship, performance, and enjoyment. In: 6th International Conference on Digital Libraries for Musicology, pp. 84–88 (2019)
18. Mäkelä, E., Törnroos, J., Lindquist, T., Hyvönen, E.: WW1LOD: an application of CIDOC-CRM to world war 1 linked data. IJDL **18**(4), 333–343 (2017)
19. McCrae, J.P., et al.: The linked open data cloud. Lod-cloud. net (2019)
20. Mountantonakis, M.: Services for Connecting and Integrating Big Numbers of Linked Datasets, vol. 50. IOS Press (2021)
21. Mountantonakis, M., et al.: Extending VoID for expressing connectivity metrics of a semantic warehouse. In: PROFILES@ ESWC (2014)
22. Mountantonakis, M., Tzitzikas, Y.: On measuring the lattice of commonalities among several linked datasets. Proc. VLDB **9**(12), 1101–1112 (2016)
23. Mountantonakis, M., Tzitzikas, Y.: Scalable methods for measuring the connectivity and quality of large numbers of linked datasets. J. Data Inf. Qual. (JDIQ) **9**(3), 1–49 (2018)
24. Mountantonakis, M., Tzitzikas, Y.: Large-scale semantic integration of linked data: a survey. CSUR **52**(5), 1–40 (2019)
25. Mountantonakis, M., Tzitzikas, Y.: Content-based union and complement metrics for dataset search over RDF knowledge graphs. ACM JDIQ **12**(2), 1–31 (2020)
26. Mountantonakis, M., Tzitzikas, Y.: How your cultural dataset is connected to the rest linked open data. In: Proceedings of the TMM-CH2021, Communications in Computer and Information Science, Athens, Greece, pp. 12–15 (2021)
27. Mountantonakis, M., Tzitzikas, Y.: LODChain, April 2022. https://doi.org/10.5281/zenodo.6467419
28. Nayak, A., Božić, B., Longo, L.: Linked data quality assessment: a survey. In: International Conference on Web Services, pp. 63–76. Springer (2021). https://doi.org/10.1007/978-3-030-96140-4_5
29. Nečaskỳ, M., Škoda, P., Bernhauer, D., Klímek, J., Skopal, T.: Modular framework for similarity-based dataset discovery using external knowledge. Data Technol. Appl. **56**(4), 506–535 (2022)
30. Otero-Cerdeira, L., et al.: Ontology matching: a literature review. Expert Syst. Appl. **42**(2), 949–971 (2015)
31. Paris, P.-H.: Assessing the quality of owl:sameAs links. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 11155, pp. 304–313. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98192-5_49
32. Pietriga, E., et al.: Browsing linked data catalogs with LODAtlas. In: International Semantic Web Conference, pp. 137–153. Springer (2018). https://doi.org/10.1007/978-3-030-00668-6_9
33. Rebele, T., Suchanek, F., Hoffart, J., Biega, J., Kuzey, E., Weikum, G.: YAGO: a multilingual knowledge base from Wikipedia, Wordnet, and Geonames. In: Groth, P., et al. (eds.) ISWC 2016. LNCS, vol. 9982, pp. 177–185. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46547-0_19
34. Rietveld, L., Beek, W., Schlobach, S.: LOD Lab: experiments at LOD scale. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9367, pp. 339–355. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25010-6_23

35. Sabou, M., Onder, I., Brasoveanu, A.M.P., Scharl, A.: Towards cross-domain data analytics in tourism: a linked data based approach. Inf. Technol. Tour. **16**(1), 71–101 (2016). https://doi.org/10.1007/s40558-015-0049-5
36. Sierman, B., Teszelszky, K.: How can we improve our web collection? An evaluation of webarchiving at the KB national library of the Netherlands (2007–2017). Alexandria **27**(2), 94–107 (2017)
37. Tzitzikas, Y., et al.: Methods and tools for supporting the integration of stocks and fisheries. In: International Conference on Information and Communication Technologies in Agriculture, Food & Environment, pp. 20–34. Springer (2017). https://doi.org/10.1007/978-3-030-12998-9_2
38. Umbrich, J., Hogan, A., Polleres, A., Decker, S.: Link traversal querying for a diverse web of data. Semant. Web **6**(6), 585–624 (2015)
39. Valdestilhas, A., Soru, T., Nentwig, M., Marx, E., Saleem, M., Ngomo, A.-C.N.: Where is My URI? In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 671–681. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_43
40. Valdestilhas, A., Soru, T., Ngomo, A.C.N.: CEDAL: time-efficient detection of erroneous links in large-scale link repositories. In: Proceedings of the International Conference on Web Intelligence, pp. 106–113 (2017)
41. Vandenbussche, P.Y., Umbrich, J., Matteis, L., Hogan, A., Buil-Aranda, C.: SPARQLES: monitoring public SPARQL endpoints. Semant. Web **8**(6), 1049–1065 (2017)
42. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Silk-a link discovery framework for the web of data. In: LDOW (2009)
43. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledge base. Commun. ACM **57**(10), 78–85 (2014)
44. Wang, X., Cheng, G., Pan, J.Z., Kharlamov, E., Qu, Y.: BANDAR: benchmarking snippet generation algorithms for (RDF) dataset search. IEEE Trans. Knowl. Data Eng. (2021). https://ieeexplore.ieee.org/document/9477056
45. Wiśniewski, D., Potoniec, J., Ławrynowicz, A., Keet, C.M.: Analysis of ontology competency questions and their formalizations in SPARQL-OWL. J. Web Semant. **59**, 100534 (2019)
46. Yochum, P., Chang, L., Gu, T., Zhu, M.: Linked open data in location-based recommendation system on tourism domain: a survey. IEEE Access **8**, 16409–16439 (2020)
47. Yumusak, S., Dogdu, E., Kodaz, H., Kamilaris, A., Vandenbussche, P.Y.: SpEnD: linked data SPARQL endpoints discovery using search engines. IEICE Trans. Inf. Syst. **100**(4), 758–767 (2017)
48. Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., Auer, S.: Quality assessment for linked data: a survey. Semant. Web **7**(1), 63–93 (2016)