IDOL: Comprehensive & Complete LOD Insights

Ciro Baron Neto
Universität Leipzig, Institut für Informatik, AKSW, http://aksw.org
cbaron@informatik.uni-leipzig.de

Dimitris Kontokostas
Universität Leipzig, Institut für Informatik, AKSW, http://aksw.org
kontokostas@informatik.uni-leipzig.de

Amit Kirschenbaum
Universität Leipzig, Institut für Informatik, AKSW, http://aksw.org
amit@informatik.uni-leipzig.de

Gustavo Correa Publio
Universität Leipzig, Institut für Informatik, AKSW, http://aksw.org
gustavo.publio@informatik.uni-leipzig.de

Diego Esteves
Universität Bonn, Smart Data Analytics Research Group, SDA, http://sda.tech
esteves@cs.uni-bonn.de

Sebastian Hellmann
Universität Leipzig, Institut für Informatik, AKSW, http://aksw.org
hellmann@informatik.uni-leipzig.de

ABSTRACT

Over the last decade, we observed a steadily increasing amount of RDF datasets made available on the web of data. The decentralized nature of the web, however, makes it hard to identify all these datasets. Even more so, when downloadable data distributions are discovered, only insufficient metadata is available to describe the datasets properly, thus posing barriers on its usefulness and reuse. In this paper, we describe an attempt to exhaustively identify the whole linked open data cloud by harvesting metadata from multiple sources, providing insights about duplicated data and the general quality of the available metadata. This was only possible by using a probabilistic data structure called Bloom filter. Finally, we published a dump file containing metadata which can further be used to enrich existent datasets.

KEYWORDS

RDF, Bloom Filter, Linked Open Data, Dataset Overlap

ACM Reference format:
Ciro Baron Neto, Dimitris Kontokostas, Amit Kirschenbaum, Gustavo Correa Publio, Diego Esteves, and Sebastian Hellmann. 2017. IDOL: Comprehensive & Complete LOD Insights. In *Proceedings of Semantics2017, Amsterdam, Netherlands, September 11–14, 2017,* 8 pages.
DOI: 10.1145/3132218.3132238

1 INTRODUCTION

Linked Open Data (LOD) comprises an unprecedented volume of structured data on the Web. The LOD cloud grew up from a handful of datasets in 2007 to thousands in recent years. There have been efforts to create diagrams of the LOD cloud and depict the connections among different datasets. The latest effort was in 2014 [17]. However, the data sources are continuously increasing in number, size, and means of publication, thus, identifying a dataset has become a very challenging task.

Semantics2017, Amsterdam, Netherlands

DOI: 10.1145/3132218.3132238

So far, the main source of dataset metadata were metadata registries like the Comprehensive Knowledge Archive (CKAN) engine[1], especially the http://datahub.io CKAN instance given as recommendation for LOD. For the creation of the first version of the LOD cloud, dataset maintainers were requested to create an entry for their dataset landing page and define a set of required and optional dataset metadata. The effort was driven by the LATC[2] and Planetdata[3] EU projects, creating and cleaning an initial metadata collection. However, metadata is not systematically maintained in a sustanaible manner and models such as the one used by `datahub.io` lack granularity. For example, the *DBpedia* entry still marks the 3.7, which dates back to 2010, as the latest release[4], and lists one large tar file (no further metadata is available, and not all *DBpedia* release files are in the archive). The problem got amplified when other metadata registries were created that describe both new and overlapping datasets, making the exact mapping of downloadable files and versions to its dataset grouping complex. On top of that, there are now meta-registries (registries of registries) that further increase the difficulty of accessing dataset metadata without any homogenization.

In recent years, the semantic web technologies and tools have matured and new standardized vocabularies like DCAT [10], VoiD [1], and DataID [7] have vastly improved to capture the metadata of a dataset. We argue, however, that the approaches described above have not proven to be effective, we need to define a new paradigm on how to identify and describe datasets on the data web and how to make this information discoverable. Human intervention is obviously a required step but should be minimized to the extent possible.

In this paper, we employ automatic methods to gather a comprehensive and up-to-date dataset list of the whole LOD cloud. We achieve this by harvesting all existing metadata registries, as well as registries of metadata registries. We then stream, explore and analyze all dataset RDF contents. This paper has two main goals: First, to assess RDF metadata in terms of the content quality, and, secondly, to explore the metadata registries providing insights about how much duplicated content exists across datasets. Detecting duplicate data is crucial as it allows us to obtain information on how

[1] See http://ckan.org
[2] Website defunct, cf. instead http://aksw.org/Projects/LATC
[3] See http://www.planet-data.eu/
[4] https://datahub.io/dataset/dbpedia

much the current representations of the LOD-cloud are inflated by redundant data. Our coverage reaches dozens of billions of triples, and in order to not exceed main memory and space limitations, we focused on using probabilistic space-efficient data structures. Therefore, we use Bloom filters to compute subset overlaps across different datasets. Finally, we enrich the harvested metadata with our analysis and re-publish them through a dump file endpoint with standard vocabularies.

The rest of the paper is structured as follows: In Section 2 we explain the necessary background concepts. In Section 3 we describe all the data sources used in this paper . Section 4 provides details on the proposed implementation. We discuss our results in Section 5 and conclude in Section 6.

2 BACKGROUND

2.1 Bloom Filters

Bloom filters (BF) [3] play a major role in our approach, since they are used for subset detection. A Bloom filter is a compact, probabilistic data structure designed to check the membership of an element x in a set S, i.e. the lookup operation. BF is a type of approximate member query (AMQ) filter since this data structure has 100% recall rate (*false negative* (fn) matches are impossible), while a small percentage of *false positives* (fp) is condoned and the fp margin of error can be adjusted in advance. The *false positive probability* (fpp) is calculated according to the size of the distributions. Equation 1 defines the fpp value used in our experiments. A fpp of $0.9/distributionSize$ guarantees an expected value (EV) of finding 0.9 links per distribution that are not links (false positives).

$$fpp = \begin{cases} 0.9/distributionSize, & \text{if } size > 100000, \\ 0.0000001, & \text{otherwise.} \end{cases} \quad (1)$$

In order to have a fixed fp rate, the length of the structure must grow linearly with the number of elements. The total number of bits m for the desired number of elements n and fp rate p, is defined as:

$$m = -\frac{n \ln p}{(\ln 2)^2} \quad (2)$$

An optimal number of hash functions is given by:

$$k = (m/n) \ln 2. \quad (3)$$

These methods are based on the optimal number of hash functions, since reducing the number of hashes would significantly decrease the BF precision. Space and time advantages of this probabilistic data structure are more coherent in this scenario compared to more commonly-used data structures, such as binary search trees, hash tables, arrays, or linked lists. In fact, the reason why BF was preferred over other data structures is that, as shown in Table 1, it provides a constant run-time for *Lookup* and *Insertion* operations (regardless of the number of elements, depending only on the number of hashes), as well as an efficient *space complexity* with a low memory footprint. In [5], the quality assessment of linked data detecting duplicated instances was done using BF Randomised Load Balanced Biased Sampling based Bloom Filter (RLBSBF). Loizos et. al. [13] state that *union* and *intersection* operations are also applicable to Bloom filters. The basic idea of an *intersection* operation is to perform a bitwise `AND` operation between the bits of the filters. After the operation, the likelihood of all bits be set to true can be estimated by:

$$\left(\left(1 - \left(1 - \frac{1}{m}\right)^{kn_1}\right) * \left(1 - \left(1 - \frac{1}{m}\right)^{kn_2}\right)\right)^k \quad (4)$$

In the best case scenario, however, the false positive probability of an intersection will be 0.953156 (using the optimal number of hashes and bits). A better precision can be achieved by oversized filters, i.e. adding fewer elements than n. Clearly, this leads to a major drawback w.r.t. the memory footprint. A second concern about the intersection operation is that the filters should have equivalent size rendering filters oversized for datasets with less than n elements, and for larger datasets, multiple filters would be used. The problem with this approach is that the false positive value propagates proportionally with the number of intersections needed to compare large datasets (e.g., for comparing two datasets with three filters each, nine intersection operations are required). Table 1 provides a comparison of bloom filters to other related data structures.

2.2 Dataset, Subset and Distribution Definition

We define the terms *dataset*, *subset* and *distribution* with the DCAT [10] and VoID [1] vocabularies to clarify the variables used in this paper.

- ID: a dataset, described by `void:Dataset` or `dcat:Dataset`;
- S_{ID}: the set of subsets, described by `void:subset` of a given dataset ID
- $\langle s, p, o \rangle$: the `RDF` triple which represents the subject s, predicate p and object o for a given relation.
- d_n: the n-th distribution consisting of a set of `RDF` triples.
- D_{ID}: the set of distributions, described by `dcat:distributions` of the dataset ID .
- $S_{d_S \to d_T}$: the subset of existing triples common to two distributions, having d_S as a source distribution and d_T as a target distribution. We define that a subset occurs from a distribution d_S to a distribution d_T whenever d_S contains one or more $t_S = \langle s_S, p_S, o_S \rangle$ and d_T contains $t_T = \langle s_T, p_T, o_T \rangle$ such that $t_S = t_T$.

2.3 Survey of Metadata

Vocabularies like DCAT [10], VoID [1] and DataID [7] can be used to define *dataset* metadata, and provide information such as *subsets*, *distributions*, license, dataset title, etc. A *subset* is a distinct part of a *dataset* that can be differentiated for a number of reasons, such as differences in provenance, publication dates, accessibility or language[5]. *Distributions* describe the specific files or resources by which the datasets might be accessed or acquired[6]. These resources can be dump files, `SPARQL CONSTRUCT` queries or a `SPARQL` endpoint. A thorough metadata example comes from *DBpedia*, where DataID[7] is used to describe multiple datasets in multiple formats and multiple languages.

Online repositories such as CKAN often provide metadata regarding dataset description, format, indegree/outdegree, creator,

[5] See http://www.w3.org/TR/void/#subset

[6] See http://www.w3.org/TR/vocab-dcat/#class-distribution

[7] See http://downloads.dbpedia.org/2016-04/2016-04_dataid_catalog.ttl and http://wiki.dbpedia.org/projects/dbpedia-dataid

Table 1: Average run-time, space complexity and feature comparison between different data structures, where: y = maximum length of the string, k = number of hash functions, m = number of bits, n = number of elements in the set and fpp = false positive probability.

Algorithm	Lookup	Insert	Space Complexity	Intersection	Precision
Radix tree	$O(y)$	$O(y)$	$O(n)$	$O((n1)(n2)))$	*determ.*
AVL tree	$O(\log n)$	$O(\log n)$	$O(n)$	$O((n1)(n2)))$	*determ.*
Hash table	$O(1)$	$O(1)$	$O(n)$	$O(n)$	*determ.*
Quotient filter	$O(\log m)$	$O(n)$	$\approx O(n \log_2(1/fpp)) * 1.2$	X	*probab.*
Bloom filter	$O(k)$	$O(k)$	$O(n \log_2(1/fpp))$	$O(m)$	*probab.*

maintainer, etc. Nevertheless, the metadata is in many cases insufficient or unreliable, leading to several drawbacks. Since certain types of data are manually inserted by the dataset maintainer, there is no guarantee that the provided data is accurate, complete or up to date. With that said, we can classify the metadata definitions into three distinct categories: manual metadata, heuristic metadata, and analytical metadata.

2.3.1 Manual Metadata. Certain types of metadata cannot be automatically generated and, thus, depend on human interactions. These metadata are often required for the dataset maintainability. For example, if we analyze the Dublin Core Metadata Specification[8], we can find solid examples of properties that have a high impact on the dataset maintenance and can only be created manually. Properties like `dc:license`, `dc:creator` and `dc:contributor` are important for the situations where a user needs information which cannot be derived from the metadata description, such as requests for dataset error correction or content integration/improvement.

2.3.2 Heuristic Metadata. Important metadata can also be extracted through the analysis of the dataset content. This process is done by machine learning or rule-based approaches, and most of the time extracting data, using supervised methods. Whilst machine learning has an important role in the Linked Data domain, some drawbacks should be considered, such as its limited precision and required resources. Therefore, using supervised methods requires a considerable amount of datasets to be used as training sets in order to achieve sufficient precision and recall. In [12], the authors use dataset features (e.g., Class Names, Property `URI`s, text from `rdfs:label`, etc.) to discover dataset topics or categories. The achieved accuracy is up to 81%. Likewise, in [16] the authors aim to extract topics not from datasets, but rather from ontologies and vocabularies.

2.3.3 Analytic Metadata. The third type of metadata is the one which can be automatically generated with 100% of accuracy since it is deterministic. Usually, this kind of metadata is technical and related to the dataset structure and can be easily measured. Examples of such metadata, which can be generated on-the-fly, are links between datasets, number of triples and file statistics (e.g., file size and format).

[8]See http://dublincore.org/

3 DATA SOURCES AND REPOSITORIES

We have retrieved `RDF` data from 8 data sources. For each of them, we fetched all datasets (regardless of their format) and identified the ones containing `RDF` data. Most files were compressed resources (e.g., `zip` files) where multiple files are bundled together. In such cases, we extracted and analyzed all files. We noticed that low-quality metadata is a common issue in all repositories, as noted already by [4]. Although the metadata was manually inserted by the dataset creator or maintainer, it is not guaranteed that the provided data is accurate. For instance, multiple unnecessary variations of `MIME-types` were found (e.g., `ttl`, `turtle`, `rdf/turtle`, etc.). To correctly identify the RDF serialization format, we used regular expressions in order to normalize these fields and then parsed the file contents with different `RDF` parsers.

3.1 Data sources

DBpedia Datasets. The first data source we crawled was *DBpedia*[9]. The *DBpedia* community maintains the DataID project, which aims at describing datasets in a uniform way with a considerable amount of details. Hence, we crawled the DataID description file[9] to gain easy access to all `RDF` files to be streamed as well as additional metadata, e.g., subsets, language, etc. The dataset consists of 476 subsets (separated out by languages and categories) and 7,229 distributions.

The **LOD laundromat** [2] initiative aims at republishing other people's "*dirty*" `RDF` data, improving data quality and making it available for reuse. Moreover, the process involves the detection of the serialization format, filtering duplicated triples (within the same distribution), syntax error detection, and others. The *clean* data is finally republished in a uniform serialization format, and either a `SPARQL` endpoint or dump files are provided. Additionally, the data can be accessed as HDT or using Linked Data Fragments [19]. The pipeline for acquiring data from *LOD laundromat* consists of the following steps: 1) reading and parsing the metadata file[10]; 2) for each dataset found, stream the clean data; 3) based on the original dataset download `URI`, we fetch metadata in any of the CKAN repositories analyzed. The last phase is necessary since *LOD laundromat* only provides analytical metadata (such as dataset size, the number of triples, etc.) and it does not keep metadata which

[9]See http://downloads.dbpedia.org/2016-04/2016-04_dataid_catalog.ttl
[10]See http://download.lodlaundromat.org/dump.nt.gz

cannot be automatically generated (Section 2.3.1). It is important to emphasize that this repository has a dual purpose, i.e., it serves both as a metadata repository and as a dataset endpoint repository. More specifically, based on the metadata provided by the repository it is possible to access the dataset in its original dataset location or access a local copy. We detected 1,489 datasets, 1,259 subsets and 34,969 distributions.

The **Registry of Research Data Repositories** (re3data.org) is a repository of repositories [15]. It contains over 400 repositories which are based on different software, from MySQL databases to CKAN. For the scope of this work, we considered only repositories powered by CKAN for two main reasons: firstly, the most well-known RDF repositories (i.e., *publicdata.eu* and *datahub.io*) are based on this technology; secondly, CKAN provides an API which allows a client to easily query for datasets and resources, providing a centralized way to insert and access data. We fetched a total of 20 CKAN repositories, including *datahub.io*, *publicdata.eu* and *healthdata.gov*. We detected 2,098 datasets, 973 subsets and 11,930 distributions.

Linked Open Vocabularies (LOV) Repository [18] provides a dataset which contains over 500 vocabularies and ontologies. Moreover, it is popular within the community, and a substantial amount of vocabularies has been added since 2011. We access *LOV* via a dump file[11] which uses N-Quad to describe all vocabularies. Similarly to *LOD Laundromat*, this repository also has a dual purpose, i.e., serving as a metadata repository and as a dataset endpoint repository. We detected 579 datasets (ontologies and vocabularies), 6 subsets and 579 distribution (one for each dataset).

The **LOD Cloud Diagram** [17] created a snapshot[12] of the current LOD Cloud alongside an image depicting the interconnections among different datasets. They define datasets according to the top-level domain. The approach is not executed frequently as manual work is involved in the analysis. They provide a zipped archive [13] containing DCAT metadata of the crawled datasets. It was possible to find 1,303 datasets distributed in 2,830 subsets and 2,262 distributions. Notice that the number of subsets is bigger than the distributions, which means that some subsets are pointing to the same distributions files.

Linghub [11] is a portal that aggregates and indexes linguistic datasets and exposes metadata under a common interface. *Linghub* is widely used by the NLP community, since the list of corpora is retrieved from repositories such META-SHARE[14], Clarin VLO[15], and LRE-Map[16]. Furthermore, *Linghub* provides both a SPARQL endpoint and a dump file containing metadata of the resources. The metadata is described using DCAT and VoID vocabularies, thus, it is possible to access the RDF resources reaching properties like `dcat:accessURL`. *Linghub* has 175 datasets, 607 subsets and 640 distributions.

CKAN.org provides a list of instances[17] available on the web. Although there is no easy way to query all instances, we created an HTML parser which fetches the repositories' URLs. We were able to load 147 instances and we make the list available at our GitHub webpage. As Re3Data, repositories like *datahub.io* and *publicdata.eu* were available here. The list of catalogs can be found on our GitHub webpage[18]. We could fetch 147 repositories, 7,043 datasets, 5,438 subsets and 17,993 distributions.

Sparqles.okfn.org SPARQL Endpoint Status *SPARQLES*[19] monitors SPARQL endpoints collected from *datahub.io* providing status of availability, performance and interoperability. *SPARQLES* provides an API[20] in which users can access more than 500 SPARQL endpoints. From *SPARQLES*, we could fetch 549 datasets and 6,143 distributions. The process of distribution detection for SPARQL endpoints is described in section 4.

3.2 Related Work and Other Data Sources

We list other data sources here, which we considered for analysis, but were eventually not included, since most of them are redundant as they are built upon the above mentioned data sources or are not relevant.

LODstats retrieves data from CKAN repositories, and at the time of writing, *LODstats* reports 9,960 datasets. *LODSats* provides statistical data about the overall number of datasets and vocabulary utilization. According to their report around 79% of the datasets have errors or are not accessible[21], therefore, a SPARQL endpoint[22] containing the dataset descriptions is available. *LODSats* was not included in our experiments, since it explores data from CKAN datasets which is already analyzed in this work.

Swoogle [23] is a crawler-based indexing and retrieval system which provides a search engine over RDF documents by means of an inverted keyword index and a relational database. Since this work was first published in 2004, it can be considered one of the first attempts to building a full search interface of documents published in the Semantic Web. This approach was not included since, at the moment of this writing, we could not find a working SPARQL endpoint or RDF dump files or any other published data.

UniProt [24]is an example of a domain-specific dataset. The SPARQL endpoint holds data of automatically annotated uncharacterized protein sequences and has a remarkable size containing over 20 billion triples. The data is available as SPARQL endpoint and Dump files. Considering that this repository is specific for a unique domain, we are not analyzing it, although this dataset might be included in a future version of this work.

SWSE The Semantic Web Search Engine (SWSE) [8] is a complete platform that provides several tools for entity search over instance data, like crawlers, ranking algorithms, reasoning, etc. When combined, the whole architecture provides a high-performance and scalable search system. Again, we could not evaluate the approach since there was no accessible SPARQL endpoint or RDF dump file.

[11] See http://lov.okfn.org/lov.nq.gz
[12] See http://lod-cloud.net/
[13] See http://data.dws.informatik.uni-mannheim.de/lodcloud/2014/ISWC-RDB/
[14] See http://www.meta-net.eu/meta-share
[15] See https://www.clarin.eu/content/virtual-language-observatory
[16] See http://www.resourcebook.eu/
[17] See http://ckan.org/instances/#
[18] See https://github.com/AKSW/IDOL/tree/master/json_resources
[19] See http://sparqles.ai.wu.ac.at/
[20] See http://sparqles.ai.wu.ac.at/api
[21] in comparison, we were actually able to access around 81% of all distributions
[22] See http://stats.lod2.eu/sparql
[23] See http://swoogle.umbc.edu/
[24] See http://www.uniprot.org/format/uniprot_rdf

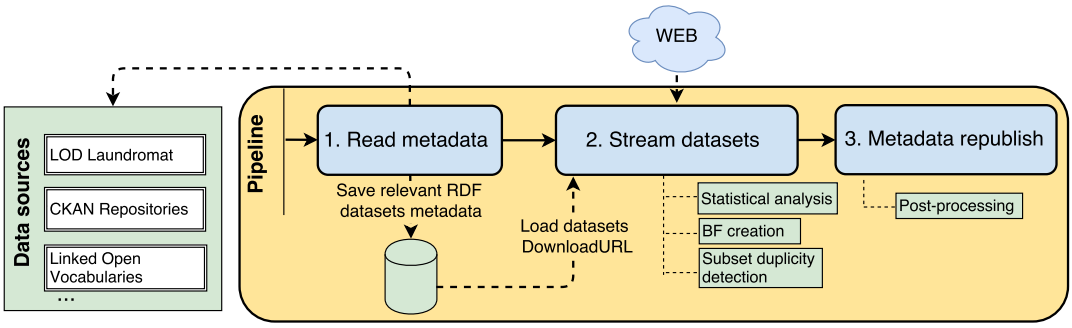

Figure 1: A pipeline showing an overview of IDOL architecture.

4 IMPLEMENTATION

Figure 1 depicts a high-level overview of the data flow in our architecture.

4.1 Metadata extraction

We start by retrieving metadata from all data sources described in Section 3.1. There are different methods to access the metadata, e.g., REST services, SPARQL endpoint, VoID files, depending on the various data sources. Parsers were customized accordingly to retrieve metadata records. We utilized repositories powered by CKAN (e.g., datahub.io), which partially implement the DCAT data model, thus facilitating the parsing process. In order to increase our coverage, we filtered properties which may indicate that the record is a URL to access a dataset. When the repository retrieves RDF description files, we explore properties such as `dcat:downloadURL`, `dcat:accessURL`, `void:dataDump`. In some cases, repositories provide REST APIs for retrieving objects whose elements are key-value pairs, for instance, JSON objects. In such cases, a set of regular expressions is used in order to identify URLs of datasets or distributions.

SPARQL endpoints are also considered, and are analyzed in two steps. First, a query is used to retrieve all named graphs available in the triple store, and second, for every graph (which is now considered as a distribution) we run paginated queries as a CONSTRUCT graph in order to further retrieve all triples. The pagination size was different for each endpoint: We started with page size of 1,000 and gradually increased it, depending on the endpoint response time.

4.2 Streaming datasets

Once metadata is acquired, all the datasets or distributions identified are streamed and processed. Distributions with the same URL, as well as distributions with different serialization formats for the same URL are not streamed twice. For data sources that have dual purposes (e.g., *LOD laundromat* and *Linked Open Vocabularies*) we stream datasets from the repositories rather than from the original locations. There are two main reasons for that: first, the repositories usually have high availability, which ensures that all datasets should be accessible, and second, in many cases (as in *LOD Laundromat*, for instance) the data provided has a better quality since it has already been pre-processed. As data start to come in from a *source distribution* $d_{\mathcal{S}}$, each triple $t_i = \langle s_i, p_i, o_i \rangle$ extracted is processed by a three-stage pipeline:

1. Statistical Analysis. The first stage retrieves and organizes the basic data. For instance, we break down the triples and extract the fully qualified domain names (FQDN) of the subjects and objects (and keep counters in case of redundancies), we count blank nodes, literals, distinct subjects and predicates, and other resources.

2. Bloom filter creation. A Bloom filter is created on the fly ($BF_{d_{\mathcal{S}},T}$) which stores all *triples* T of the current *source distribution*. It is important to emphasize that Bloom filters hash the triples, thus, regardless of the triple size (e.g., triples containing long strings as literals) the hash representing the triple will have constant size. Furthermore, these filters will later represent the current distribution as a *target distribution* for content duplication detection.

3. Duplicate content detection In order to find which *target distribution* ($d_{\mathcal{T}}$) might contain the same triple t_i we load into the main memory all filters of target distributions $BF_{d_{\mathcal{T}},T}$ where $\exists t \in d_{\mathcal{T}}.(FQDN(t_i) = FQDN(t))$ where t is a triple in d_t. The reason that we check FQDN beforehand, is to avoid loading filters that describe $d_{\mathcal{T}}$ that do not even have common namespaces with $d_{\mathcal{S}}$. Detecting duplicate content is performed by looking up the current triple (t_i) against all the $BF_{d_{\mathcal{T}},T}$ distribution filters.

Another solution for detecting content duplication would be to get the value of $BF_{d_{\mathcal{S}},T} \cap BF_{d_{\mathcal{T}},T}$. In fact, that was our first try. However, as stated in section 2.1, the fpp increases proportionally with the number of intersection operation, making the direct intersection operation ineffective.

4.3 Metadata Regeneration

Finally, we share the extracted metadata by republishing it to a SPARQL endpoint. The new metadata generation complies with the DataID[25] ontology. We chose this format because its coverage allows to describe multiple levels of relations among distributions, datasets and subsets. DataID reuses vocabularies such as VoID, Prov-O[26] and SKOS. Currently, we republish dataset properties like `dcat:distribution`, `dcat:dataset`, `void:subset`, `void:linkset`, and others. We provide a dump file[27] with the available metadata.

A final consideration about the implementation is that depending on the number of datasets streamed, the growing amount of

[25] See http://wiki.dbpedia.org/projects/dbpedia-dataid
[26] See http://www.w3.org/TR/prov-o/
[27] See https://github.com/AKSW/IDOL/blob/master/dump.nt

filters might become an issue. Therefore, in order to avoid unnecessary overheads, we can considerably increase the throughput by performing lookup operations in multiple hosts. Furthermore, the data is stored using MongoDB which has native ability to scale horizontally. The only bottleneck we encountered was the poor endpoint performances.

All experiments were performed using Intel(R) Xeon(R) CPU X5650, 64GB of memory and 25TB of SSD in RAID 5. IDOL was written in Java using MongoDB v.3.4 as the database. Bloom filters were stored using GridFS[28] over MongoDB. It is important to stress that, although our model reads and retrieves RDF data, it does not store any RDF. Our implementation creates RDF on the fly reading documents from MongoDB and using Apache Jena to create RDF models. Hence, no triplestore is required to be installed. Two Bloom filters implementations were used[29,30]. The IDOL project is open-source, and the source code, documentation, as well as other resources can be found on our GitHub web page[31]. Furthermore, we exported the configuration of the experiments based on the MEX vocabulary [6] and stored these in the WASOTA repository [14], following best practices in terms of reproducibility of experiments.

5 RESULTS AND DISCUSSION

Table 3 shows the general size of the retrieved dataset distributions. As can be seen, 91% of the distributions have less than 1 million triples, 7% have between 1 and 100 million, and only 2% are considered large, containing more than 100 million triples. With that said, we describe now how much duplicated data was found, and the general quality of the metadata.

5.1 Overview of repository data

Table 2 provides an overview of the processed repositories, datasets and distributions. For each repository described in section 3.1, separate or custom parsers were implemented to consume the repository metadata. In the following, we describe each field of the table.

Number of repositories (Rep.): is the number of repositories available on the data source. Only *CKAN.org* and *RE3Data* contain a list with more than one repository or catalog. Some repositories were not accessible at the time this paper was written.

Datasets and Subsets: A fundamental requirement is to identify whether distributions are part of a larger subset or dataset. While some sources such as DBpedia were providing this kind of metadata, we approximated subsets for other repositories (cf. Section 5.2)

Distributions (Dist) and Accessible Distributions (AD): The distributions are either dump files or graphs from SPARQL endpoints. Accessible distributions are the ones which were accessible at the time of writing. Inaccessible distributions includes the ones with HTTP response 4XX, 5XX and with authentication-request.

Triples (T), Distinct Triples(DT) and Blank Nodes (BN): The total number of triples of each data source at the end of the processing; it is a simple triple count. *DT* is the number of distinct triples detected per data source, i.e. triples which are replicated across datasets are not taken into consideration here. Finally, *BN* is the number of blank nodes. Notice that we discarded blank node component in order to count the distinct triples.

Size and Bloom filter size: The sum of the uncompressed size of all distributions (all distributions were transformed into N-TRIPLES format) and the sum of all Bloom filters of the data source. Bloom filters tend to be more effective when large datasets are used. As will be discussed, data sources with a large number of small datasets usually have bigger filters.

5.2 Subset detection

Some repositories did not provide enough metadata for us to differentiate when two or more distributions belong to the same dataset. Therefore, we did our best effort analyzing the downloadURL of multiple distributions. When multiple downloadURL are within the same FQDN (Fully Qualified Domain Name), we consider that these distributions belong to the same dataset. For instance, *http://example.com/d1* and *http://example.com/d2* are two different distributions which belong to the same dataset in *example.com*. Moreover, the distribution location also gives important clues about whether the distribution is part of a subset, in particular we assume that distributions within the same directory tree normally belongs to the same subset.

5.3 Overlap analysis

Table 4 shows the amount of overlap between the distributions. 18.1% of all distributions streamed have at least 80% of data overlapped with another distribution. On the other hand, 81.1% of the distributions have less than 20% of duplicated data or are completely unique.

5.4 Discussion

Now, we discuss the columns **T** (triples), **DT** (distinct triples), **BN** (blank nodes), size and **BF** size from table 2. *DBpedia* dataset contains 19.8 billion triples, where 19.7 billion (99%) are distinct. Bloom filters were effective for representing the whole dataset using only 1.2% of the dataset's original size. For *LOV*, only 1.7% of the triples are duplicated, which is expected since each vocabulary or ontology describe their own domain. Therefore, the filters have 11.4% of the dataset size, as they are not so effective for small files. *CKAN.org* and *RE3 data* data have both 17% of duplicated data, respectively. This large amount of overlap is due to the fact that many datasets are described twice, for instance, datahub.io (which is present in both data sources) describes multiple versions of *DBpedia*. For both data sources, Bloom filter size is around 0.7% of the sum of the size of the datasets (again, Bloom filters tend to be more efficient with big sets).

LOD Laundromat contains over 27.7 billion triples, whereas only 98% are distinct. The filter size is 1.9% of the dataset size. *LOD-cloud* was indexed with 2.5 billion triples where 80% of them are unique. *Linghub* contains 346 million triples, where 87,9% of them are unique. Finally, we could fetch over 550 million triples from *SPARQLES*, and 94,4% of them are unique.

[28] See https://docs.mongodb.com/v3.0/core/gridfs/
[29] See https://github.com/google/guava
[30] See https://github.com/Baqend/Orestes-Bloomfilter
[31] See https://github.com/AKSW/IDOL/

Source	Rep.	Datasets	Subsets	Dist.	AD	T	DT	BN	Size	BF size
DBpedia (2016-10)	1	1	476	8,859	8,729	19.8B	19.7B	99k	3.3TB	41GB
LOV	1	579	6	579	579	654k	643k	162k	35MB	4.0MB
CKAN.org	126/147	7,043	5,439	17,993	13,591	6.3B	5.19B	314M	829GB	5.4GB
RE3 data	16/20	2,098	973	11,930	9,735	5.9B	4.93B	310M	774GB	5.1GB
LOD Laund.	1	1,489	1,259	34,969	33,619	27.7B	27.1B	362k	4.2TB	79GB
LOD-Cloud	1	1,303	2,830	2,262	741	2.5B	2.1B	256M	191GB	339MB
Linghub	1	175	607	640	402	346M	304M	2.6M	26GB	101MB
Sparqles	1	549	-	6,143	3,753	550M	519M	41M	92GB	189MB
Total	174	13,237	11,590	83,375	71,149	65.3B	62.0B	1.2B	9.4TB	120GB
Distinct total	163	-	-	63,194	58,152	56.1B	56.1B	-	-	-

Table 2: Overview of sources used to acquire datasets: Rep. is the number of repositories (accessible/available), datasets, subsets, Dist. are distributions (including SPARQL endpoints), AD is accessible distribution, T triples, DT is distinct triples in the data source, BN is blank nodes, Size of uncompressed file size, and last, the storage size of Bloom Filters.

Size (triples)	Distributions
<10k	44%
10k - 1M	47%
1M - 100M	7%
100M - 1B	1%
> 1B	1%

Table 3: Overview of the size of the distributions.

Overlap	Distributions
80-100%	18.1%
60-80%	0.8%
40-60%	0.0%
20-40%	0.0%
0-20%	81.1%

Table 4: Overlap found vs. amount of distributions.

	L	NP	W
Access	9.7%	8.7%	18.4%
Structural	22.3%	25.2%	32%
License	54.3%	1%	-
Informational	63%	-	-
Provenance	79.5%	-	33%

Table 5: Detected metadata: L lack of data, NP no pattern and W wrong.

In total, we have found 83,375 distributions of which 71,149 were accessible. We streamed over 65.3 billion triples. In addition to the intra-repository duplication check, which reduced the amount of triples to 62.0B (95%), we did an additional cross-repository check that amounted to a total of 56.1 billion distinct triples for the whole web of data (85.9%).

5.5 Metadata Quality

We assessed the quality of the retrieved metadata from all data sources and identified three different cases of quality issues: a) the lack of a specific property using standard vocabularies, b) the property can be found, however, has erroneous data, either wrong datatype or inappropriate content, c) some property exists with the metadata information, but it uses not standardized vocabularies and can only be discovered by regex patterns (e.g."license", "description") or manually inspecting the data. For our assessment, we classified metadata in five different groups. The *access* group provides information about the access layer of a dataset, for instance, dump file location or SPARQL endpoints. The *structural* group provides data for dataset size, serialization format, and compression format. The *licensing* group provides data about the dataset license. The *provenance* group shows data about the dataset origin or derivation and, finally, the *informational* group presents dataset title, description, label and others.

Table 5 shows an overview of the metadata obtained. For the access layer, 9.7% of the datasets had no data indicating a URI or any address where the dataset could be obtained. 8.7% of the access data could be found through regular expressions. Of the existing metadata, for 18.4% of the data location URIs we received either HTTP 4xx or HTTP 5xx error . Within the structural layer, 22.3% of the distributions had no data w.r.t. size and format and 32% were incorrect when compared to the data. 25.2% of the structural data were found through regular expression (e.g., serialization and compression format were extracted from the URI). 54.3% of the datasets had no license description at all. 63% had no informational data (more precisely, from this 63%, 71% have no description and 29% have no title or label), and 79.5% of the datasets have no provenance information. We could not validate this further as such information

is considered manual metadata and authoritatively defined by the publisher.

6 CONCLUSIONS

In this paper, we provide a comprehensive and as complete as possible, insight of the web of data overcoming many heterogeneous technical barriers. We performed this by harvesting dataset metadata from a thorough list of metadata registries, processing all RDF data. Through this process we observed a significant amount of duplicated data, reducing the de-facto amount of available triples from 65.3 Billion to 56.1 Billion unique statements (85.9%) spread across 71,149 accessible and distinct files. Based on our theoretical assessment of data structures, Bloom filters were the best option and we give concrete insight on the parameters to choose, resulting in 120GB of compressed Bloomfilters, thus being space-efficient and also time-efficient (analysis of all data takes 5 days on our –not overpowered– machine).

While we managed to retrieve, analyse and normalise metadata for the retrieved data, we would like to stress two very important points: As previous work pointed out and has been confirmed by our findings, metadata is of poor quality. This has as a consequence a large impact on the actual usability and interpretation of data, preventing the generation of valuable structure across datasets. In this work, we have investigated how to create complete and accurate measures of data duplication, which is a necessary prerequisite for future work focusing on cross-dataset manual metadata error correction as well as generation of analytic and heuristic metadata, in particular accessibility (ease of retrieval), linking equivalent subsets, and proper versioning/provenance tracing.

Acknowledgment. This paper‘s research activities were funded by grants from the FP7 & H2020 EU projects ALIGNED (GA-644055), LIDER (GA-610782), FREME (GA-644771), from the project Smart Data Web BMWi project (GA-01MD15010B) and CAPES and CNPq foundation (scholarships 13204/13-0 and 201808/15-3).

REFERENCES

[1] K. Alexander and M. Hausenblas. Describing linked datasets - on the design and usage of void, the vocabulary of interlinked datasets. In *In Linked Data on the Web Workshop (LDOW 09)*, 2009.

[2] W. Beek, L. Rietveld, H. R. Bazoobandi, J. Wielemaker, S. Schlobach, T. Tudorache, A. Bernstein, C. Welty, C. Knoblock, D. Vrandečić, P. Groth, N. Noy, K. Janowicz, and C. Goble. Lod laundromat: A uniform way of publishing other people's dirty data. In *The Semantic Web – ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, Cham, 2014. Springer International Publishing.

[3] B. H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13(7), July 1970.

[4] M. Brümmer, C. Baron, I. Ermilov, M. Freudenberg, D. Kontokostas, and S. Hellmann. DataID: Towards Semantically Rich Metadata for Complex Datasets. In *Proceedings of the 10th International Conference on Semantic Systems*, 2014.

[5] J. Debattista, S. Londoño, C. Lange, and S. Auer. Quality assessment of linked datasets using probabilistic approximation. In *Proceedings of the 12th European Semantic Web Conference on The Semantic Web. Latest Advances and New Domains - Volume 9088*, pages 221–236, New York, NY, USA, 2015. Springer-Verlag New York, Inc.

[6] D. Esteves, D. Moussallem, C. B. Neto, T. Soru, R. Usbeck, M. Ackermann, and J. Lehmann. Mex vocabulary: a lightweight interchange format for machine learning experiments. In *Proceedings of the 11th International Conference on Semantic Systems*, pages 169–176. ACM, 2015.

[7] M. Freudenberg, M. Brummer, J. Rucknagel, R. Ulrich, T. Eckart, D. Kontokostas, and S. Hellmann. The metadata ecosystem of dataid. In *10th International Conference on Metadata and Semantics Research*, 2016.

[8] A. Hogan, A. Harth, J. Umbrich, S. Kinsella, A. Polleres, and S. Decker. Searching and browsing linked data with swse: The semantic web search engine. *Web semantics: science, services and agents on the world wide web*, 9(4), 2011.

[9] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 2014.

[10] F. Maali and J. Erickson. Data Catalog Vocabulary (DCAT). W3C recommendation, W3C, Jan. 2014.

[11] J. P. McCrae and P. Cimiano. Linghub: a Linked Data based portal supporting the discovery of language resources. In A. Filipowska, R. Verborgh, and A. Polleres, editors, *SEMANTiCS*, CEUR Workshop Proceedings. CEUR-WS.org, 2015.

[12] R. Meusel, B. Spahiu, C. Bizer, and H. Paulheim. Towards Automatic Topical Classification of LOD Datasets. In C. Bizer, S. Auer, T. Berners-Lee, and T. Heath, editors, *LDOW@WWW*, CEUR Workshop Proceedings, 2015.

[13] L. Michael, W. Nejdl, O. Papapetrou, and W. Siberski. Improving distributed join efficiency with extended bloom filter operations. In *21st International Advanced Information Networking and Applications (AINA-07)*. IEEE, 2007.

[14] C. B. Neto, D. Esteves, T. Soru, D. Moussallem, A. Valdestilhas, and E. Marx. Wasota: What are the states of the art? In *SEMANTiCS (Posters, Demos, SuCCESS)*, 2016.

[15] H. Pampel, P. Vierkant, F. Scholze, R. Bertelmann, M. Kindling, J. Klump, H.-J. Goebelbecker, J. Gundlach, P. Schirmbacher, and U. Dierolf. Making Research Data Repositories Visible: The re3data.org Registry. *PLoS One*, 8(11), Nov 2013.

[16] C. Patel, K. Supekar, Y. Lee, and E. K. Park. OntoKhoj: A Semantic Web Portal for Ontology Searching, Ranking and Classification. In *Proceedings of the 5th ACM International Workshop on Web Information and Data Management*. ACM, 2003.

[17] M. Schmachtenberg, C. Bizer, and H. Paulheim. Adoption of the linked data best practices in different topical domains. In *The Semantic Web – ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, Cham, 2014. Springer.

[18] P.-Y. Vandenbussche, G. A. Atemezing1, P. Maria, and B. Vatant. Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. *Semantic Web Journal*, 2015.

[19] R. Verborgh, M. Vander Sande, P. Colpaert, S. Coppens, E. Mannens, and R. Van de Walle. Web-Scale Querying through Linked Data Fragments. In *Proceedings of the 7th Workshop on Linked Data on the Web*, Apr. 2014.