Wouter Beek and Javier David Fernandez Garcia and Ruben Verborgh

LOD-a-lot: A single-file enabler for data science

Book Section (Accepted for Publication)
(Refereed)

LOD-a-lot: A Single-File Enabler for Data Science

Wouter Beek, Javier D. Fernández, Ruben Verborgh

Please cite as:

Wouter Beek, Javier D. Fernández, and Ruben Verborgh. 2017. LOD-a-lot: A Single-File Enabler for Data Science. In Proceedings of the 13th International Conference on Semantic Systems (Semantics2017), Rinke Hoekstra, Catherine Faron-Zucker, Tassilo Pellegrini, and Victor de Boer (Eds.). ACM, New York, NY, USA, 181-184. DOI: https://doi.org/10.1145/3132218.3132241

LOD-a-lot: A Single-File Enabler for Data Science

Wouter Beek
Dept. of Computer Science,
VU University Amsterdam, NL
w.g.j.beek@vu.nl

Javier D. Fernández
Vienna U. of Economics and Business
Complexity Science Hub Vienna, AU
javier.fernandez@wu.ac.at

Ruben Verborgh
IDLab, ELIS Department,
Ghent University – imec, Belgium
ruben.verborgh@ugent.be

ABSTRACT

Many data scientists make use of Linked Open Data (LOD) as a huge interconnected knowledge base represented in RDF. However, the distributed nature of the information and the lack of a scalable approach to manage and consume such Big Semantic Data makes it difficult and expensive to conduct large-scale studies. As a consequence, most scientists restrict their analyses to one or two datasets (often DBpedia) that contain at most hundreds of millions of triples. *LOD-a-lot* is a dataset that integrates a large portion (over 28 billion triples) of the LOD Cloud into a single ready-to-consume file that can be easily downloaded, shared and queried with a small memory footprint. This paper shows there exists a wide collection of Data Science use cases that can be performed over such a *LOD-a-lot* file. For these use cases *LOD-a-lot* significantly reduces the cost and complexity of conducting Data Science.

CCS CONCEPTS

•Theory of computation → Data structures design and analysis; •Information systems → *World Wide Web;*

1 INTRODUCTION

The *Linked Open Data (LOD) Cloud* [3] materializes the idea of using the Web as a huge shared space of knowledge, where people and machines publicly share and interconnect semi-structured data (typically as RDF). The Linked Data and Data Science communities have increasingly common fields of interest with cross-cutting perspectives and challenges, such as large-scale reasoning, Machine Learning, graph analysis, and Big Data management.

Although data scientists are increasingly using Linked Open Datasets, we see that analyses and evaluations are only performed with a *very limited number of datasets*, which might negatively impact the *external validity* of the research in question [16]. The lack of variety in datasets involved in Data Science analyses has many root causes: data quality, data discovery, data availability, and others. One reason is the *complexity and cost of hosting sustainable and scalable triple stores* [19]. On top of this comes the cost of evaluating distributed queries in a fast and scalable way [14, 11].

Recently, three complementary initiatives are promoting a scalable and low-cost consumption of Linked Datasets, impacting the way data scientists perform large-scale analyses and evaluations. First, the Header Dictionary Triples (**HDT**) [6] format represents RDF in a compressed file that enables basic query functionality. HDT files are less compact than traditional text compression but, thanks to internal indexes, enable efficient triple pattern queries. Second, the Triple Pattern Fragments (**TPF**) interface [21] proposes to alleviate the traditional burden of LOD servers by moving part of the query processing to clients. TPF allows simple triple patterns to be queried, where results are retrieved incrementally through paginated *fragments*. Complex SPARQL queries can be executed on top of TPF by a client who takes an active role in joining the results of subqueries. Given HDT's fast and low-cost triple pattern support, many public TPF interfaces use an HDT backend. Third, **LOD Laundromat** [2] crawls, cleans and republishes more than 650K LOD datasets, collected through popular data catalogs like Datahub, as HDT datasets, serving a TPF endpoint for each of them on a single server, which would not be possible with the more expressive SPARQL endpoint interface. Although this has significantly reduced the cost of Linked Data publishing and consumption, data scientists who wish to run large-scale analyses need to query many TPF endpoints and integrate the results.

LOD-a-lot[1] [5] goes one step further and integrates the 650K LOD datasets in LOD Laundromat into a *single*, ready-to-consume HDT file, also serving the corresponding integrated TPF interface over *commodity hardware*. In this paper we identify several categories of use cases that previously required an expensive and complicated setup, but that can now be run over a cheap and simple *LOD-a-lot* file. This paper shows that a wide collection of Data Science use cases can be performed over *LOD-a-lot*. We expect that, for these foreseen use case as well as for several unforeseen ones, *LOD-a-lot* will significantly reduce the cost and complexity of conducting Data Science with Linked Data.

2 CAPABILITIES OF *LOD-A-LOT*

The *LOD-a-lot* HDT file contains the 28B unique triples that are crawled by LOD Laundromat. The *LOD-a-lot* file is 304GB in size, using 133GB for compressing the IRI and literal dictionary and 171GB for storing the triples' graph structure. An additional index file of 220GB, which further optimizes querying [13], can be created on demand. The memory footprint required for exposing *LOD-a-lot* through an online endpoint, or for querying it locally, is only 15.7GB (≈ 3% of the total dataset size). This gives access to 28B triples that can be efficiently queried for every Triple Pattern.

The capabilities of *LOD-a-lot* can be split into three categories: it can enumerate various types of terms (Section 2.1), it can resolve Triple Pattern queries (Section 2.2), and it can return various data metrics (Section 2.3).

[1] See https://datahub.io/dataset/lod-a-lot

Semantics2017, Amsterdam, Netherlands

978-1-4503-5296-3/17/09…$15.00
DOI: 10.1145/3132218.3132241

2.1 Term enumeration

Since an RDF term can appear in either the (s)ubject, (p)redicate, (o)bject, or (g)raph position, we can summarize the positional occurrence of each term in the LOD Cloud as a subset of $\{s, p, o, g\}$. In terms of these denotations, *LOD-a-lot* contains the following six indexes: $\{i\}$, $\{l\}$, $\{s\}$, $\{p\}$, $\{o\}$, and $\{s, o\}$. A term can belong to multiple indexes, e.g., literals are also object terms. RDF terms syntactically belong to either of the following groups: blank nodes, IRIs, or literals. To improve usability, *LOD-a-lot* Skolemizes all blank nodes into well-known IRIs (possible in RDF 1.1), which ensures that all terms are universally unique. Since HDT is a *triple* storage format, *LOD-a-lot* does not store graph terms.[2]

Based on its explicit indexes, *LOD-a-lot* can uniquely enumerate all IRIs, literals, names (IRIs and literals), subjects, predicates, objects, nodes (subjects and object), sources (terms appearing only as subject), and sinks (terms appearing only as object). Because of its uniqueness guarantee, these *LOD-a-lot* enumerators support many term-based use cases in Data Science, as well as more complex use cases in which term enumeration plays a crucial role (Section 3).

2.2 Querying

LOD-a-lot can be efficiently answer all queries of the form $\langle s, p, o \rangle$, where s, p, and o are RDF terms or SPARQL variables. This subset of SPARQL is called Triple Patterns (TP). For example, the following query extracts all instances $?i$ and their explicitly asserted class $?c$: $\langle ?i, \texttt{rdf:type}, ?c \rangle$. The HDT backend achieves competitive query performance by (i) making use of an indexed dictionary that assigns compact representations to all RDF terms, (ii) depending on the positions that are bound in the query, accessing the subject-based, predicate-based and/or object-based indexes that HDT provides [13], and (iii) streaming the solution bindings.

An online deployment of *LOD-a-lot* is provided through a Triple Pattern Fragments (TPF) interface which allows queries to be asked and answered over HTTP. The endpoint runs on commodity hardware (8 cores at 2.6 GHz, 32 GB RAM and a SATA HDD on Ubuntu 14.04.5 LTS). TPF queries are answered within milliseconds and return result sets of arbitrary size by using pagination.

2.3 Metrics

LOD-a-lot also exposes data metrics. Metrics can be split into two categories: on-demand metrics that are dynamically calculated, and precomputed metrics that are stored and retrieved when needed.

The same indexes that are used for querying the data can also be used to estimate the number of solutions for a Triple Pattern query. For querying, the whole result set must be iterated through, but the result set size can be estimated without iteration. This allows on-demand metrics to be retrieved for each Triple Pattern. Because SPARQL endpoints *do* iterate through full result sets in order to calculate Triple Pattern cardinalities, they are typically slower in generating on-demand metrics than HDT.

The Triple Pattern Fragments server uses HDT on-demand metrics in order to communicate to the client the total number of results. Based on a TPF request, the client retrieves a first *fragment* response that contains the first 100 results of the full result set, together with metadata that expresses the estimated total result set size. Based on this estimate, the client is informed about the approximate number of subsequent requests that is needed to obtain the full result set. In addition to these on-demand metrics, HDT also stores metadata of the data stored in *LOD-a-lot*. This includes precomputed metrics (e.g., number of triples, number of literals).

[2] *LOD-a-lot* should be thought of as one big integrated graph. Provenance information can be retrieved through the LOD Laundromat infrastructure [16].

3 USE CASES FOR DATA SCIENCE

With the capabilities described in Section 2, *LOD-a-lot* is able to support a surprisingly wide range of Data Science use cases.

Query planning. On-demand metrics of estimated result set size (Section 2.3) can be used in query rewriting. Specifically, SPARQL Basic Graph Pattern evaluation can be optimized by reordering based on result set size. More advanced heuristics, such as the computation of characteristic sets [15], can be performed efficiently on top of *LOD-a-lot*, given that they only require to scan the data or perform specific queries to retrieve structural properties of the data (Section 2.2).

Enumerating schema. It is surprisingly difficult to extract the schema (or TBox) from a Linked Dataset: SPARQL queries of the form '`select distinct ?p { ?s ?p ?o }`' do not yield any results on most systems (resulting in a time-out), or they return only an initial segment of the actual result set. In *LOD-a-lot*, the set of properties can be efficiently enumerated (Section 2.1). Furthermore, the set of classes can be enumerated as well, by matching the TPs $\langle ?c, \texttt{rdfs:subClassOf}, ?d \rangle$, $\langle ?p, \texttt{rdfs:domain}, ?c \rangle$, $\langle ?p, \texttt{rdfs:range}, ?c \rangle$, and $\langle ?i, \texttt{rdf:type}, ?c \rangle$. Combining these into one query yields only a modest number of duplicate occurrences, e.g., when a class appears both as the domain and as the range of some property. By using these queries the schema of the entire LOD Cloud can be extracted. It is also possible to only extract parts of the schema that are of particular interest. For example, the subclass hierarchy of classes in the PROV vocabulary (like `prov:Entity`) can be extracted.

Obtaining statistics. By using the various term enumerators (Section 2.1) in combination with on-demand metrics (Section 2.3), *LOD-a-lot* allows statistics about the LOD Cloud to be retrieved. For example, the list of most used predicate terms together with the number of triples in which they occur. These statistics are obtained by combining the predicate term enumerator $\{p\}$ together with an on-demand metric lookup for the TP $\langle ?s, p, ?o \rangle$ (where p is a predicate term). Precomputed metrics can be retrieved from *LOD-a-lot* as well, resulting in a dataset metrics overview that is similar to a VoID description (e.g., number of triples, number of unique object terms).

Generating specialized indexes. Based on the indexes that are already provided by *LOD-a-lot* (Section 2.1), new indexes can easily be created. Firstly, the *LOD-a-lot* enumerators guarantee uniqueness, which is expensive to implement in SPARQL. As mentioned above, enumerating the distinct predicate terms in a SPARQL endpoint is often quite challenging. Secondly, *LOD-a-lot* does not enforce limits on the number of terms that can be retrieved from its enumerators. This is important because specialized indexes can be large, easily surpassing common SPARQL result set limits (typically 10,000). An example of a specialized index is obtained by combining the literal enumerator with a datatype

IRI filter that extracts all and only dates (`?lex^^xsd:date`). Another example is a literal enumerators combined with a filter that extracts all and only language-tagged strings in the German language (`?lex@de`). Furthermore, for each occurrence in the new index, the estimated number of relevant statements can be retrieved (Section 2.3). This allows us to find dates that occur most often, and German names or phrases that are commonly used.

Identity closure. The *explicit extension* of a property is the set of pairs of terms for which that property is asserted to hold. The explicit extension of the identity relation can be extracted from *LOD-a-lot* with TP query $\langle ?x, \texttt{owl:sameAs}, ?y \rangle$ (Section 2.2). The *implicit extension* of the identity relation is the explicit extension closed under equivalence (reflexivity, symmetricity, transitivity). This closure can be calculated by using *LOD-a-lot*, by storing a mapping from terms t to sets of identical terms $[\![t]\!]_=$. Solution bindings for the variables $?x$ and $?y$ in the above TP query result in adding new mapping pairs $t \mapsto \{t\}$, or in performing an ordered set merge between two existing pairs: $t \mapsto [\![s]\!] \cup [\![t]\!]$. After all merges have been performed, the remaining mapping implements the identity closure according to the semantics of `owl:sameAs`.

Archival context. When Linked Datasets are archived, they are linked to resources that exist at that time. For example, an archived dataset from 2014 may link to IRIs from the 2014 edition of DBpedia. In order to property capture that dataset, the 2014 edition of DBpedia should be archived as well. With *LOD-a-lot* it is possible to store and retrieve the context in which a dataset should be used, including linksets and external datasets.

Graph navigation. Graph navigation techniques play an important role in extracting value from the LOD Cloud, for instance by linking IRIs within a given dataset to IRIs that denote the same thing in other datasets. In order to extract additional value from linking, the link to the other dataset has to be followed and the properties of the linked-to IRI have to be retrieved. This graph navigational technique is known as 'Follow-Your-Nose'. *LOD-a-lot* allows a large subset of the LOD Cloud to be efficiently traversed. Evaluating a TP query (Section 2.2) can also be though of as traversing an edge in the integrated LOD Cloud graph.

Random walks. One specific form of graph navigation that is of particular importance in Data Science is the random walk. Calculations of node popularity (e.g., PageRank) often involve performing very many of these walks. In addition, random walks are used as one of the main sampling techniques in graph-based Machine Learning. Specifically, random walks that start at a given node n are used as a means of estimating edge/node co-occurrence statistics for n [17]. *LOD-a-lot* allows random walks to be extracted by using queries $\langle ?o?e?n \rangle$ and $\langle ?n?e?o \rangle$ (Section 2.2) to find all edge/node pairs (e, o) that are accessible from node n, from which a pair can be randomly picked.

Data quality metrics. Data quality can significantly impact knowledge-intensive tasks like automated reasoning and ontology-based data access. There are also domain in which data quality is of critical importance, e.g., Bioinformatics or cadastral base registries. Data cleaning crucially depends on a preparatory phase in which metrics are calculated over the data in order to quantify the level of data quality and in order to detect those areas in which quality is lacking [22]. Because *LOD-a-lot* has built-in support for metrics (Section 2.3) and can stream result sets of arbitrary size (Section 2.2), it is better suited towards calculating data quality metrics than SPARQL. An example of a data quality metric is checking whether RDF literals (lex, i) with lexical form lex and datatype IRI i denote values that actually belong in the indicated value space. $(1, \texttt{xsd:int})$ is valid but $(1.0, \texttt{xsd:int})$ is not [1].

Analyzing inconsistencies. An RDF graph is inconsistent if some of its statements contradict each other. For example, the following SPARQL query reveals multiple inconsistencies in well-known RDF datasets such as DBpedia[3]:

```
select ?i { ?c owl:disjointWith ?d . ?i a ?c, ?d }
```

Dealing with inconsistent data is achieved by (i) detecting the inconsistency, (ii) repairing it when possible or (iii) trying to reason with the inconsistency [10, 12]. The size and query capabilities of *LOD-a-lot* (Section 2.2) allow such inconsistencies to be measured at the level of the LOD Cloud. Specifically, *LOD-a-lot* allows the detection of inconsistencies that are not present in individual datasets, but that do arise when datasets are combined.

Studying structural properties. Relatively little is known about the structural properties of the Semantic Web. Early work in this area has observed the presence of power-law distributions and other network-based features, such as clustering coefficient and path lengths, over individual datasets of (up to) millions of triples [4, 9, 7]. It is not yet known whether the structural properties of the LOD Cloud are the same as the structural properties of individual datasets. Many practical applications can benefit from a better understand of the structural properties of the Semantic Web: query optimization, data summarization, data visualization. It is currently very costly to calculate such properties. *LOD-a-lot* democratizes large-scale access to the LOD Cloud, and has the capabilities required for calculating large-scale structural properties. The complexity of finding the right data sources and the overhead of querying them efficiently have been taken care of by the centralized term indexes (Section 2.1) and the integrated graph structure (Section 2.2).

Question answering. With the increasing number of high quality datasets, the Semantic Web is seeing a renewed interest in Question Answering (QA) over graph data [18]. Traditional graph-based query answering systems are restricted to using one graph, often DBpedia, as the main knowledge base. This implies that query answering does not work well over domain-specific knowledge, because the background knowledge is then lacking. It also implies that alternative views of the same content are not presented. *LOD-a-lot* provides a much large, but also a much more heterogeneous knowledge graph. *LOD-a-lot* includes generic as well as domain-specific datasets. Furthermore, the *LOD-a-lot* term index (Section 2.1) supports auto-completion, which is useful when typing queries. Graph-based query answering is often based on calculating the shortest path distance between nodes, which can be efficiently computed over HDT [8].

Federated querying. While *LOD-a-lot* provides a centralized approach for querying the LOD Cloud, it can be used in combination with other web services that support federated querying. Triple Pattern queries can be initially performed on *LOD-a-lot* (Section

[3] The resulting inconsistencies of this query in DBpedia are at https://goo.gl/sDsihJ.

2.2). Once a solution is retrieved LOD Laundromat can be consulted to find the original dataset from which the data was scraped, from which more up-to-date answers may be obtained. In this scenario *LOD-a-lot* acts as an index or summary that can be used for federation against individual endpoints.

The availability of *LOD-a-lot* provides a useful base case against which federated approaches can be compared. For example, LOD Laundromat publishes 650K Triple Pattern Fragments endpoints: one for each dataset. The LOD Cloud contains the original locations where data is (and in some cases: was) published. Querying each of these deployment has different costs and benefits in terms of performance, reliability, expressiveness, and staleness. Furthermore, it is possible to try out different partitions of *LOD-a-lot* and see how this impacts state-of-the-art federated approach. This is especially interesting given the competitive results of federation over TPF interfaces compared to SPARQL endpoints [21].

Versioning. Having a single-file dump of the LOD Cloud makes it easy to provide different versions of the LOD Cloud. Complexity in versioning arises when delta's have to be calculated on-the-fly and/or when data is spread over multiple backends and multiple files. With *LOD-a-lot*, providing multiple versions of *LOD-a-lot* simply involves publishing one file per timestamp. HDT files can be made accessible through Triple Pattern Fragments in combination with the Memento protocol for versioning [20]. Such versions can be used for historical comparisons, as well as historical analyses of the evolution of the Semantic Web over time. This can provide valuable insights in the changes of the LOD Cloud in terms of size, variety, quality, structure, etc.

4 CONCLUSION

Data Science often involves conducting analyses over large data collections. Such Data Science use cases are not very well served by today's SPARQL endpoints, which are relatively complicated and costly to setup for very large data collections. In practice, SPARQL endpoints enforce limits on the number of results that can be retrieved or involved in an aggregate query. There are many use case in Data Science, in fact, many of the use cases presented in this paper, that go beyond such limits. In addition to size restrictions, SPARQL endpoints are known to have low availability, which further hampers conducting Data Science analyses.

In order to enable conducting large-scale and cost-effective Data Science, this paper has presented a wide variety of use cases in which *LOD-a-lot* can be used. *LOD-a-lot* is a single file that contains a large copy of the Linked Open Data (LOD) Cloud, to the extent that it is crawled and cleaned by the LOD Laundromat. *LOD-a-lot* consists of over 28 billion RDF triples that are stored in a compressed and self-indexed HDT format, which allows (i) term enumeration, (ii) Triple Pattern Fragment (TPF) matching, and (iii) on-demand and precomputed metrics. *LOD-a-lot* can be hosted against relatively low hardware costs: 524GB disk and 15.7GB memory, which – at the time of writing – cost 305 euro.

Future work focuses on working closely with data scientist in order to facilitate use and adoption of *LOD-a-lot*. Several use cases presented in this paper are already being run over *LOD-a-lot*. With its triple-level index, *LOD-a-lot* stays close to the official semantics of RDF, where meaning is expressed in a triple graph with model theoretic semantics. However, there are use cases in which a data scientist wants to go beyond the standard semantics and take the original context of use into account, including who makes which assertion, and how often a proposition is asserted by different publishers. For such use cases *LOD-a-lot* must be combined with LOD Laundromat, which stores the required provenance information.

ACKNOWLEDGMENTS

The work of Wouter Beek is part of the research programme MaestroGraph (612.001.552), financed by the Dutch Organization for Scientific Research (NWO). The work of Javier Fernández is funded by the Austrian Science Fund: M1720-G11 and the European Union's Horizon 2020 research and innovation programme grant 731601, WU Post-doc Research Contracts. Ruben Verborgh is a postdoctoral fellow of the Research Foundation – Flanders.

REFERENCES

[1] W. Beek, F. Ilievski, J. Debattista, S. Schlobach, and J. Wielemaker. *Literally* Better: Analyzing and Improving the Quality of Literals. *Semantic Web*, 2017.
[2] W. Beek, L. Rietveld, H. R. Bazoobandi, J. Wielemaker, and S. Schlobach. LOD Laundromat: A uniform way of publishing other people's dirty data. In *Proc. of ISWC*, pages 213–228. 2014.
[3] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data: The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.
[4] L. Ding and T. Finin. Characterizing the Semantic Web on the Web. In *Proc. of ISWC*, pages 242–257, 2006.
[5] J. D. Fernández, W. Beek, M. A. Martínez-Prieto, and M. Arias. Lod-a-lot: A queryable dump of the LOD cloud (2017). In *Proc. of ISWC*, 2017. http://purl.org/HDT/lod-a-lot.
[6] J. D. Fernández, M. A. Martínez-Prieto, C. Gutiérrez, A. Polleres, and M. Arias. Binary RDF representation for publication and exchange (HDT). *Web Semantics: Science, Services and Agents on the World Wide Web*, 19, 2013.
[7] J. D. Fernández, M. A. Martínez-Prieto, P. de la Fuente Redondo, and C. Gutiérrez. Characterizing RDF datasets. *Journal of Information Science*, 2016.
[8] E. Filtz, V. Savenkov, and J. Umbrich. On finding the k shortest paths in RDF data. In *Proc. of Intelligent Exploration of Semantic Data*, 2016.
[9] W. Ge, J. Chen, W. Hu, and Y. Qu. Object link structure in the semantic web. *The Semantic Web: Research and Applications*, pages 257–271, 2010.
[10] J. Grant and A. Hunter. Measuring inconsistency in knowledgebases. *Journal of Intelligent Information Systems*, 27(2):159–184, 2006.
[11] O. Hartig and G. Pirrò. A Context-based Semantics for SPARQL Property Paths over the Web. In *Proc. of ESWC*, pages 71–87, 2015.
[12] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Inconsistency-tolerant semantics for description logics. In *International Conference on Web Reasoning and Rule Systems*, pages 103–117. Springer, 2010.
[13] M. A. Martínez-Prieto, M. Arias, and J. D. Fernández. Exchange and Consumption of Huge RDF Data. In *Proc. of ESWC*, pages 437–452, 2012.
[14] I. C. Millard, H. Glaser, M. Salvadores, and N. Shadbolt. Consuming multiple linked data sources: Challenges and experiences. In *Proceedings of the First International Conference on Consuming Linked Data-Volume 665*, 2010.
[15] T. Neumann and G. Moerkotte. Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins. In *Proc. of ICDE*, 2011.
[16] L. Rietveld, W. Beek, and S. Schlobach. LOD Lab: Experiments at LOD Scale. In *Proc. of ISWC*, pages 339–355, 2015.
[17] P. Ristoski and H. Paulheim. Rdf2vec: Rdf graph embeddings for data mining. In *International Semantic Web Conference*, pages 498–514. Springer, 2016.
[18] C. Unger, C. Forascu, V. Lopez, A.-C. N. Ngomo, E. Cabrio, P. Cimiano, and S. Walter. Question answering over linked data (qald-4). In *Working Notes for CLEF 2014 Conference*, 2014.
[19] P.-Y. Vandenbussche, J. Umbrich, L. Matteis, A. Hogan, and C. Buil-Aranda. SPARQLES: Monitoring public SPARQL endpoints. *Semantic Web Journal*, Preprint(Preprint):1–17, 2017.
[20] M. Vander Sande, R. Verborgh, P. Hochstenbach, and H. Van de Sompel. Towards sustainable publishing and querying of distributed Linked Data archives. *Journal of Documentation*, 73(6), 2017.
[21] R. Verborgh, M. Vander Sande, O. Hartig, J. Van Herwegen, L. De Vocht, B. De Meester, G. Haesendonck, and P. Colpaert. Triple Pattern Fragments: a Low-cost Knowledge Graph Interface for the Web. *JWS*, 37–38:184–206, 2016.
[22] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, and S. Auer. Quality assessment for linked data: A survey. *Semantic Web*, 7(1):63–93, 2016.