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 University of Economics and Business, 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 RDF 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, locally or online, 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.

KEYWORDS

Data Science, Data Access, Linked Open Data

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, in practice, analysis and evaluations are performed with only 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 main reason is the *complexity and cost of hosting sustainable, scalable and centralized triple stores* [19]. On top of this comes the cost of evaluating distributed queries in a fast and scalable way [11, 14].

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 onto clients. TPF allows simple triple patterns to be queried on the server, where results are retrieved incrementally through paginated *fragments*. Complex SPARQL queries can be executed on top of TPF by a client, who takes a leading role in query planning in order to join the results of subqueries. Given HDT's fast and low-cost triple pattern support, many public TPF interfaces uses an HDT backend. Third, **LOD Laundromat** [2] crawls, cleans and republishes more than 650K LOD datasets 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, still, 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*. The HDT file represents all 28B triples in 304GB, using 133GB for compressing the IRI and literal dictionary and 171GB for the triple structures. Note that, thanks to the self-indexed representation of these components, HDT can efficiently resolve subject-based TPs (i.e. TPs where the subject is bounded). The so-called HDT-Focused on Querying (HDT-FoQ) [13] extends HDT with two additional indexes that enable predicate and object-based access. These indexes are distributed in a supplementary file within *LOD-a-lot*, which takes an additional 220GB. Overall, the RAM footprint required for exposing *LOD-a-lot* through an online endpoint, or for querying *LOD-a-lot* locally, is only 15.7GB ($\approx$ 3% of the total dataset size) for 28B triples. 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. Note that *LOD-a-lot* does not natively expose the same functionality as a full-blown database suite, mainly offering TPF, although there exists a Jena adapter that offers full SPARQL[2]. Despite these limitations, this paper shows a wide collection of Data Science use cases that can be performed over *LOD-a-lot*. We expect that, for these foreseen use case as well as for future ones, *LOD-a-lot* will significantly reduce the cost and complexity of conducting Data Science using Linked Open Data.

2 CAPABILITIES OF *LOD-A-LOT*

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).

SEMANTiCS, Amsterdam, The Netherlands
2017. 978-x-xxxx-xxxx-x/YY/MM…$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

[1] See https://datahub.io/dataset/lod-a-lot
[2] See http://www.rdfhdt.org/manual-of-hdt-integration-with-jena/

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 addition RDF terms can syntactically belong to exactly one of the following groups: (b)lank nodes, (i)ris, or (l)iterals. 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. Blank nodes are only unique within the scope of the document in which they appear, which causes problems when sharing, reordering and recombining data documents. Specifically, in order to concatenate two RDF documents that contain blank nodes one has to consistently relabel them to avoid inadvertently changing the data. (People are known to skip this step in practice.) For these reasons, *LOD-a-lot* skolemizes blank nodes into IRIs. Because *LOD-a-lot* abstracts away from the context of use, it does not allow graph term occurrences to be retrieved. Based on its explicit indexes, the following terms can be *uniquely* enumerated by *LOD-a-lot*: IRIs, literals, names (IRIs and literals), subjects, predicates, objects, nodes (terms appearing as subject, object, or both), sources (terms appearing only as subject), sinks (terms appearing only as object). Because *LOD-a-lot* provides all of these these enumerations with a uniqueness guarantee, it supports many term-based use cases in Data Science, as well as more complex use cases in which term enumeration plays a crucial role (see Section 3).

2.2 Querying

LOD-a-lot can be queried for triple pattern queries, i.e., all queries of the form $\langle s, p, o\rangle$, where s, p, and o are RDF terms or variables. For example, to extract all instances $?i$ and their explicitly asserted class $?c$ from the LOD Cloud the query $\langle ?i, \texttt{rdf:type} ?c\rangle$ is issued. The HDT backend achieves competitive querying performance for triple patterns by (i) making use of the indexed dictionary of terms to retrieve the compact representation of each query term, (ii) accessing the corresponding subject-based, predicate-based or object-based index in HDT-FoQ [13] (depending on the bounded terms of the query), and (iii) streaming the resulting terms satisfying the user query (using the HDT dictionary of terms to retrieve the uncompressed string of each term). A deployment of *LOD-a-lot* is exposed through a TPF interface on commodity hardware (8 cores at 2.6 GHz, 32 GB RAM and a SATA HDD on Ubuntu 14.04.5 LTS), and can resolve TPF queries (with 100 results as page size) at the level of milliseconds.

2.3 Metrics

In addition to enumerations and queries, *LOD-a-lot* also exposes several metrics over the data. We can split metrics over datasets into two categories: on-demand metrics that are dynamically calculated, and precomputed metrics that are stored and retrieved when needed.

HDT provides a lazy-evaluation strategy for Triple Pattern queries. By using its built-in triple indexes [13], HDT is able to partially resolve such queries in order to find the exact range of index positions where the result set can be found. Although this whole range must be iterated through in order to obtain the full result set, the length of this range alone already provides an approximation of the size of the result set. In this way, cardinality estimates for every Triple Pattern can be requests as on-demand metrics from *LOD-a-lot* and are calculated within milliseconds (which is faster than iterating through the full result set, especially when this is very large). Because SPARQL endpoints *do* iterate through full result sets in order to calculate Triple Pattern cardinalities, they are typically much slower than HDT. Unfortunately this is an inherent property of the SPARQL 1.1 query language which has only standardized calculation and retrieval of exact counts.

The Triple Pattern Fragments server uses these on-demand metrics in order to communicate to the client how many results there are in total. 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 result set size. Based on this estimation, the client is made aware about the number of subsequent requests that is needed in order to obtain the full result set. In addition to these on-demand metrics, HDT also stores metadata of the information in *LOD-a-lot*, which includes precomputed metrics.

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. The estimated result set size (Section 2.3) can be used in query planning to heuristically determine how query engines, such as a Triple Pattern Fragments client, should reorder SPARQL queries' Basic Graph Patterns to speed up query evaluation. 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, 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, such as the most used predicates and the number of triples in which they occur. These statistics are obtained by combining the predicate term enumerator $\{p\}$ together with a TP estimation lookup of the form $\langle ?s, p, ?o \rangle$ (where p is a predicate term). In addition to these on-demand metrics, the precomputed metrics can be retrieved from the *LOD-a-lot* metadata, resulting in a dataset metrics overview that is similar to that of 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 because of the following two reasons. 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 rather 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). In this way we can find which dates occur more often, or which German names or phrases 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. For example, the explicit extension of the identity relation can be extracted from *LOD-a-lot* by using the Triple Pattern query $\langle ?x, \texttt{owl:sameAs}, ?y \rangle$ (Section 2.2). Because all RDF terms can be uniquely enumerated (Section 2.1), it is also possible to calculate the *implicit extension* of the identity relation, which includes the pairs $\langle t, t \rangle$ for each term t (reflexivity). Using both the TP query and the reflexive enumeration, the full implicit extension of identity, i.e., closed under equivalence, can be calculated by using a key/value store. In this key/value store, a value is an ordered set of terms that are considered identical, and a key is the first term that was added to the identity set. Based on each pair belonging to the explicit `owl:sameAs` extension, either a new (singleton) identity set is created, or two existing identity sets are merged (using a set merge).

Graph navigation. Graph navigation techniques play an important role in extracting value from the LOD Cloud by linking IRIs within a given dataset to IRIs that denote the same thing in other datasets. In order to extract the 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 the Follow-Your-Nose approach. One simple but common approach occurs when IRI from a given dataset are linked to IRIs that denote the same thing in DBpedia. Because DBpedia is a multi-lingual dataset, it is often possible to extract language-tagged labels from it that make resources easier to identify and understand by human users.

Random walks. One specific form of graph navigation that is of particular importance in Data Science is the random walk. Calculation 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 for 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 random pair can be picked.

Analyzing inconsistencies. In general, 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 rdf:type ?c, ?d }`. Although many practical use cases disregard the potential 'messiness' of the data, particular scenarios such as automatic reasoning and ontology-based data access or areas where data quality is of critical importance, such as bioinformatics, rely on a higher data quality. Thus, dealing with such inconsistent data is mainly 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) position itself as a perfect test bench for measuring the overall LOD data (in)consistency, to study its roots and the degree of inconsistency introduced when all LOD datasets are integrated (w.r.t. the inconsistency of the individual datasets that can be accessed via LOD Laundromat), and to evaluate conflict resolution strategies and methods to reason in the presence of inconsistencies. In addition, *LOD-a-lot* also enables other quality metrics analysis, such as the practical use of literals [1].

Studying structural properties. In spite of the emerging need for characterizing the structural properties of real-world RDF data (e.g. for query optimization, data summarization and data visualization), the complexity of the task makes it difficult to perform studies at web scale. Initial 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 datasets that consist of (only) millions of triples [4, 7, 9]. *LOD-a-lot* democratizes the access to a large portion of the LOD Cloud, and serves the required facilities to perform such study efficiently: On the one hand, the complexities of the terms and the graph structure are isolated thanks to the internal organization of HDT. This facilitates to perform independent studies on both components while increasing the scalability of the approach (e.g. the connectivity of the nodes can be measured on the graph structure, obviating the common verbosity of RDF terms). On the other hand, the *LOD-a-lot* query facilities (Sections 2.2 and 2.3) serve the required functionality to inspect network-based metrics.

Question answering. The practical development of the Semantic Web and the increasing presence of structured data and knowledge bases has brought a renewed interest in Question Answering (QA) [18]. Most of the current systems use DBpedia as the main knowledge base, analyzing the different paths to resolve a given query. In this area, *LOD-a-lot* provides a huge interconnected knowledge base where entities (Section 2.1) and connections (Sections 2.2 and 2.3) can be found in an efficient and scalable way. In addition, such connections are often based on shortest path distance, which can be efficiently computed with HDT as well [8].

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

Federated querying. The original setup of the LOD Laundromat was large-scale federation: each of the 650K datasets has its own TPF endpoint. This contrasts with the LOD cache, where all datasets from the LOD cloud are brought together in a single endpoint. Both were due to necessity: generating a single HDT file for all LOD cloud triples required too many resources (before *LOD-a-lot*), and 650K SPARQL endpoints on a single machine require too many resources as well (and this is still the case). However, with *LOD-a-lot*, we now have such a single dataset, which means that we can benchmark the extremes (querying over 650K datasets versus querying over 1 dataset) and everything in between. For instance, we could test different partitionings of *LOD-a-lot* and see how this impacts state-of-the-art federation. This is especially interesting given the competitive results of federation over TPF interfaces compared to SPARQL endpoints [21]. Also, using *LOD-a-lot*, we can prepare an index or summary which can be used for federation against individual endpoints.

Versioning. Having a single-file dump of the LOD cloud also opens the door for straightforward versioning, which is more difficult with a more complex data infrastructure. With HDT, providing multiple versions of *LOD-a-lot* simply involves one file per timestamp, which can be made accessible through TPF and the Memento protocol [20]. Such versions can be used for historical comparisons, as well as historical analyses of the evolution of any of the previously mentioned aspects 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 costly to setup over very large data collection. In practice, SPARQL endpoints also enforce limits on the amount of results that can be retrieved as well as on the maximum number of intermediary results that is allowed to be combined in an aggregation. There are ample use case, in fact: many of the use cases described in the above, that go beyond such limits. In addition to size restrictions, many SPARQL endpoints have low availability which can further complicate conducting Data Science analyses.

In order to further enable and improve 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 serves (i) term enumeration, (ii) Triple Pattern Fragment (TPF) matching, and (iii) on-demand and precomputed metrics. Due to the Linked Data Fragments publishing approach, *LOD-a-lot* can be queried locally/offline as well as online over HTTP. *LOD-a-lot* can be hosted using relatively low hardware costs: 0.5TB disk and 15GB memory, which are commonly available in modern web servers. This allows multiple use cases for data science to be efficiently performed at LOD scale, such as query planning, schema enumeration, inconsistency analysis or query answering.

Our future work first focuses on working closely with data scientist in order to facilitate the use of *LOD-a-lot*. Several of the use cases detailed in this paper have already been run, or are currently being run, over *LOD-a-lot*. The current version of *LOD-a-lot* abstracts away from the context in which assertions have been made, i.e., the graph term for quadruples, as well as the original download location where the data was scraped from. Semantically speaking, this means that *LOD-a-lot* stays close to the official semantics of RDF, where meaning is expressed in a model theory in which one assertion of a proposition carries the same meaning as one hundred assertions of that same proposition, and where the authority of who makes a certain assertion does not matter semantically. 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 made which assertion in the LOD Cloud, and how often a proposition is asserted by different data publishers. For such use cases *LOD-a-lot* can be combined with LOD Laundromat, which does store such provenance information.

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. Springer, 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), 2017.

[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.