Strengthening LargeRDFBench for Interoperable Federated SPARQL Evaluation

Bryan-Elliott Tam[1,*], Muhammad Saleem[2] and Ruben Taelman[1]

[1]Department of Electronics and Information Systems, Ghent University – imec
[2]AKSW, University Leipzig, Germany

Abstract

LargeRDFBench is one of the most comprehensive benchmarks for evaluating federated SPARQL query engines, combining a large collection of real, interlinked datasets with a rich query suite that has made it a reference point for the community. Evaluations of federated engines are published by comparing engine results against the benchmark's expected results, so it is important that the results of the benchmark are themselves reproducible. Moreover, several of its data dumps violate the RDF specifications, so only engines that parse RDF leniently can host them, and its expected results are distributed in an ad hoc format. We systematically identify and categorize these data-quality issues and repair them, producing standards-conformant serializations of every affected dataset. Furthermore, we re-encode the benchmark's expected results in the W3C SPARQL 1.1 Query Results JSON Format and correct their discrepancies. We contribute a standards-compliant edition of LargeRDFBench, produced by a reproducible cleaning pipeline, together with its expected query results in a standard, machine-verifiable format. Every dataset now parses under strict, specification-compliant RDF parsers, and the expected results are machine-verifiable through a standard format, extending the benchmark's reach to the full range of conformant engines while staying faithful to the original data. Reproducing the expected results end-to-end with an independent implementation uncovers corruption in the published reference, and discrepancies between our results and the original ones, some of which are not trivial to resolve or remain open questions. We further perform a preliminary comparison, not previously explored, of ASK- and COUNT-based source selection in the FedX algorithm. This work strengthens an already valuable community resource by aligning its artifacts with the RDF standards. In doing so, we broaden the set of engines that can be fairly and reproducibly compared. We also raise the question of how the results of federated queries under automatic source selection can be made reproducible.

Keywords

Scientific Reproducibility, Benchmark, Linked Data, RDF, RDF Repairs, Interoperability

CEUR-WS.org/Vol-4322/paper_4.pdf

1. Introduction

Answering difficult questions over the Web of data frequently requires combining facts that reside in separate, independently published datasets. In the life sciences, for instance, a single question may link a drug in DrugBank to the pathways it participates in from KEGG and the chemical entities that characterize it in ChEBI, while a general-knowledge question may join an entity in DBpedia with its geographic context in GeoNames. Federated query processing serves precisely this need, by evaluating a single SPARQL query across multiple endpoints at once and integrating their data at query time without centralization [1]. In practice, replicating every relevant dataset into a single store is often untenable, because the sources sit behind separate administrative boundaries, carry incompatible licenses [2], and can be too voluminous to copy [3].

Reproducibility is a long-standing concern across the sciences [4], and computer science is no exception [5]. Benchmarking is an essential part of developing and comparing SPARQL engines and federation algorithms. Benchmarks are standardized and reproducible scenarios that capture either realistic workloads or interesting choke points in query processing. For a benchmark to serve as a basis for comparison, independent implementations must be able to ingest both its data and the scenarios it

DMKG 2026: 2nd International Workshop on Data Management for Knowledge Graphs, October 25, 2026, co-located with ISWC 2026, Bari, Italy

*Corresponding author.
bryanelliott.tam@ugent.be (B. Tam); saleem.muhammd@gmail.com (M. Saleem); ruben.taelman@ugent.be (R. Taelman)
0000-0003-3467-9755 (B. Tam); 0000-0001-9648-5417 (M. Saleem); 0000-0001-5118-256X (R. Taelman)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-09

prescribes, a property we call the interoperability of the benchmark.

The quality of published Resource Description Framework (RDF) data is a studied concern. The literature contains surveys that organize such concerns into taxonomies of quality dimensions, among them syntactic validity [6]. Large-scale empirical studies have in turn measured how far published data deviates from the standards, and cataloged the recurring errors [7, 8]. Existing tooling addresses such defects at authoring time, where a language server flags syntax mistakes before publication [9], or at ingestion time, where tools such as LOD Laundromat [10] republish cleaned copies of existing datasets. The latter proceeds by exclusion, however, keeping only documents whose source URL parses under RFC 3986 [11].

This matters especially for a federated benchmark, whose datasets are distributed across multiple endpoints that may each run a different engine. Such experiments usually compare client-side federation engines, yet the same datasets could equally be used to compare the server-side engines that host them. In either case, every hosting engine must be able to ingest the benchmark's data, which requires that data to be interoperable. Interoperable data also lets the benchmark be embedded in automated processes such as Continuous Integration (CI) pipelines that track an engine's performance over time.

LargeRDFBench [12], which extends the earlier FedBench [13] suite, is a widely used benchmark for SPARQL endpoint federation. More recent benchmarks such as FedShop [14] instead generate synthetic sources to stress-test how federation engines scale with the number of endpoints, a concern orthogonal to the standards-conformance of real, published data that we address here. LargeRDFBench provides 13 real datasets, most of them interlinked, together with 32 queries and their expected results. Several of these datasets, however, deviate from the RDF standard, so an engine that parses RDF strictly cannot ingest them. This is not harmless even for client-side evaluation, since we find the benchmark's expected results to depend on the engine chosen to host the data as well as on the federation semantics the client itself assumes. It also narrows the set of engines able to host the benchmark and rules out any that parse RDF strictly, undermining interoperability and the reproducibility of results using independent implementations. The expected results compound the problem, since they are published in a custom format that must be parsed with custom tooling, even though standard result formats already exist [15]. They also contain discrepancies with respect to the source datasets, introduced by the tooling that produced them.

In this paper, we identify and categorize these standards violations and repair them with a reproducible pipeline, producing a standards-conformant edition of LargeRDFBench whose datasets all load under a strict RDF parser and whose expected results are re-encoded in a standard format, archived on Zenodo so that it can be downloaded directly [16].[1] Running the modernized benchmark end-to-end in a federated engine, we validate the expected results against the source datasets and correct their discrepancies, backing each correction with a reproducible verification script. This validation also reveals a more subtle threat to reproducibility, as the engine hosting the data can itself change the answers the benchmark returns, independently of the client-side federation algorithm under test. It further shows that reproducing such results with an independent implementation is difficult, since FedQPL [17], the reference formalism for automatic source selection, is defined under set semantics and so leaves the real SPARQL execution of automatic source selection ambiguous. We also explore preliminarily whether the underlying engine running a SPARQL endpoint might have an impact, by comparing ASK- and COUNT-based source selection in the FedX algorithm [1], as implemented in the Comunica query engine [18], over endpoints hosted by QLever [19], a comparison not previously explored. Because endpoints implement ASK and COUNT differently, this evaluation raises, without settling, the question of how the hosting engine shapes federated performance, a server-side view that current benchmarking overlooks and that the interoperable benchmark now makes possible.

[1] https://doi.org/10.5281/zenodo.22140177

2. Modernization Process

We develop a pipeline that transforms the datasets of LargeRDFBench into an interoperable edition. The pipeline is open source[2] and fully reproducible, using GNU make[3] as its orchestrator. It first downloads the original datasets, then cleans them with software we developed, which also records the repairs it applies as per-dataset statistics in CSV form. We then use Semantic Operation Pipeline (SOP) [20] to serialize every cleaned dataset as N-Triples, and to confirm that each is syntactically valid. Next, we re-encode the benchmark's expected results in the W3C SPARQL 1.1 Query Results JSON Format with an open-source tool we built for LargeRDFBench[4], applying to them the same repairs as to the datasets so the two stay aligned. We also drop the original results' unbound placeholder, an unbound `OPTIONAL` variable recorded as the literal `'null'`, which a standards-compliant engine never returns, leaving the variable unbound instead. We verify that each cleaned dataset preserves the triple count of its original and that each converted result set preserves its number of solutions, emitting a CSV report of this check. We then convert each dataset into Header information, a Dictionary, and the actual Triples structure (HDT) [21], a compressed RDF format, and run simple `ASK` queries over it with the Comunica SPARQL engine [18], which parses only valid RDF and so gives an engine-level check that the data loads. The pipeline thus yields the standards-conformant datasets, their HDT counterparts, and the expected results in SPARQL Results JSON. To make the interoperable edition directly available to the community, we archive it on Zenodo under a persistent identifier [16]: the repaired datasets, their HDT indexes, the queries and the expected results can be downloaded and used as they are, without running the pipeline. We have additionally opened a pull request contributing these artifacts to the official LargeRDFBench repository.[5]

2.1. Taxonomy of the Repairs

We group the standards violations we found into three classes according to the part of the RDF syntax they breach, and repair each with a deterministic, reproducible pipeline.[6] We present the three classes in turn, giving for each the repairs it requires and a representative example. Table 1 reports how often each repair fired across the benchmark's datasets.

The first class concerns Internationalized Resource Identifiers (IRIs) that violate the IRI syntax of RFC 3987 [23], which builds on the URI generic syntax of RFC 3986 [11]. Characters that an IRI may not contain, such as spaces, control characters and brackets, are percent-encoded; for example, `ATFIP1[V]` becomes `ATFIP1%5BV%5D`. A colon in the authority[7] that is not followed by a decimal port is encoded to `%3A`, since RFC 3986 permits only digits there. A term written without a scheme is made absolute against the dataset's base IRI; for instance, `<bio2rdf_dataset:...>` becomes `<http://bio2rdf_dataset%3A...>`. Leading and trailing spaces are stripped.

The second class concerns string literals that a strict parser rejects. A literal broken across physical lines is rejoined with the breaks escaped as `\n`, since a quoted literal may not contain a raw newline in N-Triples [24]; a value spanning three lines, for example, becomes `"line one\nline two\nline three"`. Turtle [25] does admit raw newlines, but only inside a long string delimited by `"""`, a form N-Triples does not provide, so escaping is required for the serialization we produce. Both serializations also require every document to be encoded in UTF-8 [24, 25], so a value that UTF-8 cannot express cannot appear in a conformant file. A character too large for a single UTF-16 code unit is represented as a surrogate pair, and some literals carry such a character as its two surrogate halves rather than as the character itself. Because UTF-8 cannot encode surrogates and requires them to be decoded to their

[2] https://github.com/shape-federated-queries/large-rdf-bench-modernization
[3] https://www.gnu.org/software/make/
[4] https://github.com/shape-federated-queries/large_rdf_bench_result_json_format
[5] https://github.com/dice-group/LargeRDFBench/pull/4
[6] The pipeline additionally applies two repairs for tooling compatibility rather than standards conformance. It replaces NUL bytes with spaces, which HDT's `rdf2hdt` [22] cannot ingest, and converts single-quoted RDF/XML attributes to double quotes, which some parsers reject. Both are omitted from Table 1.
[7] The authority as defined by the *Uniform Resource Identifier (URI): Generic Syntax* specification [11].

Table 1
Standards-conformance repairs applied to LargeRDFBench, aggregated over its 13 datasets. Counts are per repair type, so a single term may receive several.

Repair	Count	Datasets
IRI		
Illegal characters percent-encoded	93,587	7
Relative CURIE-like terms absolutised	1,145	1
Invalid authority colons encoded	1,147	2
Leading/trailing spaces stripped	76	3
Literal		
Multiline literals joined	10	1
UTF-16 surrogate pairs combined	279	1
Unquoted literals quoted	940	2
Language		
Malformed language tags fixed	9	1

character number first [26], we recombine the halves; the pair `\uD835\uDFB1`, for example, becomes the single character U+1D7B1. A bare alphabetic object, which Turtle reads as the start of a prefixed name [25], is quoted; for example, `t:chromosome X` becomes `t:chromosome "X"`.

The third class concerns language tags that do not conform to BCP 47 [27]. Such a tag is reduced to its primary language subtag; the malformed `fr_1793`, for example, becomes `fr`.

3. Experiment

The primary purpose of this experiment is to validate the integrity of the modernized benchmark.[8] We then illustrate the kind of investigation the now standards-conformant, interoperable benchmark makes possible. As an example, we compare the ASK- and COUNT-based source-selection strategies of the FedX algorithm [1]. FedX determines which endpoints can contribute to a triple pattern by sending each of them an ASK query, which an endpoint may answer as soon as it finds one match. The COUNT variant replaces these ASK queries with COUNT queries, which ask the same question but also return cardinality information that the query planner would otherwise have to estimate. We report *query planning time* and *query execution time* for both variants. Throughout, we refer to queries by their LargeRDFBench identifiers [12], where each query can be consulted in full.

We ran the experiment on 14 machines of the imec Virtual Wall testbed, one for each of the 13 endpoints and one client, communicating over the testbed's dedicated internal network, so the times we report are those of a local-area deployment rather than of the public Web. Every endpoint is served by QLever [19], and the client evaluates the queries with the FedX algorithm as implemented in Comunica [18], under both its ASK- and COUNT-based source selection. Each machine has the same specification: two hexa-core Intel Xeon E5645 (2.4 GHz) CPUs, 24 GB of RAM, and Ubuntu 20.04 LTS 64-bit. We did not run the full query suite, but its simple (S) and complex (C) queries, and allowed each 30 minutes before aborting it.

We corrected a pervasive defect in the original expected results, accented characters recorded as `?` or the replacement character U+FFFD, verified against the source data. With this correction applied, every query our engine answers matches the expected results, except three: S7, C8 and C7. For S7, the triple pattern binding `"California"` is answered by both NYT and GeoNames; the original results record a single solution, whereas our engine returns two. FedQPL [17], the reference formalism for query plans that automatic source selection produces, requires the answer to a query over a federation to be the one obtained by evaluating it over the set union of the data of all federation members [17, Def. 3], that is, over the federation seen as a single knowledge graph, in which the triple asserted by

[8]The experiment is reproducible and open source: https://github.com/shape-federated-queries/ask-count-fedx-large-rdf-bench

Table 2
Median, minimum and maximum per-query planning and execution time on the QLever-hosted federation; the last row is the per-query ASK/COUNT ratio (dimensionless).

	Planning (ms)			Execution (s)		
	Med	Min	Max	Med	Min	Max
ASK	274	149	665	0.6	0.3	1727.6
COUNT	340	226	2183	1.8	0.3	32.8
ASK/COUNT	0.84	0.07	1.04	0.79	0.15	123.62

both sources occurs once. Its operators are correspondingly defined over sets of solution mappings [17, Def. 6], whereas SPARQL operators evaluate under bag semantics, and the formalism proposes no union operator for SPARQL that would bridge the two, so how to reconcile them is not settled. A bag semantics for FedQPL is left to future work, and no published extension develops one. The consequence for reproducibility is significant. No established formalism describes how federated (real) SPARQL queries under automatic source selection are actually executed, so the answers an engine returns cannot be checked against any definition. We therefore report the discrepancy rather than resolve it. For C8, the source data contains the author label both with and without a trailing space (`"Eyal Oren "` and `"Eyal Oren"`), so our engine returns both where the original results kept only the trimmed form. For C7, the difference comes from the hosting engine, not the answer set. QLever canonicalizes numeric literals, casting `xsd:float` and `xsd:double` to `xsd:decimal`. SPARQL evaluates over RDF terms, and the dataset asserts the coordinate under both `xsd:float` and `xsd:double`, whose use in RDF is itself a documented source of data-quality problems [28]. The two are therefore distinct solutions, yet QLever collapses them into a single value absent from the data, returning fewer results than the term-based semantics prescribe. Losing valid solutions this way bears directly on interoperability and reproducibility, as the same benchmark then returns different answers across engines. A proof file for each case is provided in the analysis repository. [9]

Comparing the two source-selection strategies, ASK is faster on the median (Table 2), but only in aggregate. The per-query ASK/COUNT execution ratio ranges from 0.15 to 123.62, so on some queries ASK is instead orders of magnitude slower. This wide spread makes the strategy, and its interaction with the hosting engine, worth investigating, though beyond this paper's scope.

4. Conclusion

Reproducibility is a cornerstone of science, and of benchmarking in particular, yet a benchmark can uphold it only if every conformant implementation is able to run it. Several of LargeRDFBench's datasets deviate from the RDF standard and its expected results are distributed in an ad hoc format, so we repaired both with a deterministic, reproducible pipeline that yields standards-conformant datasets, their HDT counterparts, and the expected results in the W3C SPARQL 1.1 Query Results JSON Format. We also validated end-to-end in a federated engine, correcting encoding discrepancies in the reference along the way.

Because every conformant engine can now host the data, the repaired benchmark supports reproducible cross-engine comparison. Reproducing the expected results nonetheless exposed three obstacles to it, of increasing depth. The first is the ingestion of strings, where, for example, trailing spaces kept by one tool and trimmed by another change which literals exist. This is trivial to resolve once noticed. The second is that an engine may materialize entailments as it ingests the data. Collapsing `xsd:float` and `xsd:double` into `xsd:decimal`, for example, can change the query results, including the multiplicity of the bag of solution mappings. The third is the absence of a fundamental definition, since for federated queries under automatic source selection no formalism describes what a SPARQL engine

[9] https://github.com/shape-federated-queries/large-rdf-bench-result-analysis

should compute. FedQPL [17], which formalizes the plans that automatic source selection produces, defines their execution over sets of solution mappings, whereas SPARQL evaluates under bag semantics. Extending FedQPL, or establishing another formal definition of federated queries with automatic source selection under bag semantics, is in our view an important open problem for reproducible federated evaluation. Our comparison of ASK- and COUNT-based source selection opens a further direction, as the strategy's cost varies significantly across queries and depends on how the hosting engine evaluates ASK and COUNT, a real-world constraint on federation that current benchmarking practice overlooks. We leave a systematic multi-engine evaluation to future work.

Acknowledgments

This research was supported by Serendipity Engine (Research Foundation - Flanders (FWO) grant number S006323N). Ruben Taelman is a postdoctoral fellow of the Research Foundation – Flanders (FWO) (1202124N).

Declaration on Generative AI

During the preparation of this work, the authors used LLM technologies, particularly models from the provider OpenAdapter (such as Mistral Large, Qwen2.5-Coder, Kimi K2, DeepSeek-R1, and MiniMax-M2.5), in order to: Drafting content and Code generation. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] A. Schwarte, P. Haase, K. Hose, R. Schenkel, M. Schmidt, Fedx: Optimization techniques for federated query processing on linked data, in: The Semantic Web – ISWC 2011, volume 7031, 2011, pp. 601–616.

[2] B. Moreau, P. Serrano-Alvarado, Ensuring License Compliance in Linked Data with Query Relaxation, 2021, pp. 97–129.

[3] T. Lubiana, L. Rasberry, D. Mietchen, The wikidata query service split and its impact on the scholarly graph, in: Posters, Demos, Workshops, and Tutorials at the 21st International Conference on Semantic Systems (SEMANTiCS 2025), volume 4064, 2025. URL: https://ceur-ws.org/Vol-4064/PD-paper3.pdf.

[4] M. Baker, 1,500 scientists lift the lid on reproducibility, Nature 533 (2016) 452–454.

[5] C. Collberg, T. A. Proebsting, Repeatability in computer systems research, Communications of the ACM 59 (2016) 62–69.

[6] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, S. Auer, Quality assessment for linked data: A survey: A systematic literature review and conceptual framework, Semantic Web 7 (2015) 63–93. URL: https://journals.sagepub.com/doi/abs/10.3233/SW-150175. arXiv:https://journals.sagepub.com/doi/pdf/10.3233/SW-150175.

[7] A. Hogan, A. Harth, A. Passant, S. Decker, A. Polleres, Weaving the pedantic web, in: LDOW, 2010. URL: https://api.semanticscholar.org/CorpusID:5880345.

[8] A. Hogan, J. Umbrich, A. Harth, R. Cyganiak, A. Polleres, S. Decker, An empirical survey of linked data conformance, Journal of Web Semantics 14 (2012) 14–44.

[9] Vercruysse, Arthur and Rojas Melendez, Julian Andres and Colpaert, Pieter, The semantic web language server : enhancing the developer experience for semantic web practitioners, in: Curry, Edward and Acosta, Maribel and Poveda-Villalón, Maria and van Erp, Marieke and Ojo, Adegboyega and Hose, Katja and Shimizu, Cogan and Lisena, Pasquale (Ed.), The Semantic Web : 22nd European Semantic Web Conference, ESWC 2025, Proceedings, Part II, volume 15719, 2025, pp. 210–225. URL: http://doi.org/10.1007/978-3-031-94578-6_12.

[10] W. Beek, L. Rietveld, H. R. Bazoobandi, J. Wielemaker, S. Schlobach, Lod laundromat: A uniform way of publishing other people's dirty data, in: P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. Knoblock, D. Vrandečić, P. Groth, N. Noy, K. Janowicz, C. Goble (Eds.), The Semantic Web – ISWC 2014, 2014, pp. 213–228.

[11] T. Berners-Lee, R. Fielding, L. Masinter, Uniform Resource Identifier (URI): Generic Syntax, Technical Report 3986, Internet Engineering Task Force (IETF), 2005. URL: https://www.rfc-editor.org/rfc/rfc3986.

[12] M. Saleem, A. Hasnain, A.-C. N. Ngomo, Largerdfbench: A billion triples benchmark for sparql endpoint federation, Journal of Web Semantics 48 (2018) 85–125.

[13] M. Schmidt, O. Görlitz, P. Haase, G. Ladwig, A. Schwarte, T. Tran, Fedbench: A benchmark suite for federated semantic data query processing, in: International Semantic Web Conference, 2011, pp. 585–600.

[14] M.-H. Dang, J. Aimonier-Davat, P. Molli, O. Hartig, H. Skaf-Molli, Y. Le Crom, Fedshop: a benchmark for testing the scalability of sparql federation engines, in: International Semantic Web Conference, 2023, pp. 285–301.

[15] A. Seaborne, SPARQL 1.1 Query Results JSON Format, W3C Recommendation, World Wide Web Consortium (W3C), 2013. URL: https://www.w3.org/TR/sparql11-results-json/.

[16] B.-E. Tam, M. Saleem, R. Taelman, LargeRDFBench interoperable edition: RDF datasets, HDT indexes, queries and expected results, 2026. URL: https://doi.org/10.5281/zenodo.22140177. doi:10.5281/zenodo.22140177.

[17] S. Cheng, O. Hartig, Fedqpl: A language for logical query plans over heterogeneous federations of rdf data sources, in: Proceedings of the 22nd International Conference on Information Integration and Web-Based Applications & Services, iiWAS '20, Association for Computing Machinery, New York, NY, USA, 2021, pp. 436–445. URL: https://doi.org/10.1145/3428757.3429120. doi:10.1145/3428757.3429120.

[18] R. Taelman, J. Van Herwegen, M. Vander Sande, R. Verborgh, Comunica: a modular sparql query engine for the web, in: Proceedings of the 17th International Semantic Web Conference, 2018. URL: https://comunica.github.io/Article-ISWC2018-Resource/.

[19] H. Bast, B. Buchhold, Qlever: A query engine for efficient sparql+ text search, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 647–656.

[20] P.-A. Champin, sop: Semantic operation pipeline, https://github.com/pchampin/sophia-cli, 2026.

[21] J. D. Fernández, M. A. Martínez-Prieto, C. Gutiérrez, A. Polleres, M. Arias, Binary rdf representation for publication and exchange (hdt), Journal of Web Semantics 19 (2013) 22–41. URL: https://www.sciencedirect.com/science/article/pii/S1570826813000036.

[22] rdfhdt, hdt-cpp: C++ implementation of the HDT compression format, https://github.com/rdfhdt/hdt-cpp, 2026.

[23] M. Duerst, M. Suignard, Internationalized Resource Identifiers (IRIs), Technical Report 3987, Internet Engineering Task Force (IETF), 2005. URL: https://www.rfc-editor.org/rfc/rfc3987.

[24] D. Beckett, RDF 1.1 N-Triples, W3C Recommendation, World Wide Web Consortium (W3C), 2014. URL: https://www.w3.org/TR/n-triples/.

[25] D. Beckett, T. Berners-Lee, E. Prud'hommeaux, G. Carothers, RDF 1.1 Turtle: Terse RDF Triple Language, W3C Recommendation, World Wide Web Consortium (W3C), 2014. URL: https://www.w3.org/TR/turtle/.

[26] F. Yergeau, UTF-8, a transformation format of ISO 10646, Technical Report 3629, Internet Engineering Task Force (IETF), 2003. URL: https://www.rfc-editor.org/rfc/rfc3629.

[27] A. Phillips, M. Davis, Tags for Identifying Languages, Technical Report 5646, Internet Engineering Task Force (IETF), 2009. URL: https://www.rfc-editor.org/rfc/rfc5646.

[28] J. M. Keil, M. Gänßinger, The problem with xsd binary floating point datatypes in rdf, in: P. Groth, M.-E. Vidal, F. Suchanek, P. Szekley, P. Kapanipathi, C. Pesquita, H. Skaf-Molli, M. Tamper (Eds.), The Semantic Web, 2022, pp. 165–182.