Base Platform for Knowledge Graphs with Free Software

Simon Bin, Claus Stadler, Norman Radtke, Kurt Junghanns, Sabine Gründer-Fahrer and Michael Martin

Institute for Applied Informatics (InfAI), Leipzig

Abstract

We present an Open Source base platform for the CoyPu knowledge graph project in the resilience domain. We report on our experiences with several tools which are used to create, maintain, serve, view and explore a modular large-scale knowledge graph, as well as the adaptions that were necessary to enable frictionless interaction from both performance and usability perspectives. For this purpose, several adjustments had to be made. We provide a broad view of different programs which are of relevance to this domain. We demonstrate that while it is already possible to achieve good results with free software, there are still several pain points that need to be addressed. Resolution of these issues is often not only a matter of configuration but requires modification of the source code as well.

Keywords

Free software, Architecture, Platform, RDF Knowledge Graphs, Crisis Informatics

1. Introduction

Semantic knowledge graphs (KGs) nowadays not only have become a key asset for search engines but are at the centre of numerous applications, for instance, in data analytics, question answering, recommendation systems, and decision support.

Current interest from research communities as well as industries and administration rests on the capability of KGs to capture comprehensive machine-readable knowledge in application scenarios and their strengths in integrating, managing and exploiting information from heterogeneous data sources at scale. One especially interesting application scenario for KGs is crisis and resilience research. Semantic KGs can play a crucial role in increasing transparency of, for instance, economic value chains and in understanding the complex mechanisms of crisis factors at a global level.

Although methodologies and strategies for building knowledge graphs vary based on the specific conditions and requirements of individual use cases, the creation, utilisation, and maintenance of KGs involve several common necessities and corresponding implementation phases. These processes have been discussed for more than twenty years [1, 2, 3, 4]. While general approaches for KG development and maintenance are proposed, limited guidance regarding the selection of efficient tools for implementing these methodologies is available.

Second International Workshop on Linked Data-driven Resilience Research (D2R2'23) co-located with ESWC 2023, May 28th, 2023, Hersonissos, Greece

sbin@informatik.uni-leipzig.de (S. Bin)

Knowledge graph engineering must address additional challenges in today's rapidly evolving landscape. Firstly, contemporary knowledge graphs should adhere to Linked Open Data requirements, such as the FAIR [5] data principles (Findable, Accessible, Interoperable, and Reusable) and the 5-star Open Data Model.[1] These standards ensure that the resulting knowledge graphs can be easily discovered, understood, and integrated with other datasets, promoting data sharing and collaboration across various domains. Secondly, knowledge graph engineering projects often involve teams with diverse backgrounds and expertise levels in ontology authoring. Consequently, the tools used for KG engineering must cater to both ontology experts and domain-specific specialists, facilitating effective *real-time* collaboration and allowing all team members to contribute meaningfully to the development and maintenance of the knowledge graph.

We aim to create a knowledge graph-based platform for managing and exploring supply chain data. The platform will offer customised views for scientists, engineers, and decision-makers to foster collaboration and interaction with the knowledge graph. Our goal is to enable users to analyse complex data, make informed decisions, and contribute to a resilient economy.

2. Related work

Recent studies [6, 7, 8, 9, 10] have shown that there are many different approaches for building specific KGs. Limitations are often faced regarding the approaches' scalability, metadata and ontology management, entity resolution and fusion, incremental updates, and quality assurance [11].

While certain commercial tools like metaphactory[a] or the Enterprise Knowledge Graph Platform[b] claim to offer comprehensive, all-in-one solutions, free and open-source alternatives often only address specific aspects. Consequently, users seeking to employ free and open-source tools must combine multiple applications and platforms to arrive at a complete knowledge graph platform.

When embarking on the creation of a custom stack for knowledge graph and ontology engineering, a valuable starting point for discovering relevant tools are link collections. Two notable lists in this domain include the Awesome Knowledge Graph[2] and the Awesome Semantic Web[3] repositories on GitHub. However, it is crucial to be aware that despite being curated, these lists may still suffer from link rot, leading to outdated or non-functional resources. The previously popular list hosted on the W3C website[4] is rather outdated, with only 4 out of 21 listed tools still operable at the time of writing in the category of "RDF or OWL browser" (LodView[c], Rhizomer [12], Structured Data Linter, and VocBench [13]). Refer to Table 1 for an overview of the tools mentioned in this paper.

[1] https://5stardata.info/
[2] https://github.com/totogo/awesome-knowledge-graph
[3] https://github.com/semantalytics/awesome-semantic-web
[4] https://www.w3.org/2001/sw/wiki/Category:RDF_or_OWL_Browser

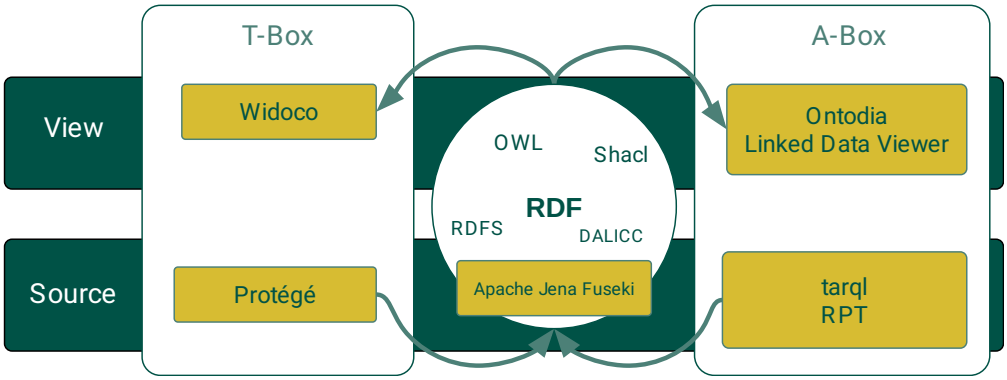

Figure 1: Overview of our solution architecture

3. Solution and other tools

In the following, we will show how we tackle the previous concerns with free and open-source tools. Our solution and the utilised tools are shown in Figure 1.

Ontology authoring. Part of creating knowledge graphs involves creating a schema (ontology) for your data. We used the venerable Protégé [14] for this purpose. We looked but failed to find other comparable OWL[5] authoring tools. While working on the ontology together with the partners, we found the best working approach to discuss one step of ontology evolution in a group and then designate one person to implement the change. Afterwards, the change could be reviewed on a source code control system. For the latter, it should also be ensured that all engineers are using the same version of Protégé, this will ensure that the textual difference of changes to the ontology is minimal and limited to the actual changes implemented.

> The commercial tool TopBraid Composer[d] might provide an alternative, but we did not evaluate it (the free edition would crash when trying to open our ontology). VocBench might also be a candidate (we did not try it.)

To document and visualise our ontologies, we used the Widoco [15] documentation tool in an automated fashion from the command line. Widoco extends LODE and also includes WebVOWL [16], a graph-based visualisation of the ontology document. You can inspect the result on ‹https://schema.coypu.org/›. We have created a *pipeline* that updates our documentation whenever the ontology files are changed in the Git source code repository on GitLab.[6]

[5]Web Ontology Language, https://www.w3.org/TR/owl2-overview/
[6]Source code for the pipeline script on gitlab.com/coypu-project/coy-ontology in the supplements folder

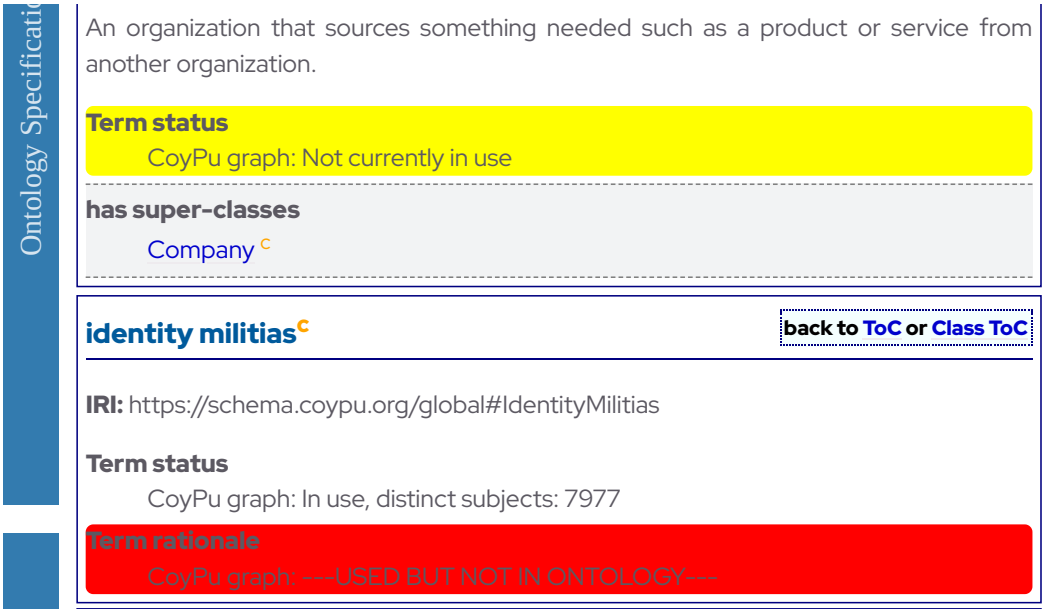

Figure 2: CoyPu Ontology documentation enriched with usage information, hinting at incomplete ontology

> Potential alternatives for Widoco include pyLODE[e] or JOD[f]. In our initial attempt, JOD did not readily handle multiple ontologies and there was no way to reference the entities in the documentation. It was also hiding too much information behind tabs, obstructing the view at a glance. PyLODE did not support our annotations out of the box. These issues might be fixable by spending some time on the templates, however.

The availability of the ontology documentation helps in locating the classes and properties to use when creating the mappings (see Mapping structured data to RDF data). To further enhance the usefulness of our documentation for the modelling workflow, we enrich the ontology documentation with usage statistics from the actual instance data (see Figure 2). In the example, the term "identity militias" has been found in use in our knowledge graph, but not yet described in the ontology, whereas the term "Customer" is contained in the ontology but not (yet) used in the knowledge graph. Calculation of the usage statistics is done using the RDF Processing Toolkit (RPT)[g] with a set of VoID [17]-generating SPARQL queries.[7]

> Automated pipelines that run automatically whenever changes to your ontology are committed to a source code control system, require integration with the source code control system. There are two common ways to realise pipelines. One is to use the platform-specific CI/CD pipelines,[8] the other is using a Webhook event[9] to trigger the start of an external pipeline.

[7] https://gitlab.com/coypu-project/dataset-stats/-/tree/main/compact
[8] Forgejo/Woodpecker: https://woodpecker-ci.org/docs/usage/intro,
GitLab: https://docs.gitlab.com/ee/ci/quick_start/,
GitHub: https://github.blog/2022-02-02-build-ci-cd-pipeline-github-actions-four-steps/

To further increase the quality of our ontology, we have started to implement RDFUnit [18] rules that check the ontology details and version. Another standard with growing adoption is the Shapes Constraint Language (SHACL),[10] which might be validated for example using pySHACL [19] or Jena SHACL[h].

Mapping structured data to RDF data. A big part of our knowledge graph is not our original creation, but rather the mapping and integration, combination, and refinement of existing data sources. Most of the data sources we used are structured data, for example in XML, GML, JSON, CSV formats or data returned by Web APIs. Other parts of the data were also extracted and semantified from unstructured or semi-structured data like news articles.

Several different tools were used by the different project partners, for example Tarql[i], RPT or Morph-KGC [20]. Each tool was chosen by the familiarity of the expert user and applicability to the data source that is to be mapped. Tarql for instance is an excellent choice to map CSV to RDF. RPT on the other hand can easily process CSV, JSON, and XML input files as well as web APIs and remote services using SPARQL, and Morph-KGC can map CSV or connect to SQL databases using RML.

> **i** There is a plenitude of mapping tools that can be found. Other projects that might be worth investigating are SPARQL-Generate [21], RML [22], SPARQL Anything [23].

We found the concise syntax of SPARQL as used in Tarql and RPT to be very refreshing compared to the rather detailed way in which R2RML mapping documents are written. Especially the way to register custom functions in RML is quite heavy-handed, whereas RPT allows defining such functions using either JavaScript or Java annotations.[11] We also tested some commercial web-form based mapping editor, but found it to be rather cumbersome when having to edit and refine the mappings compared to a good old programmers' text editor.

Further tools used by our partners include Named entity recognition from text using Falcon 2.0 [24] as well as many custom-written programs.[12] For some of these, the Python RDFLib[j] was a popular choice when producing RDF data.

Dereference and visualise the Graph. In our experience in the *CoyPu* project, we found it extremely helpful to grasp the Semantic Web concepts when the IRIs are dereferenceable, i.e. you can open the IRIs of your graph entities in the browser. Users and experts alike want to see and understand what is "in" the graph. The *classes and properties* are dereferenceable through the use of Widoco and publishing the ontology documentation at the same location as the Ontology IRI. To make the *data* dereferenceable and browseable, tools like Trifid[k] and LodView can be used. It was easy to start Trifid with the Docker image provided by the authors. However, the default view is a bit bland. There is a great overview of tools presented in [25], but we found that at the time of writing only LodView and Trifid were still available.

[9] https://progrium.github.io/blog/2007/05/03/web-hooks-to-revolutionize-the-web/
[10] https://www.w3.org/TR/shacl/
[11] https://docs.oracle.com/javase/6/docs/technotes/guides/language/annotations.html
[12] https://docs.coypu.org/ExternalRepositories.html

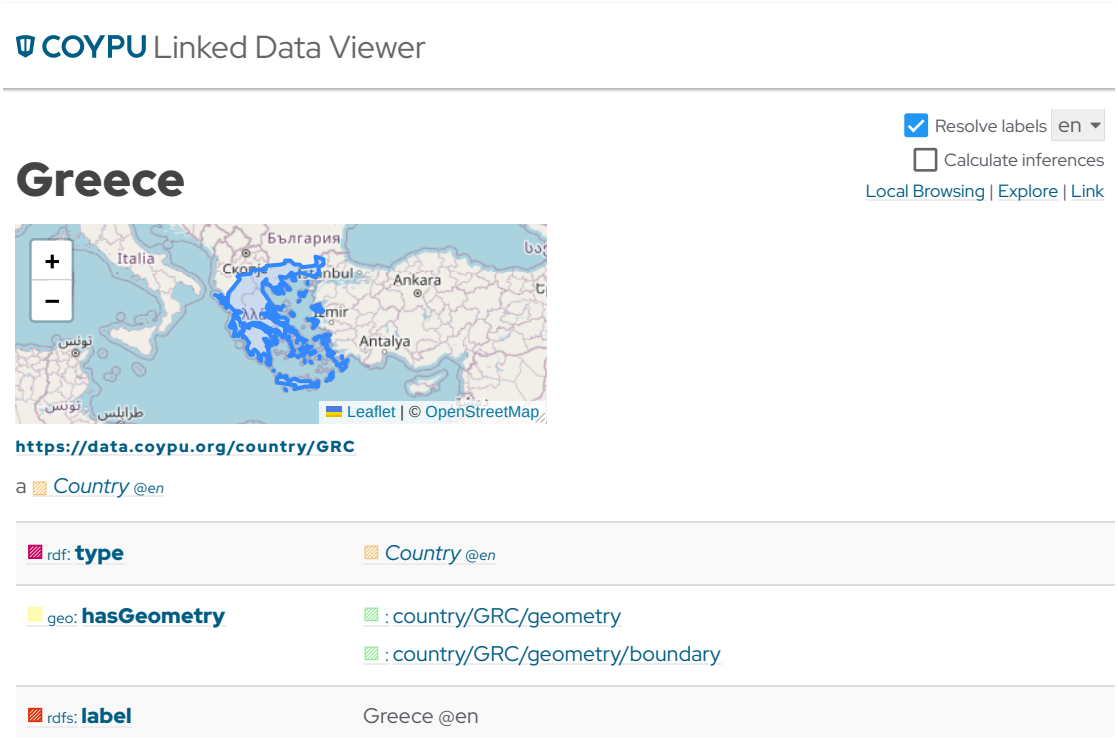

Figure 3: Linked data viewer showing some subject triples

Our solution for IRI dereferencing (see Figure 3) is built on top of the Trifid rendering component[1]. Compared to Trifid, we have extended the code for full client-side querying, a map widget for resources with geospatial data, rendering of inverse relations (similar to LodView), source attribution (useful to know which data set the entity is from), lazy pagination for many property values, label resolution and language switching, colour cards for namespaces and optional inferencing. Furthermore, we support browser-based authentication for password-protected graph databases.

To make the graph explorable and the graph-based nature understandable, we have found the Ontodia Graph Explorer (see Figure 4) [26]. It allows one to interactively add existing entities from the knowledge graph to a drawing area on the screen, and will automatically add the links between entities. It can also show the Ontology schema in a tree fashion and will show a list of instances of the selected ontology class. We refined the graph explorer with a geo-spatial map widget and the possibility to hide obstructive properties. Further, we included meta-data schema and concept schema in the class tree. As a convenience, we have also integrated a link into the Linked Data Viewer which will add the current entity to the graph explorer, and we have made it possible to share created knowledge graph diagrams with other users through custom links.

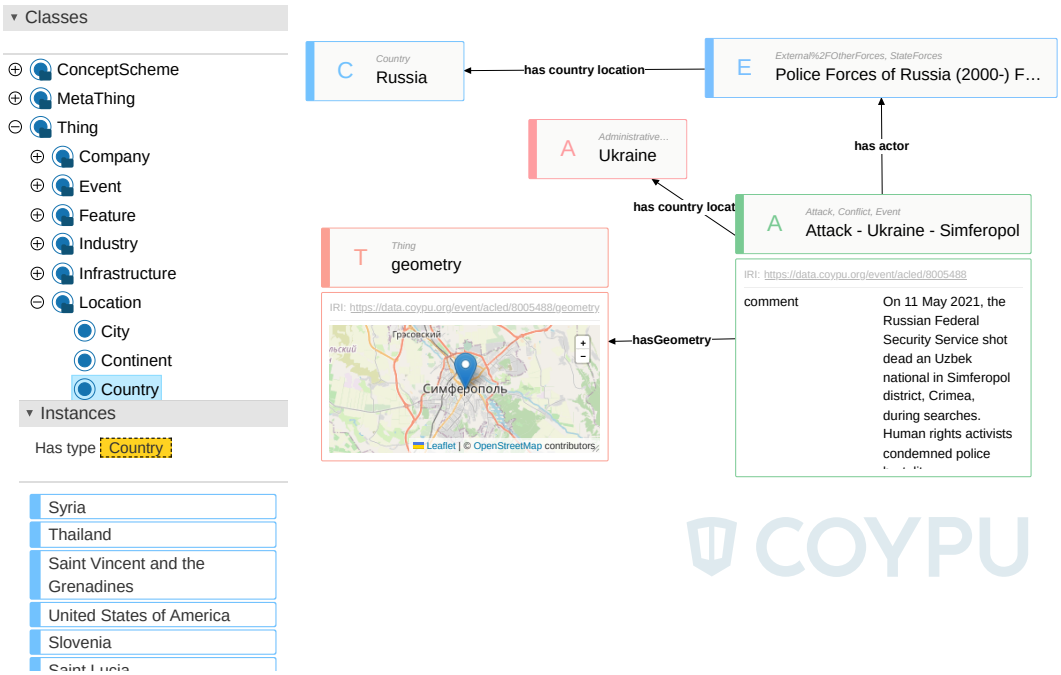

Figure 4: Ontodia Graph Explorer with Map widget

> LODmilla [27] would be another knowledge graph browser, but it does not receive much attention these days. Another tool for exploring knowledge graphs is the RDF surveyor [28]. It extends the Trifid/LodView concept with more exploratory options. (We did not evaluate these tools.)

For interactive testing of SPARQL queries and their results, we found YASGUI [29] and GeoYASGUI [30] very convenient. Thanks to the integration of Google Charts and Openstreetmap[13] map widget, it is possible to present and verify the SPARQL results of geospatial data and numeric data easily.

Documentation and Dataset catalogue. To provide project documentation, we make use of Just the Docs[m]. The documentation can be written in Markdown in our source control system and is then deployed on ‹https://docs.coypu.org/› using a pipeline. The configuration has been extended with a “Try it!” link on SPARQL queries,[14] so that these can be tried out directly on the Jena Fuseki User Interface.

Additionally, a tabular listing of our data catalogue content is published at ‹https://datasets.coypu.org/› by automatically executing SPARQL queries on our knowledge graph and rendering the result to Markdown. This is done by feeding the SPARQL CSV output [31] into `csvlook`[n].

The data catalogue itself is hand-created in RDF/Turtle format and collects all the details about

[13] https://www.openstreetmap.org, implemented with Leaflet: https://leafletjs.com/
[14] https://gitlab.com/coypu-project/skynet/platform-exhibits/-/tree/main/just-the-docs

the externally sourced data that is being used in the *CoyPu* project. This includes licensing information, links to the data source, the target RDF graph where this data set is loaded and a link to the Data Licenses Clearance Center [32][15] (DALICC) licence where available. We plan to record further meta-data into our catalogue, such as the tools involved in transforming the original data into RDF data.

> The DALICC offers a RESTful web service that supports *automated clearance of rights* thus supporting the *legally secure and time-efficient re-utilisation of third-party data sources.* The service is also named DALICC and the source code can be found on GitHub. It is possible to run your own instance of DALICC but some features like the usage of a remote SPARQL endpoint are missing at the moment. We also consider extending the service with a feature that helps to examine the rights and permission of data, derived from multiple datasets (e.g. via SPARQL queries).

Hosting and serving the knowledge graph. We use Apache Jena [33] Fuseki backed by Apache Jena TDB2, both combined making a full-fledged SPARQL Endpoint for scripts and programs to work with. Getting started is mostly straightforward, for a simple deployment you can download the Apache Jena Fuseki release and start it (Java is required).

Extensive customisation has been added to our Jena Fuseki. First of all, we configured geo-spatial[16] and Lucene [34] text index in Fuseki.[17] The endpoint for the Graph Store Protocol[18] was moved from the Fuseki-default to a `/data` sub-path, to stop inadvertent dumps of the full graph. The endpoint has been access-restricted using Apache Shiro that is included in Fuseki[19] (we hope to make it publicly available at a later date).

We identified a performance bottleneck involving SPARQL `FROM` queries and implemented a `FROM`-as-`GRAPH` filter for Jena. Concerning the geospatial data, we added GeoSPARQL functions such as `lat`, `lon` (to access the latitude and longitude of a point), `centroid` and `aggUnion` (from the upcoming GeoSPARQL 1.1 standard [35]) and `simplifyDp` (to simplify a geometry) or `lineMerge`[20] functions, which we have suggested to the Open Geospatial Consortium for consideration. Additionally, the geospatial index was reprogrammed to work on a per-graph level rather than on the whole data set, and we also implemented a Fuseki service to recompute the geo-index of individual graphs at run-time[o]. The initial loading time of the index was also heavily improved due to a custom serialisation format. To aid in RDFS[21] reasoning and `owl:sameAs` [36] resolution, we have implemented two new SPARQL-`SERVICE` based RDFS and sameAs inferencers. They are also conveniently exposed in our linked data viewer (see Dereference and visualise the Graph). These additional functions are implemented in our JenaX[p] extension modules for Apache Jena.

[15] https://www.dalicc.net/
[16] https://jena.apache.org/documentation/geosparql/geosparql-assembler
[17] https://jena.apache.org/documentation/query/text-query.html#text-dataset-assembler
[18] https://www.w3.org/TR/sparql11-http-rdf-update/
[19] https://jena.apache.org/documentation/fuseki2/fuseki-security.html
[20] https://github.com/opengeospatial/ogc-geosparql/issues/402
[21] http://www.w3.org/TR/rdf-schema/

Server Infrastructure. The knowledge graph and the associated processes cannot conveniently operate on free resources. To deploy our knowledge graph, we have a physical server with 250 GiB main memory, of which the Apache Jena Fuseki server is using 50 GiB. According to one of the Jena maintainers,[22] per TDB2 database approx. 2 GiB heap should be assigned (we have configured 10 databases). We have configured the Java heap space to 32 GiB[23] (after having experimented with smaller values, but those were found insufficient when executing larger `CONSTRUCT` or `UPDATE` queries).

The on-disk database files take currently 200 GiB of disk space, which amounts to 700 million ($7 \cdot 10^8$) triples spread across 39 graphs. For the spatial index for GeoSPARQL, 300 MiB of disk space is used for our 4 million ($4 \cdot 10^6$) geometries. The database files are stored on 4 SATA QLC SSDs operating in a software RAID. Fuseki is accessing the database files using memory-mapped I/O. Scanning through the whole index takes 200 seconds and scanning the whole literal table takes 50 minutes.

To create the initial database, we used RPT to convert all the output from our various mapping tools into N-Quads format and then the `xloader` script included in Fuseki. We did this after testing all the various loaders provided by Fuseki and determining that this script works best for our hardware/configuration. Altogether, it takes 8 hours for our system to convert the source RDF files into a Fuseki TDB2 database. The break-down is as follows: 3h 20m for mapping the data, 4h 20m for loading into TDB2, 1 minute to calculate the statistics for the query optimiser, 7 minutes to calculate the geospatial index for GeoSPARQL queries, 20 minutes to calculate a Lucene index for full-text search on the RDFS labels and just short of 1 minute to reload the database. (All times, sizes, and triple counts are approximate.)[24]

4. Conclusion

Many free and open-source software components have been created by the community when it comes to knowledge graph serving and preparation. Nevertheless, it can be challenging to set up the individual software components and configure them in the right way to work together. We had to manually configure nearly all of the components, which requires a diverse skill set ranging from Docker, YAML, JSON, and Bash, to SPARQL, JavaScript, and Java.

The remaining pain points are the long loading times and the query performance for more complex SPARQL queries as well as missed automatic optimisations when using the full-text search. Other issues could be improved, such as the per-graph Geo-index, but could be further enhanced for example by implementing a self-updating geo-index.

We gave impulses to heavily optimise the speed of certain specific queries such as the named graph list[27] and raised other issues as we met them. Other improvements have been contributed

[22] https://lists.apache.org/thread/43vo5pdgfy0mst2pl6ppyyf65bwf7yb2

[23] `java -Xmx` command line flag

[24] An initial stumbling block for us was that the server was originally provided to the project with a ZFS RAID-1 filesystem [37] on Ubuntu. Despite database-specific tuning[25] we were unable to make it work satisfactorily. The system Input-output load became unbearably high so we had to reinstall the system using the ext4[26] filesystem.

[25] https://openzfs.github.io/openzfs-docs/Performance%20and%20Tuning/Workload%20Tuning.html

[26] https://www.kernel.org/doc/Documentation/filesystems/ext4.txt

[27] https://github.com/apache/jena/pull/1655

directly by us, such as speed-ups for certain path query patterns, geospatial queries, or a whole new bulk SPARQL service and cache implementation. Yet other improvements such as the per-graph geo-index are still work-in-progress in our fork of Apache Jena, or part of our JenaX extension modules for Apache Jena. We plan to upstream our changes if possible, and to provide a docker compose file for easy reproducing of the set-up described in this paper.

Acknowledgments

The authors acknowledge the financial support by the Federal Ministry for Economic Affairs and Energy of Germany in the project Coypu (project number 01MK21007[A-L]).

References

[1] M. Gruninger, Methodology for the design and evaluation of ontologies, in: International Joint Conference on Artificial Intelligence, 1995.
[2] N. Noy, Ontology development 101: A guide to creating your first ontology, 2001.
[3] M. C. Suárez-Figueroa, A. Gómez-Pérez, M. Fernández-López, The neon methodology for ontology engineering, in: Ontology Engineering in a Networked World, 2012.
[4] S. Auer, L. Bühmann, C. Dirschl, O. Erling, M. Hausenblas, R. Isele, J. Lehmann, M. Martin, P. Mendes, B. Nuffelen, C. Stadler, S. Tramp, H. Williams, Managing the life-cycle of linked data with the lod2 stack, volume 7650, 2012.
[5] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al., The fair guiding principles for scientific data management and stewardship, Scientific data 3 (2016) 1–9.
[6] A. Hur, N. Janjua, M. Ahmed, A survey on state-of-the-art techniques for knowledge graphs construction and challenges ahead, in: 2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), IEEE, 2021, pp. 99–103.
[7] P. Hitzler, A review of the semantic web field, Commun. ACM 64 (2021) 76–83.
[8] X. Zhu, Z. Li, X. Wang, X. Jiang, P. Sun, X. Wang, Y. Xiao, N. J. Yuan, Multi-Modal Knowledge Graph Construction and Application: A Survey, IEEE Transactions on Knowledge and Data Engineering (2022) 1–20.
[9] V. Ryen, A. Soylu, D. Roman, Building Semantic Knowledge Graphs from (Semi-)Structured Data: A Review, Future Internet 14 (2022) 129.
[10] G. Tamašauskaitė, P. Groth, Defining a Knowledge Graph Development Process Through a Systematic Review, ACM Trans. Softw. Eng. Methodol. 32 (2023) 27:1–27:40.
[11] M. Hofer, D. Obraczka, A. Saeedi, H. Köpcke, E. Rahm, Construction of knowledge graphs: State and challenges, Semantic Web (2023) 1–43.
[12] R. García, J.-M. López-Gil, R. Gil, Rhizomer: Interactive semantic knowledge graphs exploration, SoftwareX 20 (2022) 101235.
[13] A. Stellato, M. Fiorelli, A. Turbati, T. Lorenzetti, W. Van Gemert, D. Dechandon, C. Laaboudi-Spoiden, A. Gerencsér, A. Waniart, E. Costetchi, et al., Vocbench 3: A collaborative semantic web editor for ontologies, thesauri and lexicons, Semantic Web 11 (2020) 855–881.

[14] M. A. Musen, The protégé project: a look back and a look forward, AI matters 1 (2015) 4–12.
[15] D. Garijo, Widoco: a wizard for documenting ontologies, in: The Semantic Web–ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II 16, Springer, 2017, pp. 94–102.
[16] S. Lohmann, V. Link, E. Marbach, S. Negru, Webvowl: Web-based visualization of ontologies, in: Knowledge Engineering and Knowledge Management: EKAW 2014 Satellite Events, VISUAL, EKM1, and ARCOE-Logic, Linköping, Sweden, November 24-28, 2014. Revised Selected Papers. 19, Springer, 2015, pp. 154–158.
[17] K. Alexander, R. Cyganiak, M. Hausenblas, J. Zhao, Describing linked datasets, Linked Data on the Web Workshop (LDOW 09), in conjunction with 18th International World Wide Web Conference (WWW 09) 538 (2009).
[18] D. Kontokostas, P. Westphal, S. Auer, S. Hellmann, J. Lehmann, R. Cornelissen, Databugger: A test-driven framework for debugging the web of data, in: Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion, WWW Companion '14, International World Wide Web Conferences Steering Committee, 2014, pp. 115–118.
[19] A. Sommer, N. Car, pySHACL, 2022. doi:10.5281/zenodo.7059600.
[20] J. Arenas-Guerrero, D. Chaves-Fraga, J. Toledo, M. S. Pérez, O. Corcho, Morph-kgc: Scalable knowledge graph materialization with mapping partitions, Semantic Web (2022).
[21] M. Lefrançois, A. Zimmermann, N. Bakerally, A SPARQL extension for generating RDF from heterogeneous formats, Proc. Extended Semantic Web Conference (ESWC) (2017).
[22] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, R. Van de Walle, Rml: A generic language for integrated rdf mappings of heterogeneous data., Ldow 1184 (2014).
[23] E. Daga, L. Asprino, P. Mulholland, A. Gangemi, Facade-x: an opinionated approach to sparql anything, Studies on the Semantic Web 53 (2021) 58–73.
[24] A. Sakor, K. Singh, A. Patel, M.-E. Vidal, Falcon 2.0: An entity and relation linking tool over wikidata, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 3141–3148.
[25] B. Regalia, K. Janowicz, G. Mai, Phuzzy. link: A sparql-powered client-sided extensible semantic web browser., in: VOILA@ ISWC, 2017, pp. 34–44.
[26] D. Mouromtsev, D. S. Pavlov, Y. Emelyanov, A. V. Morozov, D. S. Razdyakonov, M. Galkin, The simple web-based tool for visualization and sharing of semantic data and ontologies., in: ISWC (Posters & Demos), 2015.
[27] A. Micsik, Z. Tóth, S. Turbucz, Lodmilla: Shared visualization of linked open data, in: Theory and Practice of Digital Libraries–TPDL 2013 Selected Workshops: LCPD 2013, SUEDL 2013, DataCur 2013, Held in Valletta, Malta, September 22-26, 2013. Revised Selected Papers 3, Springer, 2014, pp. 89–100.
[28] G. Vega-Gorgojo, L. Slaughter, B. M. Von Zernichow, N. Nikolov, D. Roman, Linked data exploration with rdf surveyor, IEEE Access 7 (2019) 172199–172213.
[29] L. Rietveld, R. Hoekstra, The yasgui family of sparql clients 1, Semantic Web 8 (2017) 373–383.
[30] W. Beek, E. Folmer, L. Rietveld, J. Walker, Geoyasgui: The geosparql query editor and

result set visualizer, The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 42 (2017) 39.
[31] A. Seaborne, Sparql 1.1 query results CSV and TSV formats. W3C recommendation, 2013.
[32] T. Pellegrini, V. Mireles, S. Steyskal, O. Panasiuk, A. Fensel, S. Kirrane, Automated rights clearance using semantic web technologies: The dalicc framework, Semantic Applications: Methodology, Technology, Corporate Use (2018) 203–218.
[33] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, K. Wilkinson, Jena: implementing the semantic web recommendations, in: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, 2004, pp. 74–83.
[34] A. Białecki, R. Muir, G. Ingersoll, L. Imagination, Apache lucene 4, in: SIGIR 2012 workshop on open source information retrieval, 2012, p. 17.
[35] N. J. Car, T. Homburg, Geosparql 1.1: Motivations, details and applications of the decadal update to the most important geospatial lod standard, ISPRS International Journal of Geo-Information 11 (2022) 117.
[36] B. Motik, Y. Nenov, R. Piro, I. Horrocks, Handling owl:sameAs via rewriting, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 29:1, 2015.
[37] J. Bonwick, M. Ahrens, V. Henson, M. Maybee, M. Shellenbaum, The zettabyte file system, in: Proc. of the 2nd Usenix Conference on File and Storage Technologies, volume 215, 2003.

Table 1 Overview of the tools mentioned in this work; [+]repository updated after May 2022

Ref.	Name	Category	Licence	Repository / Website
	Commercial			
a	metaphactory			metaphacts.com
b	Enterprise Knowledge Graph Platform			eccenca.com
d	TopBraid Composer	Authoring		franz.com/agraph/tbc
	Free software			
[13]+	VocBench	Authoring	BSD-3-Clause	vocbench.uniroma2.it
[14]+	Protégé Version 5.6.1	Authoring	BSD-2-Clause	protege.stanford.edu
[15]+	Widoco Version 1.4.17	Docu.	Apache-2.0	github.com/dgarijo/Widoco
e+	pyLODE	Docu.	GPL-3.0 / BSD-3-Clause	github.com/rdflib/pyLODE
f+	JOD	Docu.	MIT	github.com/eccenca/jod
m+	Just the Docs	Docu.	MIT	github.com/just-the-docs/just-the-docs
n+	csvkit / csvlook	Docu.	MIT	csvkit.readthedocs.io
g+	RDF Processing Toolkit	Mapping+	Apache-2.0	github.com/SmartDataAnalytics/RdfProcessingToolkit
i	Tarql	Mapping	BSD-2-Clause	tarql.github.io
[20]+	Morph-KGC	Mapping	Apache-2.0	morph-kgc.readthedocs.io
[21]+	SPARQL-Generate	Mapping	Apache-2.0	ci.mines-stetienne.fr/sparql-generate
[22]+	RML	Mapping	MIT	rml.io
[23]+	SPARQL Anything	Mapping	Apache-2.0	sparql-anything.cc
[24]	Falcon Version 2.0	NER	MIT	labs.tib.eu/falcon/falcon2
[18]+	RDFUnit	Quality	Apache-2.0	github.com/AKSW/RDFUnit
[19]+	pySHACL	Quality	Apache-2.0	github.com/RDFLib/pySHACL
h+	Jena SHACL	Quality	Apache-2.0	jena.apache.org/documentation/shacl/
c+	LodView	Deref.	MIT	github.com/LodLive/LodView
k+	Trifid	Deref.	Apache-2.0	github.com/zazuko/trifid
l+	Linked Data Viewer	Deref.	MIT	github.com/AKSW/Linked-Data-Viewer
[12]+	Rhizomer	Explore	GPL-3.0	rhizomik.net
[26]	Ontodia Graph Explorer	Explore	LGPL-2.1+ changes on	github.com/zazuko/graph-explorer https://github.com/AKSW/graph-explorer
[27]	LODmilla	Explore	Apache-2.0	lodmilla.sztaki.hu github.com/dsd-sztaki-hu
[28]	RDF surveyor	Explore	Apache-2.0	tools.sirius-labs.no/rdfsurveyor github.com/guiveg/rdfsurveyor
[29]+	YASGUI	Explore	MIT	yasgui.triply.cc
[30]	GeoYASGUI	Explore	MIT	github.com/Triply-Dev/YASGUI.YASR-deprecated/tree/geo
j+	RDFLib Version 6.2.0		BSD-3-Clause	rdflib.readthedocs.org
[33]+	Apache Jena Version 4.8.0		Apache-2.0 work-in-progress on	jena.apache.org https://github.com/AKSW/jena
o+	Geo-Service for Jena Fuseki		Apache-2.0	github.com/AKSW/fuseki-mods
p+	JenaX		Apache-2.0	github.com/Scaseco/jenax