Knowledge Graph (R)Evolution and the Web of Data

Katja Hose

Aalborg University, Denmark
khose@cs.aau.dk

Abstract. When querying knowledge over the Web we typically consider the Web of Data to be a static point of reference that is always available and that never changes. However, when actually running queries "in the wild", we encounter a broad range of problems; spanning from the (un)availability of entire knowledge graphs (and their SPARQL endpoints) to outdated references between knowledge graphs and beyond that, we are almost entirely missing out on the availability of previous versions of knowledge graphs and provenance metadata about them. This position paper discusses these issues in context and sketches some of the solutions to mitigate them. In particular, this paper first discusses approaches to keep knowledge graphs available for continuous and scalable querying and afterwards presents an approach that enables community-driven updates so that mistakes can be corrected or missing information can be added. Then, the paper highlights what we can learn from RDF archiving solutions to better support evolving knowledge graphs. And finally, the paper puts these aspects into perspective and provides an outlook to open challenges and future work.

1 Introduction and motivation

The Web of Data provides access to vast amounts of (semi-)structured data. Building upon Semantic Web standards and Linked Open Data principles [6], the Linked Open Data Cloud[1] is continuously growing and consists of a multitude of sources providing access to knowledge graphs (encoded in RDF) from very diverse domains spanning government, geography, life sciences, linguistics, media, cross-domain, publications, social networking, and user-generated data. To enable efficient query answering over these sources, providers maintain SPARQL endpoints (typically one per knowledge graph), which receive SPARQL queries as input, evaluate them over the local knowledge graph, and return answers to the queries as output. To exploit the full potential of this wealth of publicly available information, it is often necessary to answer queries over the combined data of multiple sources while exploiting the links between them (federated query processing).

However, an essential shortcoming of state-of-the-art approaches and solutions is that knowledge graphs, and along with them the Web of Data, are considered to be mostly static, always available, and rarely evolving. In reality though, the knowledge is subject to constant and continuous evolution; more information becomes available (advances

[1] https://lod-cloud.net/

in science, improved information extraction, new domains, etc.), some knowledge disappears (outdated information, knowledge provider no longer offers the service, etc.), erroneous information is corrected (or not), links and identifiers are changing, systems crash or become temporarily overloaded and unavailable, etc.

Obviously, there are plenty of different aspects, research questions, and proposed solutions in this context that are worth pursuing. This paper focuses on a subset of them and briefly highlights a few particular fundamental research questions and solutions that we have been working on in my lab. These questions are:

- How can we reduce the query load at an endpoint to keep it responsive and available? (Section 2)
- How can we keep knowledge graphs available despite the original sources becoming unavailable? (Section 3)
- How can we correct erroneous information in a community-driven and decentralized way? (Section 4)
- How can we support an evolving Web of Data? (Section 5)

These are the same questions that I discussed in my keynote titled "How can we fix the Web of Data?" at the MEPDaW Workshop 2021. To conclude, Section 6 puts these questions into perspective and provides an outlook to open challenges and future work.

2 Load balancing and endpoint availability

One of the frequently encountered issues that we experience when processing queries on the Web of Data is that response times heavily depend on the resources currently available at the source that hosts the queried knowledge graph. Typically, the source offers access via a SPARQL endpoint that is capable of executing complex and arbitrary SPARQL queries. Obviously, this can easily become a bottleneck; when expensive queries are running concurrently, there are only little resources left to answer other queries. As a consequence, response time and throughput suffer – and in the worst case – the endpoint might become so overloaded that it crashes and becomes unavailable until the provider notices and fixes the issue.

One way to counteract this issue is to reduce the load at the server and instead push some of it to the client that issued the query. Over the years, several approaches have been proposed that are trying to find a trade-off between the extremes of (a) having only downloadable dumps and doing everything on the client and (b) SPARQL endpoints doing everything on the server. Key methods cover (i) partitioning and compression of the data on the server so that smaller amounts of data have to be accessed and transferred, (ii) preemption [12] so that not all the results for a query are computed but only a subset unless the client asks for more results, and (iii) restricting the range of supported queries on the server to those that can be executed efficiently, e.g., triple patterns [21] or star patterns [1].

WiseKG [5] is the latest approach in this field, builds upon these key methods, and is able to dynamically allocate query processing tasks to server or client depending on the current query load on the server, i.e., if the server is overloaded, more tasks are pushed to the clients that issued the queries, which then have to download portions of the data and compute joins locally. For this purpose, the server partitions the data into smaller partitions based on star patterns and similar predicates so that all subgraphs matching similar star patterns are grouped together and compressed using HDT [7]. The server optimizes a query by decomposing it into star-pattern-based subqueries, optimizes the join order, and decides which subqueries should be executed at the server and which ones at the client. While there is still room for optimization, e.g., cost model, query decomposition, data partitioning, statistics, etc., experimental results have shown that especially the dynamic workload-aware shifting of tasks from server to client increases WiseKG's throughput in comparison to other approaches.

Until just a few years ago, the standard approach for querying remote knowledge graphs used to be a SPARQL endpoint. Today, the landscape has broadened up; we no longer have only endpoints but also other approaches including TPF [21] and WiseKG [5]. All of these have different strengths and weaknesses that also depend on the type of query that we want to execute [14]. To optimize and efficiently process queries in this heterogeneous landscape [13], future work needs to develop efficient query processing strategies over these heterogeneous interfaces that can exploit their strengths and avoid their weakness.

3 Keeping knowledge graphs available despite failing original sources

Another reason why query processing on the Web of Data is often not a reliable service is that it totally relies on the services offered by the data providers: Web interfaces with downloadable dumps, SPARQL endpoints, dereferenceable URIs/IRIs, etc. Studies [20] and monitoring services[2] have shown that these Web interfaces, especially SPARQL endpoints, are often not available, i.e., there is no guarantee that the data or query interface necessary to answer a query is actually available when needed. The reasons for this are manifold; for instance, as discussed in the previous section, an endpoint might be overloaded and become unresponsive. Sometimes endpoint crashes are not detected for longer periods of time and sometimes endpoints are permanently taken offline, e.g., when the grant that funded an academic project ended. The underlying issue is that maintaining a SPARQL endpoint requires considerable resources in terms of hardware and computing power but also in terms of human resources and server administration and maintenance. And since knowledge graphs on the Web typically come as (Linked) Open Data providing the service for the public does not generate any financial income.

To keep the information available, we can rely on file sharing principles and P2P systems. In particular, unstructured P2P systems are an interesting foundation where – instead of servers representing endpoints – we have servers representing independent clients (aka peers or nodes), each sharing some own data as well as copies of data from other peers. One such system designed for sharing knowledge graphs, PIQNIC [2], first splits a large dataset into smaller fragments, for example by predicates of the triples, that are easier to share and process. To ensure that the data remains available, the fragments are replicated at multiple peers so that a certain number of copies of each dataset are available in the network. Since there is no global knowledge in P2P systems, each peer only knows a couple of other peers (neighbors); to keep such a network of peers stable

[2] http://www.lodhub.aau.dk/

and connected in the presence of peers joining and leaving the network, peers regularly exchange information about their neighbors and update their connections. It might, for instance, be advantageous to have a direct connection to a peer that has related data, e.g., in the sense of "joinability" so that the datasets of the peers can be joined in a query to produce results.

For query processing and optimization, it is of course beneficial to have access to statistics about datasets and information about which datasets are available at which peer. Each peer then maintains an index containing such information about its neighbors, i.e., it captures which fragments (predicates) of a dataset are stored at which neighbors and what URIs/IRIs these fragments contain [3]. Such indexes can, for instance, be based on Bloom Filters and used to estimate the size of a join result based on the degree of overlapping bits representing URIs/IRIs in the fragments. They can also be used to decide whether a pair of fragments can produce join results at all so that the join can be entirely pruned from the query execution plan.

Evaluation results show that such kind of indexes capturing not only fragments but also contained URIs/IRIs substantially increase query performance. And if we replicate all fragments at ca. 5% of the peers in the network, then we can lose more than 50% of the peers until we start even noticing the effect in result completeness. Of course, query execution time and throughput also benefit from a higher degree of replication and locality.

There are still many challenges to explore in future work, incl. complex queries, alternative types of index structures, and forms of partitioning and allocation. In particular, workload-aware approaches that can help tune the system for a particular query workload are interesting and promising areas of future work.

4 Facilitating community-driven updates

Another fundamental issue we encounter often is that some knowledge graphs are not up-to-date; they (i) contain erroneous information, e.g., due to an error in the information extraction pipeline, (ii) contain outdated information, e.g., the president of a country changes once in a while or links to other knowledge graphs might no longer be accurate if the other side changed the URIs/IRIs, or (iii) lack important information, e.g., links to other sources or the nickname, nationality, etc. of a person.

With the current architecture of the Web of Data, there is no way for consumers to update a knowledge graph other than contacting the provider hoping that the feedback will be taken into account. This, however, is a difficult and tedious endeavor. Hence, we proposed a community-driven architecture and methods giving communities of users the opportunity to maintain and update knowledge graphs: ColChain [4] is a system that builds upon P2P architectures (as mentioned above) and exploits blockchain technology to facilitate community-driven updates while keeping track of and enabling query access to older versions of a knowledge graph.

So, the core idea is to define communities of users and peers that can propose updates to a knowledge graph and together decide, e.g., by voting using a majority consensus protocol, whether to accept an update or not. If an update is accepted, it needs to be propagated throughout the network to update all other copies. During query processing, an older copy of a knowledge graph can be recovered by going through the materialized version of the knowledge graph and the change sets of the updates. In this sense, it is possible to support "time-travel queries" by computing the result for a given query based on the state of a set of knowledge graphs at a particular point in time.

Our experimental results show that there is not much overhead for query processing that is caused by extending the basic P2P approach with the blockchain and community functionality. Since only the latest version of a knowledge graph is materialized, answering queries accessing older versions takes a bit longer since the required versions of the knowledge graph are created on the fly.

Future work will consider alternative ways of defining communities and voting strategies, e.g., not giving all peers/users an equal weight in the voting process. In particular, individual update regimes per knowledge graph might be required since some publishers might wish to retain some special rights, such as a veto for updates. Of course, another important area of future work is to consider materializing multiple versions of a knowledge graph to improve query performance (see also Section 5).

5 Supporting an evolving Web of Data

Current approaches for managing and processing RDF knowledge graphs typically optimize for one of two extremes: either the data is considered to be entirely static or it comes in a stream setting. There is not much work on use cases in between that could support evolving knowledge graphs with arbitrary (slow) rates of updates. Well-known examples of such evolving knowledge graphs are DBpedia, YAGO, and Wikidata; over the years, they grew by capturing more knowledge, there were changes to the schemas and ontologies, URI/IRI naming schemes have changed, etc.

One of the basic challenges is to define objective measures that capture the characteristics of the evolution over time [8, 16]. Proposed measures compare consecutive pairs of revisions and analyze the differences between them. Such measures range from low-level measures (incl. growth, additions, deletions of triples and vocabulary elements) to high-level measures (incl. affected entities, types, literals, ontologies) trying to capture the semantics and provide deeper insights of what the changes mean.

Another challenge is to support multiple versions of a knowledge graph, which is the goal of so-called RDF archiving systems [16]. These systems are not necessarily part of the Web of Data (although they could be used as backends for SPARQL endpoints) but approach the problem from a different angle and support functionalities that standard approaches for knowledge graph management and querying do not consider. The straightforward case are queries against a single (current, past) version of a knowledge graph. But there are other types of interesting queries, e.g., in which revisions of a knowledge graph does a query produce results, in which revisions were certain entities added or deleted, or in which version was there a relationship between two entities or was there ever. These types of queries obviously require comparisons between multiple revisions, which the research community has not yet much paid much attention to.

Existing RDF archiving systems make some basic design choices, e.g., storing independent copies per revision, storing snapshots and delta chains, or tagging triples with timestamps. Some systems then make use of metadata encoding to capture which triples belong to which revision – this ranges from reification to using provenance ontologies (PROV-O) to capture arbitrary types of metadata. Another common way is to repurpose named graphs to model the revisions so that triples become quads where the fourth column can be used to tag the revision without "breaking" the triple format. Other approaches simply extend the layout and add more columns. There are also other important design choices, which involve whether multiple graphs are supported within the same system, or whether concurrent updates are important.

To truly support (slowly) evolving knowledge graphs on the Web, future work has to design systems that can serve as backends to access the Web of Data by combining the obtained insights from triple stores and RDF archives. As a first step, the evolution measures can be used to identify evolution patterns that can then be used as guidance to choose and design an appropriate data layout and data structures [15, 18, 19]. Afterwards, we need to develop query optimizers and efficient query processing strategies for the special types of queries in this setup.

6 Conclusion

This position paper is centered around the question of how to mitigate the current challenges of the Web of Data and in particular those caused by its evolving knowledge graphs. Several challenges in this context have been discussed in more detail: (i) improving the availability of endpoints by better sharing load between server and client, (ii) lowering the burden of and the dependence on the data provider by introducing a P2P-style system using replication, (iii) supporting community-driven updates in a decentralized setup and enabling knowledge graph evolution by introducing blockchain principles, and (iv) enabling a broader range of functionality over evolving knowledge graphs by fusing advances in RDF archiving systems and efficient data management and query processing in triple stores.

To truly support knowledge graph evolution, there are plenty of open research challenges within these areas that researchers are only just beginning to explore, e.g., supporting different types of time-travel queries, scalable query processing and optimization over heterogeneous RDF interfaces, incorporating different standards of encoding knowledge graphs and queries (property graphs vs. RDF, Cypher vs. SPARQL) but also recent developments on SHACL/SheX constraints and shapes in general [17]. Moreover, to help users understand and exploit the information contained in complex knowledge graphs, we need to develop efficient ways for graph exploration [11] and extend them to support evolving graphs. And then, there is also the issue of trust and how to reassure a user that systems and answers to queries can be trusted. A foundation of trust in this sense is provenance and metadata management, which covers not only approaches to capture metadata about triples and knowledge graphs, such as reification and RDF-star [9], but also approaches to capture workflow provenance and lineage to explain how knowledge graphs were generated, processed, and integrated, and finally also approaches to provide explanations on how the system arrived at a certain answer to a query [10].

Acknowledgements. This research was partially funded by the Danish Council for Independent Research (DFF) under grant agreement no. DFF-8048-00051B and the Poul Due Jensen Foundation.

References

1. Aebeloe, C., Keles, I., Montoya, G., Hose, K.: Star Pattern Fragments: Accessing Knowledge Graphs through Star Patterns. CoRR **abs/2002.09172** (2020)
2. Aebeloe, C., Montoya, G., Hose, K.: A Decentralized Architecture for Sharing and Querying Semantic Data. In: ESWC. Lecture Notes in Computer Science, vol. 11503, pp. 3–18. Springer (2019)
3. Aebeloe, C., Montoya, G., Hose, K.: Decentralized Indexing over a Network of RDF Peers. In: ISWC. Lecture Notes in Computer Science, vol. 11778, pp. 3–20. Springer (2019)
4. Aebeloe, C., Montoya, G., Hose, K.: ColChain: Collaborative Linked Data Networks. In: WWW. pp. 1385–1396. ACM / IW3C2 (2021)
5. Azzam, A., Aebeloe, C., Montoya, G., Keles, I., Polleres, A., Hose, K.: WiseKG: Balanced Access to Web Knowledge Graphs. In: WWW. pp. 1422–1434 (2021)
6. Berners-Lee, T.: Linked Data Design Issues, http://www.w3.org/DesignIssues/LinkedData.html
7. Fernández, J.D., Martínez-Prieto, M.A., Gutiérrez, C., Polleres, A., Arias, M.: Binary RDF representation for publication and exchange (HDT). J. Web Semant. **19**, 22–41 (2013)
8. Fernández, J.D., Umbrich, J., Polleres, A., Knuth, M.: Evaluating query and storage strategies for RDF archives. Semantic Web **10**(2), 247–291 (2019)
9. Hartig, O.: Foundations of RDF★ and SPARQL★: An Alternative Approach to Statement-Level Metadata in RDF. In: AMW (2017)
10. Hernández, D., Galárraga, L., Hose, K.: Computing How-Provenance for SPARQL Queries via Query Rewriting. Proc. VLDB Endow. **14**(13), 3389–3401 (2021)
11. Lissandrini, M., Mottin, D., Hose, K., Pedersen, T.B.: Knowledge Graph Exploration Systems: are we lost? In: CIDR (2022)
12. Minier, T., Skaf-Molli, H., Molli, P.: SaGe: Web Preemption for Public SPARQL Query Services. In: WWW. pp. 1268–1278. ACM (2019)
13. Montoya, G., Aebeloe, C., Hose, K.: Towards Efficient Query Processing over Heterogeneous RDF Interfaces. In: ISWC (Best Workshop Papers). Studies on the Semantic Web, vol. 36, pp. 39–53. IOS Press (2018)
14. Montoya, G., Keles, I., Hose, K.: Analysis of the Effect of Query Shapes on Performance over LDF Interfaces. In: QuWeDa@ISWC. CEUR Workshop Proceedings, vol. 2496, pp. 51–66. CEUR-WS.org (2019)
15. Pelgrin, O., Galárraga, L., Hose, K.: Efficient In-memory Indexing for Metadata-augmented RDF. In: MEPDaW@ISWC. CEUR Workshop Proceedings, CEUR-WS.org (2021)
16. Pelgrin, O., Galárraga, L., Hose, K.: Towards fully-fledged archiving for RDF datasets. Semantic Web **12**(6), 903–925 (2021)
17. Rabbani, K., Lissandrini, M., Hose, K.: Optimizing SPARQL Queries using Shape Statistics. In: EDBT. pp. 505–510. OpenProceedings.org (2021)
18. Sagi, T., Lissandrini, M., Pedersen, T.B., Hose, K.: A Design Space for RDF Data Representations. The VLDB Journal (2022)
19. Taelman, R., Mahieu, T., Vanbrabant, M., Verborgh, R.: Optimizing storage of RDF archives using bidirectional delta chains. Semantic Web (2021)
20. Vandenbussche, P., Umbrich, J., Matteis, L., Hogan, A., Aranda, C.B.: SPARQLES: Monitoring public SPARQL endpoints. Semantic Web **8**(6), 1049–1065 (2017)
21. Verborgh, R., Sande, M.V., Hartig, O., Herwegen, J.V., Vocht, L.D., Meester, B.D., Haesendonck, G., Colpaert, P.: Triple Pattern Fragments: A low-cost knowledge graph interface for the Web. J. Web Semant. **37-38**, 184–206 (2016)