Querying Distributed Heterogeneous Linked Data Interfaces through Reasoning

Joachim Van Herwegen

Supervised by Ruben Verborgh and Erik Mannens
Data Science Lab (Ghent University - iMinds)
Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
joachim.vanherwegen@ugent.be

Abstract. Linked Data can be distributed through multiple interfaces on the Web, each of them with their own expressivity. However, there is no generic client available that can handle querying over multiple heterogeneous interfaces. This increases the complexity of combining datasets and designing new interfaces with traditional approaches. Rule-based reasoning is going to be explored to combine different interfaces without intervention of a human developer. Using an iterative approach to extend Linked Data interfaces, I will evaluate different querying set-ups for the SPARQL language on performance, completeness and correctness. Preliminary results look promising: preparatory research on query optimization using light-weight Linked Data interfaces has shown there is still room for optimization. I aim to design a generic Linked Data querying engine, capable of handling different interfaces, that can easily be extended. As my PhD is still in an early phase, I hope to narrow the scope in the next months, based on feedback of the Doctoral Consortium.

1 Problem Statement

Nowadays there are multiple ways to access Linked Data. SPARQL endpoints and data dumps are the most popular methods [10] but alternative options such as Triple Pattern Fragments [30] are gaining in popularity. These options do not use different querying methods besides SPARQL, resulting in diverging access interfaces. Not all SPARQL endpoints support the full range of SPARQL functionality [9], meaning even a set of SPARQL endpoints can already have heterogeneous interfaces. This complicates querying datasets: different client implementations are necessary to query the full spectrum of available data.

For my PhD, I consider using rule-based RDF reasoning to solve this problem. Reasoning is an important component of the Semantic Web: it allows us to draw *conclusions* from the available *data*. In the case of executing Linked Data queries the *data* is the actual query that needs to be executed, together with intermediate results and possibly additional metadata provided by the endpoints. The *conclusion* is then how to answer this query using the available methods, meaning reasoning can be used to solve these queries by leveraging all its advantages such as pattern matching and ontology comprehension.

2 Relevancy

Having heterogeneous Web APIs has always been a problem of the Semantic Web [29]. The diversity in interfaces makes it hard to create the originally envisioned agents [7] capable of interfacing with this Web of knowledge. Querying Linked Data is now starting to face a similar problem with an increase of available interfaces [9,10,30]. For Web APIs, attempts have been made to solve this problem by introducing generic description languages [27]. However, for Linked Data query interfaces there is no similar technology. This is not helped by the fact that new interfaces are still being developed [24, 26]. A generic, interface-independent solution to this problem is thus required so the Linked Data is not splintered over the Web.

3 Related Work

To create the proposed system I will make use of the extensive amount of research that has already been done in multiple fields, which I categorize in Linked Data Fragments, SPARQL querying, federated SPARQL and rule-based reasoning.

3.1 Linked Data Fragments

The main goal behind Linked Data Fragments [30] is to provide a uniform way to talk about the available Linked Data interfaces. Specifically, it suggests that there are many possible interfaces with varying complexity on either the server or client side between a simple data dump and a fully fledged SPARQL endpoint, thus providing a base for the work required.

One additional contribution was the Triple Pattern Fragments (TPF) [30] interface. This interface aims to reduce the server load of executing SPARQL queries by only supporting single triple pattern requests and offloading the remaining execution complexity to the client.

Several extensions have already been suggested for the TPF, two of which I contributed to [24, 26], again moving some complexity back to the server without having too much impact on the performance. This effectively increases the number of available interfaces to access Linked Data, making even more apparent the requirement for a generic method to query these.

3.2 SPARQL

SPARQL [11] is the current standard for querying RDF [13] data. It allows the user to request Linked Data results through SPARQL endpoints. Although the SPARQL functionality is described in the specification, it is not fully supported by all of these endpoints [9].

Much research has already been done on how to execute queries efficiently. This can be a complex problem depending on the operators used in the query [17]. Several optimizations have already been proposed, such as focusing on rewriting

SPARQL algebra [21], ordering operations based on selectivity [23] or creating an optimization model for the entire SPARQL execution process [12].

For my suggested approach, I will base the query execution on this already existing research. Many of these optimizations are dataset independent and focus solely on the query composition itself, allowing for re-use in any SPARQL query engine.

3.3 Federated SPARQL

Federated querying aims to tackle the problem of having to use multiple data sources to answer a (SPARQL) query. Two major components of this are source discovery and source selection [14]. For the work described in this paper, source selection is the most important part, on which a lot of research has already been done [18]. This is a complex issue with a lot of separate components that can be optimized [19].

3.4 Rule-based Reasoning

Notation3 (N3) [6] is an extension of the RDF model. It offers several additional features such as the possibility to refer to graphs, write logical rules that can be used for reasoning, and express proofs of these reasoning results. There are several reasoners supporting N3, with the main ones being cwm [5] and EYE [28].

An important part of reasoning over Linked Data are the OWL ontologies [4], which allow for a description of conclusions that can be drawn from Linked Data. Traditionally, the reasoning over OWL ontologies is done by Description Logic based reasoners using the tableaux algorithm [22]. However, one of the OWL profiles [16], OWL-RL, was designed to support rule-based reasoners in coping with OWL ontologies. The main reasons I choose for rule-based reasoning are performance [3] and the proof generation done by N3 reasoners, since I propose to base query plans on the generated proofs.

4 Research Questions

My main research question is:

- *How can reasoning improve query plans for accessing distributed heterogeneous Linked Data interfaces?*

Important corresponding sub-questions are:

- *How can we describe* SPARQL *queries and* LDF *endpoints formally in* RDF*?*
- *How can we execute and optimize* SPARQL *queries using these descriptions?*
- *How can we extend this process to federated querying?*

5 Hypotheses

My hypotheses are the following:

- *The decision generated by the reasoner corresponds to the valid query plan and the proof is an explanation of why the plan was chosen.*
- SPARQL *queries can be described with an* RDF *representation of the corresponding* SPARQL *algebra. An ontology can be made to describe the expressiveness of common Linked Data interfaces.*
- *The proposed system will provide both complete and correct results.*
- *Existing research in source selection for federated querying can be applied to support multiple data sources while maintaining comparable execution times to existing frameworks.*

6 Preliminary results

A sub-problem of querying heterogeneous interfaces is executing a SPARQL query on a TPF platform: multiple calls are necessary to fully execute the query and performance can differ greatly based on the chosen plan [30]. There have already been several publications providing optimized solutions for this problem [1, 25], indicating that there is definitely room for optimization. In my paper [25] I focused on how the number of requests can be optimized and how TPF data can be joined more efficiently to increase query answering performance. Many of the underlying ideas can be reused in the framework suggested here.

Related to that is using reasoning to combine multiple services to produce a requested result as described for the RESTdesc structure [27]. This is a framework that formally describes web APIs and uses reasoning to create workflows through these APIs. Results there have shown that this algorithm can scale for many services and proof lengths, meaning that the proposed system could handle larger queries and datasets.

7 Approach

I propose an iterative approach to tackle this problem, which will start out with a single interface such as TPF and only the base features of a SPARQL query, such as BGP matching. This reduces the initial complexity of the problem but still requires several necessary components to be built.

SPARQL query description Before any reasoning can be done, the reasoner has to understand the query. This means it needs an RDF representation of the actual query. There already are ontologies supporting this, such as SPIN. These focus on the actual SPARQL representation though. I propose to work with SPARQL algebra instead. The algebra breaks a query down to its base components, that can then be assigned to interfaces able to solve them, and to represent that algebra in RDF.

Interface description To reason over heterogeneous interfaces there needs to be a way to describe their functionality: the reasoner has to understand which parts of a query this specific interface can answer. This needs to be done in such a way that the functionality can be matched with query components as described above. I am currently investigating the Hydra vocabulary which was designed to describe web services [15]. Some parts of the vocabulary are already used as metadata in TPF endpoints so those can be reused. An essential part of my research in respect to interfaces descriptions is related to the rules that combine Hydra with the SPARQL algebra components, indicating what the response looks like when calling a specific interface.

Initial plan optimization There already exist multiple optimization techniques for SPARQL queries as mentioned in Section 3.2. Optimizations that focus on SPARQL query rewriting will be highly relevant, while those depending on local indexes on the data are less likely to be used. In the case of TPF and possibly other interfaces we are provided with additional metadata which can also influence the query plan. In the case of a single SPARQL endpoint the plan is rather simple: send the query to that endpoint. However, in case of multiple endpoints or more restrictive interfaces the query will need to be split up into multiple calls.

Adaptive query optimization In case the query execution is split up in multiple calls, these will provide intermediate results, which can also influence the query plan. If results indicate that a component is more selective it is more interesting to focus on that one first for example. This means that the ordering of the plan is also quite important: the results of one component of the query influence the bindings for another component if they share a common variable, making that one more selective. Since it is impossible to completely determine this in advance, the system will have to adapt at runtime to those intermediate results. Additional rules will be necessary to support this, allowing the reasoner to take this information into account.

Extending the system Once the system works for TPF interfaces and simple queries, it will be extended to additional interfaces and SPARQL operators. The challenge there will be to make sure all interface functionality can be described adequately. It is impossible to predict all the necessary components since it also needs to support future interfaces, but the system should be modular enough that it can be extended once new descriptions are required.

Multiple data sources So far I have only described solutions with a single interface which require multiple requests to execute a query. In that case the single interface provides all the necessary data to solve that query. However, this is no longer the case when we use the interfaces of multiple sources. To solve this the descriptions need to be extended with additional metadata describing the contents of the endpoint where possible. Here I will make use of the already existing work in federated querying.

8 Evaluation Plan

The proposed system will be evaluated by comparing it to existing querying technologies. Since I suggested an iterative approach, starting with a more simple system which is then extended, the evaluations will also be iterative: each new extension will require a complete evaluation to determine its influence on the complete system. I plan to make use of existing SPARQL benchmarks [2, 8, 20]. The evaluation will be done in multiple steps to make sure all bases are covered for having a functioning query engine.

Correctness The first step of the evaluation will be verifying the correctness of the algorithm: make sure the returned results are actually correct answers to the query.

Completeness The second important component to have a working SPARQL engine is making sure it returns all the results. This might not always be necessary, especially in the case of federated queries, but for the initial implementation on a single data source with simple queries the answer needs to be complete.

Optimization Once the framework passes the two steps mentioned above, I will investigate the performance: how does it compare to existing technologies and SPARQL engines? In the case of a single TPF interface there are already multiple implementations available for comparison [1, 25, 30]. The case of a single SPARQL endpoint is trivial since that simply involves sending the query to the endpoint. Once multiple data sources are used there are already several existing implementations to compare with [18]. There are multiple features that need to be covered in performance tests, such as response time, bandwidth requirements and the influence of the number or size of datasets.

9 Reflections

My first paper on query optimization for TPF [25] introduced me to the vast amount of work that has been done in the field of SPARQL query optimization. One problem with TPF is that since it is a new technology, new optimizations would have to be researched since it provided a novel way to execute SPARQL queries. It is possible that much of the original research on SPARQL query optimization can be reused, but this has to be investigated on a case by case basis. This inspired me to propose a framework for generic SPARQL query execution.

There are many challenges that need to be tackled before the idea can come to fruition. One of the main hypotheses I build upon is the ability to create adequate rules and descriptions so query plans can be generated using a reasoning engine. Working with RESTdesc has shown me that reasoning can be quite powerful for the Semantic Web. In case I am unable to find an adequate solution, the system

proposed can still be built in a non-reasoning environment, although it would lose all reasoning advantages.

The second big hurdle is the combination of multiple interfaces. Although source selection has already been extensively researched as shown in Section 3.3, the combination of both different sources and different interfaces provides a new level of complexity. Assuming the remainder of the framework works correctly, providing a solution will definitely be possible. However, providing an acceptably optimized solution will be much harder. It is still unclear to me how to best tackle this, which is why I look forward to discussing this during the Doctoral Consortium.

References

1. Acosta, M., Vidal, M.E.: Networks of linked data eddies: An adaptive web query processing engine for RDF data. In: The Semantic Web-ISWC 2015, pp. 111–127 (2015)
2. Aluç, G., Hartig, O., Özsu, M.T., Daudjee, K.: Diversified stress testing of RDF data management systems. In: The Semantic Web–ISWC 2014, pp. 197–212 (2014)
3. Arndt, D., De Meester, B., Bonte, P., Schaballie, J., Bhatti, J., Dereuddre, W., Verborgh, R., Ongenae, F., De Turck, F., Van de Walle, R., Mannens, E.: Ontology reasoning using rules in an eHealth context. In: Rule Technologies: Foundations, Tools, and Applications. Lecture Notes in Computer Science, vol. 9202, pp. 465–472. Springer (Jul 2015)
4. Bechhofer, S.: OWL: Web ontology language. In: Encyclopedia of Database Systems, pp. 2008–2009. Springer (2009)
5. Berners-Lee, T.: cwm - a general-purpose data processor for the semantic web. Tech. rep. (2000), `https://www.w3.org/2000/10/swap/doc/cwm.html`
6. Berners-Lee, T., Connolly, D., Kagal, L., Scharf, Y., Hendler, J.: N3logic: A logical framework for the world wide web. CoRR abs/0711.1533 (2007)
7. Berners-Lee, T., Hendler, J., Lassila, O., et al.: The semantic web. Scientific american 284(5), 28–37 (2001)
8. Bizer, C., Schultz, A.: The Berlin SPARQL benchmark. International Journal on Semantic Web and Information Systems 5(2), 1–24 (2009)
9. Buil-Aranda, C., Hogan, A., Umbrich, J., Vandenbussche, P.Y.: SPARQLWeb-querying infrastructure: Ready for action? In: Proceedings of the 12th International Semantic Web Conference (Nov 2013)
10. Ermilov, I., Martin, M., Lehmann, J., Auer, S.: Linked open data statistics: Collection and exploitation. In: Knowledge Engineering and the Semantic Web, pp. 242–249. Springer (2013)
11. Harris, S., Seaborne, A.: SPARQL 1.1 query language. Recommendation, World Wide Web Consortium (Mar 2013), `http://www.w3.org/TR/sparql11-query/`
12. Hartig, O., Heese, R.: The SPARQL query graph model for query optimization. In: The Semantic Web: Research and Applications, pp. 564–578. Springer (2007)
13. Hayes, P., Patel-Schneider, P.: RDF 1.1 semantics. W3C recommendation, W3C (Feb 2014), http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/
14. Heath, T., Bizer, C.: Linked data: Evolving the web into a global data space. Synthesis lectures on the semantic web: theory and technology 1(1), 1–136 (2011)
15. Lanthaler, M., Gütl, C.: Hydra: A vocabulary for hypermedia-driven Web APIs. In: Proceedings of the 6th Workshop on Linked Data on the Web (May 2013)

16. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C., et al.: OWL 2 web ontology language: Profiles. W3C recommendation 27, 61 (2009)
17. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. In: International semantic web conference. vol. 4273, pp. 30–43. Springer (2006)
18. Rakhmawati, N.A., Umbrich, J., Karnstedt, M., Hasnain, A., Hausenblas, M.: Querying over federated SPARQL endpoints - A state of the art survey. CoRR abs/1306.1723 (2013)
19. Saleem, M., Khan, Y., Hasnain, A., Ermilov, I., Ngonga Ngomo, A.C.: A fine-grained evaluation of SPARQL endpoint federation systems. Semantic Web (Preprint), 1–26 (2015)
20. Schmidt, M., Hornung, T., Meier, M., Pinkel, C., Lausen, G.: SP^{2}Bench: A SPARQL performance benchmark. In: Semantic Web Information Management, pp. 371–393. Springer (2010)
21. Schmidt, M., Meier, M., Lausen, G.: Foundations of SPARQL query optimization. In: Proceedings of the 13th International Conference on Database Theory. pp. 4–33. ACM (2010)
22. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semantics: science, services and agents on the World Wide Web 5(2), 51–53 (2007)
23. Stocker, M., Seaborne, A., Bernstein, A., Kiefer, C., Reynolds, D.: SPARQL basic graph pattern optimization using selectivity estimation. In: Proceedings of the 17th international conference on World Wide Web. pp. 595–604. ACM (2008)
24. Van Herwegen, J., De Vocht, L., Verborgh, R., Mannens, E., Van de Walle, R.: Substring filtering for low-cost Linked Data interfaces. In: The Semantic Web – ISWC 2015. Lecture Notes in Computer Science, vol. 9366, pp. 128–143. Springer (Oct 2015)
25. Van Herwegen, J., Verborgh, R., Mannens, E., Van de Walle, R.: Query execution optimization for clients of triple pattern fragments. In: Proceedings of the 12$^{\text{th}}$ Extended Semantic Web Conference (Jun 2015)
26. Vander Sande, M., Verborgh, R., Van Herwegen, J., Mannens, E., Van de Walle, R.: Opportunistic Linked Data querying through approximate membership metadata. In: The Semantic Web – ISWC 2015. Lecture Notes in Computer Science, vol. 9366, pp. 92–110. Springer (Oct 2015)
27. Verborgh, R., Arndt, D., Van Hoecke, S., De Roo, J., Mels, G., Steiner, T., Gabarró Vallés, J.: The pragmatic proof: Hypermedia API composition and execution. Theory and Practice of Logic Programming (2016)
28. Verborgh, R., De Roo, J.: Drawing conclusions from Linked Data on the Web. IEEE Software 32(5), 23–27 (May 2015)
29. Verborgh, R., Mannens, E., Van de Walle, R.: Bottom-up web APIs with self-descriptive responses. In: Proceedings of the First Karlsruhe Service Summit Research Workshop (Feb 2015)
30. Verborgh, R., Vander Sande, M., Hartig, O., Van Herwegen, J., De Vocht, L., De Meester, B., Haesendonck, G., Colpaert, P.: Triple Pattern Fragments: a low-cost knowledge graph interface for the Web. Journal of Web Semantics 37–38, 184–206 (2016)