9 Combining Graph Queries with Graph Analytics

Dan Brickley (Google Research - Mountain View, US), Aidan Hogan (IMFD, DCC, University of Chile - Santiago de Chile, CL), Sebastian Neumaier (Wirtschaftsuniversität Wien, AT), and Axel Polleres (Wirtschaftsuniversität Wien, AT)

9.1 Introduction

The topics of data analytics and querying in the context of Knowledge Graphs have been addressed as part of two separate fields. However, most data processing pipelines using Knowledge Graphs require interleaving analytical and query tasks. While there exists infrastructure (languages, tools, algorithms, optimisations, etc.) for performing queries and analytics as separate processes, currently there does not exist an infrastructure for integrating the two. Still, many conceptual questions that a domain expert may wish to ask imply such a combination. There are several applications for combining analytical algorithms and querying relevant to Knowledge Graphs, for example:

- Ranking query results (e.g., ordering solutions based on the centrality of nodes in a graph),
- selecting topical sub-graphs to query (e.g., performing community detection to run queries on parts of a graph relevant to a given topic),
- exploratory search (e.g., finding weighted shortest paths between pairs of nodes returned as results for a query),
- dataset search (e.g., considering various graph metrics to select an external dataset suitable to querying), or
- data quality issues (e.g., analysing the connectivity of the graph).

These examples illustrate that in some cases we may wish to query the results of an analytical process, in other cases we may wish to perform analysis on the results of a query, or in other cases still, we may wish to interleave various query/analysis steps.

In this chapter we attempt to identify some essential steps towards combining graph querying and analytics in terms of useful features. We briefly discuss the state-of-the-art standard technologies to implement these features. We shall also turn towards questions about how to implement those features in a scalable manner, and missing bits and pieces with respect to these standard technologies. With currently available technologies we likely end up with the necessity to store and transfer graphs between different systems and stores to enable all of these features, due to the non-availability of a single system and engine to implement them. We argue that (extensions of) RDF and SPARQL seem to be the most suitable anchor points as a crystallisation point to enable such interchange and integration of query and analytics features.

9.2 Potential Starting points & Prior attempts

Although there have been proposals of various languages for querying graphs [2], including for example Cypher [11] and G-CORE [1], in the Semantic Web community, SPARQL [6] has been set as the standard query language for (Knowledge) Graphs, until now remaining the only graph query language backed by a standardisation body and implemented by numerous

engines. The following discussion thus focuses on SPARQL, though the topics covered generalise also to other query languages for graphs, such as those mentioned.

Since the original standardisation of SPARQL [9], the scientific community has proposed lots of useful extensions for this language in terms of analytics and data processing features and combinations with other languages. This has led to SPARQL 1.1. which was a conservative extension of features agreed, by the W3C, to be key to the future of the language, essentially taking on board and consolidating the most urgent of these proposed features.

However, as for specific connections to graph analytics, apart from basic path query and aggregation features, many issues and in meanwhile urgent features remain unaddressed. In particular, core features relating to graph algorithms and network analysis have not found their way into the standard, despite being part of many typical (knowledge) graph processing pipelines.

While there have been attempts to combine SPARQL with other Turing-complete languages, e.g. Spark, Gremlin [3], XSPARQL [4], which would allow to address and implement all such features – herein, we rather aim at investigating which are the core features and tasks that typically are needed and that would deserve to be added as first-class citizens (or built-ins) in such a language.

Likewise, extensions of typical analytics languages like R [12], working on data frames, have simple libraries to import/incorporate SPARQL results tables as such data frames, but not allowing per se the reuse of analytical results as graphs again in a SPARQL-like query language, nor providing an integrated graph analytics and query language.

Also, potentially interesting starting points are widely-used graph analysis systems outside of the Semantic Web world; to name a few, e.g.: Shark [14], that allows to run SQL queries and sophisticated analytics functions; Google's Pregel [8], a system to efficiently process large graphs (of billions of vertices and trillions of edges) which powers Google's PageRank; as well as frameworks built on top of Apache Spark [13, 5], as well as various academic projects such as Signal-Collect [10].

9.3 Motivating Examples

Before continuing, we enumerate some motivating examples that help to illustrate the importance of considering queries and analytics in a unified framework. We will consider a hypothetical Knowledge Graph of bibliographical data considering scientific articles, the articles they cite, where they were published, their authors, their fields, and relations between fields. Potentially relevant questions on such a Knowledge Graph include:

- Find sub-communities of Computer Science in Mexico.
- Find the most important papers in AI published in IJCAI.
- Find connections (paths) from researchers in UChile and UBA.

Such questions involve goals that are naturally expressed through queries (Computer Science papers, authors in Mexico, papers in IJCAI, researchers in UChile, etc.) and goals that are naturally expressed through analytics (sub-communities, important papers, connections). Inspecting these questions, we can see that querying and analytical goals are interleaved, where we may wish to analyse a graph produced as the result of a query, or querying a graph enriched with the results of analyse, or any such combination.

Rather than pushing data between separate querying and analytical frameworks, the goal would be to combine both into one framework, allowing for the design of a unified language,

hybrid algorithms optimised to consider all goals, as well as practical tools, interfaces and implementations.

Graph Analysis Requirements:

Let us consider some of the common types of algorithms used in the graph analytics community that could be interesting to combine with queries in a unified framework.

Centrality. Centrality of graphs can serve as indicators of finding the most important/most influential vertices of a graph. As an example, centrality measure would allow, e.g., an analysis of the most influential papers in a network of publications, cross-citations, and co-authorships (given the above example bibliographic Knowledge Graph).

Community structure/detection. A graph is said to have communities if there are densely connected structures that can be grouped in node subsets. Community detection algorithms, such as minimum-cut algorithms, allow to discover these sub-communities, which, for instance, could relate to a connected sub-community of researchers, given the above example.

Paths/Flows. A path in a graph generically denotes a connection between two nodes that may traverse multiple edges. Various technical definitions exist that restrict the set of valid paths between such nodes, including simple paths that do not visit the same node twice, or regular path queries that restrict the labels of edges that can be traversed by the path [2]. Additionally, extensions of such regular path queries with more complex conditions on properties have been defined, which are particularly important when dealing with graph data beyond "flat" RDF, such as property graphs that express provenance or other contextual information along the edges.

Vertex similarity. There exist measures for "vertex similarity" that capture the relatedness of nodes in a graph by considering the neighbours they have in common and/or the specificity of the paths that exist between them. These methods allow to understand what connects nodes, and, thereafter, in what ways they are similar.

Connectivity/Spanning trees. The connectivity of a graph – defined as the number of vertices or edges that need to be removed to disconnect the graph – in the context of Knowledge Graphs allows to analyse the resilience and (un)reachability of components. Also, related to the connectivity is the spanning tree of such a graph.

9.4 Semantic Graph Analytics

There are various data models that can be used to describe "graphs", including, for example, directed-edge labelled graphs, property graphs, and so forth. However, many of the traditional graph algorithms – though their generality and usefulness have been well-established in a variety of domains – are proposed and studied for simple graphs or directed graphs without labels. Hence the question arises of how to adapt and apply these algorithms to other structures; there may be many options to "project" out a directed graph from a more complex Knowledge Graph model, where each such projection may yield radically different results; respectively, depending on how the Knowledge Graph is stored, computing such a projection and transforming it into a format amenable to these algrotithms itself might impose a significant effort.

Aside from structure, Knowledge Graphs often embed semantics of domain terms expressed, for example, using formal model theory. Such semantics then permit reasoning

methods that allow for transforming or extending graphs in a manner that preserves truth (i.e., applying inference); examples include subclass reasoning, or inferences over transitive or inverse properties, identity reasoning, and so forth [2]. Applying analytics before or after such transformations may again yield radically different results, and hence it is important to study such differences, and to study (and justify/evaluate) which transformations better reflect the real-world phenomena under analysis.

Projections here can involve "Inference-based transformations", i.e. materialisation or core-reduction (e.g. removing transitives or inverse edges to reduce a graph to its raw form, resolving non-unique names by choosing canonical representatives for an equivalence class) wrt. semantic rules (related to e.g. spanning trees computation). That is, when doing analytics one often needs to be aware that "semantically equivalent" graphs (with respect to the chosen KG semantics) may behave fundamentally differently when taken as inputs for graph analytics steps.

9.5 Conclusions and Next Steps

To present the herein discussed topics to a broader and appropriate audience we plan to submit an extended version of this report as a position paper to the upcoming W3C Workshop on Web Standardization for Graph Data.[8] The scope of the Workshop includes requirements for graph query languages and different kinds of reasoning in graph database systems. Also, it aims at bringing together the adjacent worlds of RDF and Property Graphs (cf. for instance [7]), to achieve productive and interoperable boundaries, and foster information exchange and mutual awareness.

Acknowledgements

Axel Polleres' and Sebastian Neumaier's work is supported by the EC under the H2020 project SPECIAL and by the Austrian Research Promotion Agency (FFG) under the projects "CitySpin", "Communidata" and "DALICC". Aidan Hogan is funded by the Millennium Institute for Foundational Research on Data (IMFD) and Fondecyt Grant No. 1181896.

References

1 Renzo Angles, Marcelo Arenas, Pablo Barceló, Peter A. Boncz, George H. L. Fletcher, Claudio Gutierrez, Tobias Lindaaker, Marcus Paradies, Stefan Plantikow, Juan F. Sequeda, Oskar van Rest, and Hannes Voigt. G-CORE: A core for future graph query languages. In Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein, editors, *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 1421–1432. ACM, 2018.

2 Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoc. Foundations of modern query languages for graph databases. *ACM Comput. Surv.*, 50(5):68:1–68:40, 2017.

3 Apache TinkerPop. TinkerPop3 Documentation v.3.2.5. http://tinkerpop.apache.org/docs/current/reference/, June 2017.

4 Stefan Bischof, Stefan Decker, Thomas Krennwallner, Nuno Lopes, and Axel Polleres. Mapping between RDF and XML with XSPARQL. *J. Data Semantics*, 1(3):147–185, 2012.

[8] https://www.w3.org/Data/events/data-ws-2019/

5 Ankur Dave, Alekh Jindal, Li Erran Li, Reynold Xin, Joseph Gonzalez, and Matei Zaharia. Graphframes: an integrated API for mixing graph and relational queries. In *Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems, Redwood Shores, CA, USA, June 24 - 24, 2016*, page 2, 2016.

6 Steve Harris and Andy Seaborne. SPARQL 1.1 Query Language. W3C Recommendation, 2013.

7 Olaf Hartig. Reconciliation of RDF* and Property Graphs, Technical Report, *CoRR abs/1409.3288*, 2014.

8 Grzegorz Malewicz, Matthew H. Austern, Aart J. C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, pages 135–146, 2010.

9 Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. W3C Recommendation, 2008.

10 Philip Stutz, Daniel Strebel, and Abraham Bernstein. Signal/Collect12. *Semantic Web* 7(2): 139–166, 2016.

11 The Neo4j Team. The Neo4j Manual v3.0. http://neo4j.com/docs/stable/, 2016.

12 The R Foundation. The R Project for Statistical Computing. https://www.r-project.org, since 1992.

13 Reynold S. Xin, Joseph E. Gonzalez, Michael J. Franklin, and Ion Stoica. Graphx: a resilient distributed graph system on spark. In *First International Workshop on Graph Data Management Experiences and Systems, GRADES 2013, co-loated with SIGMOD/PODS 2013, New York, NY, USA, June 24, 2013*, page 2, 2013.

14 Reynold S. Xin, Josh Rosen, Matei Zaharia, Michael J. Franklin, Scott Shenker, and Ion Stoica. Shark: SQL and rich analytics at scale. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 13–24, 2013.