Keyword Search on RDF Graphs

Dennis Dosso
Department of Information Engineering,
University of Padua
Padua, Italy
dennis.dosso@phd.unipd.it

ABSTRACT

The Resource Description Framework (RDF) is a family of specifications developed and supported by the W3C consortium to represent information in the Web[1]. During the last years, RDF has gained popularity in many domains such as medicine and cultural heritage as a representation format for heterogeneous structured data on the Web [2]. RDF graphs can be interrogated by queries expressed with the SPARQL language. To write queries in this language can become very difficult. Users are required to know the language and the structure of the underlying dataset in order to write correct queries. Thus, the need for a system of keyword search for these graphs. Keyword search permits users to express their information need via a query in natural language, in a Google-like fashion.

Keyword search over large knowledge bases can become difficult both in therms of memory and time required to answer to a single query.

In this abstract, we discuss the experience of designing and implementing keyword search algorithms over big RDF databases.

KEYWORDS

Graph Databases, RDF databases, Keyword Search

1 KEYWORD SEARCH

The authors in [4] used a subset of IMDB and LibraryThing to create RDF datagraphs, which are relatively small (less than 1M triples). In [5], the authors adopted the databases used in [3] which don't exceed the 2M tuples. The proposed IMDB subset has only 1.6 million tuples, while the Mondial dataset has only 17K tuples. The database provided by IMDB on their website has over 100 millions of triples[2] once parsed translated in RDF. LinkedMDB, another database about films which is RDF native, has 7 millions of triples[3]. DisGeNET, a medical database about gene-disease associations, has more than 40 millions of triples[4] in its version 5. Other databases reported in [1] have billions of triples. With these numbers, we need to seriously think about performances both in time and space in order to create applications that can work in real-world scenarios.

The algorithms proposed in [4] and [5] are based on the visit of the graph and in the creation of potential answer documents that are ranked and proposed to the user. The time and the memory required to explore the graph, discover the answer graphs and rank them become big with huge graphs. A query can require up to an hour and half of time in order to be answered on a database like LinkedMDB. This, in an on-line scenario, can become unbearable.

It is necessary to re-think the architectural infrastructure and the algorithmic approach in order to deal with time and memory. This implies that algorithms cannot use only RAM, but also secondary memory. They should leverage on Triple Stores as the ones provided by libraries like Blazegraph[5]. Relational databases can be used as a support to perform the more computational expensive queries thanks to indexes. This requires additional storage memory for the necessary tables and their indexes, and the study of the correct relational schema to support the future queries and algorithms.

A possible approach to deal with the problem of time is to study algorithms that work off-line, without a user query. Their aim will be to extrapolate useful information that can be used to speed-up the on-line phase.

We are currently working on an algorithm that searches off-line possible *good subgraphs*, which are subgraphs that are composed by triples that encompass the same topic, preferably with a minimum presence of noise, that is triples that are about a different argument. These graphs are more easily explorable and manageable than the whole graph and can be queried faster with standard or adapted IR methods. Hopefully, they are also good answers for the different information needs of the user, which are expressed through keyword queries.

One key aspect is also to guarantee the reproducibility of the systems and the corresponding experiments. In order to do so, it is necessary to keep track of different things: the relational schema, the code, the support files, the structure of the file system, and also the time and memory required by the algorithms on the different queries.

REFERENCES

[1] Hannah Bast, Björn Buchhold, and Elmar Haussmann. 2016. Semantic Search on Text and Knowledge Bases. *Foundations and Trends in Information Retrieval* 10, 2-3 (2016), 119–271. https://doi.org/10.1561/1500000032
[2] Peter Buneman, Susan B. Davidson, and James Frew. 2016. Why data citation is a computational problem. *Commun. ACM* 59, 9 (2016), 50–57. https://doi.org/10.1145/2893181
[3] Joel Coffman and Alfred C Weaver. 2010. A framework for evaluating database keyword search strategies. In *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, 729–738.
[4] Shady Elbassuoni and Roi Blanco. 2011. Keyword search over RDF graphs. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 237–242.
[5] Yosi Mass and Yehoshua Sagiv. 2016. Virtual Documents and Answer Priors in Keyword Search over Data Graphs.. In *EDBT/ICDT Workshops*.

[1] https://www.w3.org/RDF/

This work was supported by the CDC-STARS project and co-funded by UNIPD.

[2] https://datasets.imdbws.com/

[3] http://www.linkedmdb.org/

[4] http://www.disgenet.org/web/DisGeNET/menu/rdf

DESIRES 2018, August 2018, Bertinoro, Italy

[5] https://www.blazegraph.com/