Keyword-Based Semantic Search Engine Koios++

Björn Forcher[1], Andreas Giloj[1], and Erich Weichselgartner[1]

Leibniz Institute for Psychology Information,
Germany, email: firstname.lastname@zpid.de

Abstract. In this paper, we describe the keyword-based semantic search engine *KOIOS++* which interprets the keywords and computes a set of SPARQL queries. The special feature of *KOIOS++* is that it leverages not only the class hierarchy but also the property hierarchy. The algorithm and data structures of *KOIOS++* are based on a well-established approach that we extended by minor adjustments of the data structures and a sophisticated weighting strategy.

Key words: keyword-based semantic search, RDFS semantics

1 Introduction

The RDF data model is a popular data model of the Semantic Web which can be easily transferred to a simple directed graph. RDFS extends RDF and provides representational constructs for describing ontologies, for instance the properties *rdfs:subClassOf* and *rdfs:subPropertyOf*. Both properties are transitive relations which are used to describe class or property hierarchies of ontologies.

Finding information in graph-shaped data, keyword search seems to be natural because it is the de facto standard for current search engines. The field of keyword-based search on graph-structured data in general, and in particular over RDF data, is a prevalent research topic and corresponding efforts can be referred in [1], [2], and [3] at different glances. Tran *et al.* [4] describe an interesting approach of keyword-based semantic search for computing the top-k ranked search results from RFD(S) graphs. They first compute queries from the keywords, allowing the users to choose one of them, and finally to process the query using the underlying database engine. Nevertheless, these approaches do not focus on the hierarchy of properties and thus, they cannot make use of the transitive tree of the *rdfs:subPropertyOf* relation. Recent publications of that group focused on the processing of the approximated top-k ranked results [5] to reduce computation time but they did not consider further aspects of the RDFS semantics. The primary motivation of our work is to make the *rdfs:subPropertyOf* available for keyword-based semantic search on graph shaped data. We extend the approach of Tran *et al.* by using a special graph mapping and a sophisticated weighting strategy which is implemented in the *KOIOS++* search engine. We claim that in this way we get a semantic benefit because we can favor certain graph patterns during the search. As a result, it is possible, for instance, to make use of the property hierarchy by leveraging the *rdfs:subPropertyOf* relation.

The paper is structured as follows. Sec. 2 introduces the search engine *KOIOS++* including its search algorithm and data structures. We conclude with a brief summary and a small outlook (Sec. 3).

2 Koios++

As mentioned in the previous chapter, the approach of *KOIOS++* is based on the work of Tran *et al.*[4] which is depicted in Figure 1. In the *Data Preprocessing* two main data structures are built from the RDF(S) data, namely *keyword index* and *summary graph*. The graph represents a summarization of the RDF(S) data comprising only structural information. Thus, it contains only classes and properties, but no literals and instances. Literals and instances are integrated at runtime by means of the keyword index. In general, the keyword index is used to map keywords to elements of the RDF(S) data. The basic thinking behind the establishment of both structures is to enable a high performance search along with scalability. The summary graph (kept in memory) contains only as much information as necessary whereas the keyword index (native database) represents an entry point containing additional information.

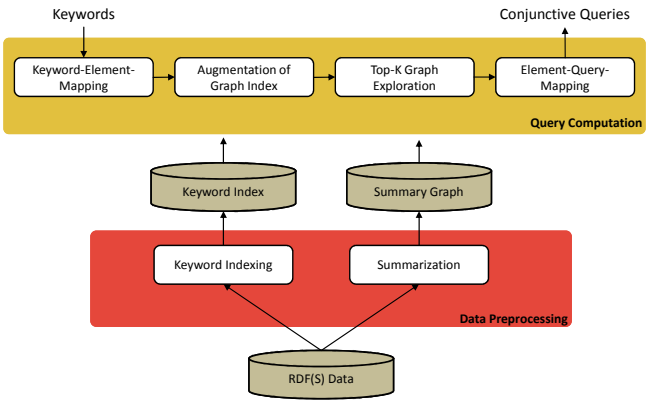

Fig. 1. Data Processing and Query Computation

One contribution of our work is the adjustment of the summary graph in order to leverage both, the class hierarchy and the property hierarchy. Figure 2 is intended to point out the main difference between the two approaches. The figure contains some example triples and the corresponding summary. In contrast to the summarization of Tran *et al.* a triple with two instances is mapped to a directed edge. Source and target correspond to the classes of the instances and the property of the triple is noted as label of the edge. In our approach every argument of a triple is mapped to a typed node (class or property node). The node of the subject and the node of the predicate are connected by a directed edge without label. The same applies for predicate and object. The only exception to that rule are triples including a 'rdfs:subClassOf' or 'rdfs:subPropertyOf'

predicate. In these cases the nodes are directly linked with an edge.
For computing SPARQL queries, the loose keywords are first disassembled into its constituent parts. For simplicity, let the keywords already be separated. Thus, there is a set of h terms $T = \{t_1, ..., t_h\}$, whereas $t_i \in T$ is either a single word or a multi-word expression. In the following step, the terms are mapped to elements of the input RDF(S) data which are called *M-Resources*. In general, a term $t_i \in T$ is mapped to a set of j resources $R_i = \{r_{i_1}, ..., r_{i_j}\}$. As follows, there are h sets of M-Resources $R_1, ..., R_h$ that are used for further processing. The resource sets R_1 to R_h were distributed to h threads and in each thread Z_i a graph exploration is performed on the prepared (augmented) summary graph for each M-Resource $r_{i_q} \in R_i$. Hence, many paths were explored starting from r_{i_q}. In case there is a resource r_c that is reached by any path in each thread a connecting subgraph G_s of the knowledge base can be constructed consisting of h paths. The resource r_c is called *C-Resource* which connects all paths with one another. The outcome of the graph search algorithm is a set of weighted subgraphs $G_1, ..., G_q$, whose size can be restricted by an upper weight limit and a general time limit. The weighting strategy can be separated into two parts. The static part weights all nodes and edges in the preprocessing step (further information is presented in Tran *et al.* [4]). The dynamic part of the weighting takes place at the runtime of the system and concerns visited paths only. The weight w_{p_n} for a new path p_n is based on the path before p_o, the new edge e_a and node v_b, and the resulting path pattern m_n: $w(p_n) = w(p_o) + w(e_a) + w(v_b) + w(m_n)$. The last steps of *KOIOS++* are straightforward. For each subgraph $G_1, ..., G_q$ a conjunctive SPARQL query is constructed and presented as semantic network to the user. Subsequently, the user can select relevant queries to retrieve the corresponding answer from the triplestore.
Consider the example keywords 'person' and 'worked'. The first one is mapped to the class 'zpid:Person' and the second one is mapped to the property 'zpid:WorkedOn'. The exploration may track two paths starting from both nodes which may end up in the nodes 'zpid:Librarian' or 'zpid:Author'. As follows, two different SPARQL queries can be constructed.
The presented data structure has an inherent disadvantage because the queries may contain statements that are not included in the origin data, for instance, 'a librarian wrote an article'. This information is not necessarily incorrect, but if this relation is not entailed in the triplestore unnecessary graph traveling is done. As follows, the algorithm gets expensive and time-consuming. This is one reason, why we integrated the dynamic weighting as described above. If the graph traveling ends in a path p_n with an unwanted pattern the additional weight $w(m_n)$ is set to infinite (if not unwanted $w(m_n) = 0$). That means that the new path is (very likely) not considered for further exploration and query construction. Thus, the described graph mapping in combination with the dynamic weighting enables the integration of the property hierarchy without constructing unwanted SPARQL queries.
However, this kind of weighting strategy could also be used to foster ($w(m_n) < 0$) certain graph patterns which is important for explanation scenarios.

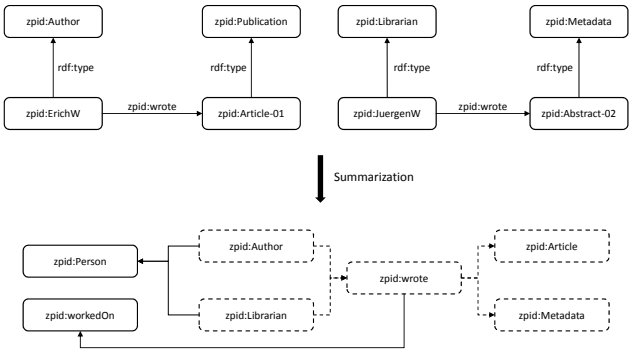

Fig. 2. Summarization of some triples

3 Conclusion

In this paper, we presented the semantic search engine *KOIOS++* that enables a keyword-based search on RDF(S) triple stores. It interprets the keywords and computes a set of SPARQL queries which can be selected by users to search the triple store. The RDF(S) data is mapped to a graph structure which is explored for the computation. In contrast to other approaches a predicate is not mapped to an edge, it is mapped to a node. To avoid unnecessary workload a sophisticated weighting strategy is applied. In particular, the strategy takes place at runtime whereas the exploration of new graph elements is based on the previous explored elements (conditional search). The presented approach enables not only heuristic reasoning on class hierarchies but also on the property hierarchies.
The next step of our work is to integrate SPARQL operators into our approach. Thus, it becomes possible to use words, such as "greater" or "smaller".

References

1. Achiezra, H., Golenberg, K., Kimelfeld, B., Sagiv, Y.: Exploratory keyword search on data graphs. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. SIGMOD '10, New York, USA, ACM (2010)
2. Cappellari, P., De Virgilio, R., Maccioni, A., Roantree, M.: A path-oriented rdf index for keyword search query processing. In: Proceedings of the 22Nd International Conference on Database and Expert Systems Applications - Volume Part II. (2011)
3. De Virgilio, R., Maccioni, A., Cappellari, P.: A linear and monotonic strategy to keyword search over rdf data. In Daniel, F., Dolog, P., Li, Q., eds.: Web Engineering. Springer Berlin Heidelberg (2013)
4. Tran, D.T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In: Proc. of the 25th Intern. Conference on Data Engineering (ICDE'09), Shanghai, China (2009)
5. Wagner, A., Bicer, V., Tran, T.: Pay-as-you-go approximate join top-k processing for the web of data. In Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A., eds.: The Semantic Web: Trends and Challenges. Springer International Publishing (2014)