Semantics-driven Keyword Search over Knowledge Graphs

Leila Feddoul[1,2] [0000−0001−8896−8208]

[1] Heinz Nixdorf Chair for Distributed Information Systems,
Friedrich Schiller University Jena, Jena, Germany
leila.feddoul@uni-jena.de
[2] Institute of Data Science, German Aerospace Center (DLR), Jena, Germany
leila.feddoul@dlr.de

Abstract. With the continuous growth of data represented as knowledge graphs, advanced access techniques are in great demand. Keyword search is a convenient and simple method to retrieve information. It can be used as an alternative to structured query languages, since it does not require technical expertise. In the context of knowledge graphs, keyword search aims to find candidate subgraphs that connect the query keywords and thus answer user queries. Efficiently finding candidates and ranking them effectively is still a challenging task. The goal of this work is to automatically find appropriate subgraphs. The focus is particularly on enhancing the accuracy of subgraph ranking by leveraging semantic information combined with other factors. Preliminary results demonstrate that the combination of *importance-based* and *semantics-based* metrics is promising compared to purely structural techniques.

Keywords: Keyword search · Knowledge graph · Graph traversal.

1 Problem statement

Knowledge graphs (KGs) have become useful resources to enhance search. In the context of keyword search, query elements are generally associated with entities from the graph. They are not considered as mere literals anymore but acquire their semantic identity including relations with other graph entities. KGs are often available in form of RDF triples. Queries using SPARQL require not only the construction of complex queries but also the knowledge of underlying data and schema. Those drawbacks make it not suitable for most end-users.

Keyword search is a common and user-friendly paradigm for querying text and structured data. It could be used and adapted to enable non-expert users to explore RDF data. Users enter terms to describe their information need and the system returns items deemed relevant. A possible answer is usually a subgraph that connects nodes corresponding to the input query keywords. Those answers (bindings) are directly retrieved or fetched using a generated query that corresponds to a conceptual answer subgraph (subgraph template).

The main challenges in both approaches are: (i) accurate mapping of keywords to graph elements, (ii) developing efficient approaches that scale to large

KGs to retrieve candidate subgraphs (bindings or subgraph templates), and (iii) providing strategies to rank the candidates with respect to their relevance to the query. Many research efforts addressed the scaling aspect, whereas candidates ranking is still not exhaustively studied in the literature. Existing ranking factors are mostly based on structural (e.g., number of nodes), statistical (e.g., edge frequency), or textual (e.g., node literal description) graph properties. They do not leverage the semantic nature of KGs. However, the relationships between different concepts can provide valuable insights for the ranking of candidates.

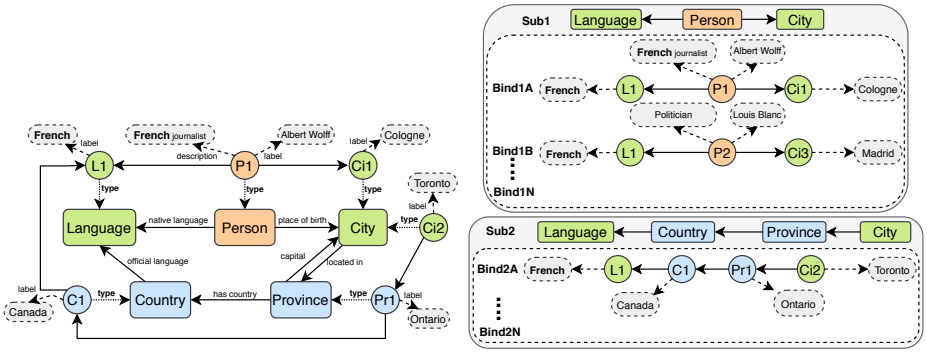
For this work, we assume that keyword-graph element mappings and the type of entities to retrieve, referred to as target, are determined beforehand. Generating such mappings and answer type prediction are considered out of scope for now. The initial aim is not performing a Google-like generic search, but allowing domain specific lookups where possible targets are predefined by the system administrator (e.g., a product (target) search engine, where other entities like vendors should not appear as results of a query). Instead, we focus on the second as well as the third challenge and aim at (1) automatically finding candidate subgraph templates and thus their bindings, (2) studying how semantic, structural, statistical, and textual properties of a a template and its bindings could be combined to better judge its relevance and (3) determining to which extent lower ranked templates still add relevant results without loosening the relation between search query and results. Thus, we attempt to improve the effectiveness of keyword search systems by incorporating ranking criteria that also leverage information that can be inferred from the semantics of the subgraphs.

2 Importance

By providing techniques that efficiently retrieve good quality results with respect to the user query, we allow end-users to access, explore and analyze semantic web resources. Furthermore, we make use of KGs to implicitly trying to understand what the query as a whole actually means and thereby achieve at least a modest progress in interpreting user intention. By using keyword search as an easy, quick and familiar query method, the time and effort needed to analyze the underlying structure of the data and to learn new technologies will be reduced.

3 Related Work

Keyword search over structured data. We consider efforts from keyword search over both relational databases and KGs relevant, as both perceive data as a graph. For KGs, the definition of the graph structure is straightforward. For relational databases, nodes are tuples and edges are given by foreign key relations. *Graph-based* methods [1–3] apply graph traversal algorithms on the whole data graph (instance level) to directly find final answer subgraphs, whereas *schema-based* approaches [4–6] operate either on an existing schema, or generate a graph summary using available instance level data. They retrieve generic subgraph templates whose instance level bindings represent the final answer subgraphs. Both



(a) RDF schema with some instances. (b) Subgraphs with bindings

Fig. 1: Example RDF graph and possible answer subgraph templates to the query $Q = \{French\ (keyword),\ cities\ (target)\}$ together with some bindings.

categories use heuristic algorithms to find the top-k connected structures. While *graph-based* methods mostly focus on on how to speed up exploration and do not rely on a schema, *schema-based* approaches construct a summary graph to improve scalability. However, the price is being schema-dependent and there is a risk of losing important relations during summarization.

Ranking criteria for subgraphs. We classify the approaches used to rank candidate subgraphs as follows: *Compactness-based* [4,5] consider the size of subgraphs as a quality criterion and their scores are given by the number of nodes or edges. However, they fail to distinguish between subgraphs having the same number of elements. *Importance-based* methods assign an importance score to graph elements based on different criteria. In [6] the importance of nodes/edges is defined by the total number of entities/edges clustered to a class node/summary graph edge. Others use PageRank [7] assigning higher scores to graph elements referenced by many other important elements. [1] assigns weights to nodes considering that nodes that have more pointers (in-degree) get higher scores. InfoRank [8], a weighted variant of PageRank, calculates edge weights as the sum of informativeness of the respective subject and object. Informativeness is defined as the number of owned literals. One of the shortcomings of *importance-based* approaches is that relations appearing frequently or having more pointers are not necessarily more semantically relevant. *Textual-based* methods usually use matches between the textual content of subgraphs and keyword query terms often adapting TF/IDF to graphs. The different approaches vary in what they consider as a document (e.g., text of tuple in [9]). Those methods try also to include the structural information of the subgraph by dividing by the number of nodes. *Virtual document-based* approaches [10, 11] extract a set of subgraphs from the original graph, map each subgraph to a document (e.g., node text), and finally perform a keyword search. *Profile-based* methods [12] use user profiles to reduce the search space and provide results closer to the user intent.

Running example. We illustrate the difference among existing approaches using the sample graph given in Figure 1a. It features five concepts (e.g., *Person*), their instances (e.g., *P1*), and relations. Figure 1b shows two possible answer subgraph templates (*Sub1* and *Sub2*) to the query *Q = {French (keyword), cities (target)}*, that are found using a *schema-based* method, where the most intuitive information need is *Cities located in a French-speaking country*. *Sub1* retrieves *cities that are place of birth of a person whose native language is French*, whereas *Sub2* retrieves *cities that are located in a province within a French-speaking country*. Each template has a list of possible bindings (e.g., *Bind1A*).

Compactness-based methods will rank *Sub1* over *Sub2*, since they primarily rely on the size of the subgraph. However, *Sub1* will also return cities located in non-French speaking countries (e.g., Cologne in *Bind1A*). *Textual-based and Virtual document-based* approaches rely on the textual content of the subgraphs. In graphs, textual information is usually rather scarce. Classes descriptions are also quite generic, so using them does not allow to distinguish sufficiently between *Sub1* and *Sub2*. Instance level bindings provide a more nuanced picture though, so aggregated scores of bindings could rank the templates themselves. Yet, *Sub1* will likely be ranked over *Sub2*, as the term *French* appears quite often in the descriptions of *Person*-instances. *Importance-based* methods depend for example on the number of occurrences of nodes/edges. Our example contains more instances of *Person* and their relations than *Countries* and *Provinces*. This places a higher importance on relations like *place of birth* and thus ranks *Sub1* over *Sub2*. In general, all mentioned approaches omit the semantics of relations. To the best of our knowledge, semantic metrics for judging subgraphs relevance are not considered by existing methods. To address the mentioned limitations, we propose a ranking method that takes advantage from the semantics of relations.

4 Research Questions

From the previous discussion, we identify the following research questions:

RQ1. How to automatically find non-redundant, concise subgraph templates and thus bindings connecting the query keywords in a reasonable time that permits live user interaction?

RQ2. How to measure the quality of the candidates by leveraging semantic information to rank them by relevance to the user query?

RQ3. How many templates are required to meaningfully answer the query?

In order to answer those questions, we establish the following assumptions:

(RQ1) ASM1.1. A mapping between keywords and graph elements already exists. Each keyword is assigned to one corresponding element in the data graph.

(RQ1) ASM1.2. A target class for the search is given for each query.

(RQ1) ASM1.3. Heuristic algorithms can approximate the set of relevant subgraph templates. A pruning phase should reject subgraphs with superfluous parts (e.g., edge can be removed without disconnecting keywords and target) and perform a redundancy check. This prevents replacing other relevant subgraphs that may contribute better to the completeness of the final query result.

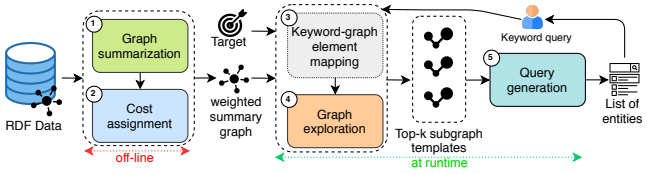


Fig. 2: Overall overview of the proposed approach.

(RQ2) ASM2. Combining metrics to exploit the semantic relations between entities with other factors such as structural, statistical, or textual characteristics can achieve a good relevance ranking. One example is the pairwise semantic relatedness between graph nodes under the intuition that *jumping to a semantically related node could increase the possibility to lead to highly relevant results.*

(RQ3) ASM3. Expanding the list of used templates may contribute new suitable results. We generate more complete results by considering not just the most relevant templates, but also others with high relevance. Evaluations using real-world datasets and suitable gold standards are needed to identify which and how many templates should be considered to add relevant items to the result.

5 Preliminary results

Methodology. To address the research questions, we propose the approach depicted in Figure 2. The workflow is divided into two phases: *off-line* and *runtime*. The *off-line* phase takes the original RDF graph, summarizes it, and assigns costs to its edges based on query-independent cost functions. Our work pertains to the *schema-based* category. We assume that the input data is available in form of triples together with *rdf:type* relations and the summary graph is created using them. Instances of the same type are clustered into a node in the summary graph together with their relations. Entity-literal and *rdf:type* relations are excluded. The former are always leaf nodes and thus do not contribute to the exploration.

The *runtime* phase takes the weighted summary graph, user keywords (classes or instances given by IRIs), and the target. The first step is to retrieve the corresponding node in the summary graph for each keyword. Afterwards, the graph exploration automatically attempts to find paths that connect keywords and target. We use an adapted and improved version of the backward search based on [6]. The basic algorithm has the drawback of generating redundant subgraphs, as different connecting nodes can result in the same subgraph being found. Thus, we add a redundancy check after finding a *connecting node* (connects keywords and target). Another shortcoming is that some generated subgraphs have superfluous parts. We believe that those subgraphs could sometimes be higher ranked and thus prevent more informative subgraphs to appear within the top-k. To cope with this problem, after finding a connecting node we first check if the subgraph contains superfluous parts, if yes we remove them and perform a redundancy check before adding it to the candidates list.

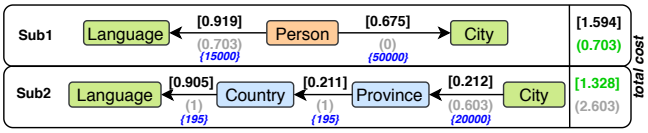


Fig. 3: Possible relevant subgraphs to $Q = \{French\ (keyword),\ cities\ (target)\}$. Each subgraph edge has a $CostFreq_{EdgeAB}$ (round brackets), $CostRel_{EdgeAB}$ (square brackets), and $\#triples_{EdgeAB}$ (braces).

The exploration returns a list of subgraph templates that serve as input for the query generation step that consists of translating templates to queries (e.g., SPARQL). Query results for each template correspond to a collection of bindings that could be again ranked based on query-dependent functions (e.g., textual). The system's final output is a list of entities to be displayed to the user. The described workflow allows to automatically generate candidate subgraphs given a target and a set of keywords and thus approaches **RQ1**.

Results. To address **RQ2**, we run our pipeline using different cost functions, and investigate how the returned templates are rated. First, an *importance-based* cost, $CostFreq_{EdgeAB} = 1 - tripleFreq_{EdgeAB}$ where $tripleFreq_{EdgeAB}$ is given by Equation 1. Here frequent triples are more important, therefore they are given lower cost and will be preferred during the exploration.

$$tripleFreq_{EdgeAB} = \frac{\#triples_{EdgeAB} - MinTripleCount}{MaxTripleCount - MinTripleCount} \tag{1}$$

Second, a *semantics-based* cost given by $CostRel_{EdgeAB} = 1 - SemRel(A, B)$ where $SemRel(A, B)$ is the pairwise semantic relatedness of two connected nodes A and B in the subgraph.

We concretely use the semantic relatedness defined in [13] calculated over WordNet [14] and the graph in Figure 1a with triple counts. With this, we want to verify **(RQ2) ASM2** under the intuition that *jumping to a semantically related node could increase the possibility to lead to highly relevant results.*

Figure 3 shows two generated possible relevant subgraph templates (*Sub1* and *Sub2*) to a given query Q. We analyze subgraph total costs (sum of edges costs) using the previous metrics. Since the overall intention is not allowing discovery by assigning higher scores (lower costs) to less predictable results, we expect that *Sub2* should have lower cost in comparison to *Sub1*. Consequently, we deduce that the frequency of appearance is not a good indicator of relevance and semantic relatedness seems more promising. However, semantic relatedness still cannot distinguish between subgraphs having the same nodes, but different edges (e.g., in Figure 1a there are two relations between city and province).

For this issue, we propose a new cost function combining semantic relatedness and triple frequency. This function favors frequent edges in cases of equal semantic relatedness and is given by Equation 2, where $0 < \beta \leq 1$ is a factor to adjust the effect of $tripleFreq_{EdgeAB}$ on $SemRel(A, B)$. Instead of a cost of 0.212 using semantic relatedness alone, this new cost function results in

$Cost_{locatedin} = 0.482$ and $Cost_{capital} = 0.598$ (with $\#triples_{Capital} = 1720$ and $\beta = 0.5$) and thus allows to differentiate between both subgraphs.

$$Cost_{EdgeAB} = 1 - SemRel(A, B) \cdot (\beta^{1-tripleFreq_{EdgeAB}}) \quad (2)$$

6 Evaluation

To evaluate the effectiveness of the current approach (**RQ2**), we need a test collection including a KG, a set of queries, and human relevance judgments. This test collection should provide mappings between possible result entities and the KG and at best a known target for each query. Determining the ground truth based on the automatic generation of a single SPARQL query (one template) like in QALD-2[3] is not suitable, since our approach uses multiple subgraph templates and thus queries. One suitable benchmark could be [15], which provides a large number of queries over DBpedia together with their relevance judgments collected via crowd sourcing. Another possible benchmark is [16] which uses three datasets and provides 50 queries for each dataset. The used datasets are available in form of relational databases and have to be converted to RDF. Both test collections could be fitted to our requirements, but still should contain at least some queries that are complex enough in the sense of requiring subgraphs that go beyond direct relations. Another direction is to think of new ways of evaluation that are more adapted to retrieving subgraphs and not only the final generated entities. By analyzing relevant results, we expect to verify that expanded subgraph templates also contribute meaningful results (**RQ3**). The following metrics could be used to compare the result set returned by our system with the ground truth: Precision, Precision@k, Recall, F-measure, Mean average precision and Reciprocal rank. We plan also to verify that our system scales with real-world datasets with a response time that allows comfortable user interaction (**RQ1**) by measuring its execution time and memory consumption.

7 Discussion and future work

The current preliminary results indicate that semantics (e.g., semantic relatedness) are a good indicator of results relevance compared with structural metrics (e.g., triple frequency). We introduced a new ranking metric by combining *importance* and *semantics-based* criteria. Current results are promising, but still need evaluation using other datasets and queries against an appropriate ground truth. Furthermore, we need to investigate how to include other aspects in result quality assessment. The current approach requires some conditions that are not necessary available in all scenarios (*rdf:type* relations, known target, and existing keyword-graph element mappings). Therefore we could think of solutions to make the approach more unbounded. Potential questions that still need to be investigated in further studies could be: How to deal with entities without type by summary graph generation? How user intention (target) could be automatically identified? How to leverage domain knowledge to improve result accuracy?

[3] `https://github.com/ag-sc/QALD/tree/master/2`

Acknowledgment

This work has been funded by the German Aerospace Center (DLR) and is supervised by Prof. Dr. Birgitta König-Ries, Sirko Schindler and Dr. Frank Löffler.

References

1. Bhalotia, G., Hulgeri, A., Nakhe, C., Chakrabarti, S., Sudarshan, S.: Keyword searching and browsing in databases using banks. In: ICDE 2002. pp. 431–440 (2002)
2. He, H., Wang, H., Yang, J., Yu, P.S.: BLINKS: Ranked keyword searches on graphs. In: ACM SIGMOD Conference 2007. p. 305–316 (2007). https://doi.org/10.1145/1247480.1247516
3. Kasneci, G., Ramanath, M., Sozio, M., Suchanek, F.M., Weikum, G.: STAR: Steiner-tree approximation in relationship graphs. In: ICDE 2009. pp. 868–879 (2009). https://doi.org/10.1109/icde.2009.64
4. Hristidis, V., Papakonstantinou, Y.: Discover: Keyword search in relational databases. In: VLDB 2002. p. 670–681 (2002)
5. Agrawal, S., Chaudhuri, S., Das, G.: DBXplorer: a system for keyword-based search over relational databases. In: ICDE 2002. pp. 5–16 (2002)
6. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data. In: ICDE 2009. p. 405–416 (2009). https://doi.org/10.1109/ICDE.2009.119
7. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. In: WWW 1999 (1999)
8. Menendez, E.S., Casanova, M.A., Paes Leme, L.A.P., Boughanem, M.: Novel node importance measures to improve keyword search over RDF graphs. In: DEXA 2019. pp. 143–158 (2019). https://doi.org/10.1007/978-3-030-27618-8_11
9. Xu, Y., Guan, J., Li, F., Zhou, S.: Scalable continual top-k keyword search in relational databases. Data & Knowledge Engineering **86**, 206 – 223 (2013). https://doi.org/https://doi.org/10.1016/j.datak.2013.03.004
10. Mass, Y., Sagiv, Y.: Language models for keyword search over data graphs. In: WSDM 2012. p. 363–372 (2012). https://doi.org/10.1145/2124295.2124340
11. Dosso, D., Silvello, G.: Search text to retrieve graphs: A scalable RDF keyword-based search system. IEEE Access **8**, 14089–14111 (2020). https://doi.org/10.1109/access.2020.2966823
12. Sinha, S.B., Lu, X., Theodoratos, D.: Personalized keyword search on large rdf graphs based on pattern graph similarity. In: IDEAS 2018. p. 12–21 (2018). https://doi.org/10.1145/3216122.3216167
13. Lin, D.: An information-theoretic definition of similarity. In: ICML 1998. p. 296–304 (1998)
14. Fellbaum, C., George, A.: WordNet: an electronic lexical database. MIT Press (1998)
15. Hasibi, F., Nikolaev, F., Xiong, C., Balog, K., Bratsberg, S.E., Kotov, A., Callan, J.: DBpedia-Entity v2: A Test Collection for Entity Search. In: SIGIR 2017. p. 1265–1268 (2017). https://doi.org/10.1145/3077136.3080751
16. Coffman, J., Weaver, A.C.: A framework for evaluating database keyword search strategies. In: Proceedings of the 19th CIKM 2010. p. 729–738. ACM Press (2010). https://doi.org/10.1145/1871437.1871531