Keyword Search on RDF Graphs: It Is More Than Just Searching for Keywords

Kostas Stefanidis(✉) and Irini Fundulaki

Institute of Computer Science, FORTH, Heraklion, Greece
{kstef,fundul}@ics.forth.gr

Abstract. In this paper, we propose a model for enabling users to search RDF data via keywords, thus, allowing them to discover relevant information without using complicated queries or knowing the underlying ontology or vocabulary. We aim at exploiting the characteristics of the RDF data to increase the quality of the ranked query results. We consider different dimensions for evaluating the value of results and achieving relevance, personalization and diversity.

1 Keyword Search on RDF Graphs

Typically, data in knowledge bases is represented using the RDF model. In RDF, everything we wish to describe is a *resource* that may be a person, an institution, a thing, a concept, or a relation between other resources. The building block of RDF is a *triple*, which is of the form (*subject*, *predicate*, *object*). The RDF Schema (RDFS) language is used to introduce useful semantics to RDF triples. It provides a built-in vocabulary for asserting user defined schemas within the RDF model. This vocabulary can be used to specify URIs as being of a specific type (*classes*, *properties* and *instances*), to denote special relationships between URIs. The flexibility of the RDF data model allows the representation of both schema and instance information in the form of RDF triples.

The traditional way to retrieve RDF data is through SPARQL, the W3C recommendation language for querying RDF datasets. SPARQL queries are built from *triple patterns* and determine the pattern to seek for; the answer is the part(s) of the set of RDF triples that matches this pattern. The correctness and completeness of answers of SPARQL queries are key research challenges. SPARQL is a structured query language that allows users to submit queries that may precisely identify their information needs, but require users to be familiar with the syntax, and the complex semantics of the language, as well as with the underlying schema or ontology. Moreover, this interaction mode assumes that users are to some extent familiar with the content of the knowledge base and also have a clear understanding of their information needs. On the other hand, today, there is a growing interest on a keyword-based search functionality to answer the search needs of users who are looking for specific information obtained by integrating numerous and heterogeneous sources. In addition, it is important for the user to be able to define different criteria on how all the resulting information could be ranked and returned to the user.

© Springer International Publishing Switzerland 2015
F. Gandon et al. (Eds.): ESWC 2015, LNCS 9341, pp. 144–148, 2015.
DOI: 10.1007/978-3-319-25639-9_28

2 Searching

A collection of RDF triples forms an *RDF graph* that is a *directed, edge and node labeled graph*, where nodes are subjects and objects of triples, and edges between nodes are predicates. An edge between a node corresponding to a subject s and a node corresponding to an object o exists only if a triple (s, p, o) exists in the triple set. Subjects, predicates and objects are URIs, while the latter could also be literals. [5] introduces a graph data model for representing RDF instance and schema information. Each RDF schema S defines a finite set of *class names* C and *property names* P. Properties are defined using class names or literal types, so that, for each property p, the domain of property p ($domain(p)$) is a class and the range of p ($range(p)$) is either a class or a literal. The model of [5] supports class and property hierarchies and denotes them as $H = (C \cup P, <)$.

A collection $\mathcal{A}$ of RDF triples forms a directed RDF graph $G(V, \lambda_V, \lambda_P, E)$, where: (i) $V = V_s \cup V_o$ represents a set of nodes; V_s and V_o contain a node for each subject and object, resp., in the triples of $\mathcal{A}$, (ii) $\lambda_v : V \rightarrow 2^C$ is a function that assigns to each node in G a set of class names from C, (iii) $\lambda_P : E \rightarrow P$ is a function that assigns to each edge in G one property name from P, and (iv) E represents a set of edges of the form $e(v_i, v_j)$ with $v_i \in V_s$, $v_j \in V_o$ and direction from v_i to v_j. Given that v_i, v_j, e correspond to a subject s, an object o and a predicate p, resp., the edge $e(v_i, v_j)$ exists in G, if and only if, $(s, p, o) \in \mathcal{A}$. The label of e is $\lambda_P(e) = p$, where $p \in P$.

A keyword query Q consists of a set of keywords $\{w_1, \ldots, w_m\}$. The result of Q is defined as a set of subgraphs H of G that are total, i.e., every keyword of Q is contained in at least one node n, one edge e, or their labels, $\lambda_V(n)$, $\lambda_P(e)$, resp., of H, and minimal, i.e., we cannot remove a node from H and get a total graph for Q. To process keyword queries over a graph G, we construct inverted indexes that store information about keyword occurrences in the RDF triples. Typically, after retrieving, for each keyword in Q, a list of matching elements, i.e., nodes, edges or labels, in G, we perform a bidirectional expansion search strategy to join elements from different lists.

3 Ranking

Given the abundance of available information, exploring the contents of a knowledge base is a complex process that may return a huge volume of data. Still, users would like to retrieve only a small piece of it, namely the most relevant to their interests. Previous approaches on querying RDF graphs mostly focus on the Boolean answer model, where query criteria are considered as hard by default and an answer is returned only if it satisfies all the criteria. In this context, a user can face the *too-many-answers problem*, where too many results match the query. Our focus in this work is on a keyword search model over RDF graphs, where users will be able to query the graph and be presented with a top-k result set by combining the *relevance* and the *diversity* of the results, and the *preferences* of the users.

Relevance. Relevance is an important and well-studied criterion for ranking the results $R(Q)$ of a keyword query Q. A natural characterization of the relevance of a subgraph $H \in R(Q)$ is its size, i.e., the number of its nodes: the smaller the size of the graph, the stronger the connection between the query keywords, thus the larger its relevance.

For XML documents, several approaches can be exploited for efficiently ranking keyword search results based on relevance (e.g., [8]). For RDF graphs, recently, [3] proposes ranking the resulting graphs of a query Q based on a statistical language-model. Specifically, this approach assigns a score to each graph $H \in R(Q)$ that reflects the probability of generating Q given the language-model of H. In this work, we intend *not to restrict* to a specific definition of relevance: we just assume that each individual subgraph is characterized by a degree of relevance rel, and combine relevance with preferences and diversity for performing top-k computations.

Preferences. For personalizing keyword queries results, we incorporate *user preferences* [7]. In general, preferences can be distinguished between *qualitative* and *quantitative* ones. In the former, a user specifies a binary preference relation $\mathcal{P}$, $\mathcal{P} = \{(w_i \succ w_j)\}$, where $w_i \succ w_j$ denotes that the user prefers keyword w_i over w_j. Given $\mathcal{P}$, we would like to rank the graphs in $R(Q)$ of a query Q. To do this, we use the fact that, for $w_i \succ w_j$, graphs containing w_i are preferred over graphs containing w_j. One way to assign preference scores to graphs is by using the winnow operator [2]. The intuition is to give the highest score to the most preferred graphs, that is, to the graphs for which there is no other graph in $R(Q)$ that is preferred over them. So, winnow at level 1 returns the most preferable graphs. An additional application of winnow, i.e., winnow at level 2, returns the next most preferred graphs, and so forth. In general, repeated applications of winnow result in ranking all graphs in $R(Q)$. We associate a preference score $pref$ with each graph in $R(Q)$ based on the winnow level their keywords belong to. In the latter, instead of providing $\mathcal{P}$, users explicitly provide specific scores $pref$ to interesting keywords. A higher score indicates a more important keyword. Then, scores to graphs are assigned with respect to the keywords they contain. Either qualitatively or quantitatively, preferences can be defined over the keywords of the nodes, edges and labels of the graph G. To impose an ordering of results when the results do not contain keywords in $\mathcal{P}$, we use the concept of *indirect dominance* [6]. Abstractly, a graph $H \in R(Q)$ that is related to a keyword w_i, not contained in H, is preferred over a graph $J \in R(Q)$ that is related to a keyword w_j, not contained in J, if $w_i \succ w_j$.

For structured queries, [1] focuses on personalizing XML search via qualitative preferences that target at: (i) expanding or restricting the original query result by adding, removing or replacing query predicates and (ii) specifying how to rank answers. Only recently, [4] sketches an IR methodology for training RDF-specific ranking functions.

Diversity. Top-results in $R(Q)$ are often very similar to each other, since they contain the same highly relevant and preferable piece of information. Caring for the quality of the result set as a whole, in our work, besides pure accuracy

achieved by relevance and user preferences, we also consider retrieving results on a broader variety of topics, i.e., increasing the diversity of results. To achieve diversity, we target at avoiding the overlap among results, i.e., choosing graphs that are dissimilar to each other. For quantifying the overlap between two graphs, we use a Jaccard-like definition of distance, which measures dissimilarity between the keywords of the nodes that form these graphs. This objective is enhanced by taking into account the semantic similarity of classes and properties that can be inferred by the schema, or even OWL-like axioms[1] that define explicitly, for instance, disjointness of classes. Then, the diversity, *div*, of a set of graphs is computed with respect to their distances from each other.

Top-k selection. Given a budget k on the number of results for a keyword query Q, we would like to combine the order of results as indicated by the user preferences with the results relevance. Furthermore, to increase user satisfaction, we consider the results as a whole and aim at selecting those that differ from each other to achieve diversity.

This way, let f be a cost function that combines the above properties and maps a set of resulting graphs onto a score with respect to Q. If $|R(Q)| > k$, our goal is to select the subset of k results that maximizes f. Formally:

Definition 1. *Given an RDF graph G, a keyword query Q with results $R(Q)$ and a cost function f, the top-k results is the set B^* for which: $B^* = argmax_{B \subseteq R(Q), |B|=k} f(B)$.*

4 Summary

In this paper, we have shown the growing importance of introducing new keyword search mechanisms for RDF data. The user experience in querying an RDF graph can be significantly improved by exploiting different criteria for ranking query answers towards meeting the users information needs. Criteria are oriented on ranking based on relevance, preferences, as well as on the diversity of the answer set as a whole. The problem of exploring how to combine such criteria is challenging. Research progress in this area does not necessarily demand working from scratch; a different point of view on how to employ or adapt existing algorithms and techniques should also be considered.

Acknowledgments. The work of the first author is partially supported by the project "IdeaGarden" funded by the Seventh Framework Programme under grand n^o 318552.

References

1. Amer-Yahia, S., Fundulaki, I., Lakshmanan, L.V.S.: Personalizing xml search in pimento. In: ICDE (2007)

[1] http://www.w3.org/TR/owl2-profiles.

2. Chomicki, J.: Preference formulas in relational queries. ACM Trans. Database Syst. **28**(4), 427–466 (2003)
3. Elbassuoni, S., Blanco, R.: Keyword search over RDF graphs. In: CIKM (2011)
4. Giannopoulos, G., Biliri, E., Sellis, T.: Personalizing keyword search on RDF data. In: Aalberg, T., Papatheodorou, C., Dobreva, M., Tsakonas, G., Farrugia, C.J. (eds.) TPDL 2013. LNCS, vol. 8092, pp. 272–278. Springer, Heidelberg (2013)
5. Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M.: RQL: a declarative query language for RDF. In: WWW (2002)
6. Stefanidis, K., Drosou, M., Pitoura, E.: Perk: personalized keyword search in relational databases through preferences. In: EDBT (2010)
7. Stefanidis, K., Koutrika, G., Pitoura, E.: A survey on representation, composition and application of preferences in database systems. ACM Trans. Database Syst. **36**(3), 19 (2011)
8. Tao, Y., Papadopoulos, S., Sheng, C., Stefanidis, K.: Nearest keyword search in XML documents. In: SIGMOD (2011)