Understanding Linked Open Data through Keyword Searching: the KEYRY approach[*]

Sonia Bergamaschi
University of Modena and Reggio Emilia, Italy
sonia.bergamaschi@unimore.it

Francesco Guerra
University of Modena and Reggio Emilia, Italy
francesco.guerra@unimore.it

Silvia Rota
University of Modena and Reggio Emilia, Italy
silvia.rota@unimore.it

Yannis Velegrakis
University of Trento, Italy
velgias@disi.unitn.eu

ABSTRACT

We introduce KEYRY, a tool for translating keyword queries over structured data sources into queries formulated in their native query language. Since it is not based on analysis of the data source contents, KEYRY finds application in scenarios where sources hold complex and huge schemas, apt to frequent changes, such as sources belonging to the linked open data cloud. KEYRY is based on a probabilistic approach that provides the top-k results that better approximate the intended meaning of the user query.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Query formulation

General Terms

Design, Algorithms

Keywords

Semantic keyword-based searching, Intensional Knowledge

1. INTRODUCTION

Linked open data has become one of the fundamental blocks for achieving interoperability on the Web. It is a large corpus of linked information contributed by different organizations and individuals. This information is exploited during query answering to translate or associate data expressed in terms of one vocabulary, into equivalent terms of another. The translation takes place by querying the corpus. The corpus has been modeled in RDF, thus, the query language of choice is naturally SPARQL. SPARQL is a structured language, which means that the user is still required to have some idea about the structure and the semantics of the data held. The majority of linked open data is coming from well-known data sources with a publicly available schemas. Nevertheless, these schemas are large and complex, thus making their knowledge difficult to acquire.

In recent years we have witnessed an exponential increase on the amount of linked open data and the number of sources that contribute to it. Nevertheless, these sources hold complex and huge schemas, thus difficult to be understood. Users have a hard time navigating the labyrinth of this huge information and an even harder formulating the right SPARQL queries to retrieve the required links. Keyword querying is becoming an attractive alternative, since the users typically do not need to elaborate on the many technical details of the data structures. Instead they can provide through a flat list of keywords a general description of the data that is desired. To be answered over the linked data, a keyword query first has to be translated into a SPARQL query that describes its semantics. Unfortunately, the simplicity of keyword queries comes with a price. Their semantics will have to be discovered. There has been already a lot of work on answering queries over relational data [3, 6]. These techniques heavily rely on an a-priori instance-analysis that scans the whole data instance and constructs an index, a symbol table or some structure that is later used during run time to identify the parts of the database in which each keyword appears. For linked open data, this is not a feasible solution. The linked data is not a collection of data physically stored in a centralized repository. Instead, it is a federation of data sources where the links among the data values is physically distributed across the many sources. Thus, creation and maintenance of a centralized global index offline is not an efficient option.

In this work, we describe a KEYRY(from KEYword to queRY), a tool that translates keyword queries over RDF data with schema into SPARQL queries. The novelty of the approach is that it does not assume any knowledge about the data source. It rather extracts and uses semantics to discover the intended meaning of the keywords and expresses them in terms of the underlying data structures, which are then combined together to form a SPARQL query. By turning keyword queries into SPARQL queries, KEYRY allows the exploitation of linked open data using a simple and intuitive interface. Furthermore, it offers an excellent tool for the browsing and discovery of the linked open data. A quick look on the linked data cloud is enough for someone to realize its size and complexity. In that it is definitely hard to locate some specific piece of data that actually describes the intended information. With KEYRY, a user may form a keyword query with the main concepts of the information the user is looking for. KEYRY analyzes the query and generates its possible interpretations as SPARQL queries over the linked data. The user can look at the list of these queries, thus being guided in identifying the piece of data s/he had initially in mind.

In order to achieve this result, KEYRY uses the semantics provided by the data source, auxiliary semantic information that is freely available (i.e., public ontologies and thesauri), a probabilistic Hidden Markov Model (HMM) and an adapted notion of authority [5],

[*]Work partially supported by project "Searching for a needle in mountains of data" (http://www.dbgroup.unimo.it/keymantic/)

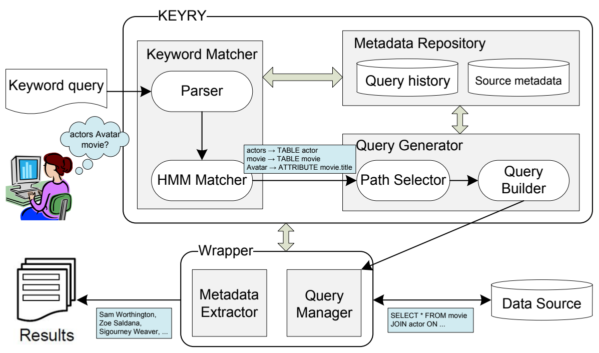

Figure 1: KEYRY functional architecture

to build a ranked list of the possible interpretations. In practice, the tool only computes the top-k most prominent interpretations and not the whole answer space. One of the innovative features of KEYRY is the fact that the keywords in a query are not seen as independent entities. During the discovery of the data source structures that serve as potential candidates for describing the intended semantics of a keyword, the mapping of adjacent keywords to other data source structures are taken into consideration. This leads to more meaningful interpretations and, at the same time, significantly reduces the search space. KEYRY is based on the design principles of Keymantic [1, 2], but its novelty lies in its HMM computational mechanism to provide a set of SPARQL candidate queries. Furthermore, while Keymantic is targeted to strictly relational data, KEYRY is specially designed to handle RDF, OWL and XML data.

2. KEYRY OVERVIEW

Figure 1 provides the general architecture of KEYRY. When a keyword query is issued, KEYRY first finds the structures that contain these keywords. A keyword in a query may in principle be found in any data source element (i.e., as the name of some data structure or a value in its respective domains). This gives rise to a large number of possible *configurations* (i.e., combinations of each keyword in a query with a data source element). Since no knowledge of the actual linked data values is assumed a-priori existing , selecting the best top-k configurations is a challenging task. Let us suppose that a user is posing the query “`movie Avatar actors`" over the Linked Movie Database[1]. The keywords in the query may be associated to different structures/data values, representing in that way different semantics of the intended meaning of the user query. For example, the user might be looking for the actors acting in the movie with title Avatar, and, consequently, the keywords `movie` and `actors` could be respectively mapped into the OWL classes `movie` and `actor` while `Avatar` could be a value of the property `title` of the class `movie`. Other semantic interpretations are also possible, e.g. `Avatar` could denote an actor, a character, or something else.

The *Keyword Matcher* is the component in charge for computing the top-k best matches of the keywords to the data. It employs a first order HMM that takes into account both the likelihood of mapping of keywords into data source structures considered in isolation and as the whole sequence of keywords. In this way, the assignment of a keyword to a data element may increase or decrease the likelihood that another keyword corresponds to some data element.

[1]http://www.linkedmdb.org/, a project that aims at publishing an open semantic web database for movies, including a large number of interlinks to several datasets on the open data cloud and references to related webpages.

In our approach, the HMM states represent data source elements while the observations describe the keywords in a query. The HMM parameters are initialized by exploiting the semantic information collected in the *Metadata Repository* including metadata about data sources structures and the history of the previous user queries. By applying the Viterbi algorithm [4] to the HMM, we find the state sequence (i.e. the data source elements) that more likely represents the intended meaning of the user query. This is achieved through an iterative process that computes, for each two consecutive elements in the observation sequence and for each state S_i, the most probable state S_j following S_i. The best solution is the path joining the state per observation having the highest sum of scores associated to the transition. The score definition takes into account the previous state score, the probability associated to the transition from the previous state to the current state, and the probability that the keyword under evaluation is one of the emissions associated to the state in exam. We extended the Viterbi algorithm by calculating the top-k solutions instead of the best one to reduce the number of generated queries. In our approach transition probabilities variate in time. At each step, the probability values are updated following two criteria: to force the association of some keywords to specific states (this happens if the state refers to structural data source elements and the similarity measure with the user keyword is greater than a predetermined threshold) and to prevent matching two keywords into the same data source element (to avoid associations of multiple values to the same data source elements, which is not possible due to the atomicity of the attribute values in a data source).

Once the matches have been identified, the possible paths joining them are computed. Different paths correspond to different interpretations. For instance, the configuration in which the keywords `movie` and `actor` are mapped into the classes `movie` and `actor`, respectively, and the keyword `Avatar` to the property `title` of the class `movie` may represent the actors that acted in the movie Avatar, or the movies where acted actors of the movie Avatar, etc., depending on the path selected and the classes involved. The path computation is the main task of the *Query Generator* module. Different strategies have been proposed in the literature to select the most prominent one, or to provide an internal ranking based on different criteria, such as the length of the paths. In KEYRY, we propose two criteria: one based on the shortest path, the second is an adaptation of the HITS algorithm [5] which classifies the relevance of the data structures involved in a path.

REFERENCES

[1] S. Bergamaschi, E. Domnori, F. Guerra, R. T. Lado, and Y. Velegrakis. Keyword Search over Relational Databases: a Metadata Approach. In *Proc. of SIGMOD 2011, Athens, Greece, June 12-16*. ACM, 2011.

[2] S. Bergamaschi, E. Domnori, F. Guerra, M. Orsini, R. T. Lado, and Y. Velegrakis. Keymantic: Semantic Keyword-based Searching in Data Integration systems. *PVLDB*, 3(2):1637–1640, 2010.

[3] S. Chakrabarti, S. Sarawagi, and S. Sudarshan. Enhancing search with structure. *IEEE Data Eng. Bull.*, 33(1):3–24, 2010.

[4] J. Forney, G.D. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268 – 278, 1973.

[5] L. Li, Y. Shang, H. Shi, and W. Zhang. Performance evaluation of hits-based algorithms. In *Communications, Internet, and Information Technology*, pages 171–176. IASTED/ACTA Press, 2002.

[6] J. X. Yu, L. Qin, and L. Chang. *Keyword Search in Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Pub., 2010.