KEYRY: A Keyword-Based Search Engine over Relational Databases Based on a Hidden Markov Model*

Sonia Bergamaschi[1], Francesco Guerra[1], Silvia Rota[1], and Yannis Velegrakis[2]

[1] Università di Modena e Reggio Emilia, Italy
firstname.lastname@unimore.it
[2] University of Trento, Italy
velgias@disi.unitn.eu

Abstract. We propose the demonstration of KEYRY, a tool for translating keyword queries over structured data sources into queries in the native language of the data source. KEYRY does not assume any prior knowledge of the source contents. This allows it to be used in situations where traditional keyword search techniques over structured data that require such a knowledge cannot be applied, i.e., sources on the hidden web or those behind wrappers in integration systems. In KEYRY the search process is modeled as a Hidden Markov Model and the List Viterbi algorithm is applied to computing the top-k queries that better represent the intended meaning of a user keyword query. We demonstrate the tool's capabilities, and we show how the tool is able to improve its behavior over time by exploiting implicit user feedback provided through the selection among the top-k solutions generated.

1 Introduction

The vast majority of existing keyword search techniques over structured data relies heavily on an a-priori creation of an index on the contents of the database. At run time, the index is used to locate in the data instance the appearance of the keywords in the query provided by the user, and then associate them by finding possible join paths. This approach makes the existing solutions inapplicable in all the situations where the construction of such an index is not possible. Examples include databases on the hidden web, or behind wrappers in integration systems. To cope with this issue we have developed KEYRY (from KEYword to queRY) [2], a system that converts keyword queries into structured queries expressed in the native language of the source, i.e., SQL. The system is based only on the semantics provided by the source itself, i.e., the schema, auxiliary semantic information that is freely available, i.e., public ontologies and thesauri, a Hidden Markov Model (HMM) and an adapted notion of authority [4]. Using this information, it builds a ranked list of possible interpretations of the keyword query in terms of the underlying data source schema structures. In practice, the tool computes only the top-k most prominent answers and not the whole answer space. One of the features of KEYRY is that the order and the proximity of the keywords in the queries play a central role in determining the k most prominent interpretations.

* This work was partially supported by project "Searching for a needle in mountains of data" http://www.dbgroup.unimo.it/keymantic.

O. De Troyer et al. (Eds.): ER 2011 Workshops, LNCS 6999, pp. 328–331, 2011.

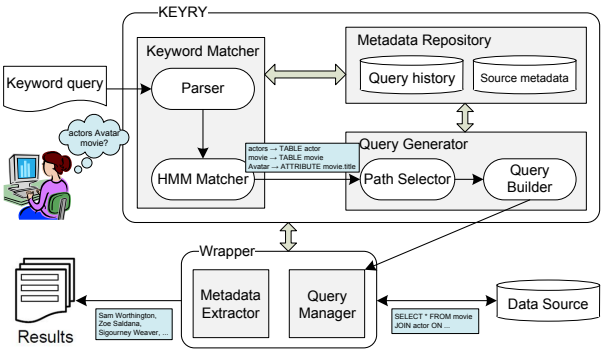

Fig. 1. KEYRY functional architecture

Apart from the obvious use case of querying hidden web sources, KEYRY finds two additional important applications. First, it allows keyword queries to be posed on information integration systems that traditionally supports only structured queries. Second, it can be used as a database exploration tool. In particular, the user may pose some keyword query, and the system will generate its possible interpretations on the given database. By browsing these interpretations, i.e., the generated SQL queries, the user may obtain information on the way the data is stored in the database.

2 KEYRY at a Glance

Interpreting a keyword query boils down to the discovery of a bipartite assignment of keywords to database structures. In some previous work we have studied the problem from a combinatorial perspective [1] by employing the the Hungarian algorithm [3]. The tool we demonstrate here is instead based on a Hidden Markov Model. A detailed description of the methodology can be found on our respective research paper [2]. A graphic illustration of the different components of the tool can be found in Figure 1.

The first step when a keyword query has been issued is to associate each keyword to some database structure. That structure may be a table name, a class, an attribute, or an actual value. Each assignment of the keywords to database structures is a *configuration*. The computation of the different configurations is done by the *Keyword Matcher* component. A configuration describes possible meanings of the keywords in the query. Given the lack of access to the instance data that we assume, finding the configurations and subsequently the interpretations of a keyword query and selecting k most prominent is a challenging task. Consider for instance the query "movie Avatar actors" over a relational database collecting information about movies. The query may be intended to find actors acting in the movie "Avatar" which means that the keywords movie and actors could be respectively mapped into the tables `movie` and `actor` while Avatar could be a value of the attribute `title` of the table `movie`. Different semantics are also possible, e.g. Avatar could be an actor or a character, or something else.

The *Keyword Matcher* is the main component of the system. It implements a Hiden Markov Model (HMM), where the observations represent the keywords and the states the data source elements. The main advantages of this representation are that the HMM

takes into account both the likelihood of single keyword-data source element associations, and the likelihood of the match between the whole keyword sequence in the query to the data source structure. In this way, the assignment of a keyword to a data source element may increase or decrease the likelihood that another keyword corresponds to a data source element. In order to define a HMM, the values of a number of parameters need to be specified. This is usually done using a training algorithm that, after many iterations, converges to a good solution for the parameter values. Therefore, finding a suitable dataset for training the HMM is a critical aspect for the effective use of HMM-based approaches in real environments. In our scenario, the parameters are initialized by exploiting the semantic information collected in the *Metadata Repository* (explained below), providing that way keyword search capabilities even without any training data, as explained in more details in our paper [2].

To obtain more accurate results, the HMM can be trained, i.e. the domain-independent start-up parameters may be specialized for a specific data source thanks to the users' feedbacks. We train KEYRY with a semi-supervised approach that exploits both the feedbacks (when available) implicitly provided by the users on the system results as supervised data and the query log as unsupervised data. By applying the List Viterbi algorithm [5] to an HMM, the top-k state sequences which have the highest probability of generating the observation sequence are computed. If we consider the user keyword query as an observation sequence, the algorithm retrieves the state sequences (i.e., sequences of HMM states representing data source terms) that more likely represent the intended meaning of the user query.

The HMM Matcher can present the computed top-k configurations to the user who, in turn, selects the one that better represents the intended meaning of his/her query. This allows the tool to reduce the number of queries to be generated (the ones related to the configuration selected) and to train the HMM parameters.

Given a configuration, the different ways that the data structures on which the keywords have been mapped can be connected, need to be found (e.g., by discovering join paths). A configuration alongside the join paths describe some possible semantics of the whole keyword query, and can be expressed into some query in the native query language of the source. Such queries are referred to as *interpretations*. As an example, consider the configuration mentioned above in which the keywords movie and actor are mapped into the tables `movie` and `actor`, respectively, while the keyword Avatar to the `title` of the table `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 tables involved. The path computation is the main task of the *Query Generator* module. Different strategies have been used in the literature to select the most prominent one, or provide an internal ranking based on different criteria, such as the length of the join paths. KEYRY uses two criteria: one is based on the shortest path and the other is using the HITS algorithm [4] to classify the relevance of the data structures involved in a path.

The tasks of generating the various configurations and subsequently the different interpretations are supported by a set of auxiliary components such as the *Metadata Repository* that is responsible for the maintenance of the metadata of the data source structures alongside previously executed user queries. KEYRY has also a set of *Wrappers* for managing the heterogeneity of the data sources. Wrappers are in charge of

extracting metadata from data sources and formulating the queries generated by KEYRY in the native source languages.

3 Demonstration Highlights

In this demonstration we intend to illustrate the use of KEYRY and communicate to the audience a number of important messages. The demonstration will consist of a number of different application scenarios such as querying the IMDB database (www.imdb.com), the DBLP collection (dblp.uni-trier.de) and the Mondial database (http://www.dbis.informatik.uni-goettingen.de/Mondial). We will first show the behavior of KEYRY without any training. We will explain how the metadata of the sources is incorporated into our tool and how heuristic rules allow the computation of the main HMM parameters. We will run a number of keyword queries against the above sources and explain the results. These queries are carefully selected to highlight the way the tool deals with the different possible mappings of the keywords to the database structures. The participants will also have the ability to run their own queries. Furthermore, we will consider the parameters obtained by different amounts of training and we will compare the results to understand how the amount of training affects the final result.

The important goals and messages we would like to communicate to the participants though the demo are the following. First, we will demonstrate that keyword-based search is possible even without prior access to the data instance, and is preferable from formulating complex queries that require skilled users who know structured query languages and how/where the data is represented in the data source. Second, we will show that using a HMM is a successful approach in generating SQL queries that are good approximations of the intended meaning of the keyword queries provided by the user. Third, we will illustrate how previously posed queries are used to train the search engine. In particular, we will show that the implicit feedback provided by the user selecting an answer among the top-k returned by the system can be used for supervised training. We will also demonstrate that, even in the absence of explicit users' feedbacks, the results computed by the tool may still be of high enough quality. We will finally demonstrate that each user query may be associated to several possible interpretations which can be used to reveal the underline database structure.

References

1. Bergamaschi, S., Domnori, E., Guerra, F., Lado, R.T., Velegrakis, Y.: Keyword search over relational databases: a metadata approach. In: Sellis, T.K., Miller, R.J., Kementsietsidis, A., Velegrakis, Y. (eds.) SIGMOD Conference, pp. 565–576. ACM, New York (2011)
2. Bergamaschi, S., Guerra, F., Rota, S., Velegrakis, Y.: A Hidden Markov Model Approach to Keyword-based Search over Relational Databases. In: De Troyer, O., et al. (eds.) ER 2011 Workshops. LNCS, vol. 6999, pp. 328–331. Springer, Heidelberg (2011)
3. Bourgeois, F., Lassalle, J.-C.: An extension of the Munkres algorithm for the assignment problem to rectangular matrices. Communications of ACM 14(12), 802–804 (1971)
4. Li, L., Shang, Y., Shi, H., Zhang, W.: Performance evaluation of hits-based algorithms. In: Hamza, M.H. (ed.) Communications, Internet, and Information Technology, pp. 171–176. IASTED/ACTA Press (2002)
5. Seshadri, N., Sundberg, C.-E.: List Viterbi decoding algorithms with applications. IEEE Transactions on Communications 42(234), 313–323 (1994)