Keymantic: Semantic Keyword-based Searching in Data Integration Systems*

Sonia Bergamaschi
University of Modena and Reggio Emilia, Italy
{fname.lname}@unimore.it

Elton Domnori
University of Modena and Reggio Emilia, Italy
{fname.lname}@unimore.it

Francesco Guerra
University of Modena and Reggio Emilia, Italy
{fname.lname}@unimore.it

Mirko Orsini
University of Modena and Reggio Emilia, Italy
{fname.lname}@unimore.it

Raquel Trillo Lado
University of Zaragoza, Spain
raqueltl@unizar.es

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

ABSTRACT

We propose the demonstration of *Keymantic*, a system for keyword-based searching in relational databases that does not require a-priori knowledge of instances held in a database. It finds numerous applications in situations where traditional keyword-based searching techniques are inapplicable due to the unavailability of the database contents for the construction of the required indexes.

1. WHY KEYMANTIC?

Keyword queries have become a popular alternative to structured query languages, since they do not require the users to have a good knowledge of the way data has been organized in the source. Keyword-based searching techniques on databases [6] and on XML documents [5] typically rely on the construction of specialized indexes on the instances. These indexes are used to identify at run-time the database objects corresponding to the keywords. Unfortunately, there are many practical application scenarios in which construction of such indexes is not possible. One such scenario is the case of integration systems that follow the virtual integration architecture. Due to the lack of a materialized global instance, the required indexes to support keyword queries cannot be constructed in advance. A solution is to construct the index based on the contents of the individual local sources. Moreover, sources do not always expose the full details of their contents to the integration system, but instead, provide access to it through predefined question/answer interfaces.

With all these in mind, we have developed *Keymantic* [1], a system that answers keyword queries over relational data systems by relying only on intensional knowledge, i.e., information extracted (or provided) by the data sources, such as, schemas, data types, lexical references and mappings, and additional knowledge that is publicly available on the Web, such as lexical resources, ontologies etc. The system uses that knowledge to translate the ambiguous keyword queries into fully specified SQL expressions.

Apart from the obvious functionality of answering keyword queries over relational systems, *Keymantic* also serves as a tool for helping users to understand how some information in which they are interested has been modelled into large complicated and unknown schemas. The user expresses his/her request of information as a keyword query. *Keymantic* tries to guess the intentions of the user by generating a list of possible interpretations in terms of the underlying database structures. So, the user can browse through the different interpretations and select the one whose semantics is closer to the intended meaning of his/her query.

The challenging task in *Keymantic* is the development of a methodology for combining different types of intensional knowledge into a coherent framework that can efficiently discover the possible interpretations of a keyword query and express them as SQL. To efficiently achieve this, it is needed to limit the number of database elements, i.e., tables, attributes, or attribute values that a keyword may correspond. This process should take into consideration the fact that any assignment of a keyword to a database term affects the assignment of the remaining keywords. One of the novelties of *Keymantic* is the use of the Hungarian algorithm adapted to deal with the above challenges.

2. MOTIVATION

Consider the case of a database system with the schema indicated in Figure 1, and the keyword query "Restaurant Naples". The intention of the query may be asking for information about a restaurant called Naples, a restaurant in the city of Naples, a restaurant located in Avenue Naples, or even restaurants that offer Naples specialities. To find the intended semantics of the keyword query, we need to discover to which element of the database each keyword corresponds. Each keyword represents a piece of information that may have been modeled in the database as a table, as an attribute, or as an atomic value of an attribute. Thus, the

*Work partially supported by project "Searching for a needle in mountains of data" (http://www.dbgroup.unimo.it/keymantic/), CICYT project TIN2007-68091-CO2-02 and EU grant ICT-215874.

Proceedings of the VLDB Endowment, Vol. 3, No. 2

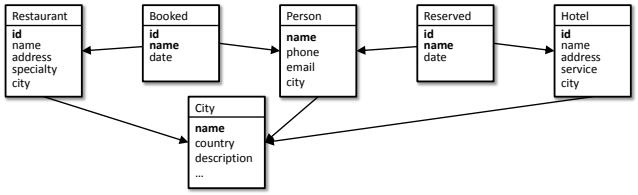

Figure 1: A fraction of a database schema.

first challenge in the translation of the possible semantics of a keyword query is to obtain the semantically meaningful assignments of each keyword in a query to some element of the database. For instance, in the above example, the keyword Naples may correspond to the value of the attribute `Name` in the table `Restaurant`, or the attribute `Name` of the table `City`, etc., while the keyword `Restaurant` may be mapped to the table with the same name.

It is generally accepted the fact that keywords in a query are not independent. It is natural to assume that they express different features of the concept that the query is about. Thus, any assumption about the semantics of one of the keywords, indirectly affects the possible semantics of the rest. For instance, in the keyword query "Restaurant name Naples", if we assume that the first keyword is assigned to the table `Restaurant`, then the keyword *name* is likely to represent the attribute `Name` of the table `Restaurant` and be mapped to it. Furthermore, assuming the previous assignments, the third keyword is likely to represent the actual name of the restaurant, thus, it should be considered to be a value of the attribute `Name` of the table `Restaurant`. This kind of interdependency among the possible mappings of the keywords in a query can significantly reduce the different assignments that have to be considered in the process of translating the keyword query into SQL.

The order of the keywords in a query is also playing an important role [4]. The intention of the query "restaurant Obama New York" is probably looking for "the restaurant called Obama in New York city", while the intention of "restaurant New York Obama" is probably for "the restaurant in New York city that Obama has visited".

An assignment of keywords to database elements is not enough to specify the full semantics of a keyword query. It is also important to understand how the database structures are related to each other. This is typically determined by the different join paths in the relational database. For instance, consider the query "Person USA" with the first keyword mapped to the table `Person` and the second to the value of the attribute `Country` in the table `City`. As it can be seen in Figure 1 there are different join paths between the table `Person` and and the attribute `Country` of the table `City`. Considering the shortest one means that the keyword query is about a person living in the USA, while considering the one that goes through `Hotel` means that the query is about persons that have reserved hotels located in the USA.

Finding the different semantic interpretations of a keyword query turns out to be a combinatorial problem which can be solved by an exhaustive enumeration of the different assignments to database structures and values. However, internal and external knowledge can help eliminating mappings that are likely not to lead to meanings intended by the user. For instance, given the keyword query `320-463-1463`, and knowing that this is typically the format of a phone number, an interpretation based on an assignment of the keyword to the attribute `Address` is unlikely to be among the ones intended by the user.

3. CHALLENGES AND SOLUTIONS

We call a *configuration* any mapping of the keywords in a query into the vocabulary of a database (V_D), i.e., table names, their attributes and their respective domains. The latter ones are referred to *value database terms*, as opposed to the *schema database terms* (the tables and the attributes).

Since each keyword can be mapped to any database term, there are $2*\Sigma_{i=1}^{|D|}|R_i| + |D|$ alternatives to which a query keyword can be mapped, with $|R_i|$ denoting the *arity* of the relation R_i and $|D|$ the number of tables in the database. Based on this, and on the fact that no two keywords can be mapped to the same database terms, for N keywords, there are $\frac{|V_D|!}{(|V_D|-N)!}$ possible configurations. The decision to map each keyword to one database term is a result of two factors. First, the desire to generate interpretations of the keyword query in the form of SPJ queries. If keywords are allowed to be mapped to different terms, this will lead to queries with disjunctions not only at the attribute level, but also at the table level, which will make factorial the number of possible interpretations. Furthermore, through studies on a number of real keyword query sets, we have found only exceptional cases in which a keyword within a query has at the same time more than one meaning, something that would mean a mapping of the keyword to more than one database terms.

Of course, not all the configurations are equal: we introduce the notion of a *weight* between a keyword and a database term to offer a quantitative measure for the relativeness of a keyword to a database term, and indirectly, the likelihood that the semantics of the keyword in the intended interpretation of the query is the one described by the respective database term. The sum of the weights of the mappings in the configuration can serve as a quantitative measure for the importance of a configuration in describing the intended keyword semantics.

The naive approach for selecting the best configurations is the computation of the score of each possible configuration and then selecting those with the highest score. Due to the amount of possible combinations, such an approach is unfeasible. The problem of computing the mapping with the maximum score without an exhaustive computation of the scores of all the possible mappings is known in the literature as the problem of *Bipartite Weighted Assignments* [3]. Unfortunately, existing algorithms to solve this problem suffer from two main limitations. Firstly, they do not consider interdependencies that may exist between the mappings, apart from the mutual exclusiveness. Nevertheless, exploiting such interdependencies may improve the performances, since keywords in a query typically all refer to the same context. Secondly, they provide the single best mapping, instead of the best ones.

To cope with the first limitation, we introduce two different kinds of weights, the *intrinsic*, and the *contextual*. Intrinsic weights measure the likelihood that a keyword should be mapped to a database term independently of the other query keywords or database terms. It is based on syntactic, semantic and structural factors such as the attribute names, relation names, etc., or other auxiliary external sources,

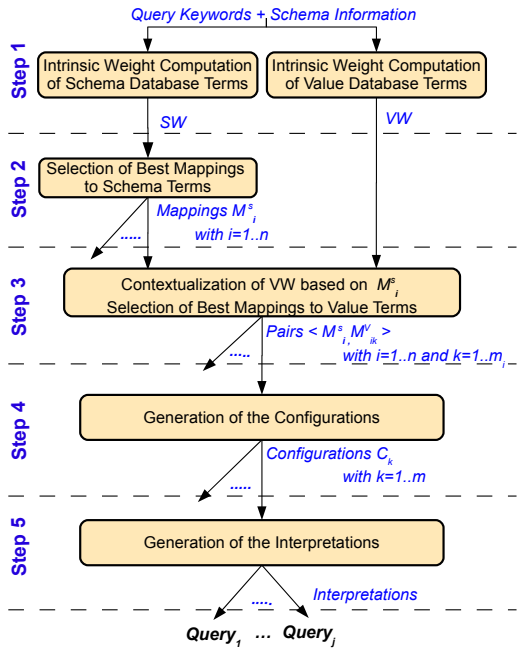

Figure 2: The keyword query translation process.

such as vocabularies, ontologies, domains, etc. On the other hand, a contextual weight is used to measure the relativeness of a keyword to a database term by taking into consideration the mappings of other keywords. This is motivated by the fact that the assignment of a keyword to a database term may increase or decrease the likelihood that another keyword corresponds to a certain database term.

To cope with the second limitation, we have developed a novel algorithm for computing the best mappings based on the Hungarian (a.k.a., Munkres) algorithm [2].

Figure 2 provides an overview of the translation process of the keyword queries to SQL. A special data structure, called *weight matrix*, is heavily used in that process. The *weight matrix* is a two-dimensional array with one row for each keyword in the keyword query, and one column for each database term. The value of a cell represents the weight between the respective keyword and the database term. Two parts (i.e., sub-matrices) can be distinguished in the weight matrix. One, denoted as SW, corresponds to the database terms related to schema elements, i.e., *schema database terms* (relational tables and attributes), and another, denoted as VW, that corresponds to attribute values, i.e., *value database terms* (the domains of the attributes).

Intrinsic Weight Computation. The weight matrix is populated with the intrinsic weights. The computation of these weights is achieved by exploitation and combination of a number of semantic techniques based on structural and lexical knowledge extracted from the data source, and external knowledge bases. The output of this step are the populated SW and VW submatrices of the weight matrix.

Selection of Best Mappings to Schema Terms. Having the populated weight matrix, a number of prominent mappings of certain keywords to schema database terms are generated. The input to this task is the SW sub-matrix and the outcome is a series of mappings M_i^S, with i=1..n. Each such mapping assigns a number of keywords to schema database terms. Those that remain unassigned will be later

Algorithm 1: Keywords to DB Terms Assignment

Input: $I(i_{ij})$ where $I \equiv SW$ or $I \equiv VW$
Output: $M^I = \{M_1^I, \ldots, M_z^I\}$: Mappings generated by I

MAPPING(I, W_{MAX})
(1) $tempM = \bigcup i_{pt} \leftarrow HUNGARIAN_{Ext} * (I)$
(2) $W \leftarrow \sum i_{pt}$
(3) $M^I \leftarrow tempM$
(4) **if** $(W > c * W_{MAX})$
(5) $W_{MAX} \leftarrow W$
(6) **while** $(W > c * W_{MAX})$
(7) **foreach** $i_{pt} \in tempM$
(8) $i_{pt} \in I \leftarrow -100$
(9) $Mapping(I, W_{MAX})$

mapped to value database terms. From the different partial mappings only those that have a score higher than a specific threshold are kept. The generation of the mappings is performed by the Hungarian algorithm that we have extended not to stop after the generation of the best mapping but to continue to the generation of the second best, the third, etc. Furthermore, some of the internal steps of the algorithm have been modified so that the weight matrix is dynamically updated every time that a mapping of a keyword to a database term is decided during the computation. This adaptation is needed to reflect the consequences of a specific assignment to the semantic relativeness of the remaining mappings to the remaining unmapped database terms. Algorithm 1 depicts the overall process of computing the set of mappings of keywords to database terms, given the weight matrix. The HUNGARIAN$_{Ext}$ expression refers to our extended version of the Hungarian algorithm. The original algorithm for rectangular matrices has a complexity $O(n^2 * m)$, where n is the number of keywords and m is the number of databases terms. Extending the algorithm to consider contextual weights brings the complexity to $O(n^3 * m^2)$ which is due to the fact that a mapping may affect any of the other mappings, thus, in the worst case, $(n-1)*(m-1)$ weight updates may need to take place. Nevertheless, this worst case rarely happens since only a subset of the matrix is updated in each iteration, and, due to the threshold, not all the possible updates are evaluated.

Contextualization of VW and selection of Best Mappings to Value Terms. As a third step, each partial mapping of keywords to schema terms M_i^S is associated to a partial mapping M_{ik}^V of the remaining keywords to value database terms. Mappings of keywords to attribute values are computed by analyzing the submatrix VW with a two-phase process. Firstly, each partial mapping M_i^S generates a new updated VW submatrix, where the intrinsic value weights VW initially computed are updated to reflect the added value provided by the mapping M_i^S of certain keywords to schema database terms. Besides, the weights in VW of the keywords that have been mapped in M_i^S are set to zero. Secondly, given an updated VW submatrix, the most prominent mappings of the unmapped keywords to value database terms are generated. This is achieved by using again the adapted method of the Hungarian algorithm. The result is a series of partial mappings M_{ik}^V, with k=1..m_i, where i identifies the mapping M_i^S on which the computation of the updated matrix VW was based. The mapping M_{ik}^V is partial because it involves only the keywords that had

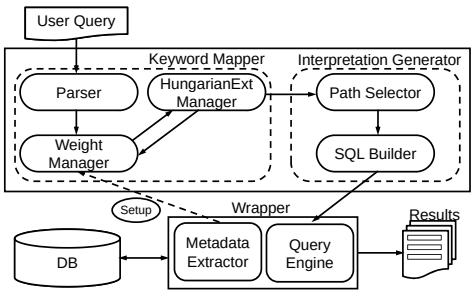

Figure 3: Keymantic architecture

not been mapped to some schema database term in M_i^S.

Generation of the Configurations. Each mapping M_{ik}^V together with its associated mapping M_i^S form a configuration C_k. The score of the configuration is the sum of the weights of the mapped pairs of keywords and database terms in M_{ik}^V and M_i^S.

Generation of the Interpretations. Having computed the best configurations, the SQL queries can be generated. Recall that a configuration consists only of mappings of the keywords to database terms. It does not specify how these terms are related to each other. Thus, in the presence of different join paths among the elements involved in a configuration, multiple queries may result from a single configuration. Different strategies can be used to select the most prominent one, or provide an internal ranking based on different criteria, such as the length of the join paths [6]. However, this is not the main focus of the current work and we will not elaborate further on it. We adopt a greedy approach that computes a query for every possible combination where the database terms involved in the configurations are related and the user may select the shorthest or another join-path among them.

4. SYSTEM IMPLEMENTATION

Keymantic consists of three main modules, as shown in Figure 3: the *Wrapper* responsible for extracting the source metadata, the *Keyword Mapper* that generates the different configurations alongside their weights, and the *Interpretation Generation* that generates the final SQL queries.

5. DEMONSTRATION HIGHLIGHTS

There are four main messages that we would like to communicate to the VLDB participants through our demonstration scenario. First, that keyword based searching can be successfully implemented even in the cases where access to the source data is not available. Second, that a combination of different metadata information provides expected translations of keyword queries into SQL interpretations. Third, that our extension of the Hungarian algorithm significantly reduces the number of possible SQL interpretations of a keyword query to those semantically meaningful. Finally, we will show that a combination of keyword queries and intensional information is effective for browsing large complicated data sources.

The demonstration will consist of two phases. In the first phase, we will load a number of different sources such as a large tourist database, IMDB, etc., and we will explain how the metadata information of the sources is incorporated into

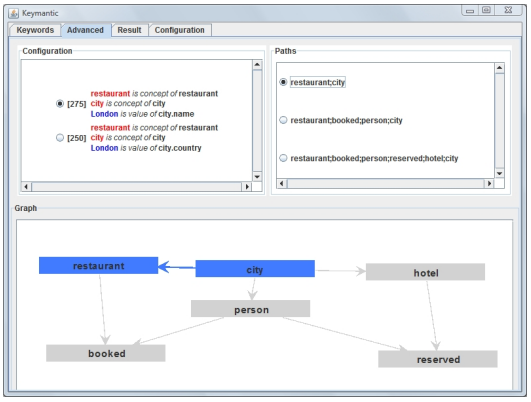

Figure 4: Part of the *Keymantic* interface.

our system. Then, we will run a number of keyword queries against these sources, and explain the results. The participants will be free to run their own queries, but a number of specially chosen keyword queries will be available to demonstrate how the system handles ambiguous queries that generate multiple possible mappings, each one with multiple possible paths associated. Query processing time will be provided alongside the different alternatives that have been considered to prove the effectiveness of the Hungarian algorithm. Queries of different sizes will be tested to also demonstrate how our algorithms scale up. The returned results to the keyword queries will be accompanied by the SQL query representing the interpretation of the keyword query that has been followed, in order to increase the confidence of the user for the quality of the returned results.

During the second phase, the users will pose keyword queries and the system will display the different interpretations without actually executing them. By browsing these interpretations, the users will be able to get a taste of the parts of the sources that are related to their interest (as described by the keyword query), something that would have been hard to do before due to the size and the complexity of the schemas. Furthermore, the users will be able to select those interpretations that they believe that more accurately reflect the semantics they had in mind when they were creating the keyword query. Thus, the user can guide the system to return results that are very related, increasing the precision of the whole process. Figure 4 illustrates how the different interpretations are displayed to the user.

REFERENCES

[1] S. Bergamaschi, E. Domnori, F. Guerra, R. T. Lado, and Y. Velegrakis. Keyword-based Searching in Databases using Intensional Knowledge. *Submitted*, 2010.

[2] F. Bourgeois and J.-C. Lassalle. An extension of the Munkres algorithm for the assignment problem to rectangular matrices. *Comm.of ACM*, 14(12):802–804, 1971.

[3] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems*. SIAM, Philadelphia, 2009.

[4] R. Kumar and A. Tomkins. A Characterization of Online Search Behavior. *IEEE Data Engineering Bulletin*, 32(2):3–11, 2009.

[5] Z. Liu, J. Walker, and Y. Chen. Xseek: A semantic xml search engine using keywords. In *VLDB*, pages 1330–1333. ACM, 2007.

[6] J. X. Yu, L. Qin, and L. Chang. *Keyword Search in Databases*. Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2009.