Keyword-based Search in Data Integration Systems*

Sonia Bergamaschi[1], Elton Domnori[1], Francesco Guerra[1], Raquel Trillo Lado[2], and Yannis Velegrakis[3]

[1] Università di Modena e Reggio Emilia, via Università 4, 41121 Modena, Italy
firstname.lastname@unimore.it
[2] SID - University of Zaragoza, María de Luna, 1, 50018 Zaragoza, Spain
raqueltl@unizar.es
[3] DISI - Università di Trento, Via Sommarive 14, 38123 Povo (TN), Italy
velgias@disi.unitn.eu

Abstract. In this paper we describe Keymantic, a framework for translating keyword queries into SQL queries by assuming that the only available information is the source metadata, i.e., schema and some external auxiliary information. Such a framework finds application when only intensional knowledge about the data source is available like in Data Integration Systems.

1 Introduction

One of the main motivations for supporting the research on data integration is to provide the user with a unique view synthesizing a set of distributed data sources. By querying a unique integrated view, the user obtains an answer that is the union of the results returned by the sources involved in the integration process. Since the benefits of this approach for end users are firm, the research community has been focused on techniques for building and querying the unified view [9] for over 20 years. Despite the results obtained in the field, data integration systems are not really permeating the real world, i.e., we register a low presence of data integration systems in real business environments. We think that one of the reasons is that querying in such environments is too complex for end users.

Data integration systems are typically queried by means of requests expressed in the native query languages (in general structured query languages such as SQL, OQL, SPARQL, ...). This is definitely a limit for a large exploitation of these systems: they require skilled users and also impose on data integration systems some of the limitations intrinsic to the query languages. A complete knowledge of the underlying data structures and their semantics is needed to formulate queries into a structured query language. Unfortunately, the former requires the user to deeply explore the structure of the source, that is an error prone and time consuming process when a source is composed of hundreds of unknown tables and attributes. The latter may be too large and too complex to be communicated to the user. Understanding the semantics conveyed by the

* Extended abstract of the paper "S. Bergamaschi, E. Domnori, F. Guerra, R. Trillo Lado, Y. Velegrakis: Keyword Search over Relational Databases: a Metadata Approach, to appear at SIGMOD 2011"

Person

Name	Phone	City	Email
Saah	4631234	London	saah@aaa.bb
Sevi	6987654	Auckland	eevi@bbb.cc
Edihb	1937842	Santiago	edibh@ccc.dd

Reserved

Person	Hotel	Date
Saah	x123	6/10/2009
Sevi	cs34	4/3/2009
Edihb	cs34	7/6/2009

Hotel

id	Name	Address	Service	City
x123	Galaxy	25 Blicker	restaurant	Shanghai
cs34	Krystal	15 Tribeca	parking	Cancun
ee67	Hilton	5 West Ocean	air cond.	Long Beach

City

Name	Country	Description
Shanghai	China	...
Cancun	Mexico	...
Long Beach	USA	...
New York	USA	...

Booked

Person	Rest	Date
Saah	Rx1	5/4/2009
Sevi	Rx1	9/3/2009

Restaurant

id	Name	Address	Specialty	City
Rx1	Best Lobster	25, Margaritas	Seafood	Cancun
Rt1	Naples	200 Park Av.	Italian	New York

Fig. 1. A fraction of a database schema with its data.

unified view means to know both the semantics conveyed by the data sources involved in the integration process and how the semantics of the local views are mapped to the integrated view. Therefore, it is clear that such a requirement may nullify the whole motivation for integrating sources: for users holding this knowledge the advantages of working with an integrated source are highly lowered.

Keyword-based searching has been introduced as a viable alternative to the highly structured query languages. A number of keyword searching systems over structured data have been developed, e.g., BANKS, DISCOVER, DBXplorer, Prècis and many others presented in various surveys [6, 11]. Their typical approach is to perform an off-line pre-processing step that scans the whole instance data and constructs an index, a symbol table or some structure which is later used during the run time to identify the parts of the database in which each keyword appears. Once they discover it, they perform a path discovery algorithm to find the different ways that these parts are connected (e.g., finding minimal joining networks, or Steiner trees).

Unfortunately, although keyword-based techniques can be very successful as local database services, they cannot be easily applied in large Data Integration Systems. One of the reasons is that the built index requires continuous maintenance since it is based on data values that may frequently change. If the index is locally stored in the source, it cannot be used to index the data of all the sources of a data integration system since it may be under the responsibility of different owners. Furthermore, in data integration systems, the data sources may not expose their whole data, but only portions of their schema, thus making impossible to build an index over the instances. On the other hand, integration systems rely on metadata, in the form of data types, lexical references, mappings extracted from the sources to be integrated, meaningful for addressing the solution of a keyword query, but that are not really exploited by the current keyword based search engines.

To overcome the above issues, we introduce Keymantic[1, 2] a new framework for keyword based searching on data integration systems that, in contrast to existing approaches, exploits intensional knowledge to transform keyword queries into semantically meaningful SQL queries that can be executed by the data integration system.

The structure of this paper is the following. Section 2 provides a motivating example, Section 3 introduces our approach and Section 4 sketches out some conclusion and future work.

2 Motivating Example

Let us assume that a virtual tourism district composed of a set of companies (travel agencies, hotels, local public administrations, tourism promotion agencies) wants to publish an integrated view of their tourism data about a location (see Figure 1). Keymantic allows users to query that data source with a two step process: firstly the keyword query is analyzed for discovering its intended meaning, then a ranked set of SQL queries, expressing the discovered meaning according to the database structure, is formulated.

Each keyword represents some piece of information that has been modeled in the database, but, depending on the design requirements of the data source, this piece might have been modeled as data or metadata. Thus, the first task is to discover what each keyword models in the specific data source and to which metadata / data may be associated to. The association between keywords and database needs to be approximate: the synonymous and polysemous terms might allow the discovery of multiple intended meanings, each one with a rank expressing its relevance. Let us consider, for example, a query consisting of the keywords "Restaurant Naples". For instance, a possible meaning of the query might be "find information about the restaurant called Naples". In this case, the former keyword should be mapped into a metadata (the table Restaurant) and the other one into a value of the attribute Name of the same table Restaurant. A user might have in mind other meanings for the same keywords, for example, "find the restaurants that are located in the Naples Avenue", or "in the Naples city", or "that cook Naples specialties". All these intended meanings give rise to different associations of the keyword `Naples`; attributes Address, City or Specialty of the table Restaurant. This example shows also that keywords in a query are not independent: we expect that the keywords express different features of what the user is looking for. For this reason we expect that in our example "Naples" is a value referring to an element of the Restaurant table. If the user had formulated the keyword query "Restaurant name Naples", the number of possible intended meanings would have been reduced, since the keywords `name` forces the mappings of Naples into the attribute Name of the table Restaurant. Notice that different intended meanings may generate a mapping from the same keyword both into metadata and into data values. For example, in our database restaurant is the name of a table, but it is also one of the possible values of the attribute Service in the table Hotel. Finally, the order of the keywords in a query is also another element to be taken into account since related elements are usually close. If a user asks for "Person London restaurant New York" one possible meaning of the query is that the user is looking for the restaurant in New York visited by people from London. Other permutations of the keywords in the query may generate other possible interpretations.

The second step in answering a keyword query concerns the formulation of an SQL query expressing one of the discovered intended meaning. In a database, semantic relationships between values are modeled either through the inclusion of different attributes under the same table or through join paths across different tables. Different join paths can lead to different interpretations. Consider, for instance, the keyword query "Person USA". One logical mapping is to have the word `Person` corresponding to the table Person and the word `USA` to a value of the attribute Country of the table City. Even when this mapping is decided, there are different interpretations of the keywords based

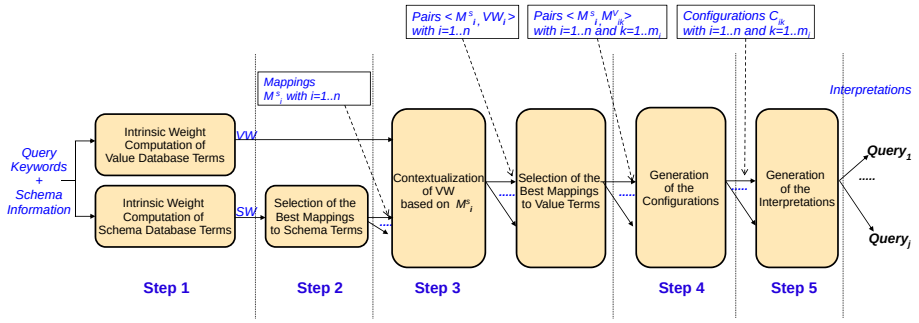

Fig. 2. Overview of the keyword query translation process

on the different join paths that exist between the tables Person and City. For instance, one can notice that a person and a city are related through a join path that goes through the City attribute referring to the attribute Name in the table City (determining which people in the database are from USA), through another path that is based on the table Hotel (determining which people reserved rooms of Hotels in USA), and also through another path that is based on the table Restaurant (determining which people are reserved a table in an American restaurant).

Finding the different semantic interpretations of a keyword query is a combinatorial problem which can be solved by an exhaustive enumeration of the different mappings to database structures and values. The large number of different interpretations can be brought down by using internal and external knowledge that helps in eliminating mappings that are not likely to lead to meanings intended by the user. For instance, if one of the provided keywords in a query is ``320-463-1463'', it is very unlikely that this keyword refers to an attribute or table name. It most probably represents a value, and in particular, due to its format, a phone number. Similarly, the keyword ``Bistro'' in a query does not correspond to a table or an attribute in the specific database. Some auxiliary information, such as a thesaurus, can provide the information that the word "bistro" is typically used to represent a restaurant, thus, the keyword can be associated to the Restaurant table.

3 From Keywords to Queries

The generation of *interpretations* (i.e. SQL queries) that most likely describe the intended semantics of a keyword query is based on semantically meaningful *configurations*, i.e. sets of mappings between each keyword and a database term. We introduce the notion of *weight* that offers a quantitative measure of the relativeness of a keyword to a database term, i.e., the likelihood that the semantics of the database term are the intended semantics of the keyword in the query. The sum of the weights of the keyword-database term pairs can form a score serving as a quantitative measure of the likelihood of the configuration to lead to an interpretation that accurately describes the intended keyword query semantics. The range and full semantics of the score cannot be fully specified in advance. They depend on the method used to compute the similarity. This

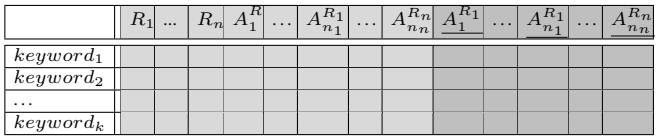

Fig. 3. Weight table with its SW (light) and VW (dark) parts

is not a problem as long as the same method is used to compute the scores for all the keywords.

The naive approach for selecting the best configurations is the computation of the score of each possible configuration and then selecting those that have the highest scores. Of course, we would like to avoid an exhaustive enumeration of all the possible configurations, and compute only those that give high scores. 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* [5]. Unfortunately, solutions to this problem suffer from two main limitations. First, apart from the mutual exclusiveness, they do not consider any other interdependencies that may exist between the mappings. Second, they typically provide only the best mapping, instead of a ranked list based on the scores.

To cope with the first limitation, we introduce two different kinds of weights: the *intrinsic*, and the *contextual* weights. Given a mapping of a keyword to a database term, its intrinsic weight measures the likelihood that the semantics of the keyword is that of the database term if considered in isolation from the mappings of all the other keywords in the query. The computation of an intrinsic weight is based on syntactic, semantic and structural factors such as attribute and relation names, or other auxiliary external sources, such as vocabularies, ontologies, domains, common syntactic patterns, etc. On the other hand, a contextual weight is used to measure the same likelihood but considering the mappings of the remaining query 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. As an example, for the keyword query ``Restaurant Name Naples'' expressed on the database in Figure 1, since the keyword ``Naples'' is right next to keyword Name, mapping the keyword Name to the attribute Name of the table Restaurant makes more likely the fact that the keyword Naples is a name value, i.e., should be mapped to the domain of the attribute Name. At the same time, it decreases its relativeness to the other database terms. To cope with the second limitation, we have developed a novel algorithm for computing the best mappings. The algorithm is based on and extends the Hungarian (a.k.a., Munkres) algorithm [4].

A visual illustration of the individual steps in the keyword query translation task is depicted in Figure 2. A special data structure, called *weight matrix* (see Figure 3), plays a central role in these steps. The *weight matrix* is a two-dimensional array with a row for each keyword in the keyword query, and a column for each database term. The value of a cell $[i, j]$ represents the weight associated to the mapping between the keyword i and the database term j. An R_i and $A_j^{R_i}$ columns correspond to the relation R_i and the attribute A_j of R_i, respectively, while a column with an underlined attribute name $\underline{A_j^{R_i}}$ represents the data values that may be contained in the column A_j of table R_i,

i.e., its domain. Two parts (i.e., sub-matrices) can be distinguished in the weight matrix. One corresponds to the database terms related to schema elements, i.e., relational tables and attributes, and the other one corresponds to attribute values, i.e., the domains of the attributes. We refer to database terms related to schema elements as *schema database terms* (SW), and to those related to domains of the attributes as *value database terms* (VW).

Intrinsic Weight Computation. The first step of the process is the intrinsic weight computation. The output is the populated SW and VW sub-matrices. The computation is achieved by the exploitation and combination of a number of similarity techniques based on structural and lexical knowledge extracted from the data source, and on external knowledge, such as ontologies, vocabularies, domain terminologies, etc. Note that the knowledge extracted from the data source is basically the meta-information that the source makes public, typically, the schema structure and constraints. In the absence of any other external information, a simple string comparison based on tree-edit distance can be used for populating the SW sub-matrix. For the VW sub-matrix the notion of *Semantic Distance* [7] can always be used in the absence of anything else. As it happens in similar situations [10], measuring the success of such a task is not easy since there is no single correct answer. In general, the more meta-information has been used, the better. However, even in the case that the current step is skipped, the process can continue with the weight matrix where all the intrinsic values have the same default value.

Selection of the Best Mappings to Schema Terms. The intrinsic weights provide a first indication of the similarities of the keywords to database terms. To generate the prominent mappings, we need on top of that to take into consideration the inter-dependencies among the mappings of the different keywords. We consider first the prominent mappings of keywords to schema terms. For that we work on the SW sub-matrix. Based on the intrinsic weights, a series of mappings M_1^S,M_2^S, ..., M_n^S, of keywords to schema terms are generated. The mappings are those that achieve the highest overall score, i.e., the sum of the weights of the individual keyword mappings. The mappings are partial, i.e., not all the keywords are mapped to some schema term. Those that remain unmapped will play the role of an actual data value and they will be considered in a subsequent step for mapping to value database terms. The selection of the keywords to remain unmapped is based on the weight matrix and some cut-off threshold. Those with a similarity below the threshold remain unmapped. For each mapping M_i^S, the weights of its SW matrix are adjusted to take into consideration the context generated by the mapping of the neighboring keywords. It is based on the observation that users form queries in which keywords referring to the same or related concepts are adjacent. The generation of the mappings and the adjustment of the weights in SW are performed by an adaptation of the Hungarian algorithm. In particular, the algorithm does not stop after the generation of the best mapping to continues to the generation of the second best, the third, etc. Furthermore, some of its internal steps 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. The output of such a step is an updated weight matrix SW_i and, naturally, an updated score for each mapping M_i^S. Given the updated scores, some mappings may be rejected. The selection is based on a threshold. There is no golden value to set the threshold value. It depends on the expectations from

the keyword query answering systems. The higher its value, the less the interpretations generated at the end, but with higher confidence. In contrast, the lower the threshold value, the more the mappings with lower confidence.

Contextualization of VW and selection of the Best Mappings to Value Terms. For each partial mapping M_i^S of keyword to schema terms generated in the previous step, the mappings of the remaining unmapped keywords to value terms needs to be decided. This is done in two phases. First, the intrinsic weights of the VW sub-matrix that were generated in Step 1 are updated to reflect the added value provided by the mappings in M_i^S of some of the keywords to schema database terms. This is called the process of contextualization of the VW sub-matrix, and it increases the weights of the respective values terms to reflect exactly that. For example, in the keyword query ``Name Alexandria'' assume that the keyword Alexandria was found during the first step to be equally likely the name of a person or of a city. If in Step 2 the keyword Name has been mapped to the attribute Name of the table Person, the confidence that Alexandria is actually the name of a person is increased, thus, the weight between that keyword and the value database term representing the domain of attribute Name should be increased, accordingly. In the second phase, given an updated VW_i sub-matrix, the most prominent mappings of the remaining unmapped keywords to value database terms are generated. The mappings are generated by using again the adapted technique 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_i was based. Given one such mapping M_{ik}^V the value weights are further updated to reflect the mappings of the adjacent keywords to value database terms, in a way similar to the one done in Step 2 for the SW sub-matrix. The outcome modifies the total score of each mapping M_{ik}^V, and based on that score the mappings are ranked.

Generation of the Configurations. As a fourth step, each pair of a mapping M_{ik}^V together with its associated mapping M_i^S is a total mapping of the keywords to database terms, forming a configuration C_{ik}. The score of the configuration is the sum of the scores of the two mappings, or alternatively the sum of the weights in the weight matrix of the elements $[i, j]$ where i is a keyword and j is the database term to which it is mapped through M_{ik}^V or M_i^S.

Generation of the Interpretations. Having computed the best configurations, the interpretations of the keyword query, i.e., the SQL queries, can be generated. The score of each such query is the score of the respective configuration. Recall, however, that a configuration is simply a mapping of the keywords to database terms. The presence of different join paths among these terms results in multiple interpretations. Different strategies can be used to further rank the selections. One popular option is the length of the join path [8] but other heuristics found in the literature [11] can also be used. It is also possible that a same interpretation can be obtained with different configurations. A post-processing analysis and the application of data-fusion techniques [3] can be used to deal with this issue. We adopt a greedy approach that computes a query for every alternative join path. In particular, we construct a graph in which each node corresponds to a database term. An edge connects two terms if they are structurally related, i.e., through a table-attribute-domain value relationship, or semantically, i.e., through a ref-

erential integrity constraint. Given a configuration we mark all terms that are part of the range of the configuration as "marked". Then we run a breath-first traversal (that favors shorter paths) to find paths that connect the disconnected components of the graph (if it is possible). Then, the final SQL query is constructed by using the "marked" database terms, and in particular, the tables for its from clause, the conditions modeled by the edges for its where clause and the remaining attributes for its select clause. After that, the process is repeated to find a different interpretation, that will be based on a different join path. The final order of the generated interpretations is determined by the way the different paths are discovered and the cost of the configuration on which each interpretation is based.

4 Conclusion and future work

We described a novel framework for keyword searching in relational databases. In contrast to traditional keyword searching techniques that have access to the actual data stored in the database, our technique uses intensional knowledge such as schema information, semantic knowledge, rules specified by users, and techniques that exploit common values and formats. The work opens many new challenging opportunities and research directions, such as the exploitation of standard modeling practices to enhance the configuration generation process and produce more meaningful configurations.

References

[1] S. Bergamaschi, E. Domnori, and Francesco. Keyword search over relational databases: a metadata approach. In *to appear in SIGMOD*. 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] J. Bleiholder and F. Naumann. Data fusion. *ACM Comput. Surv.*, 41(1), 2008.

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

[5] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems*. SIAM Society for Industrial and Applied Mathematics, Philadelphia, 2009.

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

[7] R. Cilibrasi and P. M. B. Vitányi. The google similarity distance. *IEEE Transactions on Knowledge & Data Engineering*, 19(3):370–383, 2007.

[8] Y. Kotidis, A. Marian, and D. Srivastava. Circumventing Data Quality Problems Using Multiple Join Paths. In *CleanDB*, 2006.

[9] M. Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246. ACM, 2002.

[10] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

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