A Hidden Markov Model Approach to Keyword-based Search over Relational Databases*

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

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

Abstract. We present a novel method for translating keyword queries over relational databases into SQL queries with the same intended semantic meaning. In contrast to the majority of the existing keyword-based techniques, our approach does not require any a-priori knowledge of the data instance. It follows a probabilistic approach based on a Hidden Markov Model for computing the top-K best mappings of the query keywords into the database terms, i.e., tables, attributes and values. The mappings are then used to generate the SQL queries that are executed to produce the answer to the keyword query. The method has been implemented into a system called KEYRY (from KEYword to queRY).

1 Introduction

Keyword searching is becoming the de-facto standard for information searching, mainly due to its simplicity. For textual information, keyword query answering has been extensively studied, especially in the area of information retrieval [14]. However, for structured data, it has only recently received considerable attention [5, 12]. The existing techniques for keyword searching over structured sources heavily rely on an a-priori instance-analysis that scans the whole data instance and constructs some index, a symbol table or some structure of that kind which is later used during run time to identify the parts of the database in which each keyword appears. This limits the application of these approaches to only cases where direct a-priori access to the data is possible.

There is a great deal of structured data sources that do not allow any direct access to their own contents. Mediator-based systems, for example, typically build a virtual integrated view of a set of data sources. The mediator only exposes the integrated schema and accesses the contents of the data sources at query execution time. Deep web databases are another example of sources that do not generally expose their full content, but offer only a predefined set of queries that can be answered, i.e., through web forms, or expose only their schema information. Even when the access to the data instance is allowed, it is not practically feasible in a web-scale environment to retrieve all the contents of the sources in order to build an index. The scenario gets even worst when

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

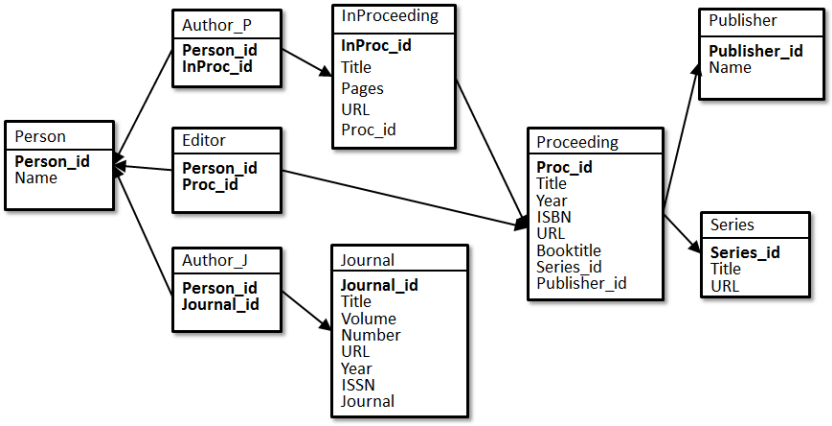

Fig. 1. A fragment of the DBLP schema

the data source contents are subject to frequent updates, as it happens, for instance, in e-commerce sites, forums/blogs, social networks, etc.

In this paper we present an approach, implemented in the KEYRY (from KEYword to queRY) prototype system, for keyword searching that does not assume any knowledge about the data source contents. The only requirement is for the source to provide a schema, even a loosely described, for its data. Semantics, extracted directly from the data source schema, are used to discover the intended meaning of the keywords in a query and to express it in terms of the underlying source structures. The result of the process is an interpretation of the user query in the form of a SQL query that will be executed by the DBMS managing the source. Notice that there are many interpretations of a keyword query in terms of the underlying data source schemas, some more likely and other less likely to capture the semantics that the user had in mind when she was formulating the keyword query. As usual in keyword based searching systems, assigning a ranking is a critical task since it avoids the users to deal with uninteresting results. One of the innovative aspects of our approach is that we adopt a probabilistic approach based on a Hidden Markov Model (HMM) for mapping user keywords into database terms (names of tables and attributes, domains of attributes). Using a HMM allows us to model two important aspects of the searching process: the order of the keywords in a query (this is represented by means of the HMM *transition probabilities*) and the probabilities of associating a keyword to different database terms (by means of the HMM *emission probabilities*). A HMM typically has to be trained in order to optimize its parameters. We propose a method providing a parameter setting not relying on any training data. In particular, we developed some heuristic rules that applied to the database schema provide the transition probabilities. Moreover, we approximated the emission probability by means of similarity measures (we use string similarity for measuring the distance between keywords and schema elements and regular expressions for evaluating the domain compatibilities, but our approach is independent of the measure adopted). Finally, we developed a variation of the HITS algorithm [9], typically exploited for ranking web pages on the basis of the links among them, for computing

an authority score for each database term. We consider these scores as the *initial state probabilities* required by the HMM.

More specifically, our key contributions are the following: (i) we propose a new probabilistic approach based on a HMM for keyword based searching over databases that does not require to build indexes over the data instance; (ii) we develop a method for providing a parameter setting allowing keyword searching without any training data and (iii) we exploit the List Viterbi algorithm that decodes the HMM in order to obtain the top-K results.

The remainder of the paper is as follows. Section 2 is an overview of our approach for keyword-based searching over databases. The problem is formalized in Section 3 and our proposal is described in Section 4. Section 5 describes related work and we conclude in Section 6 with a brief wrap up and some suggestions for future work.

2 KEYRY at a glance

A keyword in a keyword query may in principle be found in any database term (i.e., as the name of some schema term (table or attribute) or as a value in its respective domain). This gives rise to a large number of possible mappings of each query keyword into database term. These mappings are referred to as *configurations*. Since no access to the data source instance is assumed, selecting the top-K configurations that better represents the intended semantics of the keyword query is a challenging task.

Figure 1 illustrates a fragment of a relational version of the DBLP database[3]. The keywords in a query may be associated to different database terms, representing different semantics of the keyword query. For instance, a user may be interested in the papers written by Garcia-Molina published in a journal on 2011 and poses the query consisting of the three keywords `Garcia-Molina`, `journal` and `2011`. The keyword `journal` should be mapped into the table *Journal*, `2011` into the domain of the attribute *Year* in the *Journal* table, and `Garcia-Molina` should be an element of the domain of the attribute *Name* in the table *Person*. If we do not know the intended meaning of the user query, we may attribute different semantics to the keywords, e.g. `2011` might be the number of a page or part of the ISSN journal number. Not all the keywords may be mapped into all the database terms: certain mappings are actually more likely than other. Since KEYRY does not have any access to the data instance, we exploit the HMM *emission probabilities* to rank the likelihood of each possible mapping between a keyword and a database term. In our approach, we approximate the emission probabilities by means of similarity measures based on semantics extracted from the source schema (e.g. names of attributes and tables, attribute domains, regular expressions).

Moreover, based on known patterns of human behavior [8], we know that keywords related to the same topic are typically close to each other in a query. Consequently, adjacent keywords are typically associated to close database terms, i.e. terms that are part of the same table or belong to tables connected through foreign keys. For example, in the previous query the mapping of the keyword `journal` into the table *Journal* increases the likelihood that `2011` is mapped into the domain of an attribute of the table

[3] http://www.informatik.uni-trier.de/ ley/db/

Journal. The HMM *transition probabilities*, that we estimate on the basis of heuristic rules applied to the database schema, allows KEYRY to model this aspect. Section 4 describes how KEYRY computes the top-K configurations that better approximate the intended meaning of a user query.

Then, the possible paths joining the database terms in a configuration have to be computed. Different paths correspond to different *interpretations*. For example, let us consider the query "`Garcia-Molina proceedings 2011`" and the configuration that maps `proceeding` into the table *Proceeding*, `2011` into the domain of the attribute *Year* in the same table, and `Garcia-Molina` into the domain of the attribute *Name* in the table *Person*. Two possible paths may be computed for this configuration, one involving the tables *InProceeding* and *Author_P*, with the meaning of retrieving all the proceedings where Garcia-Molina appears as an author, and the second involving the table *Editor* and returning the proceedings where Garcia-Molina was an editor. Different strategies have been used in the literature to rank the interpretations. One popular option is the length of the join path, but other heuristics [12] can also be used. In KEYRY we compute all the possible paths and we rank them on the basis of their length. However, this is not the main focus of the current work and we will not elaborate further on it.

3 Problem statement

Definition 1. *A database D is a collection V_t of relational tables $R_1, R_2, \ldots, R_n$. Each table R is a collection of attributes $A_1, A_2, \ldots, A_{m_R}$, and each attribute A has a domain, denoted as $dom(A)$. Let V_a=$\{A \mid A \in R \wedge R_{\in} V_t\}$ represent the set of all the attributes of all the tables in the database and V_d=$\{d \mid d$=$dom(A) \wedge A \in V_a\}$ represents the set of all their respective domains. The* database vocabulary *of D, denoted as V_D, is the set V_D=$V_t \cup V_a \cup V_d$. Each element of the set V_D is referred to as a* database term*.*

We distinguish two subsets of the database vocabulary: the *schema vocabulary* $V_{SC} = V_t \cup V_a$ and the *domain vocabulary* $V_{DO} = V_d$ that concerns the instance information. We also assume that a keyword query KQ is an ordered l-tuple of keywords $(k_1, k_2, \ldots, k_l)$.

Definition 2. *A* configuration $f_c(KQ)$ *of a keyword query KQ on a database D is an injective function from the keywords in KQ to database terms in V_D. In other words, a configuration is a mapping that describes each keyword in the original query in terms of database terms.*

The reason we consider a configuration to be an injective function is because we assume that: (i) each keyword cannot have more than one meaning in the same configuration, i.e., it is mapped into only one database term; (ii) two keywords cannot be mapped to the same database term in a configuration since overspecified queries are only a small fraction of the queries that are typically met in practice [8]; and (iii) every keyword is relevant to the database content, i.e., keywords always have a correspondent database term. Furthermore, while modelling the keyword-to-database term mappings, we also

assume that every keyword denotes an element of interest to the user, i.e., there are no stop-words or unjustified keywords in a query. In this paper we do not address query cleaning issues. We assume that the keyword queries have already been pre-processed using well-known cleansing techniques.

Answering a keyword query over a database D means finding the SQL queries that describe its possible semantics in terms of the database vocabulary. Each such SQL query is referred to as an *interpretation* of the keyword query in database terms. An interpretation is based on a configuration and includes in its clauses all the database terms that are part of the image[4] of the query keywords through the configuration. In the current work, we consider only select-project-join (SPJ) interpretations that are typically the queries of interest in similar works [2, 7], but interpretations involving aggregations [11] are part of our future work.

Definition 3. *An* interpretation *of a keyword query* $KQ = (k_1, k_2, \ldots, k_l)$ *on a database* D *using a configuration* $f_c^*(KQ)$ *is an SQL query in the form*
`select` $A_1, A_2, \ldots, A_o$ `from` R_1 JOIN R_2 JOIN ... JOIN R_p `where` $A'_1{=}v_1$ AND $A'_2{=}v_2$ AND ... AND $A'_q{=}v_q$
such that the following holds:

- $\forall A{\in}\{A_1, A_2, \ldots, A_o\}$*:* $\exists k{\in}KQ$ *such that* $f_c^*(k){=}A$
- $\forall R{\in}\{R_1, R_2, \ldots, R_p\}$*: (i)* $\exists k{\in}KQ$*:* $f_c^*(k){=}R$ *or (ii)* $\exists k_i, k_j{\in}KQ$*:* $f_c^*(k_i){=}R_i \wedge f_c^*(k_j){=}R_j \wedge$ *exists a join path from* R_i *to* R_j *that involves* R
- $\forall$ "$A'{=}v$"$\in\{A'_1{=}v_1, A'_2{=}v_2, \ldots, A'_o{=}v_o\}$*:* $\exists k{\in}KQ$ *such that* $f_c^*(k){=}dom(A') \wedge k = v$
- $\forall k{\in}KQ$*:* $f_c^*(k){\in}\{A_1, A_2, \ldots, A_o, R_1, R_2, \ldots, R_p, dom(A'_1), \ldots, dom(A'_q)\}$

The existence of a database term in an interpretation is justified either by belonging to the image of the respective configuration, or by participating in a join path connecting two database terms that belong to the image of the configuration. Note that even with this restriction, due to the multiple join paths in a database D, it is still possible to have multiple interpretations of a keyword query KQ given a certain configuration $f_c^*(KQ)$. We use the notation $\mathcal{I}(KQ, f_c^*(KQ), D)$ to refer to the set of these interpretations, and $\mathcal{I}(KQ, D)$ for the union of all these sets for a query KQ.

Since each keyword in a query can be mapped into a table name, an attribute name or an attribute domain, there are $2\Sigma_{i=1}^{n}|R_i| + n$ different mappings for each keyword, with $|R_i|$ denoting the *arity* of the relation R_i and n 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 term, for a query containing l keywords, there are $\frac{|V_D|!}{(|V_D|-l)!}$ possible configurations. Of course, not all the interpretations generated by these configurations are equally *meaningful*. Some are more likely to represent the intended keyword query semantics. In the following sections we will show how different kinds of meta-information and interdependencies between the mappings of keywords into database terms can be exploited in order to effectively and efficiently identify these meaningful interpretations and rank them higher.

[4] Since a configuration is a function, we use the term image to refer to its output.

4 Computing configurations using a HMM

In a first, intuitive attempt to define the configuration function we can divide the problem of matching a whole query to database terms into smaller sub-tasks. In each subtask the best match between a single keyword and a database term is found. Then the final solution to the global problem is the union of the matches found in the subtasks. This approach works well when the keywords in a query are independent of each other, meaning that they do not influence the match of the other keywords to database terms. Unfortunately, this assumption does not hold in real cases. On the contrary, inter-dependencies among keywords meanings are of fundamental importance in disambiguating the keyword semantics.

In order to take into account these inter-dependencies, we model the matching function as a sequential process where the order is determined by the keyword ordering in the query. In each step of the process a single keyword is matched against a database term, taking into account the result of the previous keyword match in the sequence. This process has a finite number of steps, equal to the query length, and is stochastic since the matching between a keyword and a database term is not deterministic: the same keyword can have different meanings in different queries and hence being matched with different database terms; vice-versa, different database terms may match the same keyword in different queries. This type of process can be modeled, effectively, by using a Hidden Markov Model (HMM, for short), that is a stochastic finite state machine where the states are hidden variables.

A HMM models a stochastic process that is not observable directly (it is hidden), but it can be observed indirectly through the observable symbols produced by another stochastic process. The model is composed of a finite number *N* of states. Assuming a time-discrete model, at each time step a new state is entered based on a *transition probability distribution*, and an observation is produced according to an *emission probability distribution* that depends on the current state, where both these distributions are time-independent. Moreover, the process starts from an initial state based on an *initial state probability distribution*. We will consider first order HMMs with discrete observations. In these models the *Markov property* is respected, i.e., the transition probability distribution of the states at time $t + 1$ depends only on the current state at time t and it does not depend on the past states at time $1, 2, \ldots, t - 1$. Moreover, the observations are discrete: there exists a finite number, *M*, of observable symbols, hence the emission probability distributions can be effectively represented using *multinomial distributions* dependent on the states. More formally, the model consists of: (i) a set os states $S = \{s_i\}, 1 \leq i \leq N$; (ii) a set of observation symbols $V = \{v_j\}, 1 \leq j \leq M$; (iii) a transition probability distribution $A = \{a_{ij}\}, 1 \leq i \leq N, 1 \leq j \leq N$ where

$$a_{ij} = P(q_{t+1} = s_j | q_t = s_i) \text{ and } \sum_{0<j<N} a_{ij} = 1$$

(iv) an emission probability distribution $B = \{b_i(m)\}, 1 \leq i \leq N, 1 \leq m \leq M$ where

$$b_i(m) = P(o_t = v_m | q_t = s_i) \text{ and } \sum_{0<m<M} b_i(m) = 1$$

and (v) an initial state probability distribution $\Pi = \{\pi_i\}$, $1 \leq i \leq N$ where

$$\pi_i = P(q_1 = s_i) \text{ and } \sum_{0<i<N} \pi_i = 1$$

Based on the above, the notation $\lambda = (A, B, \Pi)$ will be used to indicate a HMM. In our context, the keywords inserted by the user are the observable part of the process, while the correspondent database terms are the unknown variables that have to be inferred. For this reason, we model the keywords as observations and each term in the database vocabulary as a state.

4.1 Setting HMM parameters

In order to define a HMM, its parameters have to be identified. This is usually done using a training algorithm that, after many iterations, converges to a good solution for the parameter values. In our approach we introduce some heuristic rules that allow the definition of the parameter values even when no training data is available. The HMM parameter values are set by exploiting the semantics collected from the data source metadata. In particular:

The transition probabilities are computed using heuristic rules that take into account the semantic relationships that exist between the database terms (aggregation, generalization and inclusion relationships). The goal of the rules is to foster the transition between database terms belonging to the same table and belonging to tables connected through foreign keys. The transition probability values decrease with the distance of the states, e.g. transitions between terms in the same table have higher probability than transitions between terms in tables directly connected through foreign keys, that, in turn, have higher probability than transitions between terms in tables connected through a third table.

The emission probabilities are computed on the basis of similarity measures. In particular two different techniques are adopted for the database terms in V_{SC} and V_{DO}. We use the well known *edit distance* for computing the lexical similarity between the keywords and each term (and its synonyms extracted from Wordnet[5]) in the schema vocabulary V_{SC}. On the other side, the similarity between keywords and the terms in the domain vocabulary V_{DO} is based on domain compatibilities and regular expressions. We use the calculated similarity as an estimate for the conditional probability $P(q_t = s_i | o_t = v_m)$ then, using the Bayes theorem, we calculate the emission probability $P(o_t = v_m | q_t = s_i)$. Note that the model is independent of the similarity measure adopted. Other more complex measures that take into account external knowledge sources (i.e., public ontologies and thesauri) can be applied without modifying the model.

The initial state probabilities are estimated by means of the scores provided by the HITS algorithm [9]. The HITS algorithm is a link analysis algorithm that calculates two different scores for each page: *authority* and *hub*. A high authority score indicates that the page contains valuable information with respect to the user query, while a high

[5] http://wordnet.princeton.edu

hub score suggests that the page contains useful links toward authoritative pages. This algorithm has been adapted to our context in order to rank the tables in a database based on their authority scores. The higher the rank, the more valuable is the information stored in the tables. For this reason, the authority score is used as an estimate of the initial state probabilities.

Adapted HITS algorithm In order to employ the HITS algorithm, we build the database graph $G_D = (V_t, E_{fk})$ which is a directed graph where the nodes are the database tables in V_t and the edges are connections between tables through foreign keys, i.e., given two tables R_i, R_j there exists an edge in E_{fk} from R_i to R_j, denoted as $R_i \rightarrow R_j$, if an attribe in R_j is referred by a foreign key defined in R_i. Let A be the n x n adjacency matrix of the graph G_D

$$A = [a_{ij}],\ a_{ij} = 1 \text{ iff } e_{ij} \in E_{fk},\ a_{ij} = 0 \text{ iff } e_{ij} \notin E_{fk}$$

In our approach we use the modified matrix B that takes into account the number of attributes minus the number of foreign keys in a table (foreign keys are considered as links).

$$B = [b_{ij}],\ b_{ij} = a_{ij} \cdot (|R_i| - ||\{R_i \rightarrow R_j, 1 \leq j \leq n\}||)$$

Let us define the authority weight vector $auth$ and the hub weight vector hub

$$auth^T = [u_1, u_2, \ldots, u_n] \text{ and } hub^T = [v_1, v_2, \ldots, v_n]$$

The algorithm initializes these vectors with uniform values, e.g., $\frac{1}{n}$, then the vectors are updated in successive iterations as follows

$$\begin{cases} auth = B^T \cdot hub \\ hub = B \cdot auth \end{cases}$$

At each iteration a normalization step is performed to obtain authority and hub scores in the range $[0, 1]$. After few iterations the algorithm converges and the authority scores are used as estimate for the initial state probabilities.

Example 1. Let us consider the database in Figure 1. This database generates 53 states[6], one for each database term. Concerning the transition probability distribution, the heuristic rules foster transitions between database terms of the same table or in tables connect via foreign key. According to this, the most probable states subsequent to the state associated to the table *Journal* are the states associated to the names and the domains of the attributes *Title*, *Volume*, *Number*, etc., then, the state associated to the table *Author_J*, subsequently, the states associated to the table *Person* and so on. We use different similarity measures for computing the emission probability distribution. Domain compatibility and regular expressions are used for measuring the probabilities of the values associated to states of terms in the V_{DO}, e.g., the possible values associated to the state representing the *Year* of the table *Journal* are numeric values between

[6] Since in this schema the primary key values are not meaningful for the user, we removed the states associated to these database terms.

1900 and 2100. The probabilities of values associated to states if terms in the V_{SC} is computed on the basis of lexical similarity computed on the term and on its synonyms, hypernyms and hyponyms. For example, we consider "article" as synonym of journal. In our experiments we noticed that, by applying the adaptation of the HITS algorithm, the states corresponding to the database terms in the tables *Journal* and *Inproceeding* obtain the highest authority scores.

4.2 Decoding the HMM

Once the parameters A, B, and Π have been defined, the resulting HMM is used to compute the top-K configurations by applying the *List Viterbi* algorithm [10], which is a generalization of the well known Viterbi algorithm [6]. The algorithm has also been applied, using a different formulation, to HMMs in order to solve the following problem:

Given a HMM λ and an observation sequence $O_l = (o_1, o_2, \ldots, o_T)$ find the ordered list of the K state sequences $\hat{Q}_l^k = (q_1^k, q_2^k, \ldots, q_T^k), 1 \leq k \leq K$ which have the highest probability of generating O_l

In other words, the algorithm generalizes the well-known Viterbi algorithm finding the top-K maximum likelihood state sequences (MLSSs) instead of the single MLSS found by the original algorithm.

Example 2. The top-2 results of the keyword query "`proceeding ER 2011`" are the ones mapping the keyword `proceeding` into the table *proceeding*, `ER` into the domain of the attribute *Title* of the table *proceeding*, and `2011` into the domains of the attributes *Year* and *ISBN*, respectively. Both the solutions have the same transition probabilities but different emission probabilities since the likelihood of the mapping of `2011` into the attribute *Year* is higher than the one into *ISBN*. Other mappings are possibles: for example the configuration that maps the keyword `proceeding` into the table *proceeding*, `ER` into the domain of the attribute *Title* of the table *proceeding*, and `2011` into the domain of the attribute *Year* of the table *Journal* has rank 9.

5 Related work

There has been already a number of different systems that consider keyword searching over structured or semi-structured data. Specifically, well-known systems of the former include BANKS [1], DISCOVER [7], DBXplorer [2], QUICK [13], SQAK [11] and many others presented in various surveys [5, 12]. Their typical approach is to perform an off-line pre-prosessing step that scans the whole data instance and constructs an index, a symbol table or some structure of that kind which is later used at run time to identify the parts of the database in which each keyword appears. After that, they perform a path discovery algorithm to find the different ways in which these tables are connected. The algorithms range from finding minimal joining networks [7] to Steiner trees [1]. In contrast to all these approaches, KEYRY is able to achieve similar results without the need of accessing and scanning the data. QUICK guides the users into a series of query refinements to find the one that describes their intended semantics, but

it assumes that there is only one such semantics, while practice has shown that there may be different alternatives. Keymantic [3, 4] has goals similar to ours but it follows a fundamentally different approach. It handles the keyword search as a bipartite graph assignment problem and finds the solutions using an extended version of the Hungarian algorithm.

6 Conclusion and future work

We described KEYRY, a probabilistic keyword-based searching system for relational databases that is based on a HMM for computing the top-K best mappings of the query keywords into the database terms. The use of HMM allows the efficient modeling of the order of the keywords in a query and the probabilities of associating them to different database terms. Although a HMM typically has to be trained to optimize its parameters, we propose a method that does not rely on any particular training.

References

1. B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, Parag, and S. Sudarshan. Banks: Browsing and keyword searching in relational databases. In *VLDB*, pages 1083–1086, 2002.
2. S. Agrawal, S. Chaudhuri, and G. Das. Dbxplorer: A system for keyword-based search over relational databases. In *ICDE*, pages 5–16. IEEE Computer Society, 2002.
3. S. Bergamaschi, E. Domnori, F. Guerra, R. T. Lado, and Y. Velegrakis. Keyword search over relational databases: a metadata approach. In *SIGMOD*. ACM, 2011.
4. 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.
5. S. Chakrabarti, S. Sarawagi, and S. Sudarshan. Enhancing search with structure. *IEEE Data Eng. Bull.*, 33(1):3–24, 2010.
6. J. Forney, G.D. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268 – 278, 1973.
7. V. Hristidis and Y. Papakonstantinou. Discover: Keyword search in relational databases. In *VLDB*, pages 670–681, 2002.
8. R. Kumar and A. Tomkins. A Characterization of Online Search Behavior. *IEEE Data Engineering Bulletin*, 32(2):3–11, 2009.
9. L. Li, Y. Shang, H. Shi, and W. Zhang. Performance evaluation of hits-based algorithms. In M. H. Hamza, editor, *Communications, Internet, and Information Technology*, pages 171–176. IASTED/ACTA Press, 2002.
10. N. Seshadri and C.-E. W. Sundberg. List viterbi decoding algorithms with applications. *IEEE Transactions on Communications*, 42(234), 1994.
11. S. Tata and G. M. Lohman. SQAK: doing more with keywords. In *SIGMOD*, pages 889–902. ACM, 2008.
12. J. X. Yu, L. Qin, and L. Chang. *Keyword Search in Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.
13. G. Zenz, X. Zhou, E. Minack, W. Siberski, and W. Nejdl. From keywords to semantic queries-incremental query construction on the semantic web. *Journal of Web Semantics*, 7(3):166–176, 2009.
14. J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys*, 38(2), 2006.