Information Systems 55 (2016) 1–19

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Combining user and database perspective for solving keyword queries over relational databases

Sonia Bergamaschi[a], Francesco Guerra[a,*], Matteo Interlandi[b], Raquel Trillo-Lado[c], Yannis Velegrakis[d]

[a] DIEF – University of Modena and Reggio Emilia, Italy
[b] UCLA – University of California, Los Angeles, USA
[c] DIIS – University of Zaragoza, Spain
[d] DISI – University of Trento, Italy

ARTICLE INFO

Article history:
Received 15 June 2014
Received in revised form
18 July 2015
Accepted 21 July 2015
Recommended by: Martin Theobald
Available online 30 July 2015

Keywords:
Keyword search over relational databases
Hidden Markov Models
Dempster–Shafer Theory
Machine learning

ABSTRACT

Over the last decade, keyword search over relational data has attracted considerable attention. A possible approach to face this issue is to transform keyword queries into one or more SQL queries to be executed by the relational DBMS. Finding these queries is a challenging task since the information they represent may be modeled across different tables and attributes. This means that it is needed to identify not only the schema elements where the data of interest is stored, but also to find out how these elements are interconnected. All the approaches that have been proposed so far provide a monolithic solution. In this work, we, instead, divide the problem into three steps: the first one, driven by the user's point of view, takes into account what the user has in mind when formulating keyword queries, the second one, driven by the database perspective, considers how the data is represented in the database schema. Finally, the third step combines these two processes. We present the theory behind our approach, and its implementation into a system called QUEST (QUEry generator for STructured sources), which has been deeply tested to show the efficiency and effectiveness of our approach. Furthermore, we report on the outcomes of a number of experimental results that we have conducted.

1. Introduction

Keyword search has become the de-facto standard for searching on the web. Structured data sources contain a vast amount of information that is significant to be available for querying. Typically, query interfaces consist of web forms that allow predefined queries to be posed on their contents. Besides, web search engines index the content of these sources (the so-called hidden web) through the results of these web form queries, seen as free text. Apart from the fact that this restricts the kind of data that can be searched, the great deal of semantic information provided by the structure of the data, e.g., the schema, is basically lost. This gave rise to a special interest in supporting keyword search over structured databases [1] in ways that are as effective as those offered on text data and at the same time exploit as much as possible the structure of the data that databases provide.

Many approaches exploit full-text search functionalities natively implemented in the DBMS, such as the `contains` function in SQL server and the `match-against` function in MySQL, to discover the attributes of the database containing

* Corresponding author.
E-mail addresses: sonia.bergamaschi@unimore.it (S. Bergamaschi), francesco.guerra@unimore.it (F. Guerra), minterlandi@cs.ucla.edu (M. Interlandi), raqueltl@unizar.es (R. Trillo-Lado), velgias@disi.unitn.eu (Y. Velegrakis).

http://dx.doi.org/10.1016/j.is.2015.07.005
0306-4379/© 2015 Elsevier Ltd. All rights reserved.

the query keywords at run-time. Then, they construct the answer set by combining tuples containing different query keywords and selecting those combinations considered most likely to be what users were looking for [2–12]. All these approaches are typically heuristic-based, without a clear specification of the steps required to answer the keyword query. In this work, we advocate that there is a need for a more principled approach to keyword searching on structured data; in particular, we believe that keyword search on structured sources requires three fundamental steps. Existing works consist of either a monolithic end-to-end solution that provides no clear distinction of these three steps, or are focused on only some of them, considering some straightforward implementation of the remaining.

The three fundamental steps we consider are first to match the keywords to the database structures, then to discover ways these matched structures can be combined, and finally to select the best matches and combinations such that the identified database structures represent what the user had in mind to discover when formulating the keyword query. The first step is focused on trying to capture the meaning of the keywords in the query as they are understood by the user, and express them in terms of database terms, i.e., the metadata structures of the databases. In some sense, it provides the *user perspective* of the keyword query and it does so by providing a mapping of the keywords into database terms. This step is referred to as the *forward analysis step* since it starts from the keywords and moves towards the database. The second step tries to capture the meaning of the keywords as they can be understood from the point of view of the data engineers who designed that database organization, and express them in semantically coherent units of database structures containing the images of the keywords specified by the first step. So, in some sense, it provides the *database perspective* of the keyword query and it does so by providing the relationships among the images of the keywords. This task is referred to as the *backward analysis step* since it starts from the database structures and moves towards the query keywords through their images. The third step provides a ranking of the coherent units of database structures that the second step produced after selecting those that are more promising, i.e., those whose semantics more likely express what the user had in mind while was formulating the keyword query.

In our previous works we have studied different aspects of the keyword search problem over relational databases. The KEYMANTIC [13,14] system was focused on the first step. It provided a solution based on a bipartite graph matching model where user keywords were matched to database schema elements by using an extension of the Hungarian algorithm. KEYMANTIC is one of the first solutions that deals with the problem of querying structural databases through keywords when there is no prior access to the database content to build any indexes, thus, relying on semantic information of the database meta-data. This feature of KEYMANTIC makes it especially appropriate for keyword-based search on federated database systems and for exploring data sources in the hidden web. KEYRY [15,16] extended KEYMANTIC by providing a probabilistic framework, based on a HMM, to match keywords into database schema elements. Both works deal with the first step of the process described previously, i.e., the user perspective step. Our experience with these systems made clear that this was not enough for a complete solution. These systems were the motivation for the principled, holistic and unified framework presented in this work.

The main contributions of the current paper are the following: (i) we introduce a principled 3-step model for the keyword search problem over structured databases; (ii) we develop two different implementations of the first step, one that exploits heuristic rules and one that is based on machine learning techniques. Both aim at finding the appropriate Hidden Markov Model specification to generate the right mapping of the query keywords into database structures; (iii) we define an implementation of the second step based on Steiner Tree discovery which exploits a mutual information-based distance as edge weight and which works at the schema level instead of the instance level; (iv) we provide a probabilistic framework founded on the Dempster Shafer Theory that is able to combine the first two steps and modalities in a way that permits the system to promptly adapt to different working conditions by selecting the best combination among them; (v) we implement all the above in a system called QUEST (QUEry generator for STructured sources) [17] and provide the details of its implementation; and finally (vi) we perform an extensive set of experiments that offer a deep understanding of the whole process, its effectiveness and efficiency.

The remainder of the paper is as follows. First, the principled 3-step approach is introduced and our proposed framework is formally defined in Section 2. The implementation of each of the three steps in our developed QUEST prototype follows in Section 3. The relationship of our framework with the related works alongside our own previous works on the topic is explained in Section 4. Finally, the results of our extensive experimental evaluation are discussed in Section 5.

2. The three-step framework

As a data model for the structured database we assume the relational model, however the framework can be easily extended to other structured models as well.

We assume an infinite set $\mathcal{A}$ of attribute names, $\mathcal{R}$ of relation names, and $\mathcal{V}$ of value domains. A *tuple* is a finite set of attribute name–value pairs $\langle A_1{:}v_1, A_2{:}v_2, \ldots, A_n{:}v_n\rangle$ where $A_i \in \mathcal{A}$, $v_i \in V_i$ with $V_i \in \mathcal{V}$, for $i=1\ldots n$, and $A_i \neq A_j$ if $i \neq j$. The *schema of the tuple* is the $\langle A_1{:}V_1, A_2{:}V_2, \ldots, A_n{:}V_n\rangle$ and its *arity* is the number n. The domain V_i is referred to as the domain of the attribute A_i and will be denoted as $Dom(A_i)$, for $i=1\ldots n$. A *relation instance* is a finite set of tuples, all with the same schema. The *schema of the relation instance* is the common schema of its tuples and its *cardinality* the number of tuples it consists of. A *relation* is a pair $\langle R, I_R\rangle$, where $R \in \mathcal{R}$, referred to as the *relation name*, and I_R is a relation instance. The *schema* of a relation $\langle R, I_R\rangle$ is the schema of its relation instance, and will be denoted as $R(A_1{:}V_1, \ldots, A_n{:}V_n)$, where the $\langle A_1{:}V_1, \ldots, A_n{:}V_n\rangle$ is the schema of the relation instance I_R. In what follows, whenever there is no risk of confusion, the name R will be used to refer to the whole relation $\langle R, I_R\rangle$. Furthermore, the indication of the domains will be omitted leading to the simplified expression of the relation schema as $R(A_1, A_2, \ldots, A_n)$. Finally, the notation $|R|$ will denote the arity of the relation R and the $|I_R|$ the cardinality of its relation instance [18].