Recommendation-based Keyword Search over Relational Databases

Haithem Ghorbel[*], Nouha Othman[*] and Rim Faiz[***],

[*]Université de Tunis, Institut Supérieur de Gestion de Tunis, LARODEC, Tunisia
gho.haithem,othmannouha@gmail.com
[***]Université de Carthage IHEC Carthage, LARODEC, Tunisia
rim.faiz@ihec.rnu.tn

Abstract. Recently, there has been a burgeoning interest in keyword search in relational databases owing to its ease of use. Although extensive research has been lately done within this context, most of this research not only requires a prior access to data which severely restricts their applicability if this condition is not verified, but also returns very generic answers. However, providing users with personalized answers has become more than ever necessary due to the overabundance of data which can be annoying for the user. The challenge to return personalized and relevant answers that satisfy users' information needs remains. Inspired by the successful application of the collaborative filtering technique in recommender systems, we propose a novel keyword-based approach to provide users with personalized results based on the hypothesis that only information on the database schema is available.

1 Introduction

Over the decades, an explosive amount of structured data has been stored in Relational Databases (RDB)s. These latter have been widely used owing to the rich information they provide including relationships between the different entities in the DB. Developing effective query methods for users to easily query huge and complex repositories without the need of technical expertise has become one of the biggest challenges of the database community (Agrawal et al., 2002; Aditya et al., 2002). The emergence of web search engines has made keyword search the most commonly used search technique. The strength of this latter is that it enables users to easily express their information needs by a few keywords without needing to know the database schema or structured query languages. Nevertheless, such a technique requires a prior access to the database content in order to build the indices that will pinpoint the different tuples associated with the keywords at run time (Bergamaschi et al., 2011). This is a considerable shortcoming since it limits its applicability if a prior access to data is not possible. Another significant limitation is that the inter-dependencies among the query keywords were ignored. Actually, the meaning of each keyword in a user's query also depends on the meaning of the others. On the other hand, with the tremendous development of information technology, the amount of data has been growing exponentially. Thus, finding the desired information in a massive database has become a crucial but also a challenging task. Recommendation Systems (RSs) are powerful tools to filter data, providing only what the user is most likely looking for. In this paper, we propose a successful attempt to combine RSs techniques and RDB to overcome the limitation of the keyword search. Our proposed approach aims at returning personalized answers when we have no prior access to the actual data stored in the database. The remainder of the paper is structured as follows: In Section 2, we review the main existing work on querying RDBs. Then, we describe our approach in Section 3. Subsequently, we present in Section 4, we report our experimental evaluation and results. Towards the end, we conclude and outline our perspectives.

2 Related work on Querying Relational Databases

Arguably, keyword search has become the standard for seeking information on the Web as it allows the user to easily formulate queries with a few keywords. However, its simplicity comes with a price; keywords are fraught with ambiguity and their intended meaning needs to be explored further (Wang et al., 2008). Over the years, advanced approaches for keyword search over documents have been proposed to return relevant answers to the user. These approaches, though, don't return good results with RDB systems (RDBs), as IR-style search considers tuples as unstructured data, while in RDBs, the retrieved information is spread among tables. Unlike textual documents, the tuples are linked through the foreign-primary key constraints. Thus, foreign-primary key paths connecting tuples that contain keywords, represent an essential ingredient for solving a keyword query over a database. Defining representation models for databases to retrieve these paths is crucial. For these reasons, the direct application of keyword-based approaches to relational databases, where information is fragmented in numerous tables, is neither efficient nor effective (Bergamaschi et al., 2014). Indeed, multiple systems were proposed in the literature, where the most popular ones are BANKS (Aditya et al., 2002), BANKSII (Kacholia et al., 2005), DBXplorer (Agrawal et al., 2002), DISCOVER (Hristidis and Papakonstantinou, 2002) and SQAK (Tata and Lohman, 2008) and the most recent one are KEYMANTIC (Bergamaschi, Domnori, Guerra, Trillo Lado, and Velegrakis, 2011), KEYRY (Bergamaschi, Guerra, Rota, and Velegrakis, 2011) and SEMINDEX (Chbeir et al., 2014). The objective of these systems is to better cover a keyword query, in order to return answers that matches the user's intent. These approaches can be classified into two broad categories: schema based and tuple-based approaches. The Schema-based approaches model the database schema as a graph, in which the nodes express database relations and edges express interdependence between primary and foreign keys. Such approaches can fulfill a keyword query by the use of the schema information to generate SQL queries in RDBs, such as in DBXPLORER, DISCOVER, PRECIS, SQAK, KEYMANTIC, and KEYRY systems. Tuple-based approaches such as BANKS and BANKS II, model the database as a data graph, wherein nodes represent the tuples and edges denote the relationships between a pair of tuples, such as foreign key or primary key dependencies. The particularity of a data graph is that nodes and edges are typically weighted, which provides users with more information on how the objects are interconnected. KEYMANTIC and KEYRY tackled the issue of keyword search over RDBs differently; they can provide answers to the user's query without the necessity of a prior access to the data stored in the database to build indices that will locate the tuples.

3 Proposed Solution

The core idea of our approach, called *DeepRec*, is to combine the keyword search over RDB with some techniques used in recommender systems. DeepRec aims at integrating some recommendation and databases concepts to get better personalized answers to a simple keyword-based query posted by a user. It provides recommendations and serendipitous answers even when no prior access to the database is allowed, relying on both schema and users information. The different components of the given approach are detailed below.

3.1 Schema Terms Matching

The first phase named *Schema Weight Computation* consists in determining which keywords match with schema terms (attributes, relations) starting from a query and the schema information of the database. Attributes and relations are considered as metadata. In order to estimate the keyword-attribute/relation distance, we opted for the Levenshtein measure which computes the minimal number of insertions, deletions and replacements needed for transforming a string X into a string Y. However, a simple string similarity between the keyword and the schema term is not enough due to the heterogeneity of the user's vocabulary. In fact, a user may use different words that do not figure in the schema information of the database. For tis purpose, we employ WordNet for Word Sense Disambiguation (WSD), so that each used keyword is compared to all the synonyms, hyponyms and hypernyms of every schema term to keep the one having the highest similarity.

The second phase named *Schema Weight Personalization* consists in updating the Schema Weight (SW) Matrix by making use of the information gathered from the users' profiles. The main idea here is to add the concept of Collaborative Filtering (CF) of the RSs; build profiles for users and use their search history as well as similar users' history to personalize results.

We compute the Personalized Schema Weight (PSW) that uses sessions information to update the SW matrix. Similar sessions are indexed in a table. Then, we calculate how many times every schema term had the maximum value, for all the queries in the similar sessions. We store the result of this computation in its specific column in the SW matrix. Each value of this column is combined with each one in its analogue column in the SW to get the new PSW values in their corresponding cells. In other words, we weight the values of the SW of each column by a variable that affects the first values depending on the number of times this column had the maximum value.

Computing the best possible matching of keywords to database terms is known as the assignment problem. The popular Munkres, a.k.a. Hungarian, algorithm (Munkres, 1957), is a possible solution to this problem but, it provides only the best matching. Bergamaschi, Domnori, Guerra, Trillo Lado, and Velegrakis (2011) adapted this algorithm to our context, to not stop after the generation of the top one mapping, but continue to generate the other best ones. Besides, the weight matrix is dynamically updated every time a mapping of a keyword to a database term is decided during the calculation.

3.2 Value Weight Contextualization

The Value Weight (VW) matrix computation is performed in the same way as in (Bergamaschi, Domnori, Guerra, Trillo Lado, and Velegrakis, 2011). The computation is mainly done within the domain information of attributes. KEYMANTIC used a semantic distance to estimate the relatedness of two concepts. Thus, every matrix cell in the VW contains a value as an indicator of the eligibility and suitability of the keyword with the attribute domain. The keywords that have already been mapped to schema in the previous step will get assigned 0 in every cell of their lines to ensure that they won't be recomputed in the VW.

After the computation of the best mapping to schema terms M_i and the value weight matrix, the VW matrix is updated according the the terms mapped to schema. In keywords queries, a keyword may refer either to a schema term or to a value in a schema term. We contextualize the value weight matrix according to terms mapped as schema terms from the PSW, taking into consideration the keywords positions in the query. The user can use more than one term to describe one concept. The basic intuition behind our method is to check for each keyword k if it corresponds to any schema term x from the mapped ones. Then, if k corresponds to a keyword mapped to a schema term x, two situations may arise: If x is a relation R, we add a weight Ω to all the attributes of R for every adjacent keywords A(k) $\cup$ B(k), otherwise, if x is an attribute A of a relation R, we increase the weights of this attribute and the related attributes (with functional dependencies) by Ω for all A(k) $\cup$ B(k) keywords neighboring k.

A(k) and B(k) are two functions that retrieve the following and preceding keywords neighboring k respectively. Ω is a variable, its value is proportional to the distance between keywords. The output of this step is a contextualized Value Weight Matrix $V_j(M_i)$. Again, we will use the extension of the Hungarian Algorithm, this time over $V_j(M_i)$, to get the best assignments.

3.3 Generation of the personalized query answers:

The $V_j(M_i)$ with its related M_i is a full matching of the keywords to DB terms, producing together a new combination named A_ij. The score of each combination is the sum of weights of the $V_j(M_i)$ and its associated M_i.

SQL queries can be achieved with the possession of the first-score combinations. Yet, this latter just reside on the keywords match to their adequate database terms, without defining the relations between the terms. Works that process under the same assumption of no prior access to data such as (Bergamaschi et al., 2011) only consider the similarity between keywords-attributes/domain of attributes' similarities, which is not always appropriate. To cope with this limitation, we take advantage of the profiles built by our approach, to personalize the answers. Giving a current user making a query in his current session, we compute the similarity of this query with all previous queries in all the sessions using the cosine similarity. The answer that gained interest of the user is credited even its associated query is fairly similar to the user's current query.

4 Experimental Evaluation

4.1 Experimental Setup

In our experiments, we used MySQL 5 as the relational database management system and WordNet 3.0[1] as a lexical database. We explained the content of the Database to no technical users without exposing its schema information. We asked them to propose keywords queries and describe what they expect as answers for their queries. Then, an expert formulates an SQL query for this purpose. We compared the results generated by our approach with those obtained by the expert. We used a fraction of the MovieLens[2] 100K database. In our tests, 18 users were involved, we group each user's queries in a single session. The number of sessions is between 1 and 3 for each user. We used 105 queries distributed among the users sessions. To evaluate DeepRec, we were based on the number of keywords and that of sessions. For the initialization, we used the Information_schema views provided by MySQL which allows to retrieve metadata about objects in the DB.

4.2 Experimental Results and Discussion

Figure 1 shows a comparison between KEYMANTIC and DeepRec in terms of the percentage of the *1st position answers*, the percentage of the *not in first position answers* and the percentage of the *not found answers* which denotes the relevant answers that the system fails to return.

Experiments showed an amelioration in the percentage of the *1st position answers* over the *not in first position answers*. However, the percentages of the *not found answers* for both approaches are quite similar. The results of success rate according to the number of sessions are given in Figure 2. For the number of keywords and its impact on the success rate, we computed the percentage of answers that were considered as relevant to the users and returned as the first answer while changing the number of keywords in the query as shown in Figure 2.

1. www.wordnet.princeton.edu
2. www.MovieLens.com

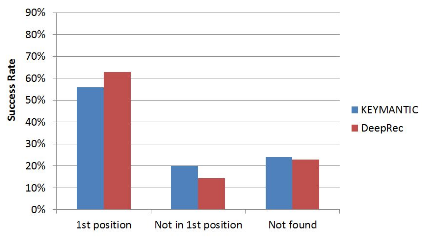

FIG. 1 – *Generated Success Rate for DeepRec and KEYMANTIC*

We remarked that no changes have been noticed for the *not found answers*. Some queries can generate the expected answer, but mostly, not in the first position, except for the ones that have been 'liked' before by the user. The variability of the processing times depends on the number of tables related to the queries. Interesting results are presented for users' interactions including sessions and queries. The number of answers responding to users' expectations has increased. As any CF based

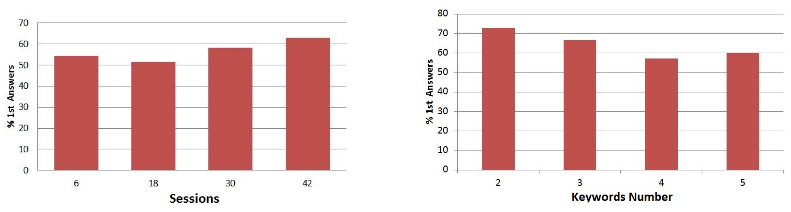

FIG. 2 – *Generated Success Rate according to Sessions and Keywords Number*

system, the more users we have and the more interactions they make with the system, the better its performance is. Experiments show that there isn't an optimal number of keywords. However, the number of keywords may increase either accuracy or serendipity depending on the terms employed by the user. The cold start problem of the CF technique is resolved by DeepRec. Furthermore, the personalization step allows to take into account every result that fits the user's request. This is very useful when a user searches an information that has previously sought. However, with KEYMANTIC, regardless the number of times the user interacted with the system, it always recomputes the answers ignoring what the user is probably expecting.

5 Conclusion

This paper addressed the problem of processing keyword queries over RDBs under the assumption that no prior access to data is possible. Our contribution consists in providing users with personalized results in this specific context by extending an existing approach with new components and resources namely, the personalization of the schema weight matrix and the answers, as well as users and information related to their interactions with the system. Beyond simply returning generic answers, our findings indicate that DeepRec provides personalized results that better fit the users' intent. Before the generation of the answers, we took advantage of the current user profile to further personalize answers based on the CF technique of RSs; favoring an answer already liked by the user. Providing personalized answers when no prior access to data is possible, makes DeepRec usable in Web databases and certain systems where building specialized indexes is not a possible option.

References

Aditya, B., G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, P. Parag, and S. Sudarshan (2002). Banks: Browsing and keyword searching in relational databases. In *Proceedings of the 28th international conference on Very Large Data Bases*, pp. 1083–1086.

Agrawal, S., S. Chaudhuri, and G. Das (2002). Dbxplorer: A system for keyword-based search over relational databases. In *ICDE*, pp. 5–16.

Bergamaschi, S., E. Domnori, F. Guerra, R. Trillo Lado, and Y. Velegrakis (2011). Keyword search over relational databases: A metadata approach. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, pp. 565–576.

Bergamaschi, S., F. Guerra, S. Rota, and Y. Velegrakis (2011). A hidden markov model approach to keyword-based search over relational databases. In *Conceptual Modeling ER 2011*, pp. 411–420.

Bergamaschi, S., F. Guerra, and G. Simonini (2014). Keyword search over relational databases: Issues, approaches and open challenges. In *Bridging Between Information Retrieval and Databases*, pp. 54–73.

Chbeir, R., Y. Luo, J. Tekli, K. Yetongnon, C. R. Ibanez, A. J. Traina, C. Traina Jr, and M. Al Assad (2014). Semindex: Semantic-aware inverted index. In *Advances in Databases and Information Systems*, pp. 290–307.

Hristidis, V. and Y. Papakonstantinou (2002). Discover: Keyword search in relational databases. In *Proceedings of the 28th international conference on Very Large Data Bases*, pp. 670–681.

Kacholia, V., S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar (2005). Bidirectional expansion for keyword search on graph databases. In *Proceedings of the 31st international conference on Very large data bases*, pp. 505–516.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics* (1), 32–38.

Tata, S. and G. M. Lohman (2008). Sqak: doing more with keywords. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 889–902.

Wang, H., K. Zhang, Q. Liu, T. Tran, and Y. Yu (2008). Q2semantic: A lightweight keyword interface to semantic search. In *The Semantic Web: Research and Applications*, pp. 584–598.

Résumé

Récemment, la recherche par mots-clés dans les bases de données relationnelles a suscité un intérêt grandissant en raison de sa facilité d'utilisation. Bien que des recherches approfondies fussent dernièrement effectuées dans ce contexte, la plupart de ces recherches non seulement nécessitent un accès préalable aux données, ce qui restreint leur applicabilité si cette condition n'est pas vérifiée, mais aussi renvoient des réponses très génériques. Cependant, fournir aux utilisateurs des réponses personnalisées est devenu plus que jamais nécessaire en raison de la surabondance de données qui peut déranger l'utilisateur. Le défi de retourner des réponses pertinentes et personnalisées qui satisfont les besoins des utilisateurs demeure. Inspiré par l'application réussie de la technique de filtrage collaboratif dans les systèmes de recommandation, nous proposons une nouvelle approche basée sur les mots-clés pour fournir aux utilisateurs des résultats personnalisés basés sur l'hypothèse que seulement une information sur le schéma de la base de données est disponible.