Volume 4, Issue 11, November 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

# An Efficient Forecasting of Difficult Keyword Queries over Database

[1]Varsha Khalate, [2]Prof. Shyam Gupta
[1]Department of Computer Engineering, SCOE, Sudumbare, India
[2]Internal Guide Department of Computer Engineering, SCOE, Sudumbare, India

***Abstract—Keyword queries on information bases offer easy accessibility to data, however usually suffer from low ranking quality, i.e., low exactitude and/or recall, as shown in recent benchmarks. it'd be helpful to spot queries that square measure probably to own low ranking quality to improve the user satisfaction. for example, the system could recommend to the user different queries for such onerous queries.. We set forth a high-principled framework and proposed novel algorithms to live the degree of the issue of a question over a dB, exploitation the ranking strength principle. supported our framework, we tend to propose novel algorithms that with efficiency predict the effectiveness of a keyword question. Our intensive experiments show that the algorithms predict the issue of a question with comparatively low errors and negligible time overheads.***

***Index Terms- Query performance, query effectiveness, keyword query, robustness, databases.***

## I. INTRODUCTION

Question interfaces (KQIs) for databases have attracted a lot of attention within the last decade because of their flexibility and easy use in looking and exploring the data. Since any entity in an exceedingly information set that contains the question keywords may be a potential answer, keyword queries typically have several potential answers. KQIs should determine the information desires behind keyword queries and rank the answers so the required answers seem at the highest of the list[3]. Unless otherwise noted, we tend to ask keyword query as question within the remainder of this paper.

Some of the difficulties of answering a query are as follows: First, unlike queries in languages like SQL, users do not normally specify the desired schema element(s) for each query term. For instance, query *Q*1: *Godfather* on the IMDB database (http://www.imdb.com) does not specify if the user is interested in movies whose *title* is *Godfather* or movies distributed by the *Godfather* company. Thus, a KQI must find the desired attributes associated with each term in the query. Second, the schema of the output is not specified, i.e., users do not give enough information to single out exactly their desired entities [4]. For example, *Q*1 may return movies or actors or producers. We present a more complete analysis of the sources of difficulty and ambiguity

There are cooperative efforts to produce standard benchmarks and analysis platforms for keyword search strategies over databases. One effort is that the data-centric

track of INEX Workshop [5] wherever KQIs square measure evaluated over the well-known IMDB information set that contains structured info regarding movies and other people in show business. Queries were provided by participants of the workshop. Another effort is that the series of linguistics Search Challenges (SemSearch) at linguistics Search Workshop, where {the information|the info|the information} set is that the Billion Triple Challenge data set at http://vmlion25.deri.de. it's extracted from completely different structured data sources over the online like Wikipedia. The queries square measure taken from Yahoo! keyword question log. Users have provided relevancy judgments for each benchmark.

These results indicate that even with structured information, finding the specified answers to keyword queries remains a tough task. additional apparently, looking nearer to the ranking quality of the most effective playacting methods on each workshops, we tend to notice that all of them have been playacting terribly poorly on a set of queries. For instance, take into account the question ancient Rome era over the IMDB data set. Users would really like to check data regarding movies that state ancient Rome. For this question , the state-ofthe- art XML search ways that we tend to enforced come rankings of significantly lower quality than their average ranking quality over all queries. Hence, some queries area unit more difficult than others. Moreover, regardless of that ranking method is employed, we tend to cannot deliver an inexpensive ranking for these queries. Table one lists a sample of such arduous queries from the 2 benchmarks. Such a trend has been additionally observed for keyword queries over text document collections .

It is necessary for a KQI to acknowledge such queries and warn the user or use various techniques like question reformulation or question suggestions . it's going to additionally use techniques like question results diversification . To the most effective of our data, there has not been any work on predicting or analyzing the difficulties of queries over databases. Researchers have projected some ways to sight tough queries over plain text document collections . However, these techniques aren't applicable to our drawback since they ignore the structure of the information. above all, as

 

mentioned earlier, a KQI should assign every question term to a schema element(s) within the information. It should additionally distinguish the specified result type(s). we tend to through empirical observation show that direct diversifications of these techniques area unit ineffective for structured data.

## II. LITERATURE SURVEY

Y. Luo, X. Lin, W. Wang, and X. Zhou,[2], In this paper, we study the effectiveness and the efficiency issues of answering top-k keyword query in relational database systems. We propose a new ranking formula by adapting existing IR techniques based on a natural notion of virtual document. Compared with previous approaches, our new ranking method is simple yet effective, and agrees with human perceptions. We also study efficient query processing methods for the new ranking method, and propose algorithms that have minimal accesses to the database. We have conducted extensive experiments on large-scale real databases using two popular RDBMSs. The experimental results demonstrate significant improvement to the alternative approaches in terms of retrieval effectiveness and efficiency.

V. Ganti, Y. He, and D. Xin,[3], Keyword search over entity databases (e.g., product, movie databases) is an important problem. Current techniques for keyword search on databases may often return incomplete and imprecise results. On the one hand, they either require that relevant entities contain all (or most) of the query keywords, or that relevant entities and the query keywords occur together in several documents from a known collection. Neither of these requirements may be satisfied for a number of user queries. Hence results for such queries are likely to be incomplete in that highly relevant entities may not be returned. On the other hand, although some returned entities contain all (or most) of the query keywords, the intention of the keywords in the query could be different from that in the entities. Therefore, the results could also be imprecise. To remedy this problem, in this paper, we propose a general framework that can improve an existing search interface by translating a keyword query to a structured query. Specifically, we leverage the keyword to attribute value associations discovered in the results returned by the original search interface

G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan[4], With the growth of the Web, there has been a rapid increase in the number of users who need to access online databases without having a detailed knowledge of the schema or of query languages; even relatively simple query languages designed for non-experts are too complicated for them. We describe BANKS, a system which enables keyword-based search on relational databases, together with data and schema browsing. BANKS enables users to extract information in a simple manner without any knowledge of the schema or any need for writing complex queries. A user can get information by typing a few keywords, following hyperlinks, and interacting with controls on the displayed results.

A. Trotman and Q. Wang [5] This paper presents an overview of the INEX 2011 Data-Centric Track. Having the ad hoc search task running its second year, we introduced a new task, faceted search task, which goal is to provide the infrastructure to investigate and evaluate different techniques and strategies of recommending facet-values to aid the user to navigate through a large set of query results and quickly identify the results of interest. The same IMDB collection as last year was used for both tasks. A total of 9 active participants contributed a total of 60 topics for both tasks and submitted 35 ad hoc search runs and 13 faceted search runs. A total of 38 ad hoc search topics were assessed, which include 18 subtopics for 13 faceted search topics. We discuss the setup for both tasks and the results obtained by their participants.

S. C. Townsend, Y. Zhou, and B. Croft,[6] We develop a method for predicting query performance by computing the relative entropy between a query language model and the corresponding collection language model. The resulting *clarity score* measures the coherence of the language usage in documents whose models are likely to generate the query. We suggest that clarity scores measure the ambiguity of a query with respect to a collection of documents and show that they correlate positively with average precision in a variety of TREC test sets. Thus, the clarity score may be used to identify ineffective queries, on average, without relevance information. We develop an algorithm for automatically setting the clarity score threshold between predicted poorly-performing queries and acceptable queries and validate it using TREC data. In particular, we compare the automatic thresholds to optimum thresholds and also check how frequently results as good are achieved in sampling experiments that randomly assign queries to the two classes.

*Clarity-score-based*: The methods based on the concept of *clarity score* assume that users are interested in a very few topics, so they deem a query easy if its results belong to very few topic(s) and therefore, sufficiently distinguishable from other documents in the collection [6], [7], [8], [9]. Researchers have shown that this approach predicts the difficulty of a query more accurately than pre-retrieval based methods for text documents [6]. Some systems measure the distinguishability of the queries results from the documents in the collection by comparing the probability distribution of terms in the results with the probability distribution of terms in the whole collection. If these probability distributions are relatively similar, the query results contain information about almost as many topics as the whole collection, thus, the query is considered difficult. Several successors propose methods to improve the efficiency and effectiveness of clarity score [7], [8].

However, one requires domain knowledge about the data sets to extend idea of clarity score for queries over databases. Each topic in a database contains the entities that are about a similar subject. It is generally hard to define a formula that partitions entities into topics as it requires finding an effective similarity function between entities. Such similarity function depends mainly on the domain knowledge and understanding users' preferences. For instance, different attributes may have different impacts on the degree of the similarity between entities.

 

## III. IMPLEMENTATION DETAILS

### 3.1 Basic Estimation Techniques:

**Data sets:** The INEX data set is from the INEX 2010 Data Centric Track [14]. The INEX data set contains two entity sets: movie and person. Each entity in the movie entity set represents one movie with attributes like title, keywords, and year. The person entity set contains attributes like name, nickname, and biography. The SemSearch data set is a subset of the data set used in Semantic Search 2010 challenge [15]. The original data set contains 116 files with about one billion RDF triplets. Since the size of this data set is extremely large, it takes a very long time to index and run queries over this data set. Hence, we have used a subset of the original data set in our experiments. We first removed duplicate RDF triplets. Then, for each file in SemSearch data set, we calculated the total number of distinct query terms in SemSearch query workload in the file. We selected the 20, out of the 116, files that contain the largest number of query keywords for our experiments. We converted each distinct RDF subject in this data set to an entity whose identifier is the subject identifier. The RDF properties are mapped to attributes in our model. The values of RDF properties that end with substring "#type" indicates the type of a subject. Hence, we set the entity set of each entity to the concatenation of the values of RDF properties of its RDF subject that end with substring "#type". If the subject of an entity does not have any property that ends with substring "#type", we set its entity set to "UndefinedType". We have added the values of other RDF properties for the subject as attributes of its entity. We stored the information about each entity in a separate XML file. We have removed the relevance judgment information for the subjects that do not reside in these 20 files. The sizes of the two data sets are quite close; however, SemSearch is more heterogeneous than INEX as it contains a larger number of attributes and entity sets.

**Query Workloads:** Since we use a subset of the dataset from SemSearch, some queries in its query workload may not contain enough candidate answers. We picked the 55 queries from the 92 in the query workload that have at least 50 candidate answers in our dataset. Because the number of entries for each query in the relevance judgment file has also been reduced, we discarded another two queries (Q6 and Q92) without any relevant answers in our dataset, according to the relevance judgment file. Hence, our experiments is done using 53 queries (2, 4, 5, 11-12, 14-17, 19-29, 31, 33-34, 37-39, 41-42, 45, 47, 49, 52-54, 56- 58, 60, 65, 68, 71, 73-74, 76, 78, 80-83, 88-91) from the SemSearch query workload. 26 query topics are provided with relevance judgments in the INEX 2010 Data Centric Track. Some query topics contain characters "+" and "−" to indicate the conjunctive and exclusive conditions. In our experiments, we do not use these conditions and remove the keywords after character "−". Some searching systems use these operators to improve search quality.

**Top-K results:** Generally, the basic information units instructured data sets, attribute values, are much shorter than text documents. Thus, a structured data set contains a larger number of information units than an unstructured data set of the same size. For instance, each XML document in the INEX data centric collection constitutes hundreds of elements with textual contents. Hence, computing Equation 3 for a large DB is so inefficient as to be impractical. Hence, similar to [13], we corrupt only the top-K entity results of the original data set. We re-rank these results and shift them up to be the top-K answers for the corrupted versions of DB. In addition to the time savings, our empirical results in Section 8.2 show that relatively small values for $K$ predict the difficulty of queries better than large values. For instance, we found that $K = 20$ delivers the best performance prediction quality in our datasets.

**Number of corruption iterations ($N$):** Computing the expectation in Equation 3 for all possible values of $\_x$ is very inefficient. Hence, we estimate the expectation using $N > 0$ samples over $M(|A| \times V)$. That is, we use $N$ corrupted copies of the data. Obviously, smaller $N$ is preferred for the sake of efficiency. However, if we choose very small values for $N$ the corruption model becomes unstable.

### 3.2 Existing Work

As per our recent literature reviews, there has not been any work on predicting or analyzing the difficulties of queries over databases. Researchers have proposed some methods to detect difficult queries over plain text document collections recently. But techniques are not applicable to our problem since they ignore the structure of the database.

There are two categories of existing methods, pre-retrieval and post-retrieval for predicting the difficulties of query. But below are limitations of this method:

- Pre-retrieval methods are having less prediction accuracies.
- Post-retrieval methods are having better prediction accuracies but one requires domain knowledge about the data sets to extend idea of clarity score for queries over databases.
- Each topic in a database contains the entities that are about a similar subject.
- Some Post-retrieval methods success only depends on the amount and quality of their available training data

### 3.2 Proposed Work

To address the issues of scalability and processing time under large datasets, we proposed new extended framework in which before going to calculate SR scores we are applying the k-means clustering to divide input dataset into number of clusters those having legitimate information's. Due to this, time required for predicting the difficult keywords over large dataset is minimized and process becomes robust and accurate.

 

**3.3 Algorithm:**

**SR(Structured Robustness) Algorithm:**

The Structured Robustness Algorithm (SR Algorithm), which computes the exact SR score based on the top *K* result entities. Each ranking algorithm uses some statistics about query terms or attributes values over the whole content of DB.

**Input:-** Query Q, Top-K result list L of Q by ranking function g , Metadata M , Inverted indexes I , Number of corruption iteration N.

**Output: -** SR score for Q.

1. SR ← 0; C← {}; //C catches $\lambda_T$ , $\lambda_S$
2. FOR i =1 → N Do
3. I' ← I; M' ← M ; L' ← L; // Corrupted copy of I ,M and L
4. For each result R in L DO
5. FOR each attribute value A in R DO
6. A' ← A; //Corrupted versions of A
7. For each keyword w in Q Do
8. Compute # of w in A' by Equation // If $\lambda_{T,w}$ , $\lambda_{S,w}$ needed but not in C, calculate and cache them
9. IF # of w varies in A' and A THEN
10. Update A' , M' and entry of w in I' ;
11. Add A' to R';
12. Add R' to L';
13. Rank L' using g, which returns L based on I' ,M'
14. SR+= Sim(L,L') ; //Sim computers Spersman correlation
15. RETURN SR ← SR / N ; //AVG score over N rounds

Each ranking algorithm uses some statistics about query terms or attributes values over the whole content of DB. Some examples of such statistics are the number of occurrences of a query term in all attributes values of the DB or total number of attribute values in each attribute and entity set. These global statistics are stored in *M* (metadata) and *I* (inverted indexes) in the SR Algorithm pseudo code.

SR Algorithm generates the noise in the DB on-the-fly during query processing. Since it corrupts only the top *K* entities, which are anyways returned by the ranking module, it does not perform any extra I/O access to the DB, except to lookup some statistics. Moreover, it uses the information which is already computed and stored in inverted indexes and does not require any extra index.

**3.4 Performance Study:**

**SR Algorithm:** We report the common computation time of SR score (SR-time) victimisation SR rule and compare it to the common question time interval (Q-time) victimisation PRMS for the queries in our question workloads. These times square measure presented in Table half-dozen for K = twenty. SR-time chiefly consists of two parts: the time spent on corrupting K results and therefore the time to re-rank the K corrupted results. we've rumored SR-time victimisation (corruption time + re-rank time) format. We see that SR rule incurs a substantial time overhead on the question process. This overhead is higher for queries over the INEX dataset, as a result of there square measure solely 2 entity sets, (person and movie), within the INEX dataset, and all query keywords within the question load occur in each entity sets.

Hence, consistent with Equation ten, each attribute worth in top K entities are going to be corrupted as a result of the third level of corruption. Since SemSearch contains much more entity sets and attributes than INEX, this method doesn't happen for SemSearch.

**QAO-Approx:** QAO-Approx on INEX and SemSearch, severally. We live the prediction effectiveness for smaller values of N exploitation average correlation score. The QAO-Approx rule delivers acceptable correlation scores and also the corruption times of regarding two seconds for N = ten on INEX and N = twenty on SemSearch. examination to the results of SR rule for N = 250 on SemSearch and N = three hundred on INEX, the Pearson's correlation score drops, because less noise is else by second and third level corruption. These results show the importance of those 2 levels of corruption.

## IV. CONCLUSION

In this paper, we analyze the characteristics of laborious queries and propose a unique framework to live the degree of issue for a keyword question over a information, considering each the structure and also the content of the information and also the question results. we have a tendency to assess our question issue prediction model against 2 effectiveness benchmarks for widespread keyword search ranking strategies. Our empirical results show that our model predicts the laborious queries with high accuracy. Further, we have a tendency to gift a set of optimizations to
minimize the incurred time overhead. we have a tendency to propose novel algorithms that expeditiously predict the effectiveness of a keyword question. Our in depth experiments show that the algorithms predict the problem of a question with comparatively low errors and negligible time overheads.

 

**REFERENCES**

[1] V. Hristidis, L. Gravano, and Y. Papakonstantinou, "Efficient IRstyle keyword search over relational databases," in Proc. 29th VLDB Conf., Berlin, Germany, 2003, pp. 850–861.

[2] Y. Luo, X. Lin, W. Wang, and X. Zhou, "SPARK: Top-k keyword query in relational databases," in Proc. 2007 ACM SIGMOD, Beijing, China, pp. 115–126.

[3] V. Ganti, Y. He, and D. Xin, "Keyword++: A framework to improve keyword search over entity databases," in Proc. VLDB Endowment, Singapore, Sept. 2010, vol. 3, no. 1–2, pp. 711–722.

[4] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword searching and browsing in databases using BANKS," in Proc. 18th ICDE, San Jose, CA, USA, 2002, pp. 431–440.

[5] A. Trotman and Q. Wang, "Overview of the INEX 2010 data centric track," in 9th Int. Workshop INEX 2010, Vugh, The Netherlands, pp. 1–32,

[6] S. C. Townsend, Y. Zhou, and B. Croft, "Predicting query performance," in Proc. SIGIR '02, Tampere, Finland, pp. 299–306.

[7] B. He and I. Ounis, "Query performance prediction," Inf. Syst., vol. 31, no. 7, pp. 585–594, Nov. 2006.

[8] K. Collins-Thompson and P. N. Bennett, "Predicting query performance via classification," in Proc. 32nd ECIR, Milton Keynes, U.K., 2010, pp. 140–152.

[9] C. Hauff, V. Murdock, and R. Baeza-Yates, "Improved query difficulty prediction for the Web," in Proc. 17th CIKM, Napa Valley, CA, USA, 2008, pp. 439–448

[10] Y. Zhou and W. B. Croft, "Query performance prediction in websearch environments," in Proc. 30th Annu. Int. ACM SIGIR, NewYork, NY, USA, 2007, pp. 543–550.

[11] C. Hauff, L. Azzopardi, and D. Hiemstra, "The combination andevaluation of query performance prediction methods," in Proc.31st ECIR, Toulouse, France, 2009, pp. 301–312.

[12] E. Yom-Tov, S. Fine, D. Carmel, and A. Darlow, "Learning to estimate query difficulty: Including applications to missing content detection and distributed information retrieval," in Proc. 28th Annu. Int. ACM SIGIR Conf. Research Development Information Retrieval, Salvador, Brazil, 2005, pp. 512–519.

[13] J. A. Aslam and V. Pavlu, "Query hardness estimation using Jensen-Shannon divergence among multiple scoring functions," in *Proc. 29th ECIR*, Rome, Italy, 2007, pp. 198–209.

[14] A. Trotman and Q. Wang, "Overview of the INEX 2010 datacentric track," in 9th Int. Workshop INEX 2010, Vugh, The Netherlands, pp. 1–32,

[15] T. Tran, P. Mika, H. Wang, and M. Grobelnik, "Semsearch ´S10,"in Proc. 3rd Int. WWW Conf., Raleigh, NC, USA, 2010.

 