# Predicting The Efficiency of Difficult Queries over Databases using SRC

**Shyamily Kuriakose, Rosna P. Haroon**

***Abstract*— Keyword queries on databases provide easy access to data, but often suffer from low ranking quality, i.e., low precision and/or recall, as shown in recent benchmarks. It would be useful to identify queries that are likely to have low ranking quality to improve the user satisfaction. For instance, the system may suggest to the user alternative queries for such hard queries. In this paper, we analyze the characteristics of hard queries and propose a novel framework to measure the degree of difficulty for a keyword query over a database, considering both the structure and the content of the database and the query results. We evaluate our query difficulty prediction model against two effectiveness benchmarks for popular keyword search ranking methods. Our empirical results show that our model predicts the hard queries with high accuracy. The proposed method use two level corruption module compare to structured robustness algorithm***

***Index Terms*— keyword query, query effectiveness , robustness.***

## I. INTRODUCTION

The data mining task is the automatic or semi-automatic analysis. It is used for large quantities of data to extract previously unknown interesting patterns. These are patterns are classified into three groups:

1. Cluster analysis: Cluster analysis includes the groups of data records of patterns that is it grouping a set of objects. These objects are lies in same group called cluster. Cluster analysis is not algorithm but it gives solution for algorithm. Such algorithm that is clustering algorithms are based on cluster model.
2. Anomaly detection: Anomaly detection is also known as outlier detection .It includes unusual records in data mining. It gives identification of data items.
3. Dependencies: Dependencies include the association rule mining. It discovers relations between variables in large databases.

Keyword query interfaces (*KQIs*) for databases have attracted much attention in the last decade due to their flexibility and ease of use in searching and exploring the data. Since any entity in a data set that contains the query keywords is a potential answer, keyword queries typically have many possible answers. KQIs must identify the information needs behind keyword queries and rank the answers so that the desired answers appear at the top of the list.

Databases contain entities, and entities contain attributes that take attribute values. 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. For example, *Q*1 may return movies or actors or producers.Keyword query interface (KQIs) evaluated on well-known IMDB data set. This data set contains structured information about movies and people in given business. IMDB database mainly contain three tables that is actor, director and movies.

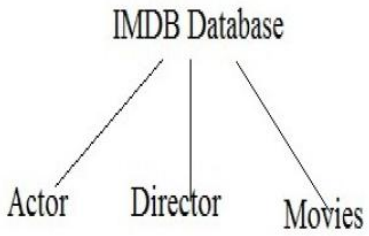


**Fig.1. IMDB database Table**

Methods uses for performance of information retrieval system are measure in relevant document and non-relevant document. These are having terms as follows:

- Precision: It is the fraction of documents retrieved that are relevant to user's information need.
- Recall: It is the fraction of the documents that are relevant to the query that are successfully retrieved to user.
- Average Precision: Precision and Recall are single value metrics that belongs to whole document list which are returned by system. Computation of Precision and recall ranks the sequence of document. Average precision computes the average value of documents.
- Mean Average Precision (MAP): MAP is used for set of of queries is the mean of the average precision scores for every query.

$$\mathbf{MAP} = \sum_{Q=0} \mathbf{AVERAGE\ (Q)\ /\ Q}$$

Where 'Q' = number of queries.

## II. RELATED WORKS

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things are satisfied,

**Revised Version Manuscript Received on October 09, 2015.**

**Kumari Shyamily Kuriakose**, Department of Computer Science and Engineering, Ilahia College of Engineering and Technology, Mulavoor, Muvattupuzha, (Kerala). India.

**Prof. Rosna P Haroon**, Department of Computer Science and Engineering, Ilahia College of Engineering and Technology, Mulavoor, Muvattupuzha, (Kerala). India.

ten next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

In this paper, we analyze the features of difficult queries over databases. It proposed novel method to detect such queries. We take advantage of the structure of the data to gain insight about the degree of the difficulty of a query given the database. We have implemented some of the most popular and representative algorithms for keyword search on databases such as SR algorithm and used them to evaluate our techniques. The results show that our method predicts the degree of the difficulty of a query efficiently and effectively.

***Process:***

1. Keyword search provides user friendly interface rather than Xpath and Xquery.
2. XML is used to store data in XML document format rather than table format.
3. XML provide security to data, user not easily recognize the XML data rather than traditional Table format.
4. User enters a keyword (i.e. Attribute, key, identifier).
5. SR scores measures the difficulty of queries over database.
6. The algorithm to compute the SR score, and parameters to tune its performance.
7. Include corruption module, compute the SR score, and parameters to tune its performance
8. It gives small time overhead compared to the query execution time.
9. Two level corruption module included in SRC Algorithm.
10. Perform prefix matching. It gives gram based, dictionary based, incremental based, neighbour based difficult queries.

## III. METHODOLOGY

### *A. Methods*

Researchers have proposed methods to predict hard queries over unstructured text documents. We can broadly categorize these methods into two groups:

1) Pre retrieval
2) Post retrieval

***Pre retrieval method*:** This method predict the difficulty of a query without computing its results. These methods usually use the statistical properties of the terms in the query to measure specificity, ambiguity, or term-relatedness of the query to predict its difficulty.

***Post-retrieval methods*:** This method utilize the results of a query to predict its difficulty and generally fall into one of the following categories.

- ***Clarity-score-based*:** The methods based on the concept of *clarity score* assume that users are interested in a very few topics
- ***Ranking-score-based:*** The ranking score of a document returned by the retrieval systems for an input query may estimate the similarity of the query and the document.
- ***Robustness-based*:** Another group of post-retrieval methods argue that the results of an easy query are relatively stable against the perturbation of queries, documents or ranking algorithms.

### *B. Algorithm*

#### *1. Structured Robustness Algorithm*

Algorithm shows 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. 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.

Fig. 2.(a) shows the execution flow of SR Algorithm. Once we get the ranked list of top K entities for Q, the corruption module produces corrupted entities and updates the global statistics of DB. Then, SR Algorithm passes the corrupted results and updated global statistics to the ranking module to compute the corrupted ranking list. SR Algorithm spends a large portion of the robustness calculation time on the loop that re-ranks the corrupted results (Line 9 in SR Algorithm), by taking into account the updated global statistics. Since the value of K (e.g., 10 or 20) is much smaller than the number of entities in the DB, the top K entities constitute a very small portion of the DB.

$$KSW = \frac{\text{Number of occurrences of a query term in all attributes values of the Particular table}}{\text{Number of occurrences of a query term in all attributes values of the DB}}$$

KSW means Keyword Specific Weight.
Algorithm 1 (Q,L,M,I,N)
Input : Query Q, top k results list L, metadata M, Inverted Index I, Number of Iterations N
Output : Structured Robustness Score of Q

Step 1 : SR = 0 , c={}
Step 2 : for i = 1 to N do
Step 3 : Corrupt the M,I,L
Step 4 : For each result R in L do
Step 5 : Calculate Attribute Specific Weight
Step 6 : Update List M,L and Attribute List
Step 7 : Add A' to R'
Step 8 : Add R' to L'

Step 9 : Calculate Rank with the help of ASW

Step 10 : Return SR = SR/N

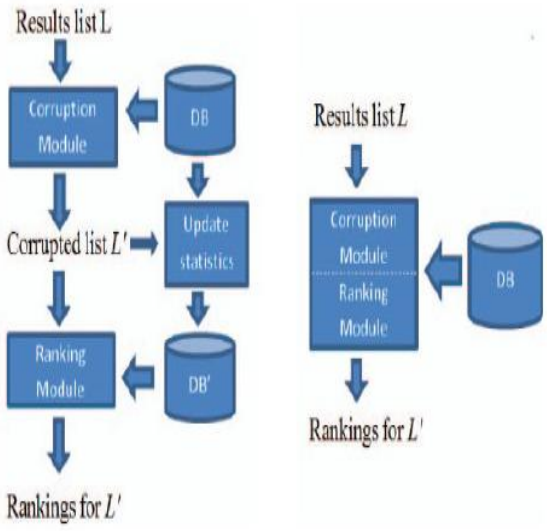


**Fig.2.Execution flow of SR algorithm And SRC (a)SR algorithm (b)SRC Algorithm**

Thus, the global statistics largely remain unchanged or change very little. Hence, we use the global statistics of the original version of the DB to re-rank the corrupted entities. If we refrain from updating the global statistics, it can combine the corruption and ranking module together. This way re-ranking is done on-the-fly during corruption. SR Corruption algorithm is illustrated in Fig. 2(b).

### *2 SRC (Structured Robustness Corruption)*

SRC Algorithm mainly depends on prefix length of query. By using this method compute gram based, neighbor based and dictionary based difficult queries. SRC can predict a word or phrase that a user may type in next based on the partial string the user has already typed. Prefix matching is the proposed solutions that use auxiliary tables as index structures and SQL queries to support search-as-you-type. Extended the techniques to the case of fuzzy queries, and proposed various techniques to improve query performance. The proposed incremental-computation techniques to answer multi keyword queries, and studied how to support first-N queries and incremental updates.

The experimental results on large, real data sets showed that the proposed techniques can enable DBMS systems to support search-as-you-type on large tables. There are several open problems to support search-as you- type using SQL. One is how to support ranking queries efficiently. Another one is how to support multiple tables.

## IV. PERFORMANCE STUDY

**SR Algorithm:** Report the common computation time of SR score (SR-time) victimization SR rule and compare it to the common question time interval (Q-time) victimization PRMS for the queries in our question workloads. These time 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. It have rumored SR-time victimization (corruption time + re-rank time) format. It can 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 Sem Search.

**SRC-Approx:** SRC-Approx on INEX and Sem Search, severally. Live the prediction effectiveness for smaller values of N exploitation average correlation score. The SRC-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.

## V. CONCLUSION

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. It 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. It 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. 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. J. Kim, X. Xue, and B. Croft, "A probabilistic retrieval model for semistructured data," in Proc. ECIR, Tolouse, France, 2009, pp. 228–239.
5. N. Sarkas, S. Paparizos, and P. Tsaparas, "Structured annotations of web queries," in Proc. 2010 ACM SIGMOD Int. Conf. Manage. Data, Indianapolis, IN, USA, pp. 771–782.
6. G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword searching and browsing in database using BANKS," in Proc. 18th ICDE, San Jose, CA, USA, 2002, pp. 431–440.
7. Manning, P. Raghavan, and H. Schütze, An Introduction to Information Retrieval. New York, NY: Cambridge University Press, 2008.
8. Trotman and Q. Wang, "Overview of the INEX 2010 data centric track," in 9th Int. Workshop INEX 2010, Vugh, The Netherlands, pp. 1–32,
9. T. Tran, P. Mika, H. Wang, and M. Grobelnik, "Semsearch ´S10,"in Proc. 3rd Int. WWW Conf., Raleigh, NC, USA, 2010.