Predicting the Difficulty of Keyword Queries

Hojage Swati B.
Department of Computer Engineering
swati.hojage93@gmail.com

Anjali Chachra
Department of Computer Engineering
anjalichachra24@gmail.com

Priyanka Patil
Department of Computer Engineering
patilpriyanka561@gmail.com

***Abstract*—** Querying using keyword is easily most widely used form of querying data. Keyword Queries on databases provide easy access to data but always suffers from low ranking quality. Current techniques for keyword search on databases may often return incomplete and imprecise results. Generally in existing system result is retrieved based on user keyword query, but it may be possible that due to low ranking the expected result is discarded. It is defined to get queries with low ranked result to improve the user satisfaction. For instance, the system may suggest to the user alternative queries for such hard queries. The aim of this paper is to show the analysis of the characteristics of hard queries and different algorithms for prediction of difficult keyword queries. The proposed system is well effective than the existing system in terms of accuracy rate, quality of result.

***Keywords*—** *Query performance, query effectiveness, keyword query, Structured Robustness (SR).*

*****

I. Introduction

Predicting the difficulty of keyword search plays an important role in information retrieval. Whenever user issues any keyword query, in response to that Keyword Query Interface(KQI) returns ranked list of keywords due to its flexibility and easy use in searching and exploring the data [1]. KQI must identify the information needs behind keyword queries and rank the answers so that the answers appear at the top of the list [1]. Databases contain entities, and entities contain attributes that take attribute values. Existing keyword search over database primarily return tuples whose attribute value contain all query keyword. Some of the difficulties of answering a query are as follows: First dissimilar queries in languages like SQL, users do not habitually stipulate the preferred schema elements for each query term. For instance,query1: Godfather on the IMDB database does not specify if the user is interested in movies whose title is Godfather or movies distributed by the Godfather corporation. It is important for a KQI to recognize such queries and warn the user or suggest alternative techniques like query reformulation or query suggestions. Keyword Query Interface in this paper uses IMDB database for evaluating the work, this database is related to movies n people in given business. IMDB database generally consist of three tables here actor, director and movies.

II. Related work

2.1 Introduction of BANKS (Browsing And Keyword Searching)system

With today's fast growth of web technology most of the users need to access the database without having detailed knowledge of schema or query language. G. Bhalotia describe BANKS, a system which does keyword based search on relational database and enables users to access or extract information in simple manner. BANKS enables a user to get information by typing a few keywords, following hyperlinks[3].

2.2 Query Performance Prediction

Query performance prediction refers to the process estimating the quality of the expected result of query .Current predictors for query performance involve the use of relevance scores, which are time consuming hence they are not very suitable for practical applications . B. He and I. Ounis proposed the list of predictors for query performance[4]. The proposed predictors are based on the features of queries. Below is the list of those predictors

2.1.1 Query Predictors:

Query length : The query length is the number of non-stop words in the query.

The distribution of informative amount in the query terms: : Given a query Q, the distribution of informative amount in its composing terms, called γ2, is represented as:

$$\gamma 2 = idfmax/idfmin \quad (1)$$

where idf max and idf min are the maximum and minimum idf (Inverse Document Frequency) among the terms in query Q respectively.

Query scope: The query scope is:

$$\omega = -\log(nQ/N) \quad (2)$$

Where nQ is the number of documents containing at least one of the query terms, and N is the number of documents in the whole collection.

Query clarity: Clarity of query is inversely proportional to ambiguity of the query. The query performance is correlated with the clarity score of the query. Researchers have shown that this approach predicts the difficulty of a query more accurately than pre-retrieval based methods for text documents.

2.3 Prediction Methods

The different approaches for the prediction of query performance are being proposed and divided into three categories: pre-retrieval predictors, post-retrieval predictors, and learning predictors.

2.3.1 Pre-retrieval predictors:

Pre-retrieval predictors can be calculated from features of the query or collection, without requiring the search system to evaluate the query itself. The information required by such prediction is obtained from various collection, document and term occurrence statistics. The researchers have proposed two categories of pre-retrieval method, first predictors that are based on similarity between queries and second predictors based on distribution of query terms.

2.3.2 Post-retrieval predictors:

Post-retrieval predictor uses the outcome from the evaluation of underlying keyword search. Post-retrieval predictors shows high level of effectiveness by processing the query first and generating the result and this result is usually the subject for future analysis. Post-retrieval methods generally falls into following categories:

- Clarity-score based
- Ranking-score based
- Robustness based

2.3.3 Learning Predictors:

Learning predictors makes the use of neural networks or support vector machine to train the examples of easy and difficult queries. The learned estimator is used then for predicting the difficulty of queries. These Predictors require suitable training dataset otherwise overhead gets incurred during the training stage.

2.4 Ranking Robustness

Ranking robustness refers to the property of a ranked list of documents [5]. Robustness shows how strong the ranking is in the presence of noise in the ranked documents. For a requested query and a ranking function the ranking robustness is calculated by comparing the ranking from the corrupted database and ranking from original database. Ranking robustness generally finds in the field of noisy data retrieval which refers to the property of ranked list of documents. The way of measuring ranking robustness is begin by calculating ranking robustness in noisy retrieval data. Next we compare the ranked document list from corrupted collection of data to corresponding ranked list using same query and ranked function. Specifically, suppose we have query Q, ranking function G and collection C. We generate corrupted collection C' by sampling from the document models of the documents in C. Then we perform retrieval on both C and C' and two ranked list L and L' are returned respectively. Finally we compute the similarity between the two rankings. Note that L is a fixed ranked list while L' is a random variable. We call the expected similarity between L and L' the robustness score and use it to measure ranking robustness. This process is illustrated in below figure.

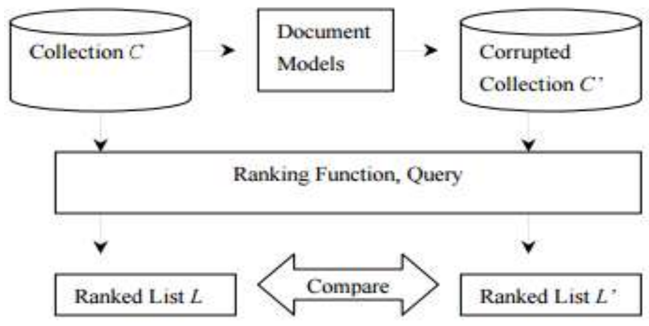


Figure 2.4.1: Robustness Score Calculation[1]

III. Proposed work

The Literature survey gives the brief idea about some query predictors used for predicting the difficulty of queries and keywords given by users. The further work is based on different algorithms which are used for prediction of difficulty of keyword queries and characteristics of hard queries.

3.1 Characteristics of hard queries over database

The queries are said to be hard or difficult when they are said to be difficult or complex for answering by Information Retrieval system. A query is said to be more difficult if its ranking over the original and its corrupted versions of the data are less matching with each other.

Following are some sources of difficulty for answering a query over given database [2].

3.1.1 A Entity Matching Many Terms:

If one or more entities in the database match the terms in a query, the query is said to be less specific and it is harder to answer correctly. For example if any user submits his query Q1: *Smith in* IMDB (Internet Movie Database).The keyword query interface must return the desire answer that should satisfied user's need. If there is more than one person named *Smith* in the IMDB database then it will be hard to predict the correct answer by the retrieval system. As opposed to query Q1, query Q2: KIM matches the smaller number of people in IMDB, so it is easier for query interface to return desired answer.

3.1.2 Attribute Level Ambiguity:

Each attribute explains the different features of an entity. If a query matches different attributes in the database that means it will have more set of answers and hence it will have higher attribute level ambiguity. For example if any keyword or entity contains in two different attributes of database then Keyword query interface must return desired attribute value for given keyword for obtaining correct result.

3.1.3 Entity Level Ambiguity:

Each entity mostly contain the data about other different entities. So, if any query matches from these different entity sets then it will be having higher ambiguity at that entity level. For example: IMBD contains two entity sets called "Movie" and "People". Consider the query Q3: 'Engineer' which is included in both Movie and People so the query becomes complex to find the proper answer as keyword interface doesn't know whether user is interested in which entity set from given one. If in case of query Q4: COMEDY matches with the entities of Movie entity set. So it will be less difficult for keyword query interface to find relevant answer as compared to query Q3.

3.3 A FRAMEWORK FOR PREDICTION

3.3.1 Noise Generation In Database:

In Order to compute the Structure Robustness Score(SR-score) for query Q over database DB given retrieval function g:

$$\begin{aligned} &SR(Q, g, DB, X_{DB}) \\ &= E\{ Sim(L(Q, g, DB), L(Q, g, X_{DB}))\} \\ &= \sum_x Sim(L(Q,g,DB), L(Q,g,x) f X_{DB}(x) \qquad (3) \end{aligned}$$

Where Sim denotes The Spearman rank correlation between ranked answer list. We need to define noise generation model

$fX_{(DB)\,(M)}$ for database DB. With the help of the combination of three corruption levels i.e on the value itself, it's attribute and its entity set each attribute value is corrupted .The corruption model have to reflected the properties about search on structure data where the statistical properties of keyword queries in attribute values, attributes and entity sets are important. The noise generated in attribute values will propagate up to their attributes and entity sets. The given example shows the necessity of generating such a noise. Let T_1 be an attribute whose attribute values are A_1 and A_2,where A1 contains w1 and A2 doesn't contain w1.In the Noisy version of T1 both A1 and A2 will contain w1.For instance, if an attribute value of attribute *title* includes keyword 'Crime Family' then Crime Family' may appear in any attribute value of *title* in corrupted version.

3.3.2 Ranking in Original and Corrupted Data:

Defining the data corruption for structured data, is the key challenge in Ranking Robustness principle. For that two different ranking algorithms have been evaluated for our effectiveness of query performance, those are PRMS [6] and IR-Style[7].

PRMS model: In PRMS model we will assume the query Q = (q1,q2.... q_m) where m is set of words and collection C on XML schema E= (E_1, E_2....En) which composed of n elements. Based on this simple schema E a document score is being formed by following weighted average og element level scores as following:

$$P(Q|d) = \prod \mu_j P_{QL}(qi|ej)$$

$$P_{QL}(qi|ej) = (1-\lambda)P(qi|ej)+\lambda P(qi|Ej)$$

where P_{QL}(qi|ej) is query likelihood score of element ej. We can adjust the parameter λ in PRMS to get the best Mean Average Precision(MAP) and for effective query performance evaluation.

IR-Style model: IR-style returns join tuples from multiple relations in database to identify tuple trees with all keywords. For given query Q BANKS uses the following scoring scheme:

Score (T,Q)={ f_r (T)+ f_n(T)+ f_p(T)} if T contains all words in Q.

where $f_r(T)$ measures how "related" the relations of the tuples of T are, $f_n(T)$ depends on the weight of the tuples of T –as determined by a Page Rank-inspired technique and $f_p(T)$ is a function of the weight of the edges of T .

3.3.3 Structure Robustness Algorithm:

This algorithm computes the exact SR score based on top K result entities for measuring difficulty of a query. It uses two datasets INEX and SEMSEARCH for evaluating query.SR algorithm first gives top k ranked list and updates global statistic stored in metadata. Then noise is generated in database which corrupts only top k entities which are returned by ranking module.SR algorithm finds corrupted result and passes it to the ranking module to generate corrupted ranking list. The SR algorithm uses Spearman rank correlation and Pearson's Correlation for computing similarity function which ranges from -1 to 1.A value close to 1 gives positive correlation result between two ranking and a value close to -1 gives negative correlation result. If no correlation it means correlation result is close to zero value.

Spearman rank correlation formula is given as:

SR (Q, g, DB, X_{DB})

$$= \sum_x \mathrm{Sim}\big(L(Q,g,DB), L(Q,g,x)\big)\mathrm{fXDI} \qquad (4)$$

Where, Q= query

g is the retrieval function

X_{DB} is noisy version of database

x is sample drawn from fX_{DB}.

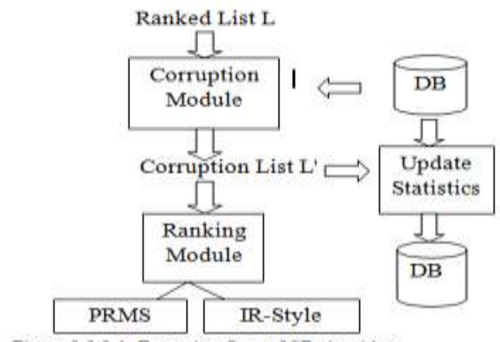


Figure 3.3.3.1: Execution flow of SR algorithm

According to the computed result we can say that SR algorithm is efficient than other algorithms used for predicting efficiency. Following table shows that Spearman correlation is more efficient than Weighted Information Gain, Clarity Score and Unstructured Robustness Method.

Table 3.3.3.1: Spearman's Correlation of Average Precision when k=10

METHOD	SR	WIG	CR	URM
INEX	0.303	0.242	0.199	0.196
SEMSEARCH	0.519	0.27	0.182	-0.012

Table 3.3.3.2: Pearson's Correlation of Average Precision when K=10

METHOD	SR	WIG	CR	URM
INEX	0.486	0.176	0.266	0.093
SEMSEARCH	0.471	0.107	0.111	0.247

Table 3.3.3.3: Effect of K on correlation of average precision and SR score using IR-Style ranking on Semsearch.

K	10	20
Pearson's Correlation	0.565	0.544
Spearman's Correlation	0.502	0.507

The drawback of SR algorithm is time consuming as it spends more time on robustness calculation. So the researchers have proposed better approach for this called as Approximation Algorithm which improves the efficiency and it is the combination of QAO-Approximation (Query Specific

Attribute Value Only Approximation and SGS Approx(Static Global State Approximation).

3.3.4 Approximation Algorithm

QOS-Approx- QOS approximation gives following observation:
The drawback of SR algorithm is solved in this approximation algorithm.QOS Approx corrupts only the attribute value that match at least one query term from given query. Hence time spent on corruption is significantly get reduced if only attribute values are corrupted. It is being observed in QOS Approx that the noise in the attribute values which contains query terms dominates the corruption effect as well as the number of attribute values which contains at least a single query terms are smaller than all attribute values in each entity.
SGS-Approx- As said earlier SR algorithm spends more time for robustness calculation which re ranks the corrupted result. Due to having smaller value of K(e.g. 10 or 20) top k entities constitute small portion of the DB. Thus global statistics remain unchanged or less changed. Hence original version of the database is used for global statistic. In SGS Approx these global statistics remains as it is. The figure below shows the execution flow of SGS Approx.

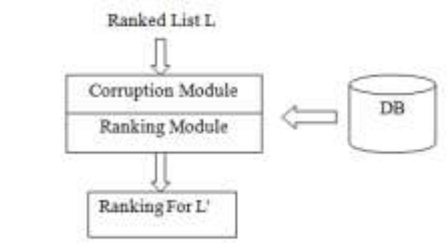


Figure 3.3.4.1: Execution Flow Of SGS-Approx

Once the ranked list of top K entities for query Q are found then corruption module produces corruption entities and global statistic is get updated accordingly.

3.4 Workflow Architecture

The workflow architecture of the whole idea is depicted in figure.

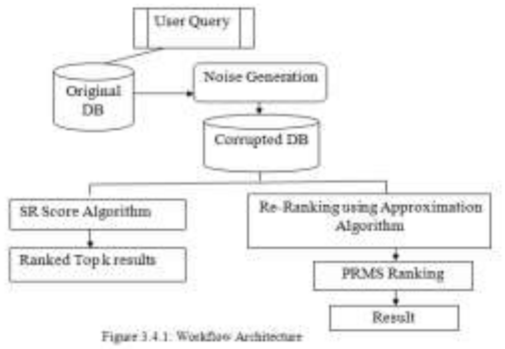
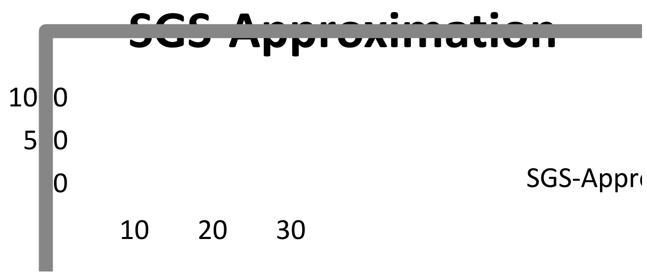


Figure 3.4.1: Workflow Architecture

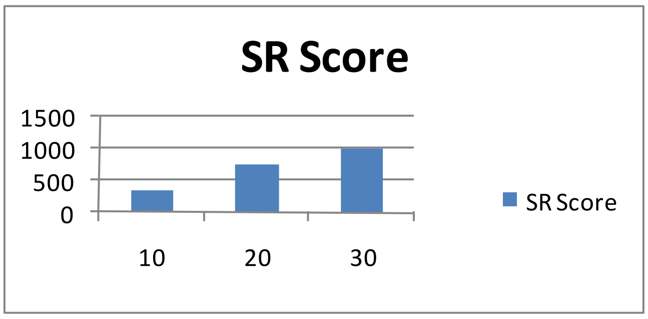
The fig above shows the complete workflow of proposed system and it shows how to detect hard queries with noise generation and produces quality result for given query. User can enter a query according to his need. The database is being corrupted by generating noise. Noise generation is nothing but addition of repeated attribute values. The entered query is searched in corrupted database. Then keywords in query are matched against each attribute and appropriate results are generated..The obtained SR score shows the complexity of query. Finally using PRMS ranked list is generated and provided to the user. The ranking gives appropriate result by presenting the result at the top of the list.

IV COMPARATIVE ANALYSIS AND RESULT

According to the existing study the average computation time required for producing SR score by Structure Robustness algorithm is a time consuming task. In this the operating cost is higher for queries over the database. Hence approximation algorithm is introduced for computing the result better than SR algorithm. The performance offered by Query-specific Attribute values Only Approximation (QAO-Approx) and Static Global State Approximation (SGS-Approx) is evaluated by parameters such as precision, time complexity, and SR score. Based on the comparison and result from the experiment it shows the proposed approach gives better result.
Comparison of SR Algorithm and Approximation Algorithm on the basis of Time **Complexity** with the help of graph is given as below:



No Of Corruption Iterations(N)
Figure 4.1: Result of SR-Score in SGS



No of Corruption Iterations (N)
Figure 4.2: Result of SR-Score

No Of Corruption Iterations(N)
Figure 4.3: Time Complexity of Query-Specific Attribute values

Table 4.1 Comparison table of SR and Approximation algorithms

Parameters	**SR Algorithm**	**QOA-Approx**	**SGS-Approx**
Time Overhead	High	Less than SR	Less than SR and QOA
Precision rate	Lower	Higher	Higher
Pearson's Correlation score on Semesearch database when k=10	0.565	0.44	0.49
Spearman's Correlation score on Semsearch database when k =10	0.502	0.5	0.56

V CONCLUSION

To demonstrate that the present forecast techniques for questions over unstructured information can't be utilized to take care of the issue of anticipating the productivity, this novel methodology is proposed for organized information. By considering the entire methodology one must say that if the SR calculation is causing all the more overhead time to process SR score which assesses the trouble of given inquiry we can give Approximation calculations as an answer for this issue which will give less overhead time for figuring the outcome. It will fulfill client's need as it gives higher expectation exactness and minimized the acquired time overhead. The finish of the structure additionally demonstrates results and investigation of examinations directed gives the consequence of expectation with low mistakes and irrelevant time overhead.

VI REFERENCES

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

[2] Shiwen Cheng, Arash Termehchy, and Vagelis Hristidis, “Efficient Prediction of Difficult Keyword Queries over Databases” IEEE Transactions on Knowledge and Data Engineering, Vol. 26, No. 6, June 2014.

[3] 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.

[4] B. He and I. Ounis, “Query performance prediction,” Inf. Syst., vol. 31, pp. 585–594, November 2006.

[5] Y. Zhou and B. Croft, ―Ranking robustness: A novel framework to predict query performance,‖ in *Proc. 15th ACM Int. CIKM*, Geneva, Switzerland, 2006, pp. 567–574.

[6] J. Kim, X. Xue, and B. Croft, "A probabilistic retrieval model for semi structured data"‖ in *Proc. ECIR*, Tolouse, France, 2009, pp. 228–239.

[7] Hristidis V., Gravano L. and Papakonstantinou Y. (2003), ‘Efficient IR-style Keyword Search over Relational Databases’, in Proc. 29 VLDB Conf., Berlin, Germany, pp. 850–861.

[8] V. Ganti, Y. He, and D. Xin,(2010) “Keyword++: A framework to improve keyword search over entity databases.

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