International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 12 (2015) pp. 31223-31232

http://www.ripublication.com

An Efficient Ranking And Predicting Based Keyword Query Interface

S.Jayasundar[1], Dr.V.N.Rajavarman[2], P.Dinesh Kumar[3]

[1]Research Scholar Dr.M.G.R Educational and Research Institute. University
[2]Professor Dr.M.G.R Educational and Research Institute. University
[3]Associate Professor, Dr.M.G.R Educational and Research Institute. University

ABSTRACT

The amount of information in the world is increasing exponentially. Keyword search has proven to be an effective method to discover and retrieve information from database as evidenced by the success of search engines. Unfortunately, much common information retrieve and management systems do not support the familiar query search that people how expect. Keyword Query Interface (KQI) technique is most common technique for access data from the database due to their flexibility and exploring the data and ease of use in searching, but it is most suffer from low precision and low ranking quality. In this paper we evaluate the difficulty of a hard keyword query over databases. And we proposed the Structural Robustness (SR) score based ranking algorithm can compute the relevance of the query. Our experiments results show that our proposed ranking algorithm efficiently to predict the difficulty of the query with higher accuracy.

Index Terms: Keyword query, Unstructured database, Robustness, Query efficiency

1. INTRODUCTION

Data mining is a field which has seen rapid advances in recent years [1] because of the immense advances in hardware and software technology which has led to the availability of different kinds of data. This is particularly true for the case of structured data, where the development of hardware and software platforms for the web and social networks has enabled the rapid creation of large repositories of different kinds of data. While structured data is generally managed with a database system, and the data is typically managed via a search engine due to the lack of

structures [2]. A search engine enables a user to find useful information from a collection conveniently with a keyword query.

Predicting the retrieval performance or determining the degree of difficulty of a query is a challenging research area which has received a lot attention recently [3] [4]. The aim is to create an efficient method (predictors) for the task, as a reliable low precision and accurate prediction mechanism would enable the creation of more adaptive and intelligent retrieval systems. Keyword query interface is a popular technique for retrieve information from database. Since any entity in a data set that contains the query keywords is a potential answer, keyword queries typically have many possible answers. The KQI must identify the information needs behind keyword queries and rank the answers so that the desired answers appear at the top of the list [5] [6]. Researchers have proposed lots of techniques to detect difficult queries over text document collections. However, these techniques are not applicable to our problem since they ignore the structure of the database. In particular, as mentioned earlier, a KQI must assign each query term to a schema elements in the database. It must also distinguish the desired result types. We empirically show that direct adaptations of these techniques are ineffective for structured data.

In this paper we evaluate difficult keyword query over databases and propose a novel method to detect such queries. We have used structured data to gain insight about the degree of the difficulty of a query given to the database. We have implemented some of the most popular and representative algorithms for keyword search on databases 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.

2. RELATED WORK

Researchers have been proposed many methods to predict the difficult query over structured and unstructured documents [7] [8]. Among recent systems that enable keyword-based search we find Microsoft DB Xplorer [Chaudhuri et al., 2006]. It uses a symbol table to store tables, columns, and rows of all data values, which is looked up during the search to identify the locations that contain all the keywords appearing in the question. Anyway there is no need to maintain a symbol table if we can rely on the actual database and its underlying metadata. The BANKS [Hulgeri et al., 2010] and Object Rank [Balmin et al., 2012] systems apply ranking to keyword search over databases: results are ranked with respect to their relevance, computed using an approach similar to PageRank in BANKS while Object Rank applies authority-based ranking. One beneficial feature of BANKS is that it also takes into account metadata while performing the search. Both these systems use graphs to model relational databases, where each node represents an object of the database. The ranked answer is a sub-graph where weighted nodes are ordered based on descending relevance. The same method is used in our approach to rank results in a way that reflects the correctness of the answers generated by the system with respect to the question, i.e. based on the number of matching substructures between the relational tree and the propositional tree.

Precis [Koutrika et al., 2009] is another system that uses both an inverted index and a directed schema graph to generate a new database and a personalized natural language synthesis of result. Besides, keyword question answering is implemented using huge inverted indexes and symbol tables that need to be rebuilt whenever the database is updated. Indeed, this approach is not suitable for very large databases where the high number of tables and rows would prohibit symbol table maintenance. In addition, it's worth noting that these systems don't consider solutions that include two tuples from the same relation. That is, they retrieve a single-value answer, while the solution is often a set of values or strings.

The main techniques in that prediction is pre retrieval output and post retrieval methods. Pre-retrieval methods [9], predict the query difficulty without computing its results. These methods are mostly used in the statistical properties of the terms in the query to measure ambiguity, or term relatedness, specificity of the query to predict its difficulty. But this method has limited prediction accuracies. Post-retrieval methods has been used in results of the query to predict its difficulty and it was having many categories.

2.1. Ranking-score-based

In this method score of a document returned by the retrieval systems for an input query and the similarity of the document is estimated. Zhou and Croft argue that the information gained from a desired list of documents should be much more than the information gained from typical documents in the collection for an easy query. They measure the degree of the difficulty of a query by computing the difference between the weighted entropy of the top ranked results' scores and the weighted entropy of other documents' scores in the collection [14].

2.2. Robustness-based

Robustness based ranking algorithm is another post retrieval method. This method that the result of an easy query is relatively stable against the perturbation of queries or ranking algorithms and some time the machine learning techniques are used to predict the hardness of difficult queries [15]. When we applied these methods to structured data they have similar limitations. Some researchers propose frameworks that theoretically explain existing predictors and combine them to achieve higher prediction accuracy [16], [17].

3. HARD QUERY DATABASE PROPERTIES

The queries which are difficult to answer correctly to the user are called hard keyword queries. The researchers propose following sources of difficulty for answering a query over a database [18] as follows

3.1. More entities matches the term in query

If more entities match the terms in a query, the query is less specific and it is harder to answer properly. For example: there are more than one person named DAVID in the IMDB database and user submits query Q1: DAVID, the keyword query interface

must resolve desire DAVID that satisfy user's information need. If the more number of people in IMDB is DAVID then it will be hard to predict the correct answer. As oppose to Q1, Q2: KIM matches the smaller number of people in IMDB, so it is easier for keyword query interface to return relevant answer.

3.2. Attribute level ambiguity

Each attribute explains a different feature of an entity and defines the context of terms in attribute values of it. If a query matches different attributes in its candidate answers, it will have a more diverse set of potential answers in database, and hence it has higher attribute level ambiguity. For example: Q3: GODFATHER, contains in title and the distributor of IMDB dataset. Keyword query interface must find out the desired attribute for GODFATHER to find correct answer. Answer for the query Q4: SPEED does not match with any instance of attribute distributor, so keyword query interface easily predict the relevant answer for this query.

3.3. Entity level ambiguity

Each entity set contains the information about a different type of entities and defines another level of context (in addition to the context defined by attributes) for terms. Hence, if a query matches entities from more entity sets, it will have higher entity set level ambiguity.

4. EFFICIENT COMPUTATION OF SR SCORE

The basic information of structured database attribute value is very lower than plain documents but the structured database contains same size of information in unstructured dataset. For instance every XML document of INEX dataset collections contains thousands of elements with textual information.

4.1. SR (Structured Robustness) Algorithm

In structured robustness algorithm which calculate SR score based on K result entity. Each ranking algorithm uses some statistics about query terms or attributes values over the whole content of DB. The global attribute data entity set stored in M (Metadata) and *I* (inverted indexes) in the SR Algorithm pseudo code.

4.2. Input

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

4.3. Output:

S R score for Q
SR ← 0; C ← {}; // C catches λ_T, λ_S for keywords in Q
FOR $i = 1 \rightarrow N$ DO
$I' \leftarrow I;\ M' \leftarrow M;\ L' \leftarrow L;$/Corrupted copy of I, M and L
FOR each result R in L DO
FOR each attribute value A in R DO

$A' \leftarrow A$;
FOR each keywords ω in Q DO
Compute # of ω in A' by equation; // ifλ_T, ω,λ_S, ω needed
But not in C, calculate and cache them
IF # of ω varies in $A'$and A THEN
UpdateA',M' and entry of ω in I';
Add A' to R' ;
Add $R'$to L':
Rank L' using g, which returns L, based on I', M' ;
Compute $SR+= Sim(L, L')$; // Sim computes spearman correlation
RETURN ← SR/N ; // AVG score over N rounds

Based on our results the following reasons the SR algorithm increases the processing query time and efficiency. First each attribute value of K result to be corrupted second one entity has many attributes so they will be increase the attribute values corruption. Third the SR ranking algorithm has to re-rank top result in N time.

5. APPROXIMATION ALGORITHMS FOR SR ALGORITHM

In this section we are proposed different approximation algorithms to increase the efficiency of structured robustness algorithm.

5.1. QAO-Approx algorithm

Query-specific Attribute values only Approximation (QAO-Approx)corrupts only the attribute values that match at least one query term. This approximation algorithm influences the following observations: The number of attribute values that contain at least one query term is much smaller than the number of allattributes values in each entity. The noise in the attribute values that contain query terms dominates the corruption effect. So we can decrease the time spent on corruption if we corrupt only the attribute values that contain query terms.

5.2. SGS-Approx algorithm

Static global stats approximation (SGS-Approx) algorithm are used to corrupt only the top-K result entities, the global DB statistics do not change much.SR Algorithm spends a large amount of the robustness calculation time on the loop that re-ranks the corrupted results (Line 13 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. 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, we can combine the corruption and ranking module together. This way re-ranking is done on-the-fly during corruption.

5.3. Combination of QAO-Approx and SGS-Approx

QAO-Approx and SGS-Approx improve the efficiency of robustness calculation by approximating different parts of the corruption and re-ranking process. Hence, we combine these two algorithms to further improve the efficiency of the query difficulty predication.

6. PERFORMANCE EVALUATION

6.1. QAO-Approx

Fig.1 shows the results of using QAO-Approx on INEX (marked-up version of the Wikipedia corpus). We measure the prediction effectiveness for smaller valuesof N using average correlation score. The QAO-Approx algorithm delivers acceptable correlation scores and the corruption times of about 2 seconds for $N = 10$. Comparing to the results of SR Algorithm for $N = 300$ on INEX, the Pearson's correlation score drops, because less noise is added by second and third level corruption. These results show the importance of these two levels of corruption.

6.2. SGS-Approx

Fig. 2 depicts the results of applying SGS-Approx on INEX. Since re-ranking is done on-the-fly during the corruption, SR-time is reported as corruption time only. As shown in Fig. 2, the efficiency improvement on the INEX dataset is slightly worse than QAO-Approx, but the quality (correlation score) remains high.

6.3. Combination of QAO-Approx and SGS-Approx

We can combine QAO-Approx and SGS-Approx algorithms to achieve better performance. Fig. 3presents the results of the combined algorithm for INEX databases. Since we use SGS-Approx, the SR-time consists only of corruption time. Our results show that the combination of two algorithms works more efficiently than either of them with the same value for N.

6.4. SR algorithm

The average computation time of SR score (SR-time) using SR Algorithm and compare it to the average query processing time (Q-time) using PRMSfor the queries in our query workloads. SR-time mainly consists of two parts: the time spent on corrupting K results and the time to re-rank the K corrupted results. We have reported SR-time using (corruption time + re-rank time) format. We see that SR Algorithm incurs a considerable time overhead on the query processing. This overhead is higher for queries over the INEX dataset, because there are only two entity sets, (person and movie), and all query keywords in the query load occur in both entity sets. Every attribute value in to K entities will be corrupted due to the third level of corruption.

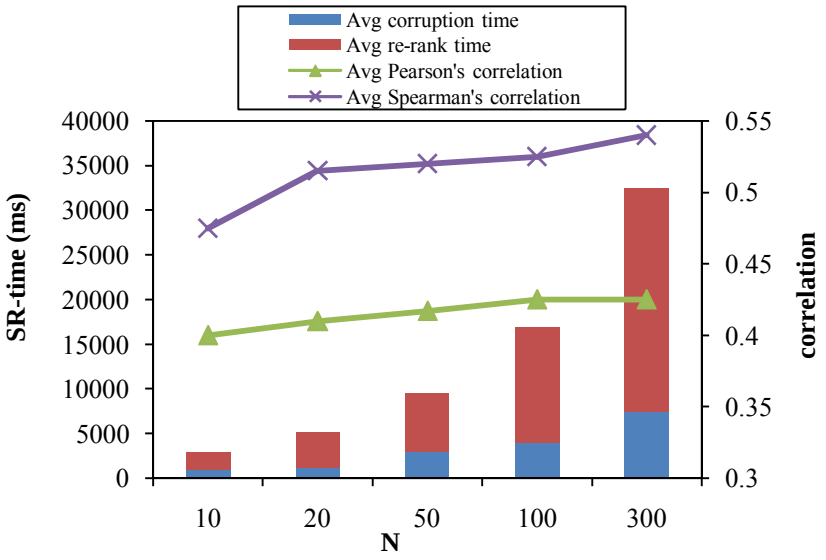

Fig. 1. Approximations in INEX dataset using QAO-Approx algorithm

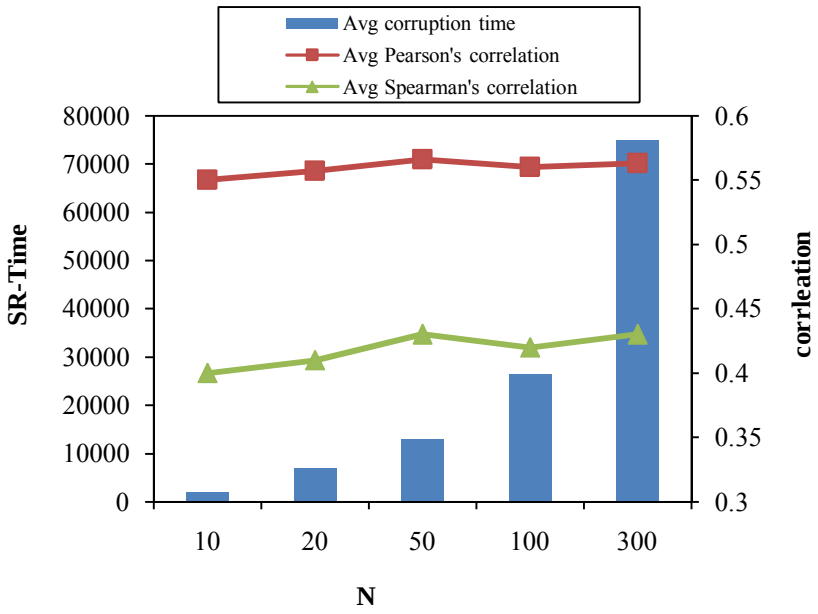

Fig. 2. Approximations in INEX dataset using SGS-Approx algorithm

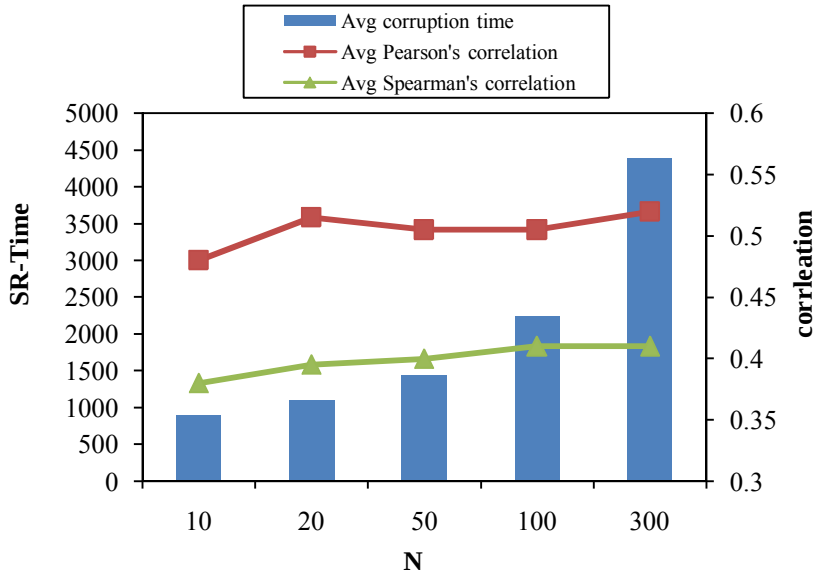

Fig. 3. Approximations in INEX dataset usingCombination of QAO and SGS algorithm

Figure 3 shows the approximations in INEX dataset using combination of QAO and SGS algorithm. According to our performance study of QAO-Approx, SGS-Approx, both datasets, the combined algorithm delivers the best balance of improvement in efficiency and reduction in effectiveness for both datasets. On both datasets, the combined algorithm achieves high prediction accuracy (the Pearson's correlation score about 0.5) with SR-time around 1 second. Using the combined algorithm over INEX when the value of *N* is set to 20, the Pearson's and Spearman's correlation scores are 0.513 and 0.396 respectively and the time decreases to about 1 second. For SR Algorithm on INEX, when *N* decreases to 10, the Pearson's correlation is 0.537, but SR-time is over 9.8 seconds, which is not ideal. Thus, the combined algorithm is the best choice to predict the difficulties of queries both efficiently and effectively.

7. CONCLUSION

In this paper we are using keyword query interface technique for difficult keyword query predicting over databases. And we proposed algorithms to measure the degree of the difficulty over unstructured data using ranking robustness principle. Finally simulation results show that our algorithm efficiently predicts the difficulty of the query with higher accuracy.

8 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] 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.

[3] W. B. Croft, D. Metzler, T. Strohma, Search Engines - Information Retrieval in Practice, Pearson Education, 2009.

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

[5] Y. Zhao, F. Scholer, and Y. Tsegay, "Effective pre-retrieval query performance prediction using similarity and variability evidence," in *Proc. 30th ECIR*, Berlin, Germany, 2008, pp. 52–64.

[6] C. Hauff, L. Azzopardi, D. Hiemstra, and F. Jong, "Query performanceprediction: Evaluation contrasted with effectiveness," in*Proc. 32nd ECIR*, Milton Keynes, U.K., 2010, pp. 204–216.

[7] B. He and I. Ounis.Inferring query performance using pre-retrieval predictors. InSPIRE'04, pages 43–54, 2004.

[8] J. He, M. Larson, and M. de Rijke.Using coherence-based measures to predict query difficulty. In ECIR'08, pages 689–694, 2008.

[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.

[10] A. Shtok, O. Kurland, and D. Carmel, "Predicting query performance by query-drift estimation," in *Proc. 2nd ICTIR*, Heidelberg, Germany, 2009, pp. 305–312.

[11] J. D. Gibbons and S. Chakraborty, *Nonparametric StatisticalInference*. New York, NY: Marcel Dekker, 1992.

[12] S. M. Katz, "Estimation of probabilistic from sparse data for thelanguage model component of a speech recognizer," *IEEE Trans.Signal Process.*, vol. 35, no. 3, pp. 400–401, Mar. 1987.

[13] C. Zhai and J. Lafferty, "A study of smoothing methods for languagemodels applied to ad hoc information retrieval," in *Proc.24th Annu. Int. ACM SIGIR Conf. Research Development Information Retrieval*, New Orleans, LA, USA, 2001, pp. 334–342.

[14] Y. Zhou and W. B. Croft, "Query performance prediction in web search environments," in *Proc. 30th Annu. Int. ACM SIGIR*,New York, NY, USA, 2007, pp. 543–550.

[15] 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.

[16] O. Kurland, A. Shtok, S. Hummel, F. Raiber, D. Carmel, andO. Rom, "Back to the roots: A probabilistic framework for queryperformanceprediction," in *Proc. 21st Int. CIKM*, Maui, HI, USA,2012, pp. 823–832.

[17] O. Kurland, A. Shtok, D. Carmel, and S. Hummel, "A Unifiedframework for post-retrieval query-performance prediction," in*Proc. 3rd Int. ICTIR*, Bertinoro, Italy, 2011, pp. 15–26.

[18] S. Cheng, A. Termehchy, and V. Hristidis, "Predicting the effectiveness of keyword queries on databases," in *Proc. 21st ACMInt. CIKM*, Maui, HI, 2012, pp. 1213-1222.

[19] A. Termehchy, M. Winslett, and Y. Chodpathumwan, "Howschema independent are schema free query interfaces?" in *Proc.IEEE 27th ICDE*, Hannover, Germany, 2011, pp. 649–660.