An Efficient Difficult Keyword Prediction using Clustered Based Model View Similarity Matrix

Palakurthi Sahitya[1], E. Deepthi[2]
Final M.Sc. Student[1], Lecturer[2]
[1, 2] M. Sc Computer Science, Chaitanya Women's PG College, Old Gajuwaka, Visakhapatnam Andhra Pradesh

Abstract:
To the best of our knowledge, there has not been any work on predicting or analysing the difficulties of queries over databases. Researchers have proposed some methods to detect difficult queries over plain 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 Keyword query interface must assign each query term to a schema element in the database. It must also distinguish the desired result type. We empirically show that direct adaptations of these techniques are ineffective for structured data. In this paper we are propose topic based cluster search algorithm for search of keyword in the database. By implementing this technique we can improve more efficiency of query oriented keyword search.

Keywords: *Keyword, Clustering Data Mining, Query Searching, Difficult Keyword.*

I. INTRODUCTION

The classical iterated query processing is easy to manipulate, the disadvantage is it gives low performance on modern CPUs due to lack of locality and frequent instruction mispredictions. Several techniques proposed in the past to improve this situation, but some techniques are frequently outperformed a user forms a query according to his information need and a number of documents are presented to the user by the retrieval system in response to the query. Calculating the process of quality outcome using Query performance prediction of a retrieval system in response to a user's query without any relevance information. Compared to the long survey of developing retrieval models to improve performance in IR, research on predicting query performance is still in its early stage. However, some associations are starting to realize the importance of this problem and a number of new methods have been proposed for prediction recently . However, accurate performance prediction with zero-judgment is not an easy task. The major difficulty of performance prediction comes from the fact that many factors, such as the query, the ranking function and the collection, have an impact on retrieval performance. Each factor affects performance to a different degree and the overall effect is hard to predict accurately. The ability to predict query performance has the potential of a fundamental impact both on the user and the retrieval system.

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. 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 Q1: 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, Q1 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 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-of the- 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 analysing 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. RELATED WORK

Y. Luo, X. Lin, W. Wang, and X. Zhou, 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,, 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, 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

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 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.

III. PROPOSED SYSTEM

The main objective of proposed system is to perform the efficient query search and reduce the time complexity of in the searching process. In this paper we are proposed an efficient query searching process i.e. topic based cluster search algorithm. By implementing this algorithm we can get efficient search result and also reduce time for searching the query. Before performing the search the query we can take sample document and search query in that documents. The implementation procedure of topic based cluster algorithm is as follows.

Text Pre-processing:

In the text pre-processing we can get only text formatted data for searching query. Before performing search operations we can get all documents and reduce all tag in that document. After getting each document text we can find out relative frequency (R_{freq}) of each document. Before finding relative frequency we also find local and global frequency of each word in the document. The local frequency (L_{freq})of each can be calculated by number of occurrence of each word in the document. After finding local frequency of each word in the document we can find out global frequency(G_{freq}) . Using both frequencies we can find out relative frequency of each document by using following formula.

$$R_{freq} = L_{freq} + G_{freq} / 2.0$$

After finding relative frequency we can calculate document weight of each document by using following formula.

N=size of each document

$L_{freq=}$ Local frequency of each word in the document

G_{freq} =Global Freqency of each document

$$\text{Weight (W)} = L_{freq} * \text{Math.Log}(N/G_{freq}) + 0.01$$

By using that formula we can calculate each document weight. After we can create MVS Matrix of each document to other documents.

Build MVS Matrix:

In the generation of MVS matrix we can calculate cosine similarity each document to other document. Based on MVS matrix we can perform the clusterization of documents. The cosine similarity of any two document can be find by using following equation.

d1= Total number of words in first document
d2= total number of words in second document
d_{prd}= d1*d2
$d1_{sqr}$ = d1*d1
$d2_{sqr}$ d2*d2

$$d_{sqrprd} = d1_{sqr} * d2_{sqr}$$

$$sim = d_{prd}/d_{sqrprd}$$

By using those formulas we find out each document cosine similarity and also we generate matrix formatted data. likewise we can calculate cosine similarity of each document to other document and arranged in the form matrix.

k means clustering algorithm for grouping related documents:

By calculating of MVS matrix we can perform the clusterization process. By performing clusterization process we can grouping all relating document into single group. Before performing clusterization we get all cosine similarity of each document to other document. Based on cosine similarity of each document we can perform clusterization process. The step of clusterization process is as follows.

1. Enter the number of cluster for performing clustering of document.
2. After that finding number of documents are available in the database.

3. Randomly choose the centroid of document based on number of clusters we want.
4. After finding centroid document we can get cosine similarity of each centroid document.
5. After that we can also get remaining document of cosine similarity.
6. Find out distance of each centroid to other document based on cosine similarity by using following formula

```
for (int i=0;i<docs.size();i++)
    {

     int minInd =0;
     double mindis=0;

     for(int j=0;j<k;j++)
     {
      double dis =
cosSim(docs.get(i),getCentriod(clusturs[j]));
       if(j==0 || mindis>dis)
       {
        minInd=j;
        mindis=dis;
       }
      }

     clusturs[minInd].add(docs.get(i));

    }.
```

By using that code we can find out related documents in a group. After grouping all related document into group perform the searching operation in those groups and get only query matched document.

Topic based searching process:

In this module we perform the searching operation of query in the document. In this we can get each cluster document and convert into text format. After that we can search each word in that cluster and find out the word id existing in that group or not. So that if the word is existing in the that cluster we display that document in the cluster. Likewise we can search all cluster document and get only the query related cluster document. By implementing those concepts we can get more effective search result and also time complexity for performing search operation.

IV. CONCLUSIONS

In this paper we are proposed a novel problem for performing effective searching operation in documents. By implementing this concept we can improve more efficiency of searching operation and also reduce time complexity. In this paper we are proposed topic based cluster searching algorithm for finding related document of query search. In this algorithm we can find out each document cosine similarity and also find out distance of centroid document to other documents. After finding distance we can perform clusterization process by using k means clustering algorithm. After performing clustering process we can perform the searching process for query. By performing query search we can get all query related documents of clusters can be display. By implementing those concepts we can improve efficiency in the searching operation.

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] J. Kim, X. Xue, and B. Croft, "A probabilistic retrieval model for semi structured 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] J. A. Aslam and V. Pavlu, "Query hardness estimation using Jensen-Shannon divergence among multiple scoring functions," in *Proc. 29th ECIR*, Rome, Italy, 2007,. 198–209.

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

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

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

[10] E. Demidova, P. Fankhauser, X. Zhou, and W. Nejdl, "DivQ: Diversification for keyword search over structured databases," in *Proc. SIGIR' 10*, Geneva, Switzerland, pp. 331–338.

[11] 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.

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

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

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

[15] 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.