COMPLEX KEYWORD QUERİES SEARCH AND PREDİCTİON OVER DATABASES

Nikita D Pattanshetti[1], Prof. Savita Bakare[2]

[1]PG Scholar, [2]Assistant Professor, Dept of CSE,

KLE Dr. M S Sheshgiri College of Engineering and Technology (KLECET),

Belagavi, Karnataka, (India)

ABSTRACT

Users query in search engines using keywords and it's a widely used form of querying. In response to a user need, or query, the task of an Information Retrieval system is to retrieve useful information from a large repository of data. Search engines usually do keyword matching only. As long as the query word is present in the document, it is fetched and presented to the user as the result of the query. If a user query is general and ambiguous, the query engine finds the query as a complex query as it cannot extract the exact result which the user expects. Search engines do not identify the needs behind the query hence most of the relevant documents are not retrieved. The difficulty of this task will be affected by various factors, relating to the system or algorithms used, to the properties of the data to be retrieved, or to the inherent difficulty of the user's information need. The effect of these factors upon retrieval performance is often referred to as query difficulty or complexity and the query is said to be a complex query. The aim of this paper is to analyze and study about the efficient query retrieval and prediction over the databases. In this approach, the Structured Robustness (SR) score algorithm is implemented to predict the effectiveness of the keyword queries over databases. Query reformulation method is used where the search engine helps the user to reformulate their query by adding more specific keywords to it. The expected outcome effectively predicts the hard queries on structured data over databases. The ranking quality of the results provides a good user satisfaction..

Keywords: Keyword Query, Keyword Query Interfaces, Query Performance, Structured Robustness (SR) Score.

I. INTRODUCTİON

Data mining is the process of extracting useful information from large volumes of data. It is also known as Knowledge Discovery in Databases (KDD) and it is used for data extraction from various databases such as relational databases, object-oriented databases, data warehouses, transactional databases etc. With the help of data mining tools we can predict behavior and future trends and make knowledge-driven decisions.

With the growth of the internet and enormous data, it is difficult for the user to get relevant documents. Information Retrieval (IR) system retrieves information from a large repository of data in response to a user query. Given a set of documents in the database and a query given by user, subset of documents relevant to the query is to be retrieved in any Information Retrieval system. In Relational databases, the data is commonly searched using Structured Query Language (SQL). Information needed to answer a keyword query is often split across the tables/tuples. If the user knows the schema of the database he can form suitable query for his needs.

In case of online databases, user usually does not have detailed knowledge of schema or query languages. Hence the desired results are not obtained.

Keyword Query Interfaces (KQI) is the type of computer-human interface used for selecting the required data. Keyword queries on databases are used to provide easy access for the data to be searched. But, these data have low ranking quality in real world scenario [1]. In response to a user's query the Search engines usually do keyword matching and return ranked list of all the documents containing the keywords specified in the query. The relevant documents may not be retrieved and/or retrieved instances may not be relevant (i.e. low precision and/or recall).

Keyword search provides an alternative and easy way of querying in relational databases. One important advantage of keyword search is users need not have prior knowledge about the structures of the underlying data or the knowledge of complex structured query languages (e.g., SQL) for querying their information needs. In the event that the query is general, it is hard to recognize the specific record on which the user is interested. The users are required to filter through a not insignificant rundown of off-subject reports. The queries which are hard to answer effectively by the Information Retrieval (IR) system are called hard queries or complex queries. Thus, the query engine should identify and search the desired attributes associated with each term given in the query. Thus, it is important to distinguish such queries that are liable to have low positioning quality with a specific end goal to enhance the user fulfillment level.

The topic of how to find data of interest of user in the World Wide Web is raised by the Web Search Problem [2]. Majority of queries requested to the Internet search engines by the users are general, few words in length. Low quality searches are many and therefore methods of dealing with the results of such queries are needed. One method is filtering of the ranked list of documents, varying from simple pruning techniques to advanced Artificial Intelligence algorithms. Although they limit the total length of the ranked list, it is difficult for users to locate the specific documents they searched for. The search engines can also help the users to refine their query by adding more specific keywords to it.

Query engine must assign each query words to schema elements in the database. In this regard, this paper we study the properties of complex queries over databases and proposes a method to detect such queries. The structure of the data in the given database is used to study about the degree of the complexity of the query. The method used predicts the degree of the complexity of a query efficiently. Structured Robustness (SR) score [1], measures the difficulty of a query based on the differences between the rankings of the same query over the original and noisy (corrupted) versions of the same database. Finally, thresholding approach is used to define query difficulty metric.

The rest of this paper is organized as follows. Section 2 describes related work. In section 3, we study the different prediction methods. Section 4, defines the properties of complex queries. In section 5, we describe the data models used. In section 6, we explain about the main methodology of our work. In section 7, we briefly explain about the result obtained. In Section 8, we summarize the main conclusions of this paper.

II. RELATED WORK

Predicting the query performance is important for an Information retrieval system. It is studied under different names like query difficulty, query complexity, query ambiguity and sometimes hard query. Existing work on

query complexity estimation can be categorized along three axes [3]. How is query complexity defined, How is query complexity predicted and How is the quality of the prediction evaluated.

2.1 Defining Query Complexity

We can define query hardness or complexity in many ways; for example, queries can be inherently complex or can be ambiguous, difficult for a particular collection of data [3]. A query can be complex in a given collection of data if it has more outcomes for single query. For example, Carmel et al. [4] considered collection query hardness by comparing the query difficulty predicted by their method to the median average precision taken over all runs submitted in the Terabyte tracks at TREC (Text REtrieval Conference) for a given query.

2.2 Predicting Query Complexity

Cronen-Townsend, Y. Zhou, and B. Croft in [5] introduced the clarity score which effectively measures the ambiguity of the query with respect to a collection. Clarity scores are computed by assessing the information theoretic distance between a language model associated with the query and a language model associated with the collection. They showed that clarity score positively correlates with average precision.

2.3 Evaluating the Quality of Query Complexity Predictions

In order to evaluate the quality of a query hardness prediction methodology, the collection hardness of a set of queries is measured and they are compared to predicted values of query hardness. These actual and predicted values are real-valued, and they are typically compared using various parametric and nonparametric statistics.

S. C. Townsend, Y. Zhou, and B. Croft in [5] use clarity score method to predict the query outcome by computing a measure of disorder between a query model and its collection model. The subsequent clarity score measures the coherence of models that are likely to generate the query. A threshold for clarity score is set between predicted queries and acceptable queries and it is validated using TREC (Text REtrieval Conference) data. They showed that clarity score measures the ambiguity of a query and it is positively correlated with average precision in a variety of TREC test sets.

The clarity score method generally assumes that the lengthier is the query, the easier it will be to evaluate. To extend idea of clarity score for queries over databases, domain knowledge about the data sets is required. Empirical studies show that these methods have limited prediction accuracies [5].

Y. Zhou and B. Croft in [6] introduced the notion of ranking robustness. It refers to a property of a ranked list of documents. It indicates how strong the ranking is in the presence of noise in the ranked documents. Robustness score, a statistical measure is used to quantify this notion. Given a query and a ranking function the ranking robustness is calculated by comparing the ranking from the corrupted dataset and ranking from original dataset. They showed that robustness score correlates with query outcomes in a variety of TREC data collections. Query difficulty prediction model for our work falls under this category.

B. He and I. Ounis, in [7] Use a set of predictors which are computed before the retrieval process takes place. IR system scans the files for the query terms and assigns a relevance score to each retrieved document. Some of the predictors are query length, which is number of words in the query, query clarity which defines the unambiguous property of query. The query scope was originally studied in [8] which defines how general or specific is a query. Some of the predictors have significant correlation with query performance. Therefore, these predictors can be applied in practical applications.

Since the predictors are generated and studied before the retrieval of results takes place, therefore domain knowledge about the data sets used is required. It requires finding a similarity function between entities that are about a similar topic. The domain knowledge and understanding users' preferences is required to understand the similarity function.

Some use IDF-related (inverse document frequency) features as predictors. He and Ounis in [9] proposed a predictor based on the IDF of the query terms. IDF-based indicators demonstrated some moderate relationship with query outcome. They estimated the clarity score model by the term frequency in the query. They also used the notion of the query scope, which is quantified as the percentage of entities that contain at least one query keyword in the collection of documents. These predictors usually do not take the retrieval algorithms into account and thus are unlikely to predict query performance well. These kinds of predictors rely on the amount and characteristics of available training.

G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, in [10] used a system which enables keyword search on relational databases considering both data and schema browsing called BANKS. It is an acronym for Browsing ANd Keyword Searching. It uses backward expanding search algorithm for finding and ranking query results. Using BANKS users can extract data without the need for writing complex queries and without any knowledge of the schema. Database is viewed as a graph where the nodes are the database tuples and the edges are application specific relationships. A user gets his information by typing a few query keywords, following hyperlinks provided, and interacts with the displayed results.

A drawback of these approaches is that a graph of the database tuples must be materialized and maintained. Once the data graph has been built, the structural information provided by the database schema is ignored. This method can be relatively slow, since a large number of tuples may be defined to be relevant to the keyword.

III. PREDICTION METHODS

Performance Prediction of complex queries is important for many reasons. From the user perspective, it gives feedback to the user to direct his search. From the perspective of retrieval system, it is important to improve the consistency and use alternative retrieval strategies to improve the performance of retrieval systems. The methods to predict complex queries can be broadly categorize into two groups: pre-retrieval and post-retrieval methods [1].

1. Pre-retrieval methods: These methods are used to predict the difficulty of a query without considering its outcomes. The properties of the terms in the query are used to measure the specificity or ambiguity of the query to predict its difficulty [11]. These methods assume that the more are the query terms, the easier the query will be. Actual studies indicate that these methods have limited prediction accuracies [5].
2. Post-retrieval methods: After query computation these methods uses the outcomes of a query to predict its difficulty and generally falls into one of the following categories.

- Clarity-score based methods consider few topics of the document which are frequently searched by user [7]. It deals with most searched topic of the document and retrieves the result from it. The degree of difficulty for query is less if result of user query is found on most searched documents otherwise high.
- Ranking score based methods defines the ranking score of the document returned by retrieval system for a user query. It estimates the similarity between a query and the document and defines the difficulty of a

query by the difference between weighted entropy score of top ranked result and the score of other documents.

- Robustness-based methods defines how robust is the query over specific document. This method defines the degree of difficulty of query by considering the robustness of the ranking over two versions of data, original version and corrupted version, and compares the top k results of same query over these two versions [12]. Degree of difference between these results defines the degree of hardness of query.

Some methods use machine learning techniques to study complex queries and its properties and then predict their hardness [13]. These methods are effective, if large amount and quality of data are available which are normally not available for many databases.

IV. PROPERTIES OF COMPLEX QUERIES ON DATABASES

The keyword queries which are difficult to answer correctly by the Information retrieval systems are called complex keyword queries or hard queries. There are basically three sources of difficulty for answering a query over a database [1]. They are as follows:

4.1 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: Consider user submits query Q1: JOEL in IMDB (Internet Movie Database) database. The keyword query interface must resolve desire "JOEL" that satisfies user's information need. If there is more than one person named JOEL 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 relevant answer.

4.2 Attribute Level Ambiguity

Each attribute explains a feature of an entity. If a query terms matches different attributes in its database then it will have a more set of answers 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: FURIOUS does not match with any instance of attribute distributor, so keyword query interface easily predict the relevant answer for this query.

4.3 Entity Level Ambiguity

Each entity set contains the data about different entities. Hence, if a query matches entities from two or more entity sets, it will have higher ambiguity at entity level. For example: IMDB has two entity sets mainly one is 'Movie', which contains the information about movie and other is 'Person', which contains the information about people who are involved in making movies. Consider the query Q5: 'MARRIED', the both Movie and Person contains MARRIED word, so the query becomes complex to find relevant answer as keyword query interface does not know whether user is interested in people who are MARRIED or the movies containing MARRIED word. In case of query Q6: COMEDY SUSPENSE 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 Q5.

V. DATA MODELS

Database is modeled as single entity set called 'Movie'. Each entity has a set of attributes that describes the entity. Each attribute contains attribute values. The database fragment is as shown in Fig 1. For instance, title and director are the attributes and 'Heartless' and 'Aditya chopra' are its attribute values respectively. The entities, which are the movie titles, are stored using set of normalized relational tables.

The model below is widely used in works on entity search [14]. It has an advantage that it can be easily mapped to both XML and relational data.

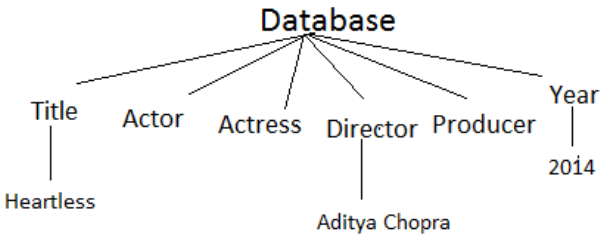

Fig 1: Database fragment

Keyword query Q is a set of keywords given by a user. An entity E from the database is the answer to Q if and only if at least one of the attribute values contains those query words. Given a database DB and a query Q, the retrieval function g(E, Q, DB) returns the relevance of entity E in DB to query Q. A keyword search system returns a ranked list of matched entities in DB called L(Q, g, DB) where entities E are placed in decreasing order of the value of g(E, Q, DB).

VI. METHODOLOGY

E.Mittendorf in [15] has shown that if a retrieval system effectively responses and ranks the results to a user query in a collection of documents, it will likewise perform well for the same query over the corrupted version that contains some errors such as repeated words. In other words, the degree of the complexity of a query is positively correlated with how robust is its ranking over the original and the corrupted versions of the data. This observation is called the Ranking Robustness Principle [1]. It defines how robust is the ranking in presence of noise.

In this paper the database is corrupted by adding repeated attributes. First we define data corruption model for structured data. For that we model a database DB using entities, attributes, and attribute values. The noisy or corrupted version of the database is created by adding repeated attribute values for entities. Given a user query Q and a information retrieval function g, we rank the candidate answers in DB to get the ranked list L. The more results in L the more complex are the query.

To categorize input queries thresholding approach is used. A query is said to be an as "easy" or "complex" query based on certain threshold [5]. This thresholding approach defines a reasonable threshold "t" for a query difficulty metric. If the difficulty metric of the query is above t, it will be considered as a complex query, and the KQI will apply further treatments like the user is asked to provide some description about the query with one or more keywords.

For a easy query with few matched entities, the retrieval system easily fetches the records from the database. For a lengthy and complex keyword query it is hard to get the accurate result over database as the retrieval system cannot identify the desired attributes for the user query. Hence multi-stage processing is required to get the proper answer for the user query.

First, Preprocessing of the data involves removal of stop words and stemming. Stopwords are normally excluded from the set of query terms. It is important to extract only the keywords from the query. Natural candidates for stopwords are articles, prepositions and pronouns. Porter stemmer is a stemming algorithm that is based on the context aware [16]. It is a process for removing the commoner morphological endings from words in English. Its main use is in normalization process which is important when using Information Retrieval systems. For instance "Movies' in a query is preprocessed as 'Movie'. It is used to increase recall without decreasing precision.

After preprocessing the important words of the query are extracted and matched against the database for accurate result. Clustering is a process of forming clusters of similar objects from a given set of inputs. In case of web search results, clustering is a way of organizing the results into a number of easily browse-able groups. For clustering of results obtained from search engine we use basic concepts of algorithm called LINGO [17]. The key idea is to first discover cluster labels and determine the data of the groups based on the labels. Two clusters labels formed are a cluster for an easy query with one or two results and another cluster for complex query for which more than two results are fetched.

Structured Robustness (SR) score is used to measure the difficulty of a query [1]. Structured Robustness Algorithm computes the exact SR score based on the results for a query. Some statistics that can be used are the number of query terms matching all attributes values of the database or number of results fetched for a query.

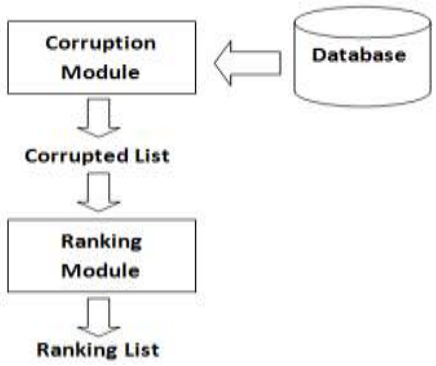

Fig 2: Structured Robustness Algorithm

The Structured Robustness algorithm is as shown in Fig 2. SR Algorithm checks for all attribute values in the results for all query terms. The corruption module checks for all the keywords in the database. A corrupted list with all possible keywords matched in database is obtained. With the user description about the query the most appropriate result is fetched and presented to the user. The ranking module ranks the results with most appropriate result at the top of the list.

We use the XML ranking method used in [18], called PRMS- Probabilistic Retrieval Model for Semistructured Data. Given a query Q, PRMS computes the relevance score of entity E based on the weighted linear combination of the relevance scores. It assigns each attribute a specific weight, which specifies its contribution in the ranking score. The algorithm for the same is given below.

Algorithm:

Input: Documents to be ranked.

Output: Ranked list of documents.

For (i=1 to n) Do

Search the documents and compute the results.

Calculate and cache them.

For (j= 1 to n)

Calculate the rank for the given query input.

Display the ranked list of documents.

For a given query, the documents fetched by the retrieval system are ranked according to the user preferences. We calculate rank for the given query input by considering the number of clicks for a document. Document with highest clicks is fetched and presented at the top of the ranking.

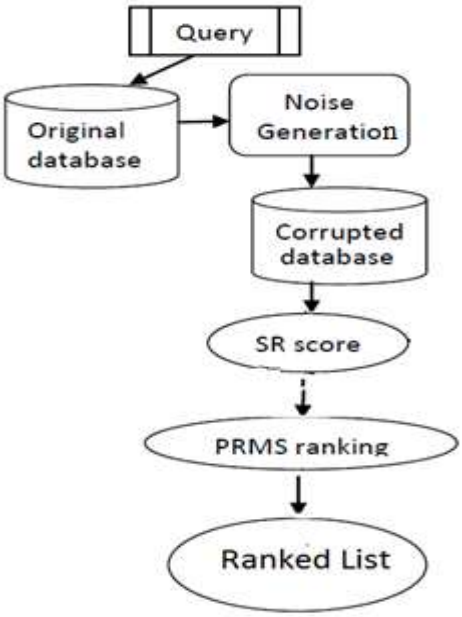

Fig 3: Work Flow Architecture

The workflow architecture of the whole idea is as shown in Fig 3. User enters a query according to his information needs. The database is corrupted by generating noise in the database. Noise generation, as mentioned earlier, is done by adding repeated attribute values. The user query is now searched in corrupted database. The keywords in the query are matched against every attribute in the database and appropriate results are fetched. The SR score obtained defines the complexity of the query. Finally using PRMS ranking a ranked list of documents is provided to the user as the search result with most searched entity is listed first. This ranking provides a good user satisfaction as the most appropriate result is present at the top of the list.

VII. RESULTS

The database used for this work contains 500 records. When a user issues a query, which is not ambiguous, relevant records were fetched by the retrieval system. For a complex query, having difficulty metric above the threshold, the user is asked to provide description about the query with one or more keywords. Based on the keywords provided by the user relevant records were fetched. Numbers of records fetched were less for easy query, where as for complex query which is ambiguous, three or more records fetched with the most relevant

record at the top of the list. The ranking quality of the results provided a good user satisfaction. The algorithm predicts the complexity of queries with fewer errors and in negligible time.

VIII. CONCLUSION AND FUTURE WORK

This paper focuses on main problem of retrieving appropriate top results for a keyword query and predicting the difficulty level of the query. In this paper, we analyze the properties of complex queries and measure the degree of complexity of a keyword query over a database. SR algorithm is used for difficult keyword queries prediction over databases. We measured the degree of the complexity of a query over a database, using the ranking robustness principle. The framework efficiently predicts the effectiveness of a keyword query.

The future work can be extending this framework to estimate query difficulty by using different entity sets and also on other ranking problems on databases. It can be extended for semi-structured keyword queries with operators and supporting phrases provided by the user.

REFERENCES

[1] 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.

[2] Eric W. Selberg. "Towards Comprehensive Web Search" Doctoral Dissertation, University of Washington, 1999. IEEE Transactions on Knowledge and Data Engineering, August 2013

[3] J. A. Aslam and V. Pavlu, "Query Hardness Estimation using Jensen-Shannon Divergence among Multiple Scoring Functions," in ECIR, 2007.

[4] Carmel, D., Yom-Tov, E., Darlow, A., Pelleg, D.: What makes a query difficult? In: SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM Press (2006) 390–397

[5] S. C. Townsend, Y. Zhou, and B. Croft, "Predicting Query Performance," in SIGIR, 2002

[6] Y. Zhou and B. Croft, "Ranking Robustness: A Novel Framework to Predict Query Performance," in CIKM, 2006.

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

[8] V. Plachouras, I. Ounis, G. Amati, and C. J. van Rijsbergen. University of Glasgow at the Web Track: Dynamic application of hyperlink analysis using the query scope. In Proceedings of the Twelth Text Retrieval Conference (TREC 2003), pages 248 - 254, Gaithersburg, MD, 2003.

[9] B.He and I.Ounis. "Inferring query performance using preretrieval predictors", In proceedings of the SPIRE 2004. Pp 43-54, 2004

[10] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword Searching and Browsing in databases using BANKS," in ICDE, 2002

[11] C. Hauff, L. Azzopardi, and D. Hiemstra, "The Combination and Evaluation of Query Performance Prediction Methods," in Advances in Information Retrieval, 2009

[12] E. Demidova, P. Fankhauser, X. Zhou, and W. Nejdl, "DivQ: Diversification for Keyword Search over Structured Databases," in SIGIR, 2010.

[13] 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 SIGIR, 2005

[14] N. Sarkas, S. Paparizos, and P. Tsaparas, "Structured Annotations of Web Queries," in SIGMOD, 2010

[15] E. Mittendorf and P. Schauble, "Measuring the Effects of Data Corruption on Information Retrieval," in SDAIR, 1996

[16] Daniel Waegel "The Porter Stemmer"

[17] Stanislaw Osinski "An Algorithm for Clustering Of Web Search Results"

[18] J. Kim, X. Xue, and B. Croft, "A Probabilistic Retrieval Model for Semistructured Data," in ECIR, 2009.