IJETIE International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue I, December 2014
ISSN: 2394 - 6598

QUERY RECOMMENDATION SYSTEM USING USERS QUERYING BEHAVIOR

V.Megha
Dept of Computer science and Engineering
College Of Engineering
Guindy, India
meghavenkat25@yahoo.com

Abstract — **In the current technology world, data growth is irresistible and managing large volumes of data becoming more challenging day by day. Developing a better search framework is equally as challenging as managing huge volume of data. In the current data search system, query optimizing is more indeed to fetch the right data from vast database using the right syntax. Hence, we developed the QRSUQB (Query Recommendation System by Using Users Querying behavior) which aids the users by making their search simple, fast and effective. QRSUQB continuously analysis the user's search query pattern and provide better options to a user, which user may find most relevant to his need. QRSUQB also use Query fragmentation and Query Ranking techniques, which will compare the user's current query behavior with previous query behaviors and provide the most relevant and ranked option to the user. And also we propose a better framework which identifies right syntax of the user'query.This brings the significant improvement in the user's search.**

***Index Terms* - Data mining, Recommendation, Personalization, Querying behavior.**

1. INTRODUCTION

The main innovation in modern human society certainly is the presence and influence of Information Technologies. Everybody is somehow directed to use computers and different technological devices for either personal, academic, professional and other needs. Wide spread of Internet has opened totally new horizons. People are able to work from home, to communicate real time with others on the different sites in the world, to do their shopping online, to search for needed information on the web and many others. Social networks are used by people of all age, so boundaries in that aspect do not exist. These, extremely numerous, groups of Information Technology users, generate very large amounts of data.

Data are stored in database management systems (DBMSs). Data belong to various types and accordingly, database users that analyze them are different. Scientists manage and analyze large volume of experimental data. Economists and financial experts process financial data. Managers deal with data about sales of their products and efficiency of their employees. There are many more examples of diverse users that work over, not necessarily, distinct data. However, all of them tend to achieve the same goal of obtaining valuable and useful information out of large data volumes [5].

Current database systems does have vast infrastructure which manages large volume of data and provides access to multiple users to analyze and explore them in various applications. Data on these applications are fast growing and users accessing these large volumes of data also increasing day by day. One among of few examples is "GOOGLE" search engine, where new data are getting populated day by day.

Despite the availability of these large volumes of data, many users often are having difficulties in understanding the underlying database schema and formulating the queries in order to fetch them the right data in an easier way since users may not be familiar with the database schema or might not have the expertise in formulating the correct queries to retrieve the information. For instance, survey conducted by "xxx" org states that, xy % of users who are using the "search engines" facing difficulties in giving the right query to fetch the data[1].

.

To address the above said problems, we designed QRSUQB (Query Recommendation System) for assisting users in the exploration of a large database with right queries to fetch the information of their need. QRSUQB assist users with personalized query recommendations by providing them options based on comparing their querying behavior with previously saved querying behaviors of multiple users. For example, if user A and user B have posed similar queries, then other queries posed by user B may be interested to user A and vice versa. In other ways, we can say that we help the user A by recommending the similar queries posed by user B during the exploration of the database.

2.RELATED WORK

QRSUQB architecture allows the user to share and use resources available efficiently without any problem between them. Several researchers have been done some works related to QRSUQB architecture they are as follows:

Khoussainova N et al. [5] proposed Snip Suggest approach which enables user when composing the query to select clause and ask for recommendations for that clause at any time. Snip Suggests goal is to recommend k features that are most likely to appear in that clause in the users intended query. System views the space of queries as a directed acyclic graph (DAG). It models each query as a set of features and every possible set of features becomes a vertex in the DAG. When asked for recommendation, Snip Suggest transforms the user’s partially written query into a set of features, which maps onto a node in the DAG. Each edge in the DAG represents the addition of a feature so the recommendation problem translates to that of ranking the outgoing edges for the vertex that corresponds to the user’s partially written query. All queries in the Query Repository that are descendants of the current vertex in the DAG are referred as the potential goals for the partially written query.

Giacometti G et al. [3] in this paper reviewed about a framework to assist non-expert users by providing personalized query recommendations. The querying behavior of the active user is represented by a set of query fragments, which are then used to identify similar query fragments in the recorded sessions of other users. The identified fragments are then transformed to interesting queries that are recommended to the active user. An experimental evaluation using real user traces shows that the generated recommendations can achieve high accuracy.

Stefanidis K et al. [8] proposed an YMAL results. This framework works with traditional select-project-join (SPJ) queries, over a database system for a set of users. Importance of a query for the user is measured by the number of times the user posed that same query. This Approaches is to generate YMAL results. Current-state approaches, which exploit the content and schema of the current query result and database instance where as History-based approaches, which exploit the history of previously submitted queries to the database system, e.g. by using query logs. Here the utility of a query for the user is equal to the number of times the user has posed that query. Recommendations generated can be either query-based YMAL results (similar to content-based recommendations), either user-based YMAL results (similar to collaborative recommendations).

Chatzopoulou et al. [1],[2] the authors present conceptual framework and its instantiation for personalized query recommendations for interactive database exploration. When user explores the database during one session, he/she is usually searching for some specific information. Due to that relation the queries of one session are usually correlated. This observation represents a possible basis for generating recommendation for the next queries. These query recommendations can be suggested to all users that have similar querying interests. To generate these personalized query recommendations for current user this framework relies on both queries of past users and queries of current user so far. Proposed instantiation of framework is based on collaborative filtering.

.

3.PROPOSED ARCHITECTURE FOR QUERY RECOMMENDATION

The major problem in recommendation engine is that improper query log filtering occurs when two users search for similar queries or identical queries. Query log is not properly updated with data incrementally for query recommendation. Query Filtering ensures the timely update of query log. Keyword-based query interface suffers from the “empty-answer” and “too-many-answers” problems and also there is a Lack of good recommendation. More accurate recommendations are provided by the use of Query Fragmentation after ranking parameters needed for location computation can cause inaccuracies in the estimated location.

Fig 1 represents the overall system architecture for QRSUQB. Initially the process starts with the user logs in the system the user can submit a SQL query to database query interface database scheme which is present in the database query interface in order to frame the SQL query most efficiently. SQL query is sent to syntaxchecker to check SQL syntax then the SQL query is fragmented and then transmitted to both the dbms and recommendation engine. The dbms requested to the database and presents the user with results and then query is fragmented and rated , creating an implicit query profile which consits of fragments and its corresponding rank .recommendation engine combine the

Fragment Name	Start Keyword	End Keyword
Attribute String	SELECT	FROM
Relational String	FROM	WHERE, GROUP BY, ORDERBY, END OF QUERY
Where String	WHERE	GROUP BY, HAVING, END OF QUERY
Group By	GROUP BY	ORDER BY, HAVING END OF QUERY
Having string	HAVING	ORDER BY, END OF QUERY

current user query and query profile and generate the set of top 'n' queries

3.1Query Fragmentation

The SQL syntax checker checks if the given input query is in SQL syntax. It also checks if the given fields match with that of the tables in the database and verifies the attributes in the database table. And then The input SQL query is fed to the Query Fragmentation algorithm. The given query is split

into fragments with respect to the keywords (select, from, where, group by, having, order by). The fragmentation process

Table 1. Parsing Keywords

is shown in table 1. The fragmented query attributes are stored in the fragment table with respect to the fragment name.

3.2Query Filter

The Query Limit is set . The active user's query is inserted into the Query log as input. The query is fragmented using the Fragmentation algorithm. The query is compared with the already recorded fragments in query log . If the queries match, the Query rating is incremented by 1. If the queries don't match, the new fragments are updated in the query log. This is done till the number of entries in the query log is within the query limit. If the number of entries is to exceed the query limit, the Query Log is full. The query is then removed from the Query log accounting to an LRU Policy.

3.3Query Recommendation Engine

The query recommendation engine gives a set of recommended queries for the given input SQL query . The input query is first fragmented and the fragmented query is stored in a table t. The Query profile from the query rating contains fragments id and rank of the queries. The fragmented query is compared with the queries in the query profile

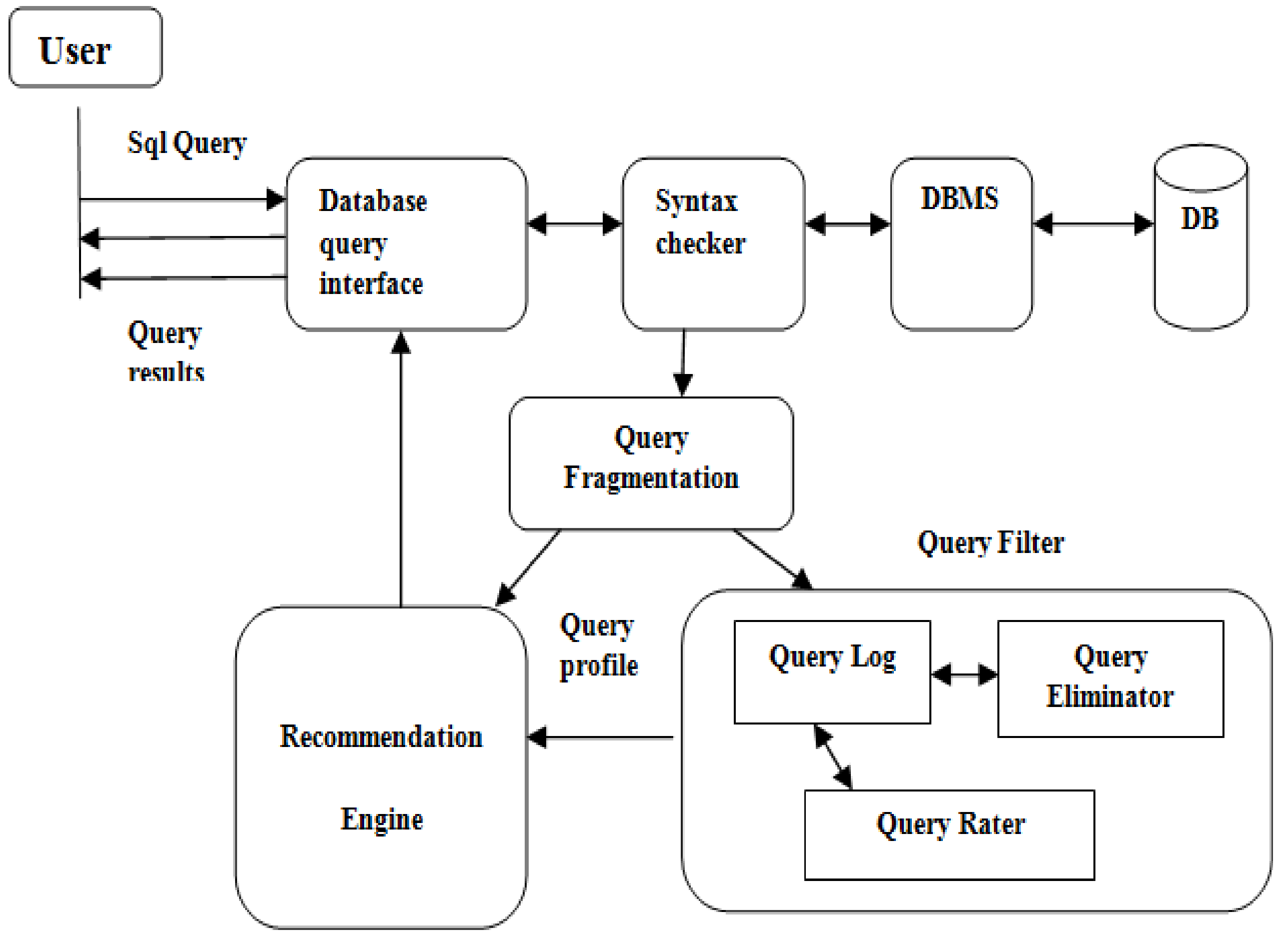

Figure 1 QRSUQB architecture

If the fragmented query matches with any of the queries in the Query profile, the rank of the queries in the query profile is checked. The top n rank queries are returned. If it does not match with any of the queries in the Query profile, the result of the input query is returned.

4.PERFORMANCE EVALUATION

In order to evaluate the various parameters of **QRSUQB,** we used the holdout set methodology the data is divided into two disjoint sets, the training set and the test set. The pair-wise fragment similarity is computed against the training set. Each user session in the test set is divided in two parts. One part is treated as the active user's queries, while the second part is treated as unseen (i.e. future) queries. Subsequently, using the active user's queries from the test set and the pre-calculated fragment-based similarities, **QRSUQB** generates a set of query recommendations.

We compare the recommended queries with the unseen queries from the test set and calculate the precision, recall and F-score for each session as shown in Equations below. We keep the precision of the recommendation that had the maximum recall value, assuming that the end user will also select only one recommended query each time.

4.1Data set description

The data set used is a movie data set, which contains wide collection of actors and actress list along with their personal details and life time achievements. When a query is placed by the user, all possible combination of information regarding to query is retrieved from the data set.

In the formula given below 1,.2 where *Fr* and *Fu* represent the fragments of the recommended and unseen queries respectively.

$$Precision = \frac{|F_r \cap F_u|}{F_r} \quad (1)$$

$$Recall = \frac{|F_u \cap F_r|}{F_u} \quad (2)$$

$$F - Score = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

.

Fig. 4.1 Average precision for various top-*k* values.

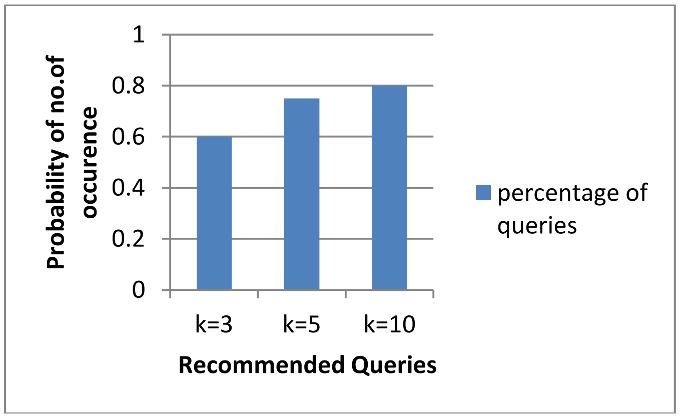

Figure 4.1 The recommended queries vs probability of no of occurrence of the query graph. Here X-Axis denoted the number of recommended queries and the Y-Axis denotes the number of times query occurred. The probability of recommended queries increases linearly.

Fig. 4.2 Average f-score for various top-*k* values

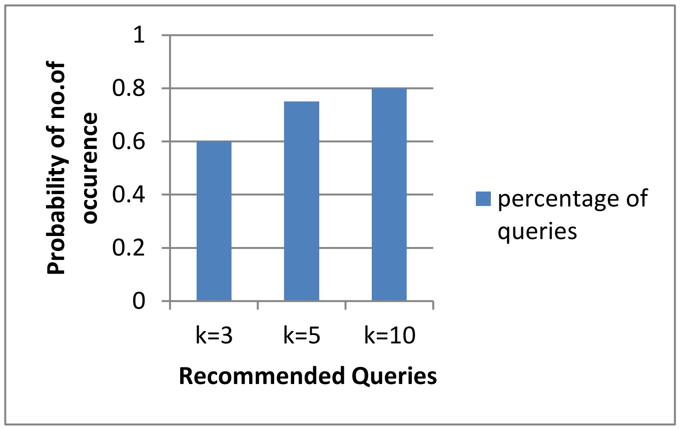

Figure 4.2 The recommended queries vs probability of no of occurrence of the query. Here X-Axis denoted the number of recommended queries and the Y-Axis denotes the number of times query occurred. The probability of recommended queries increases linearly

Figures 4.1 and 4.2 shows the effect of value of k on the average precision and F-score for the recommendations. Where k represents the no of recommended queries . If we assign k=3 the accuracy of the recommendations increases because the rate of precision is high while recall is less. as k increases the recall also increase hence the system gives better recommendation. However, for very large values of *k* (*k* > 10), the accuracy starts decreasing again.

This is completely justifiable, since when *k* is a very large number, the notion of “most similar” fragments does no longer hold and barely similar items are included in the recommendation process. QRSUQB achieves higher precisions for $k \in [5, 10]$ (0.75 and 0.8 for the two endpoints), whereas F-score is the same for both end points (0.75 and 0.76 respectively).

Fig 4.3 Accuracy chart

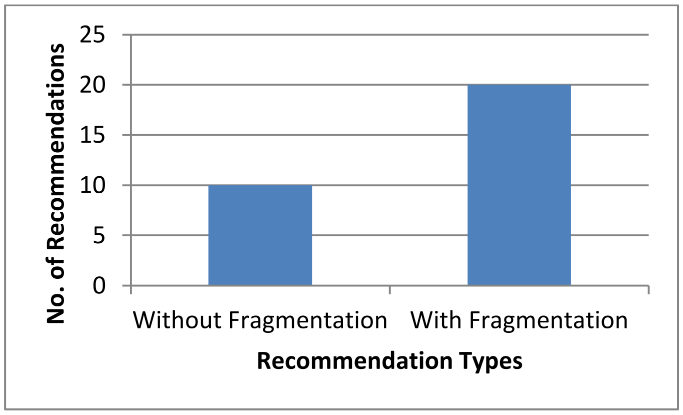

Figure 4.3 recommendation types vs no of recommendations. Here X-Axis denoted with and without fragmentation approach and the Y-Axis denotes the no of recommended queries . without fragmentation approach used is fount to increase linearly.

The figure 4.3 shows the accuracy chart for comparison of without fragmentation approach and fragment based approach by recommending the SQL queries obtained using the equations namely From this fragmented query is obtained which is more accurate when compared to the query obtained without doing fragmentation approach. This is because of the no of query suggested by the fragment method is more.

CONCLUSION AND FUTURE WORK

This project mainly focused on SQL query recommendations. Hence the fragmentation method is introduced to fragment the query then it is rated based on query matching finally it is recommended to the users of relational databases. This ensures that efficient resource sharing. .Hence, a query recommendation system using users querying behavior have been designed and implemented.

The future work on this paper is to enhance the query recommendation by design and develop a more generic and scalable system. For this work the following mechanisms are going to be followed

- Query Expansion and verification
- Query buffering and indexing
- Query clustering and ranking

REFERENCES

[1]Chatzopoulou G, Eirinaki G, Koshy M, Mittal S, Polyzotis S, Varman N, " TheQueRIE system for Personalized Query Recommendations," IEEE Data Eng. Bull., vol.34,no.2, pp 55-60, 2011.

[2]Chatzopoulou G, M. Eirinaki, and N. Polyzotis, "Collaborative filtering for interactive database exploration," in Proceeding of the 21st Intl. Conf. on Scientific and Statistical Database Management (SSDBM '09) pp 3-18, 2009.

[3]Giacometti A., Marcel P, Negre E., & Soulet A" Query recommendations for OLAP discovery driven analysis" in Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP pp 81-88, 2009.

[4]Giacometti A., Marcel P, & Negre E, "Recommending multidimensional queries," in Data Warehousing and Knowledge Discovery , pp 453-466, 2009.

[5]Khoussainova N., Kwon Y., Balazinska M, & Suciu D," SnipSuggest: context-aware autocompletion for SQL," in Proceedings of the VLDB Endowment, pp 22-33, 2010.

[6] Marcel P, & Negre E, "A survey of query recommendation techniques for data warehouse exploration," in EDA, pp 119-134, 2011.

[7] Mittal S, Varman J. S. V, Chatzopoulou G., Eirinaki M., & Polyzotis N, " QueRIE: Query Recommender System supporting Interactive Database Exploration," Data Mining Workshops (ICDMW), IEEE International Conference, pp 111-141, 2013.

[8] Stefanidis K, Koutrika G, and Pitoura G, "A survey on representation, composition and application of preferences in database systems," in ACM Trans. Database Syst., pp 28- 36, 2011.

[9] Xiwang Yang, Yang Guo, Yong Liu ,"Bayesian Inference Based Recommendation in Online Social Networks ," in IEEE Transactions on parallel and distributed systems, vol. 24, 2013.

[10]Zhiyuan Chen, Tao Li, and Yanan Sun ,"A Learning Approach To SQL Query Results Ranking Using Skyline And Users' Current Navigational Behavior," in IEEE Transactions On Knowledge And Data Engineering, vol. 25, pages 2683 – 2693,2013.