International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 6 4190 – 4193

Revisting SQL Query Recommender System Using Hierarchical Classification

Shruti Y. Patil
Department of Computer Science
S. S. G. B. College
Bhusawal, India
shrutiypatil@gmail.com

Prof. Dinesh D. Patil
Department of Computer Science
S. S. G. B. College
Bhusawal, India
dineshonly@gmail.com

Abstract— For analytical purposes, lots of data are gathered which are gathered and explored in data warehouses. Even to handle such a large data is a tough task for expert people. For non-expert users or for users who are not familiar with the database schema, handling such a voluminous data is more difficult task. The aim of this paper is to facilitate this class of users by recommending them SQL queries that they may use. By following the users past behavior and comparing them with other users, these SQL recommendations are selected. Initially, users may not know from where they can start their exploration. Secondly, users may overlook queries which help them to retrieve important data. Using hierarchical classification, the queries are recorded and compared which is then re-ranked according to relevance. Using users querying behavior, the relevant queries are retrieved. To issue a series of SQL queries, users use a query interface which aim to analyze the data and mine it for interesting information.

Keywords- *Data Mining, Data discovery, Interactive data exploration, Query personalization.*

*****

I. INTRODUCTION

Now days, database systems are more popular. These systems support for interactive exploration of large volumes of data. The example such as Genome browser which gives access to the genomic database and SkyServer contains large volumes of astronomical measurements. These databases allows user to submit queries and retrieve the results.

The need for data discovery tools increases as the data are increases vastly. Over large databases, despite the availability of querying tools, the users often have difficulties in understanding the underlying complicated schema and formulating queries. For example, the study on Hive. The data warehouse platform used in Facebook, come up with the following: Because of the heavy usage, a lot of tables are generated in data warehouse which in turn vastly increased the need for data discovery tools. Even when users have the capability to issue complex queries over large data sets, the task of knowledge discovery remains a big challenge. Moreover, a complete exploration of such databases is not practically feasible due to the continuously increasing size of the data.

To support the non-expert user for retrieving interesting information, query recommender system is used. Recommender Systems are software tools and techniques which provides suggestions to users for items to be of their use. The suggestions provided help the users in various decision-making processes, such as what items to buy, which music to listen, or what news to read. For online users, recommender systems have proven to be valuable means to deal with the information overload and have become one of the most powerful and popular tools in electronic commerce.

The interest of the user is drawn out and makes recommendations accordingly by the Query Recommender systems. Those recommended queries can be used as templates and submitted as it is instead of composing new ones or they can be further edited. In an attempt to identify previous users with similar information needs, this system continuously monitors the user's querying behavior and finds matching patterns in the system's query log. Subsequently, query recommender system uses these "similar" users and their queries to recommend queries that the current user may find interesting.

In this work, the query is stored in the query log. When the active user fired a query, his query is stored in the log and matched with the past users queries in query log with cosine similarity method. If queries are matched then those queries are recommended to active user so that they may be of user interest. This motivation draws from Web Recommender System. The principle on which the system is built is simple: If user A and user B placed the same queries then the other queries of each user may be of interest of each other.

This idea is projected with the help of collaborative filtering. A collaborative Query Management System is placed on new, large scale, shared data environments. Collaborative filtering method makes automatic predictions (filtering) about the interest of the user by collecting preferences information from many users. This method requires (1) user's active participation, (2) an easiest way to represent user's interest to the system, and (3) methods that are able to match people with similar interests. The system should also mine its query log and actively recommend queries to users, thus help them further pull the previously performed analysis.

II. RELATED WORK DONE

A keyword based query interface is provided by web databases which suffers from the empty answer or too many answers problems. In such case, it is critical to provide the correct answer to each user rather than the exact one to everyone for the same query. Preferences may be expressed qualitatively or quantitatively by embedding into relational query languages a special operator or by re-ranking or filtering the results of the original query respectively. Recently, as an additional personalization parameter, context has been added in such systems. The common denominator of these works with ours is that all can be categorized under the query personalization area.

 ISSN: 2321-8169
4190 – 4193

A multidimensional query recommendation system is already present. In this work, for generating OLAP query recommendations, the authors propose a framework for the users of a data warehouse. Although this work has some similarities to our, the techniques and algorithms used in the multidimensional development are very different to the ones we propose.

III. PROPOSED SYSTEM

Database systems provide the critical infrastructure to access and analyze large volumes of data in a variety of applications. For accessing the data from database, user uses querying tools. However new or not expert users face difficulties in understanding the underlying schema and formulating queries. This system provide help to database user, such as recommending the past user queries and also giving authority to execute or to edit recommended queries. When the structure of the queries is wrong then also our system provides similar recommend queries and gives authority to execute these queries.

The abstract framework is fundamentally a workflow, as shown in Figure 1.

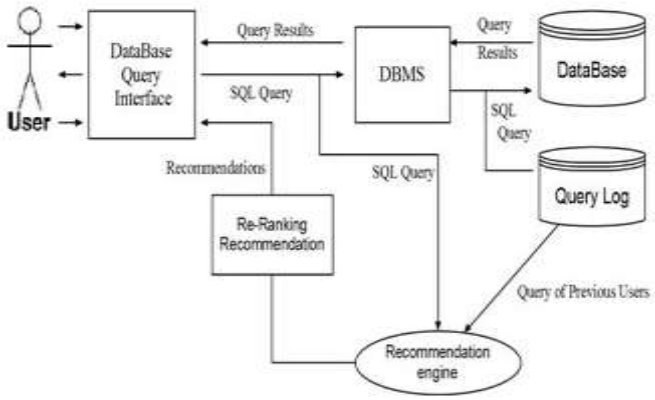

Figure 1. Architecture of Query Recommender System

The active user's queries are forwarded to both the database management system and the Recommendation Engine. The Database Management System processes each query and returns a set of results. At a time, the query is stored in the Query Log. The Recommendation Engine combines the current user's input with information gathered from the database interactions of past users, as stored in the Query Log, and generates a set of query recommendations that are returned to the user. If the query is nested then the recommended queries are re-ranked before returned to the user.

IV. FLOW OF SYSTEM

The Query recommender system recommends the query as per user interest. The flow of proposed system is as shown in Figure 2.

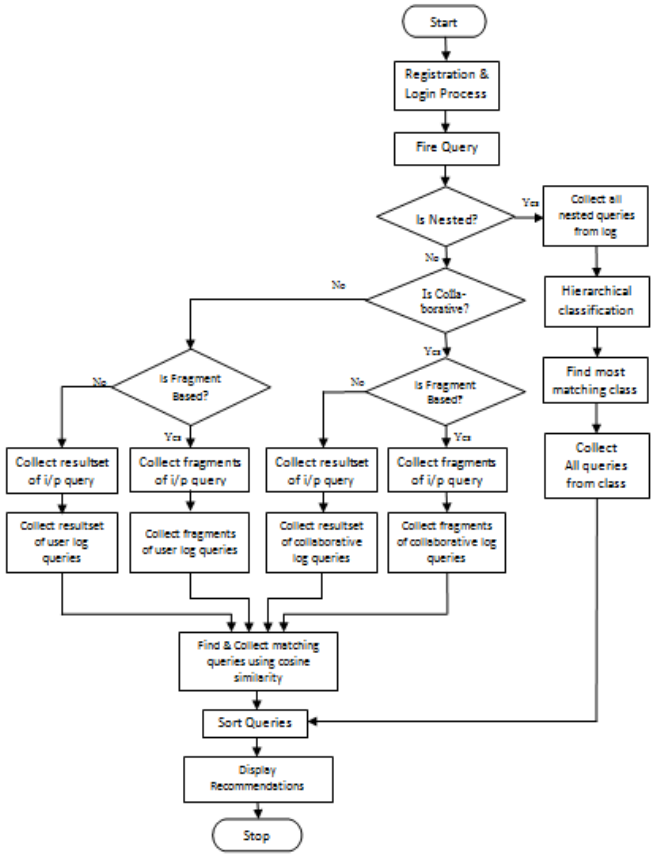

Figure 2. Flow of Query Recommender System

User needs to first register and login to a system. After that, he enters his query. If fired query is nested query then it will collect all nested queries from log. These queries are then classified using hierarchical algorithm. The most matching class will be then find out using cosine similarity method. All the queries from this class is collected and sorted. These sorted queries are then displayed as recommendations. If query is not nested then it will check for log and scheme filter. It will check that is there individual or collaborative log. It will also check for scheme filter i.e. fragment based or tuple based in combination with log filter. It will find and collect matching queries for selected filters using cosine similarity method. These queries are then sorted and display to user as recommendation.

V. IMPLEMENTATION

A. *Tuple and Fragment Based Recommendation*

In tuple based approach, the result based scheme is used. User fired query's result is stored in the user log in the form of tuples. The resultset of active user query are matched with the past user queries using cosine similarity method. The queries, whose threshold value for similarity is greater than 0.5, are then recommended to the users.

Fragment based approach is based on the pair-wise similarity of query fragments i.e. on attributes, tables, joins and predicates. We need to identify fragments that co-appear in several queries posed by same or different users.

- Query Fragmentation
  The user fired query is split into fragments with respect to the keywords such as select, from, where,

 ISSN: 2321-8169
4190 – 4193

group by, having, order by etc. These fragmented query attributes are then stored in the fragment table.

- Query Filter
  The active user's query is fragmented using fragment based approach. The query is compared with the already recorded fragments in query log. If the queries match, the fragments of old query are deleted and the fragments of new query are inserted to avoid duplication. If the queries don't match, the new fragments are updated in the query log.
- Query Suggestion Engine
  The query suggestion 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. The fragmented query is compared with the queries in log. If the fragmented query matches with the queries in log, then matched queries are recommended to the user otherwise, the result of the input query is returned.

B. Recommendation for Nested Query Using Hierarchical Classification

The recommendation for the nested queries is done by matching the query of active users with the recorded queries in query log of past users with the help of cosine similarity method.

The cluster of the most matching queries is formed for recommendation. For the clustering, we are using hierarchical classification. The proposed algorithm for hierarchical classification is as follows in Figure 3:

Input: Set Of Log Queries $Q = \{Q_0, Q_1, Q_2 \ldots Q_{n-1}\}$
Threshold thr=0.5
Output: Set Of Classes $C = \{C_0, C_1, C_2 \ldots C_{m-1}\}$

Steps:
Step 1: Create first class C_0, assign first Query Q_0 to C_0
$C_0 \leftarrow Q_0$
$C = C \cup C_0$
Step 2: for r=1 to n-1
flag=0
mostmatching= -1
for j=0 to m-1 // for remaining all Queries
Calculate CosineSimilarity sim= (Q_r, C_j)
if(sim >= thr and sim >= flag)
flag=sim
mostmatching = j

if(mostmatching == -1)
Create new class C_m
$C_m \leftarrow Q_r$
$C = C \cup C_m$
Else
$C_{mostmatching} \leftarrow Q_r$
End for
End

Figure 3 . Proposed Algorithm

VI. PROPOSED SYSTEM

The query recommender system recommends queries as per user interest. This system helps to non-experts users for analyzing and extracting the information of their interest in data warehouse.

User needs to first register himself for the use of this system. As shown in Figure 4, after clicking on the button 'Execute a Query', user can fired a query. At the time of firing of query, user gets recommendation according to his interest. He can select any of the recommended queries to execute or he can be able to edit that query as per need.

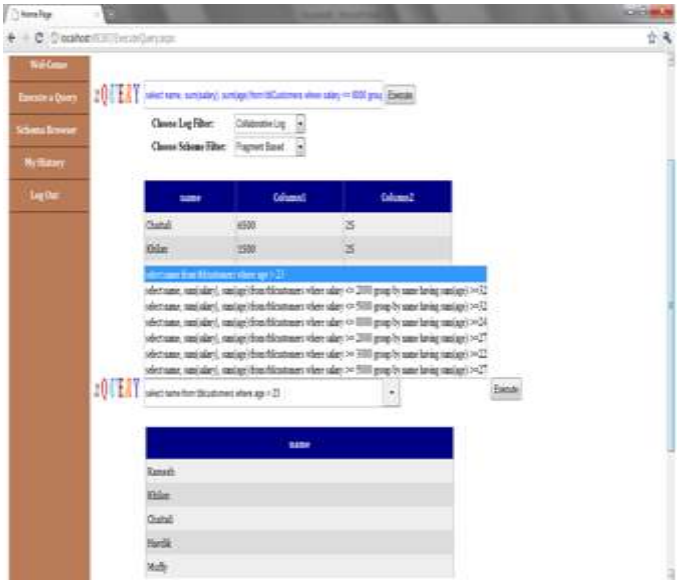

Figure 4 Execution of Query with Collaborative Log and Fragment Based Filter

When user fired nested query then it will match with database query with the help of cosine similarity method. And all the matching queries are recommended to the user Here, we are not using any filter. It is as shown in Figure 5.

The users who is first time going to fired a query may not know the definition schema of the selected table. The user first needs to select the table name of which user wants to see the definition. After clicking on the button 'Schema Browser',it will displayed the column names and data types of that column of the selected table.

If user wants to see the history of his queries i.e. if he want to see which queries he fired previously then it can shown by 'My History'. By clicking on button 'My History' , it will display all the queries fired by that user previously.

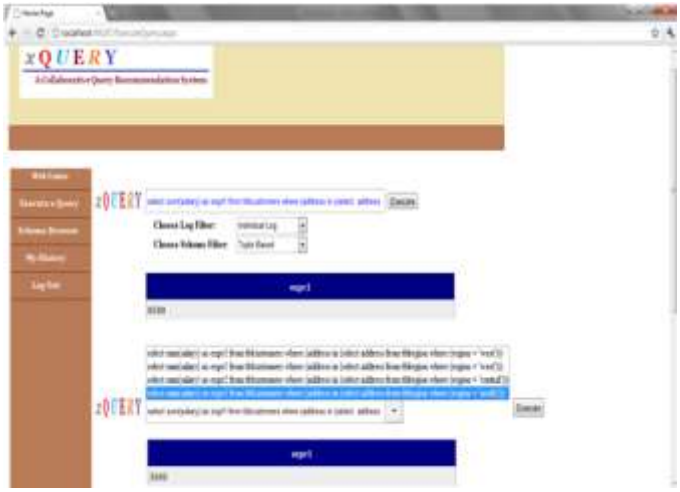

Figure 5. Recommendation for Nested Query Using Hierarchical Classificatioin

 ISSN: 2321-8169
4190 – 4193

VII. PERFORMANCE EVALUATION

In order to evaluate the performance of the proposed technique, we have used two measures which are Precision and Recall. Precision and recall are the most popular metrics which are widely used in evaluating search strategies.

Precision is defined as the ratio of the number of queries that correctly retrieved to the total number of queries retrieved.

$$\text{Precision} = \frac{\text{Number of relevant queries retrieved}}{\text{Total number of queries retrieved}}$$

While recall is the ratio of the number of queries that retrieved correctly to the total number of queries in log.

$$\text{Recall} = \frac{\text{Number of relevant queries retrieved}}{\text{Total relevant queries in log}}$$

High precision means less irrelevant queries are returned or more relevant queries are retrieved. The performance of the system, by plotting precision and recall graph is as shown in Figure 6, in which precision values are plotted against recall values.

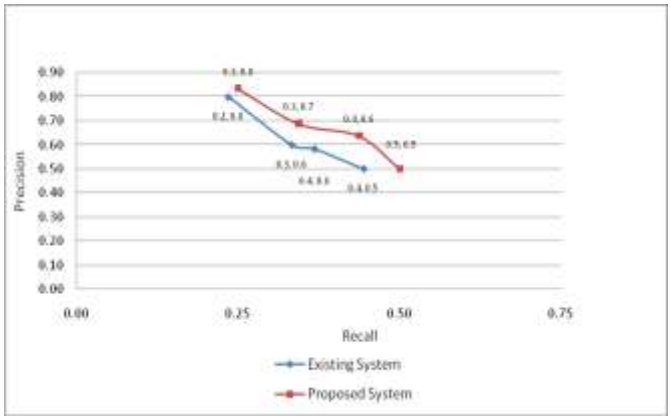

Figure 6. Comparision Graph of Existing and Proposed System

CONCLUSION

Recommendation system is a well-known field of the data mining used to pull out the essential knowledge from a huge amount of user query logs. Query recommender system supports users for analyzing and mining interesting information. A query recommendation system using fragment based approach helps the users to execute simple SQL query and proposed recommender system using hierarchical classification helps users to execute nested queries. The proposed system is shown to outperform previous fragment based approach.

ACKNOWLEDGMENT

The completion of this paper would not have been possible without the support and guidance of Prof. Dinesh D. Patil. He has been a constant of motivation and has encouraged me to pursue this project. With my deep sense of gratitude, I thank my respected teachers for supporting this of my project. This project report provides me with an opportunity to put in the knowledge of advanced technology. I thereby take the privilege opportunity to thank my guide and my friends whose helps and guidance made this study possibility.

REFERENCES

[1] Magdalini Eirinaki, Suju Abraham, NeoklisPolyzotis, and NaushinShaikh, "QueRIE: Collaborative Database Exploration" IEEE transactions on knowledge and data engineering, vol. 26, no. 7, July 2014.

[2] A. Thusooet al., "Hive - A petabyte scale data warehouse using hadoop," in Proc. IEEE 26th ICDE, Long Beach, CA, USA, Mar.2010, pp. 996–1005.

[3] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis, "Collaborative filtering for interactive database exploration," in Proc. 21st Int.Conf. SSDBM, New Orleans, LA, USA, 2009, pp. 3–18.

[4] S. Mittal, J. S. V. Varman, G. Chatzopoulou, M. Eirinaki, and N. Polyzotis, "QueRIE: A recommender system supporting interactive database exploration," in Proc. IEEE ICDM, Sydney, NSW, Australia, 2010.

[5] J. Akbarnejadet al., "SQL QueRIE recommendations," PVLDB, vol. 3, no. 2, pp. 1597–1600, 2010.

[6] E. Cohen, "Size-estimation framework with applications to transitive closure and reachability", J. Comput. Syst. Sci,vol. 55, no. 3, pp. 441–453, 1997.

[7] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," IEEE Internet Comput., vol. 7, no. 1, pp. 76–80, Jan./Feb. 2003.

[8] N. Koudas, C. Li, A. K. H. Tung, and R. Vernica, "Relaxing join and selection queries," in Proc. 33nd Int. Conf. VLDB, Seoul, Korea, 2006, pp. 199–210.

[9] B. M. X. Jin and Y. Zhou, "Task-oriented web user modeling for recommendation," in Proc. User Modeling, Edinburgh, U.K., 2005.

[10] K. Stefanidis, G. Koutrika, and E. Pitoura, "A survey on representation, composition and application of preferences in database systems," ACM Trans. Database Syst., vol. 36, no. 4, Article 19, 2011.

[11] G. Koutrika and Y. Ioannidis, "Personalized queries under a generalized preference model," in Proc. 21st ICDE, Washington, DC, USA, 2005.

[12] J. Levandoski, M. Mokbel, and M. E. Khalefa, "A Demonstration of FlexPref: Extensible Preference Evaluation Inside the DBMS Engine," in Proc. IEEE ICDE, Long Beach, CA, USA, 2010.

[13] A. Giacometti, P. Marcel, and E. Negre, "Recommending multidimensional queries," in Proc. 11th Int. Conf. DaWaK, Linz, Austria, 2009.

[14] A. Giacometti, P. Marcel, E. Negre, and A. Soulet, "Query recommendations for OLAP discovery driven analysis," IJDWM, vol. 7, no. 2, pp. 1–25, 2011.

[15] G. Koutrika, B. Bercovitz, and H. Garcia-Molina, "FlexRecs: Expressing and combining flexible recommendations," in Proc. SIGMOD Conf., New York, NY, USA, Jun. 2009, pp. 745–757.

[16] M. Drosou and E. Pitoura, "ReDRIVE: Result-driven database exploration through recommendations," in Proc. CIKM, Glasgow, U.K., 2011, pp. 1547–1552.

[17] N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu, "SnipSuggest: Context-aware autocompletion for SQL," PVLDB, vol. 4, no. 1, pp. 22–33, 2011.