QueRIE : Personalized Query Recommendation

[1]Narayan B.Vikhe, [2]Prof. M .M. Naoghare
[1]Student ME Computer Engineering, [2]Project Guide Dept.of Computer Engineering
Savitribai Phule Pune University,S.V.I.T. Chincholi Nashik, India

***Abstract* - Basically a Query is request for information from a Database. Many database systems requires you to make requests for information in the form of a stylized query that must be written in a special Query Language. Database Query Language attempts to give accurate answers to the accurate questions. For the aid of users who lack the SQL expertise we develop " QueRIE : Personalized Query Recommendation System ". Where the QueRIE holds the trace of the users querying style and finds matching patterns in the systems query log, in an attempt to recognize previous users with similar information requirements. Subsequently, QueRIE treats these similar users and their queries to commend queries that the current user may find enthralling. Here we describe an instantiation of the QueRIE framework, where the active users session is depicted by a set of query fragments. The recorded fragments are used to recognize similar query fragments in the previously recorded sessions, which are in turn collected in likely fascinating queries for the active user.**

***IndexTerms* - Data Exploration, Fragmentation, Interactive Database Exploration, Personalization, QueRIE, Query Fragment, Query Log, Query Recommendation, SQL, Tuple Based.**

I.INTRODUCTION

The database systems store large amount of data. When the number of users who use the database increases, it is critical to access large volume of data from the database systems. The users who lack familiarity with the database schema often have difficulties in retrieving relevant data. First-time users may not have the necessary knowledge to know where to start their exploration. Other times, users may simply overlook queries that retrieve important information. This work describes a framework to assist non-expert users by providing personalized QueRIE recommendations. The recommended queries are relevant to the user's information needs and can be submitted directly or be further refined. In other words, the user can use them as "templates" for query formulation instead of having to compose new ones.

QueRIE is built on a simple premise that is inspired by Web recommender systems: If users A and B have posed similar queries, then the other queries of B may be of interest to user A and vice versa [5]. In other words, we can recommend the queries of user B in order to help user A in their exploration of the database [5]. A database is an organized by collection of data. The DBMS is designed to permit the definition, the creation, querying, updatation, and the administration of the databases. DBMS which are well-known include My SQL, Oracle, Posture SQL and so many. A database is not generally portable across different DBMS, but different DBMS can interoperate by using standards such as SQL and ODBC or JDBC which in turn allows a single application to work with more than one DBMS for example, modeling the availability of seats in Railway in a way that supports finding a Train with vacancies.

II.RELATED WORK

One of the famous authors from [1] says that HADOOP is a popular open-source map implementation which is being used in Yahoo, Facebook etc. to heap and process extremely large data sets on product hardware. However reduce programming model is very low requires developers to write custom programs which are hard to conserve and reuse. Hive, an open data warehousing solution constructed on the top of HADOOP. Hive aids the queries convey in a SQL like declarative language - HiveQL, which are compiled into map-reduce jobs that are executed using HADOOP. It is mentioned in [2] that the demonstration presents QueRIE, a recommender system that assists interactive database exploration. This system focuses at supporting non-expert users of scientific databases drawing motivation from Web recommender systems, QueRIE trails the querying performance of each user and identifies potentially "interesting" parts of the database related to the corresponding data analysis task by locating those database parts that were accessed by similar users in the past. It then creates and suggests the queries that cover those parts to the user. Also [3] mentions we can use recommendation algorithms to personalize the online store for each customer. The store entirely changes based on customer attraction, by showing programming titles to a software engineer person and baby toys to a mother. There are 3 common approaches for solving the recommendation proble.

1. Traditional Collaborative Filtering,
2. Cluster Models, And
3. Search-Based Methods.

We compare these approaches with our algorithm, which is call item-to-item collaborative filtering. Unlike that of the traditional collaborative filtering, our algorithm's online computational independently of the number of the customers and number of the items in the product catalog. Our algorithm produces suggestions in real time, scales to the huge data sets, and generates high quality suggestions. And [4] find the solution for problem that we face regularly that we present a preference model that combines expressivity and concision. In addition, we assign efficient algorithms for the choice of preferences related to a QueRIE, and an

algorithm for the advanced generation of customized results, which are categorized according to the user interest. Several categories of the ranking functions are made available for this purpose. We present the outcome of experiments both the synthetic and with the real users:

a) Demonstrating the effectiveness of our algorithms.
b) Showing the ease of QueRIE personalization.
c) Providing Insight As To The Appropriateness Of The Proposed Ranking Functions.

III.PROPOSED SYSTEM

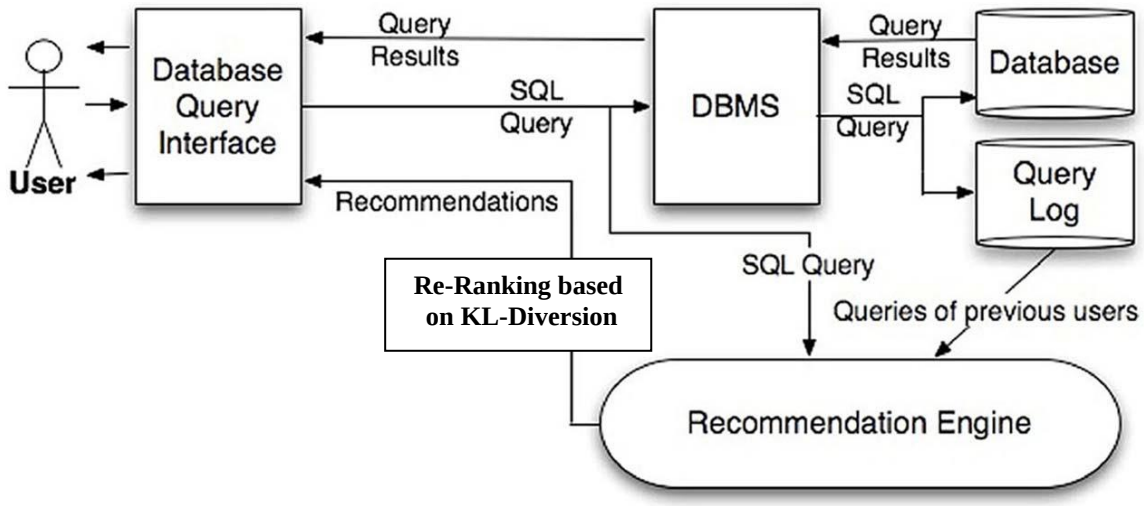


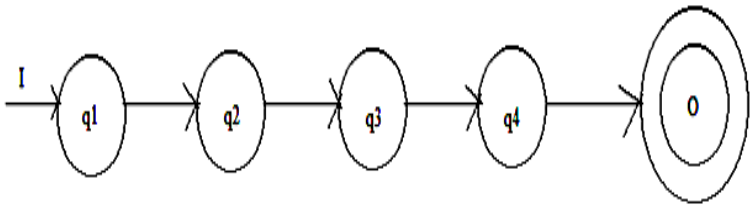
Fig 3.1 : Proposed System Architecture.

The System Architecture acts as a workflow shown in figure 3.1 above. The SQL queries of the active users are forwarded to the DBMS as well as the Recommendation Engine. The DBMS(Database Management System) processes each query and returns a set of results and at the same time, the query is heaped in the Query Log. The Recommendation Engine merge the current users input with information collected from the database interactions of past users, as recorded in the Query Log, and creates a set of query recommendations that are returned to the user. Then we consider a setting where users inquire a relational database through a sequence of SQL queries. The goal of the exploration is to find out an enthralling information or verify a particular hypothesis. The queries are composed based on this goal and reached the users overall information need. As a result, the queries reported by a user during one visit (commonly called session) to the database are tallyed, in that the user develop the next query in the sequence after having checked the results of previous queries.

Three approaches are used to generate the query recommendations:

a) Suggested Query By Dictionary Mapping
b) Suggested Query By Tuple Based Query Recommendations
c) Suggested Query By Fragment History.

IV. MATHEMATICAL MODEL



In this Query I is submitted to state q1 where the query log is maintained then it is passed to state q2 where the session summaries are extracted then in state q3 the target tupples are generated and in q4 state re-ranking is done based on Clarity score formula and the output is generated in final state O.

Input Parameter(I) :

I = I1

where I is set of Input.

I1= query which is submitted.

Functional Parameter(Q) :

Q=q1,q2,q3,q4

Where, Q is functions/process done by recommendation engine.

q1 = Maintaining of query log.

q2 = Extraction of session summary.
q3 = Generation of target tupples.
q4 = Re-ranking based on clarity
Output Parameter(O) :
O = O1
where ,O is an Output parameter.
O1 = Result generated by the Recommendation Engine.

V.IMPLEMENTATION

KL DIVERGENCE ALGORITHM

With the use of KL Divergence Theorem we will compute the clarity score of comment. To the relevancy of comment with publisize, we need to store the clarity score of comment with respect to Clarity Score Formula :

$$Clarity_q(C_i) = \sum_{w \in V_{ci}} P(w|\theta_q) \log_2 \frac{P(w|\theta_q)}{P(w|\theta_{C_i})}$$

$$P(w|\theta_q) = \frac{1}{\mathcal{Z}} \sum_{d \in \mathcal{R}} P(w|D) P(q|D)$$

and $\mathcal{Z}$ is defined as.

$$\mathcal{Z} = \sum_{D \in \mathcal{R}} P(q|D)$$

$$P(q|D) = \prod_{w \in q} P(w|D).$$

On the basis of clarity score we form the clusters based on Divergence scores.

5.2. AGGLOMERATIVE CLUSTERING ALGORITHM :

The algorithm forms clusters in a bottom up manner, as given below:

I. Initially, put each article in its own cluster.
II. Among all current clusters, pick the 2 clusters with the smallest distance.
III. Replace these two clusters with a new cluster, formed by combining the 2 original ones.
IV. Repeat the above two steps until there is only one remaining cluster in the pool.

Thus agglomerative clustering algorithm will result in a binary cluster tree with single article clusters as its leaf nodes and a root node containing all the articles.

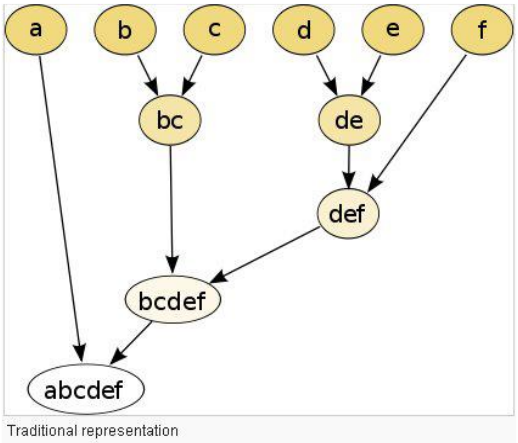


Fig 5.1: Agglomerative algorithm working system

VI. RESULT ANALYSIS

The evaluation of the proposed Querie framework is done by using the MySQL Server. The evaluation is based on the following SQL query

SELECT * FROM tblCustomers

This is given to the Database Query interface. It will generate a set of result which is returned to the user. And also a set of recommendations are generated by using the three methods viz., suggested query by dictionary mapping, suggested query by tuple based query recommendations, suggested query by fragment history. At all times, the active user is able to: (a) formulate a query from scratch, (b) select a recommended query and submit it as it is, or (c) select a recommended query and edit it before submitting it to the database. Moreover, the interface allows the user to browse the database schema, analyze and re-submit queries that were posed during her recent history. A snapshot of the QueRIE prototype is shown in Figure 6.1 The recommendations are generated in three different ways i.e., suggested query by dictionary mapping, suggested query by tuple based query recommendations, suggested query by fragment history. From this chart it is understood that the fragment based method is efficient than the other two methods. Because it takes only 0.073208363 second. But the other two methods namely dictionary mapping and tuple based query recommendations takes 0.383681081second and 0.703118613 second respectively.

Fig.6.1: QueRIE interface after a query has been submitted.

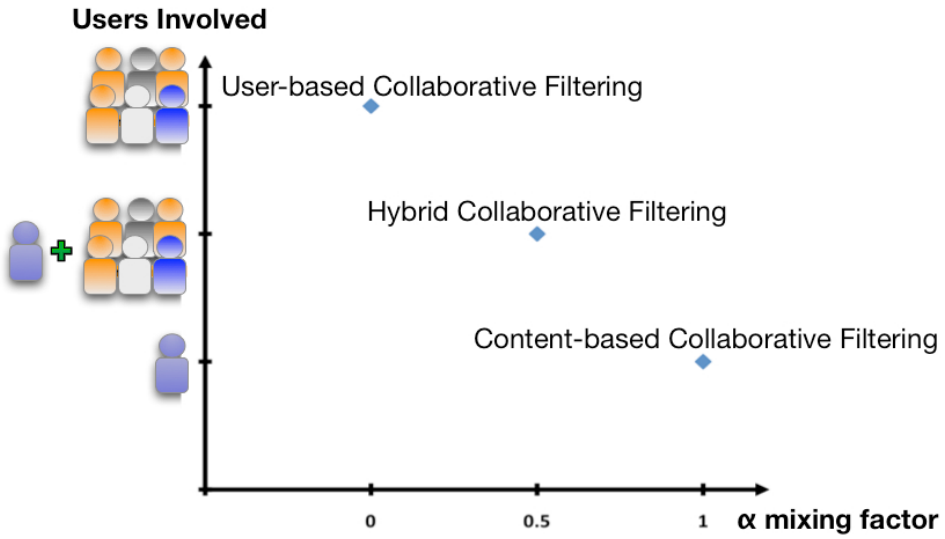


Fig. 6.2 Result Of Hybrid Collaborative Filtering & Collaborative Filtering

The big advantage of the fragment-based approach is that it can be implemented very efficiently; the space of fragments grows slowly allowing for a scalable system. The fragment-to fragment similarities can be computed offline and stored for very fast retrieval when recommendations need to be generated, leveraging all the advantages of item-to-item collaborative filtering.

CONCLUSION

There are many interesting directions we would like to explore in the future. We would like to measure the impact which the QueRIE relaxation process has in the quality of recommendations. Exploring a sequence-based approach is another interesting direction for the future work, but it needs a careful reconsideration of several aspects of our frame work. For instance, pure sequence information may not be sufficient to discover user similarities. Instead, we might have to consider the relative changes between queries in the sequence, e.g., the selection predicates becomes more selective as queries advance, in order to properly detect similarities. We also plan to focus on relational databases that have a form based interface. While that of the fragment based approach appears as a straightforward selection for such environments, new tests related to the formulation of session similarity, the synthesis of the proposal and their presentation arise. Finally, as we aim at developing a more generic and scalable system, we are currently working on combining alternative techniques for generating recommendations.

ACKNOWLEDGMENT

I wish to express my sincere gratitude to the Head Of Department Prof. S.M.Rokade of M.E. Computer Engineering Department for providing me an opportunity for presenting a project on the topic ”QueRIE :Personalized Query Recommendation”. I sincerely thank to my project guide Prof. M.M.Naoghare for her guidance and encouragement in the partial stage completion of my project work. I also wish to express my gratitude to the officials and other staff members who rendered their help during the period of my project work. Last but not least I wish to avail myself of this opportunity, to express a sense of gratitude and love to my friends and my parents for their manual support, strength, help and for everything.

REFERENCES

[1] A. Thusoo et al., Mar. 2010, pp. 9961005, Hive – A petabyte scale data warehouse using hadoop, in Proc. IEEE 26th ICDE, Long Beach, CA, USA,.
[2]S. Mittal, J. S. V. Varman, G. Chatzopoulou, M. Eirinaki, and N. Polyzotis, QueRIE: A recommender system supporting inter-active database exploration, in Proc. IEEE ICDM, Sydney, NSW, Australia, 2010.
[3] G. Linden, B. Smith, and J. York, Amazon.com recommendations: Item-to-item collaborative filtering, IEEE Internet Comput., vol. 7, no. 1, pp. 7680, Jan./Feb. 2003.
[4] J. Akbarnejad et al., “SQL QueRIE recommendations,” PVLDB, vol. 3, no. 2, pp. 1597–1600, 2010.
[5] Magdalini Eirinaki, Suju Abraham, Neoklis Polyzotis, and Naushin Shaikh. JULY 2014. ”QueRIE: Collaborative Database Exploration”. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 26, NO. 7.
[6] C.W. Bachmann (November 1973), The Programmer as Navigator, CACM (Turing Award Lecture 1973).
[7] ”DB-Engines Ranking”. January 2013. Retrieved 22 January 2013.
[8] Proctor, Seth (2013). ”Exploring the Architecture of the NuoDB Database, Part 1”. Retrieved 2013-07-12.
[9] ”TeleCommunication Systems Signs up as a Reseller of TimesTen; Mobile Operators and Carriers Gain Real-Time Platform for Location-Based Services”. Business Wire. 2002-06-24.
[10] ”database, n”. OED Online. Oxford University Press. June 2013. Retrieved July 12, 2013.
[11] IBM Corporation. ”IBM Information Management System (IMS) 13 Transaction and Database Servers delivers high performance and low total cost of ownership”. Retrieved Feb 20, 2014.