**Query Recommendations for Interactive Database Exploration**

Rajendra Desale, Vaibhav Patil, Tushar Aware

Shatabdi Institute Of Engineering, Agaskhind, Computer Department

**Abstract** - Traditional DBSMs are suited for applications in which the structure, meaning and contents of the database, as well as the questions to be asked are already well understood. There is, however, a class of applications that we will collectively refer to as Interactive Data Exploration (IDE) applications, in which this is not the case. IDE is a key ingredient of a diverse set of discovery-oriented applications we are dealing with, including ones from scientific computing, financial analysis, evidence-based medicine, and genomics. The need for effective IDE will only increase as data are being collected at an unprecedented rate. IDE is fundamentally a multi-step, non-linear process with imprecise end-goals. For example, data-driven scientific discovery through IDE often requires non-expert users to iteratively interact with the system to make sense of and to identify interesting patterns and relationships in large, amorphous data sets.

To make the most of the increasingly available complex and big data sets, users would need an expert assistant. who would be able to effectively and efficiently guide them through the data space. Having a human assistant is not only expensive but also unrealistic. Thus, it is essential that we automate this task. We propose database system terms be augmented with an automated database navigator (DBN) service that assists as a tour guide to facilitate IDE. Just like a car navigation system that offers advice on the routes to be taken and display points of interest, DBN would similarly steer the user towards interesting trajectories through the data, while highlighting relevant features. Like any good tour guide, DBN should consider many kinds of information; for instance, it should be sensitive to a user's goals and interests, as well as common navigation patterns that applications exhibit. We sketch a general data navigation framework and discuss some specific components and approaches that we believe belong to any such system.

**I. Introduction**

**A. Query Fragmentation**

The SQL syntax checker checks if the given input query is in SQL syntax. It also checks if the given `select` with that of the tables in the database and `where` the attributes in the database table. And then the input SQL query (IQ) 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`). Names are given for the fragmented queries. The fragmented query attributes are stored in the fragment table (Ft) with respect to the fragment name.

**B. Query Filter**

The query limit (QL) is set to n. The active users query (IQ) is fragmented using the FQ algorithm. The query q is compared with the already recorded fragments in query log (stack table). 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 entries in the query log are within the query limit. If the number of entries exceeds the query limit, the query log is full. The query is then removed from the query log accounting for an LRU policy.

**C. Query Suggestion Engine**

The query suggestion engine gives a set of recommended queries SQ for the given SQL query (IQ). The input query IQ is first fragmented and the fragmented query FQ is stored in a table t. The query profile QP from the query rating contains fragmentedid and rank of the queries. The fragmented query is compared with the queries in the query profile QP. If the fragmented query matches with any of the queries in the query profile QP, the fragment-based rank of the queries in the query profile QP is checked. The top n rank queries are returned as SQ. If the FQ does not match with any of the queries in the query profile QP, the result of the input query IQ is returned. We consider two different waiting schemes, a binary scheme and a weighted scheme. Both the users Q posed by user i. In binary scheme all participating fragments receive the same importance weight, regardless of whether they appear in many queries in the session or only one. In weighted scheme fragments that appear more than once in a user session will receive high weight than others. The fragment-based instantiation of the QueRIE framework works in a similar manner to the tuple based.

**II. What Is Recommendation System?**

Recommender systems or recommendation systems are a subclass of information filtering system that seek to predict the rating or preference that user would give to an item. Recommender systems have become extremely common in recent years, and are applied in a variety of applications. The most popular ones are probably movies, music, news, books, research articles, search queries, social tags, and products in general. Recommender systems typically produce a list of recommendations in one of two ways through collaborative or content-based filtering. Collaborative filtering approaches build a model from a user's past behavior as well as similar decisions made by others: the model allows us to predict items that the user may have an interest in. Content-based filtering approaches utilize a series of discrete characteristics of an item in order to
recommend additional items with similar properties. Recommender system is an active research area in the data mining and machine learning areas.

III. RELATED WORKS

A multidimensional query recommendation system is proposed in [4]. In Contextual Database Preferences it suggest that context may express conditions on situations external to the database or related to the data stored in the database. It outlines models for expressing both types of preferences. Then, given a user query and its surrounding context suggests that the size of datasets being collected and analyzed in the industry for business intelligence is growing rapidly, making traditional warehousing solutions prohibitively expensive. In this paper, we provide a comprehensive presentation of QUERIE, including an overview of previous work (tuple based instantiation).

System Architecture For Query Recommendation System

Figure 1.1: System Architecture

IV. RECOMMENDATION ALGORITHMS

The queries of each user touch a subset of the database that is relevant to the analysis the user wants to perform. We assume that this subset is modeled as a session summary $S_i$ for user $i$. We use $1, \ldots, h$ to denote the set of past users based on which recommendations are generated and 0 to identify the current user. To generate recommendations, our framework extends the summary $S_0$ of the active user to a predicted summary $S_{pred,0}$. This extended summary captures the predicted degree of interest of the active user with respect to all the parts of the database, including those that the user has not explored yet, and thus serves as the seed for the generation of recommendations. To summarize, our framework consists of three components: (a) the construction of a session summary $S_i$ for each user $i$, (b) the computation of a predicted summary $S_{pred,0}$ for the active user, based on the active users and the past users summaries, and (c) the generation of queries based on $S_{pred,0}$. Those queries will be presented to the user as recommendations. The details of each step differ for each recommendation engine. We provide a brief overview of both approaches in what follows.

V. TUPLE-BASED RECOMMENDATIONS

Session summaries :-

We define the session summary $S_i$ as a vector of tuple weights that covers all the database tuples. The weight of each vector element represents the importance of the respective tuple in the exploration performed by user $i$. For this purpose we employ two different weighting schemes which are detailed in the accompanying paper [1]. Using the session summaries of the past users, we can define the conceptual session-tuple matrix that, as in the case of the user-item matrix in web recommender systems, will be used as input in our collaborative filtering process. Computing the predicted summary. Similarly to session summaries, the extended summary $S_{pred,0}$ is a vector of tuple weights. In order to compute this summary, we assume the existence of a function $sim(S_i, S_j)$ that measures the similarity between two summaries and takes values in $[0,1]$. Using this function, we compute the extended summary as a weighted sum of the existing summaries: $S_{pred,0} = \sum_{i=0}^{h} (sim(S_0, S_i)S_i)$. The similarity function $sim$ can be realized with any vector-based metric, such as the cosine similarity measure. Generating recommendations. The final step is to generate queries that cover the interesting tuples in $S_{pred,0}$. In order to provide the users with intuitive, easily understandable recommendations, we use the queries of past users. We assign to each past query $Q$ an importance with respect to $S_{pred,0}$, computed as $rank(Q, S_{pred,0}) = sim(S_Q, S_{pred,0})$. Hence, a query has high rank if it covers the important tuples in $S_{pred,0}$. The top ranked queries are then returned as the recommendation. Accelerating the online computations. To ensure that the aforementioned approach generates real-time recommendations for the active users of a database, we need to compress the session-tuple matrix and to speedup the computation of similarities. For this reason, we employ the MinHash probabilistic clustering technique that maps each session summary $S_i$ to a signature $h(S_i)$ [2]. The Jaccard similarity between vectors is thus reduced to the similarity of their signatures: $JaccardSim(S_i, S_0) = \frac{sim(h(S_i), h(S_0))}{sim(h(S_i), h(S_0))}$.

Fragment-based recommendations

Session summaries :-

This approach is based on the pair-wise similarity of query fragments (attributes, tables, joins and predicates). We need to identify fragments that co-appear in several queries posed by different users. The session summary vector $S_i$ for a user $i$ consists of all the query fragments of the users past queries. Let $Q_i$ represent the set of queries posed by $i$ and $F$ represent the set of distinct query fragments recorded in the query logs. For a
given fragment $2$, its importance in session $S_i$ is represented by $S_i[]$ and depends on its importance in the session. We can define $SQ[]$ as a weighted or binary variable that represents the importance of $i$ in a session query $Q$.

Then, $S_i$ is defined respectively as a sum ($S_i = PQ2Q_i SQ$), or $O$-ed ($S_i = WQ2Qi SQ$). Computing the predicted summary. Using the session summaries of the past users and vector similarity metric, we construct the $(|F| |F|)$ fragment-fragment matrix that contains all similarities $\text{sim}(-, i)$, $-i \in 2 F$. The recommendation seed, modeled by $\text{Spred} 0$, represents the estimated importance of each query fragment with regard to the active users behavior $S_0$. Similarly to the item-to-item collaborative filtering approach of web recommendersystems, we employ the fragment-to-fragment similarities that are computed in the previous step: $\text{Spred} 0 [] = P-2R S0[-\iota] \sin(-, i) P-2R \sin(-, i)$.

VI. IDENTIFY, RESEARCH AND COLLECT IDEA

Collaborative Framework with User Personalization for Efficient web Search : Mining approach

Introduction

User personalization becomes more important task for web search engines. We develop a unified model to provide user personalization for efficient web search. We collect implicit feedback from the users by tracking their behavior on the webpage based on their action on the webpage. We track actions like save, copy, bookmark, time spent and logging into database, which will be used to build the unified model. Our model is used as a collaborative framework using which related users can mine the information collaboratively with little amount of time. Based on the feedback from the users we categorize the users and search query. We build the unified model based on the categorized information, using which we provide personalized results to the user during web search. Our methodology minimizes the search time and provides more amount of relevant information.

Methodology. The user interface is a tabbed web browser, which is a part of the system. Through this browser the user can provide short-term query simultaneously in multiple tabs for his information need. The user interacts with the system to give search query, to view the results and to view the ranked results. The ranking is done based on the past search behavior of the user with the system. The browser also supports providing actions like SAVE, COPY, PRINT AND E-MAIL, which depicts the importance of the web page for his need. The browser also supports displaying the importance of the page for his need. Each user-visited web page is represented by a set of index words that comes under top list. The usage time of each search query and usage-time of each visited page is calculated transparently without disturbing the user. Based on the search query, index word and usage-times, the User Conceptual Index (UCI) is calculated. The UCI can be represented mathematically as a function of weights of above parameters. The usage time directly indicates whether previous search results were relevant or irrelevant to the users information need. The search queries that have similar or related meaning are categorized to a group using word dictionary in order to avoid inconsistencies that arise in above strategies. The visited-pages that have similar related index words are also categorized to recommend the pages for a novice user. The users with similar search behavior are categorized to a group to improve the efficiency of personalization mechanisms. The pages are re-ranked by analyzing individuals behavior and are projected to them in dual window.

Some of the strategy for personalization of web search is described as follows:

1. A users search history can be collected without direct involvement.
2. The users profile can be constructed automatically from the users search history and is dynamically updated.
3. The categories that are likely to be of interest to the user are deduced based on the users search behavior.

Feature Extraction

The first step in the project is to extract the feature of the user visiting the page. The feature of a page, $P$ is defined as a set of top n frequently occurring terms. In order to extract the features, the source content of each page is extracted and it is de-tagged. From the de-tagged page, the stop words are removed and the terms in the page are extracted. From the set of terms, the top n frequently occurring terms are extracted. These n terms form as index words of the page.

Algorithms

User visiting page $Pi$

Procedure:

1. The de-tagged and stop words eliminated page $Pi$ can be represented as $IW= IW_1, IW_2, IW_3$ where $IW$ is the index word set and $F$ is the Frequency set.
2. The de-tagged page, the stop words are removed and the terms in the page are extracted. From the set of terms, the top n frequently occurring terms are extracted. These n terms form as index words of the page.

User Association Analysis

The user association is analyzed to find the similarity of search among different users. From the set of visited-pages, the actions performed by the user are monitored. From the action it is concluded whether the page is useful to the user. The order in which the page is visited is also tracked and a directed graph is constructed. The usage-time for each page forms the weight of the page.
Algorithm:

Similarity measure
Given: User behavior graph Procedure:
step1: Indexing
for i:=1 to N do
for every vertex j of the web graph
do Behavior[i][j][]:= reversed path of length l starting from j.
end
end
step2: User Sim(i,j) ,Sim:=0
for i:=1 to N do
for j:=1 to N do
let k be the smallest offset with
Behavior[i][u][k]=Behavior[j][u][k]
if such k exists then
Sim:=Sim+ck
end
end
return Sim/N
step 3: End

The above algorithm is used to identify the similarity behavior between two users. Whenever the search behavior is common, it is certain that the users might come from same source point. Thus, higher the length l in the above algorithm, greater is the similarity.

Search Query Categorization
Greater the number of times a user uses a particular SQ, greater is the interest of the user for the particular topic associated with the search query terms. If the previous search result is not relevant to the users information need, then the user might modify the SQ into their context of search. Even though the search keyword gets changed, the information need on a topic doesn’t get changed in that session. Hence the alternate keyword supplied by the user may also be intended to search exactly for same topic. So it is necessary to identify the alternate meanings of users search query, which leads to categorization of the search query. The query categorization is necessary to reduce the limitations in key word based search.

Algorithm:

SQ Categorization
Step1: Collect all the search queries given by the user in a due course of time.
Step2: Find the alternate meaning of the search query using a word dictionary.
Step3: Find whether the result of exists in the index terms set collected from user history.
Step 4: If such commonalities exist, update the TF matrix

Visited Page Categorization

Higher the similarity ranks between two users, greater the commonalities of search between them. Two pages can be said to similar even they spoke of exactly the same topic with different keywords. Hence it is necessary to identify the alternate meanings of the Index words, which leads to categorization of the visited page. The categorization of the page is used to expand the similarity rank calculation, which aids to identify common search behavior.

Algorithm:

Page Classification
Step1: Collect all the index terms of all the visited pages by the user in a due course of time.
Step2: Find the alternate meaning of the index terms using a word dictionary.
Step3: Find whether the result from step2 exists in the index terms set collected from user history.
Step 4: If such commonalities exist, update the SF matrix.

Use of Simulation software
- Jdk , JCreator , Net beans etc.
- Oracle , MySQL, WampServer etc
- SkyServer etc.
- TC for C programming

VII. CONCLUSION
Scientists need help in exploring databases Query recommendations can be an effective tool in guiding exploration. Collaborative filtering provides a natural method to generate recommendations. Experiments show promising results on real world datasets

ACKNOWLEDGMENT
With all respect and gratitude, we would like to thank all people who have helped us directly or indirectly for the completion of this project work. We express our heartily gratitude towards Prof. S. G. Chordiya for guiding us to understand the work conceptually and also for his constant encouragement to complete this Project work on Query Recommendations for Interactive Database Exploration. We also express our thanks to Prof C.N. Patki Head of Department of Computer Engineering for providing necessary information and required resources. With deep sense of gratitude we thank to our Principal Prof. Dr. Dayanand P. Joshi and Management of the A.V.E.W. Trust for providing all necessary facilities and their constant encouragement and support. Last but not the least we thank to all the Teaching and Nonteaching staff members of Computer Engineering Department for providing necessary information and required resources. We are ending this acknowledgement with deep indebtedness to our friends who have helped us.
REFERENCES


[9] Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents, Manos Papagelis, Dimitris Plexousakis.


AUTHORS

First Author – Rajendra Desale, BE (Computer Technology), Shatabdi Institute Of Engineering, Agashind, rajdesale2151@gmail.com

Second Author – Vaibhav Patil, BE (Computer Technology), Shatabdi Institute Of Engineering, Agashind, vaibhav.ps444@gmail.com

Third Author – Tushar Aware, BE (Computer Technology), Shatabdi Institute Of Engineering, Agashind, tusharaware888@gmail.com