Custom-Made Ranking in Databases Establishing and Utilizing an Appropriate Workload

Jothimani. M.[1] Mohanraj. M.[2]
[1]Professor [2]Assistant Professor
[1,2]Computer Science and Engineering Department
[1,2]PGP College of Engineering and Technology

Abstract— Custom Rating System which provides a facility to the users, that they can search and download best articles or anything on the system in the database. The article or anything can be any text content which can describe a product, a book, an institution, an application, a company or anything. This system consists of two set of users, one is the normal user and another is the administrator. The users have to register and login to the system first, in order to use the system. The users have the following privileges. Write Article and Upload Relevant Files, Post Related URL to each article for other users reference, Search and Read Article posted by other users, Rate the articles posted by other users. The articles which are written by any user are sent to the Administrator for Approval. After approval of the articles by the administrator, they are available for the users to search and download. Based on the Description provided in an article, it can be searched by any registered user on the system. The user can see the article, download a file if available and the user can rate the article based on the article. Rating can be given in terms of 1 Star to 5 Star. The users can search the article. The list of articles displayed can be sorted based on following parameters: Rating, Popularity (Number of Clicks on the Article),Relevance (Based on number of matching keywords provided),All Articles uploaded by a specific user.

Keyword: Custom ranking, databases, user similarity, query similarity, workload.

I. INTRODUCTION

THE emergence of the full of meaning has led to the proliferation of a large number of maze databases for a variety of applications (e.g., airline reservations, vehicle search, real estate scouting). These databases are typically searched by formulating query conditions on their schema attributes. When the number of results returned is large, it is time consuming to browse and choose the most useful answer(s) for further investigation. Currently, maze databases simplify this task by displaying query results sorted on the values of a single attribute. However, most maze users would prefer an ordering derived using multiple attribute values, which would be closer to their expectation. In this paper, propose a user- and query-dependent approach for ranking the results -of database queries. For a query Qj given by a user Ui, a relevant ranking function (Fxy) is identified from a workload of ranking functions (inferred for a number of user-query pairs), to rank Qj's results. The choice of an appropriate function is based on a novel similarity-based ranking model proposed in the paper. The intuition behind our approach is: 1) for the results of a given query, similar users display comparable ranking preferences, and 2) a user displays analogous ranking preferences over results of similar queries. We decompose the notion of similarity into: 1) query similarity, and 2) user similarity. While the former is estimated using either of the proposed metrics—query condition or query-result, the latter is calculated by comparing individual ranking functions over a set of common queries between users. Although each model can be applied independently, we also propose a unified model to determine an improved ranking order. The ranking function used in our framework is a linear weighted-sum function comprising of: 1) attribute-weights denoting the significance of individual attributes and 2) value-weights representing the importance of attribute values. In order to make our approach practically useful, a minimal workload is important. One way to acquire such a workload is to adapt relevance feedback techniques used in document retrieval systems. However, there exist several challenges in applying these techniques to web databases directly. Although this paper is on the usage, instead of the acquisition of such workloads, discuss and compare some potential approaches for establishing such workloads and elaborate on a learning method for deriving individual ranking functions.

1) Contributions

The contributions of this paper are:

1. In this propose a user- and query-dependent approach for ranking query results of web databases.
2. Develop a ranking model, based on two complementary measures of query similarity and user similarity, to derive functions from a workload containing ranking functions for several user-query pairs.
3. In present experimental results over the database based on the query and user dependent.
4. In this a discussion on the approaches for acquiring/generating a workload, and propose a learning method for the same with experimental results.

II. RELATED WORK:

Although there was no notion of ranking in conventional databases, it has existed in the context of information retrieval for quite some time. With the advent of the maze, ranking gained importance due to the volume of information being searched /browsed.

A. Existing System

In the existing system, the internet users cannot know the quality of the content available on the internet until they download and see them. Such facility is available only in forums, in that too the user should read the comments in order to analyse the quality of the content. In some websites, they allow the users the upload content to the website and let

the users to download the content. But, the content is not verified. The content can be a spam content which is not necessary for any user. Such content should not be permitted. There are no websites which analyse total content before uploading. Rather, after the content is uploaded, if any user report on the content, then it is deleted. Before reporting, the content could be downloaded by many users. In the current trend, user is not given many options to sort the contents that they search.

B. *Disadvantage of Existing System*

1. Users cannot know the quality of the content before downloading.
2. The spam content posted by users are not filtered from the system frequently.
3. Users are not given more options to sort the search results.

Currently, ranking has become everywhere and is used in document retrieval systems, recommended systems, maze search/browsing, and traditional databases as well. Below, relate our effort to earlier work in these areas. Given the notion of user- and query-similarity, it appears that proposal is similar to the techniques of collaborative and content filtering used in recommendation systems.

1. Users are given a lot of options to sort out the search results such as by Similarity, Popularity, Rating, etc.

The problem of integrating the information retrieval system and database systems have been attempted with a view to apply the ranking models (devised for the former) to the latter; however, the intrinsic difference between their underlying models is a major problem.

1) Relevance feedback

Inferring a ranking function by analysing the user's interaction with the query results originates from the concepts of relevance feedback in the domain of document and image retrieval systems. However, the direct application of either explicit or implicit feedback mechanisms for inferring database ranking functions; it has several challenges, and to the best of its knowledge has not been addressed in literature.

III. 3 RANKING ARCHITECTURE

The Similarity model forms the centre component of our ranking framework. When the user Ui poses the query Qj, query similarity model. As our ranking function is of the linear weighted-sum type, it is important that the mechanism used for deriving this function captures the:

1. Significance associated with the user to each attribute, i.e., an attribute-weight and
2. Users emphasize on individual values of an attribute, i.e., a value weight.

IV. SIMILARITY MODEL FOR RANKING

The concept of similarity-based ranking is aimed at situations when the ranking functions are known for a small (and hopefully representative) set of user-query pairs. At the time of answering a query asked by a user, if no ranking function is available for this user-query pair, the proposed query- and user-similarity models can effectively identify a suitable function to rank the corresponding results.

A. *Query Similarity:*

We advance the hypothesis that if Q1 is most similar to query Qy (in U1's workload), U1 would display similar ranking preferences over the results of both queries; thus, the ranking function (F1y) derived for Qy can be used to rank N1. Similar to recommendation systems, our framework can utilize an aggregate function, composed from the functions corresponding to the top-k most similar queries to Q1, to rank N1. Although the results of our experiments showed that an aggregate function works well for certain individual instances of users asking particular queries, on average across all users asking a number of queries, using an individual function proved better than an aggregate function. Hence, for the reminder of the section, we only consider the most similar query (to Q1). We translate this proposal of query similarity into a principled approach via two alternative models: 1) query-condition similarity, and 2) query-result similarity.

1) Query-Condition Similarity

In this model, the similarity between two queries is determined by comparing the attribute values in the query conditions. Consider Example 1 and the queries. Intuitively, ―Honda" and ―Toyota" are vehicles with similar characteristics, i.e., they have similar prices, mileage ranges, and so on. In contrast, ―Honda" is a very different from ―Lexus." Similarly, ―Dallas" and ―Atlanta," both being large metropolitan cities, are more similar to each other than ―Basin," a small town.

2) Query-Result Similarity

In this model, similarity between a pair of queries is evaluated as the similarity between the tuples in the respective query results. The intuition behind this model is that if two queries are similar, the results are likely to exhibit greater similarity.

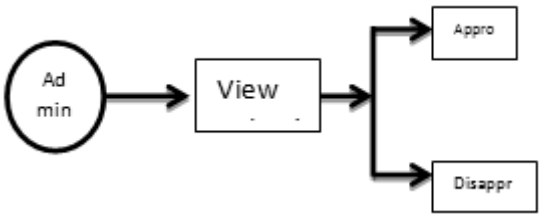

Fig. 1: Data Flow Diagram

B. *User Similarity:*

In our method for estimating user similarity, we have considered all the queries that are common to a given pair of users. This assumption forms one of our models for user similarity termed query-independent user similarity. However, it might be useful to estimate user similarity based on only those queries that are similar to the input query Q1. In other words, in this hypothesis, two users who may not be very similar to each other over the entire workload comprising of similar and dissimilar queries, may in fact, be very similar to each other over a smaller set of similar queries. We formalize this hypothesis using two different models—1) clustered, and 2) top-K—for determining user similarity. Before explaining these models, we would like to point out that given a workload, if no function exists for any query for a user, estimating similarity between that user and any other user is not possible. Consequently, no ranking is

possible in such a scenario. For the rest of the discussion, we assume that all users have at least one ranking function in the workload.

1) Query-Independent User Similarity

This model follows the simplest paradigm and estimates the similarity between a pair of users based on all the queries common to them.

2) Cluster-Based User Similarity

In order to meaningfully restrict the number of queries that are similar to each other, one alternative is to cluster queries in the workload based on query similarity. This can be done using a simple K-means clustering method. A well-established drawback of using a cluster-based alternative is the choice of K. In a database, a small value of K would lead to a large number of queries in every cluster, some of which may not be very similar to the rest, thus, affecting the overall user similarity. In contrast, a large value of K would generate clusters with few queries, and in such cases, the probability that there exist no users with any function in the cluster increases significantly.

3) Top-K User Similarity

Instead of finding a reasonable K for clustering, we propose a refinement, termed top-K user similarity. We propose three measures to determine top-K queries that are most similar to an input query (say Q1 from Example 1), and estimates the similarity between the user (U1) and every other user Strict top-K user similarity. Given an input query Q1 by U1, only the top-K most similar queries to Q1 are selected. However, the model does not check the presence (or absence) of ranking functions in the workload for these K queries.

V. EXPERIMENTAL EVALUATION

Evaluated each proposed model (query-similarity and user-similarity) in isolation, and then compared both these models with the combined model for quality/accuracy. And also they evaluated the efficiency of our ranking framework. Ideally, it would have preferred to compare our approach against existing ranking schemes in databases. However, what has been addressed in literature is the use of exclusive profiles for user-based ranking (the techniques for the same do not distinguish between queries) or the analysis of the database in terms of frequencies of attribute values for query-dependent ranking (which does not differentiate between users). for comparing with query-dependent ranking techniques, is difficult. Even, if it obtain ranking functions for different queries, all users will see the same ranking order for a given query. Thus, comparing such static ordering of tuples against our approach (that determines distinct ranking of tuples for each user and query separately) would not be a meaningful/fair comparison. Similarly, we felt that the comparing static user profiles (that ignore the different preferences of the same user for different queries) to our proposed definition of user similarity, for user-dependent ranking will not be fair. Hence, we have tried to compare the proposed user, query, and combined similarities to indicate the effectiveness of each model with respect to the other two models. Further, the context of recommendation systems is different from the one considered in this paper; hence, the direct application of these techniques for comparing against our framework was not feasible. We would also like to point out that unlike information retrieval, there are no standard benchmarks available and hence, we had to rely on controlled user studies for evaluating our framework.

A. MODULE

1) Registration
2) Login
3) Search and Read Articles
4) Rate articles
5) Write Articles and upload files
6) Sort the search results
7) Approval of Articles

1) Registration

The users must register in the system initially. The users will be provided with an user-id and password to login to the system. While registering, the details such as name, email-id, password, date of birth, gender are obtained from the user. All the fields are mandatory during registration. If the users miss any of the field, then the registration process will not be done. For achieving this Java Script validations can be done, such that error message that ―AlNecessary Fields are not entered" may be displayed on the user screen.

2) Login

The users must login to the system with the provided user-id and password. During login authentication process takes place. The user-id and password entered on the user screen is compared with the existing user-id and passwords in the Registration table of the database. If there is a match, then the user is authenticated and allowed to use the system. If there is a mismatch, then an error message that, ―Invalid User-Id or Password" will be displayed on the user screen. After successful login, the user is allowed to use the system.

3) Search and Read Articles

The users can search for any article on the system and read it. If the article consists of any download link to a file, then the users can download the article. The article can be of any category such as an organization, product, programming language, movie, book, institution, person or so. User must enter a search keyword in the search textbox, and click the search button. Then the keyword is compared with its existence in the Articles table of the database, and if match occurs, then the resulting articles are displayed on the user-screen.

4) Rate Articles

The users can rate the article that they read and downloaded. Rating can be given as 1 Star to 5 Stars. Based on the quality of the article or likeliness of the article, the users can rate the article. Such rating is stored on the database along with the article details. For every user, the rating is summed up.

5) Write Articles and Upload files

The users are allowed to write articles and upload files to the system. Such articles are sent to the administrator for approval. After approval, the articles are available for other users to search and download

6) Sort the Search Results

The users are allowed to sort the search result by various criteria. The users can sort the search result based on

1. Relevance (Default)
   The article which has more number of keywords from search Query will be displayed first.

2. Rating
   The article which got more rating will be displayed first.
3. Popularity
   The article which is read by most number of users will be displayed first.
4. User
   The articles written by a specific user are displayed.

7) Approval Articles

All the articles posted by the users will be sent to the administrator for approval. The administrator is allowed to read the full article and download the files if provided. The administrator is provided with two buttons, Approve and Reject.

Based on the content of article the administrator can approve or reject the articles. Only approved articles will be available for the users to search and read.

VI. WORKLOAD OF RANKING FUNCTIONS

Our proposed framework uses a workload of ranking functions derived across several user-query pairs. Since a ranking function symbolizes a users' specific preferences toward individual query results, obtaining such a function (especially for queries returning large number of results) is not a trivial task in the context of web databases. Furthermore, since obtaining ranking functions from users on the web is difficult (given the effort and time the user needs to spend in providing his/her preferences) determining the exact set of ranking functions to be derived for establishing the workload is important.

Probabilistic Data Distribution Difference

1. data are naturally distributed: ships and towers are at geographically separated locations
2. Ambiguity, errors, imprecise readings, and uncertainty are present in the real-time data collected, due to hazardous conditions, coarse real-time measurement and multiple readings for the same observation.
3. large amounts of data needs to be processed;

A. Proposed System

In the proposed system, all the disadvantages of the existing system are figured and rectified. Each user can upload file and write articles. Such articles are verified and approved by the administrators before they are available to other users in the system. Each user is given privilege to rate the contents available in the system. Such rating is useful for other users to analyse the quality of the article before they download them. Moreover, the user is given many options to sort the content of the search result. They users can sort the search result based on Popularity, Similarity, and Rating. If a user is interested in an article uploaded by a user say X, then the user is allowed to see all the articles written by the user X easily.

B. Advantages of Proposed System

1. Users can know the quality the content before they download them
2. Spam content is filtered before they appear in the system by the administrator. The ranking concept on the database based on the user and query similarity.

The solution for this similarity by using Top-k algorithm, using the k-nearest neighbouring algorithm to classify the data in the database in the form of cluster based on the user –query attributes. A drawback of using a cluster-based alternative is the choice of K. In database, a small value of K leads to a large number of queries in every cluster, the rest of the data are also include the similar to the user-query, without clustering affecting the overall similarity. In a large value of K would generate clusters with the minority queries, in such cases, the prospect that there exist no users with any utility in the cluster increases significantly. Workload generates the large amount of data in the database. Workload generate based on number of user enter and number of query in a specified time period. And also identify how much number of user access the same query, how much number of query access by the same user. Number of user access the same query and number of query access by the same user is used to generate the raking process based on the Top-k algorithm generating the workload to avoid the time taken to searching the user similarity and query similarity.

1. Query-Independent User Similarity
2. Cluster-Based User Similarity
3. Top-K User Similarity.

Based on the user and query similarity raking process is generated in workload. The raking based on the number of user access the same query in the specified time. This ranking process identifies the user accessing particular attributes in database.

VII. CONCLUSION

In this, proposed a user- and query-dependent solution for ranking query results for databases. We formally defined the similarity models (user, query, and combined) and presented experimental results over databases to support an analysis. We demonstrated the practicality of our implementation for real-life databases. Further, we discussed the problem of establishing a workload, and presented a learning method for inferring individual ranking functions. Our work brings forth several additional challenges. In the context of databases, an important challenge is the design and maintenance of an appropriate workload that satisfies properties of a similarity-based ranking. Determining techniques for inferring ranking functions over databases is an interesting challenge as well. Another interesting problem would be to combine the notion of user similarity proposed in work with existing user profiles to analyse if ranking quality can be improved further. Accommodating range queries, usage of functional dependencies and attribute correlations needs to be examined. Applicability of this model for other domains and applications also needs to be explored; this will improve the product developments in the other company.

REFERENCES

[1] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis, ―Automated Ranking of Database Query Results,‖ Proc. Conf. Innovative Data Systems Research (CIDR), 2003.

[2] S. Amer-Yahia, A. Galland, J. Stoyanovich, and C. Yu, ―From del.icio.us to x.qui.site: Recommendations in Social Tagging Sites,‖ Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 1323-1326, 2008.

[3] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval. ACM Press, 1999.

[4] M. Balabanovic and Y. Shoham, ―Content-Based Collaborative Recommendation,‖ Comm. ACM, vol. 40, no. 3, pp. 66-72, 1997.

[5] J. Basilico and T. Hofmann, ―A Joint Framework for Collaborative and Content Filtering,‖ Proc. 27th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval, pp. 550- 551, 2004.

