2014 Fifth International Conference on Signal and Image Processing

Similarity Based Ranking of Query Results From Real Web Databases

Harish Kumar B T
Dept. CS&E
BIT
Bangalore, India

Deepa Chowdary
Dept. CS&E
BNMIT
Bangalore, India

Vibha L
Dept. CS&E
BNMIT
Bangalore, India

Venugopal K R
Principal
UVCE
Bangalore, India

L M Patnaik
II Sc.
Bangalore, India

E-mail: harish_bit82@yahoo.com

Abstract— ***The information available in the World Wide Web is stored using many real web databases (e.g. vehicle database). Accessing the information from these real web databases has become increasingly important for the users to find the desired information. Web users search for the desired information by querying these web databases, when the number of query results generated is large, it is very difficult for the web user to select the most relevant information from the large result set generated. Users today, have become more and more demanding in terms of the quality of information that is provided to them while searching the web databases. The most common solution to solve the problem involves ranking the query results returned by the web databases. Earlier approaches have used query logs, user profiles and frequencies of database values. The problem in all of these techniques is that ranking is performed in a user and query independent manner. This paper, proposes an automated ranking of query results returned by web databases by analyzing user, query and workload similarity. The effectiveness of this approach is discussed considering a vehicle web database as an example.***

Keywords - Database; Ranking; Similarity; Spearman's coefficient; Workload

I. INTRODUCTION

Databases are searched by forming structured query language (SQL). All these database systems are based on boolean query model i.e., SQL based database, which generates all the tuples that satisfy the conditions in the query or empty tuples if the condition is not satisfied. SQL based databases fails to handle the following cases.

1. Zero result set case
2. Large result set case

A. Zero Result-set Case

The structured query language formulated by the user is very choosy (discriminating). The number of results returned by the web databases in this case may be empty or NULL. In such circumstances it is necessary to generate a ranked list of closely related tuples instead of generating zero result set.

B. Large Result-set Case

The structured query language formulated by the user is not very choosy, (discriminating) the number of results returned by the web databases may be very large. In this situation, it is necessary to rank and order the closely related tuples that best matches the user need. In this work, a user, query and workload similarity model is proposed for automated ranking of query results. The following three scenarios are used as current examples.

Example 1: Two users- a developer (*U1*) and sales executive (*U2*), put-forth the same query (*Q1*): Make="Toyota AND Location=Bangalore". The user (*U1*) typically searches for vehicles with specific color choices (e.g. only black colored vehicles) to be ranked and displayed at the top of search results. Similarly, *U2* would likely to search for vehicles with specific price. Hence, for *U2* vehicles with price<10,000$ should be displayed before the rest.

Example 2: The same user (*U2*) moves to a company for an internship and puts-forth a different query (*Q2*): "Make=Honda and Location=Delhi". Since the user has secured an internship willing to pay higher price for a lesser mileage vehicle. He would prefer vehicles with mileage<10000 to be ranked higher than others.

Example 3: Two workloads say *W1* and *W2* where W1 contains "Make = Volvo, Price = 5000$, Mileage = 10000, Location = Bangalore AND Color=RED" and *W2* contains Make=Volvo, Color=White, Price=15000$. If there exists large number of workloads similar to *W1* then the workload *W1* should be displayed before the other.

Motivation: The number of results returned by the web databases will be empty if the user is too choosy and many results if the user is not choosy. Hence an automated ranking of query results yields in an optimal solution using similarity models.

Contribution: In the present work, the query results are ranked by analyzing the user, query and workload similarity. This helps in faster retrieval of information from the web database and eliminates the zero result set problem.

The remainder of the paper is organized as follows – Section II gives the overview of the related work. Section III presents the architecture modeling. In section IV problem definition is discussed. Similarity models are presented in section V. Section VI contains the algorithms of the similarity models. Performance results are analyzed in section VII. Section VIII contains conclusion.

II. RELATED WORK

A brief survey of related work in the area of database systems despite of their efficient data management capabilities they fail to effectively handle the information retrieval queries from many emerging applications. These emerging applications pose the following data management challenge.

A. Boolean Based Query Processing

This model based on set theory and Boolean algebra is one of the traditional model for information retrieval. The query is a Boolean algebra expression using connectives. The documents retrieved are the documents that completely match the given query. Partial matches are not retrieved, the retrieved set of documents is not ordered. For each term in the query, a list of documents that contain the term is created then the lists are merged according to the Boolean operators. Boolean model is still widely used in small scale searches like searching emails, files from local hard drives or in a mid-sized library. The retrieval strategy is based on binary criteria so, partial matches are not retrieved. Only those documents that exactly match the query are retrieved. Hence, to effectively retrieve from a large set of documents users must have good domain knowledge to form good queries. The retrieved documents are not ranked. Given a large set of documents, say, at web scale, the Boolean model either retrieves too many documents or very few documents.

J. Basilico and T. Hofmann in [1] proposed a ranking model for database tuples in a query- and user-independent framework. This model relies on the availability of a workload of queries spanning all attributes and values to establish a score for a tuple. A drawback of such a workload is that in the context of web databases, user queries are restricted to a subset of the attributes that are displayed in the results. In such a setting, the workload will fail to capture user preferences towards those attributes and values that cannot be specified in the query. In contrast, we capture these preferences via users browsing choices in a query- and user-dependent setting.

M. K. Bergman in [2] presented a model for query-dependent ranking which analyzes the relationship between the query results and the tuples in the database. The drawback of this work lies in the fact that it requires the knowledge of the complete underlying database at all times to rank query results, an improbable setting for web databases that dynamically obtain data from a slew of individual sources. In contrast, we establish query-dependent ranking by analyzing the user's browsing choices and comparing different queries in terms of their similarity with each other without requiring full knowledge of the web database.

Context preferences for user-dependent ranking have been proposed however, these models require the user to specify an order/preference for the tuples in the absence of a specific query from which a global ordering across the database is obtained. In [3] and [4] the SQL query language is extended to allow the user to specify the ranking function according to their preference for the attributes.

S. Chaudri et al., in [5] defined a user relevance feedback and is employed to learn the similarity between a result record and the query for ranking in relational multimedia databases. All these approaches require considerable user input and are an arduous task for web users who have no clear idea how to assign order to tuples and/or attributes. In contrast, this work relies purely on users browsing choices that reveal their implicit ranking preferences without requiring them to have an excessive interaction with the system.

S. Gauch and M. Speretta in [6] focused on query similarity which has been widely studied in Information Retrieval and Collaborative Filtering but the database queries involving multiple combinations cannot be directly compared like IR-keyword queries.

M. Balabanovic and Y. Shoham in [7] provided an intuitive mechanism for establishing user similarity based on profiles. It involves the use of domain experts in addition to learning models to derive this similarity. Alternatively, we propose a mechanism to capture user similarity by analyzing the relationship between the users' past browsing choices.

T. Holfmann in [8] discussed the use of learning methods for deriving ranked lists which has been studied extensively in Machine Learning and Image Processing similar to these techniques; we propose a probabilistic learning method for capturing attribute preferences for web queries. Our results show that within the framework that we tested, our proposed model performed better than existing bayesian or regression models.

Chaudhuri et al. in [10] address the problem of query-dependent ranking by adapting the vector model from information retrieval. However, for a given query, these techniques provide the same ordering of tuples across all users.

The work proposed in [11] requires the user to specify an ordering across the database tuples without posing any specific query from which a global ordering is obtained for each user. A drawback in all these works is that they do not consider that the same user may have varied ranking preferences for different queries. In contrast, our framework provides an automated query- as well as user-dependent ranking solution without requiring users to possess knowledge about query languages, data models and ranking mechanisms.

The problem of integrating the information retrieval system and database systems have been attempted in [12] with a view to apply the ranking models to derive the similarity; however, the intrinsic difference between their underlying models is a major problem.

III. ARCHITECTURE MODELING

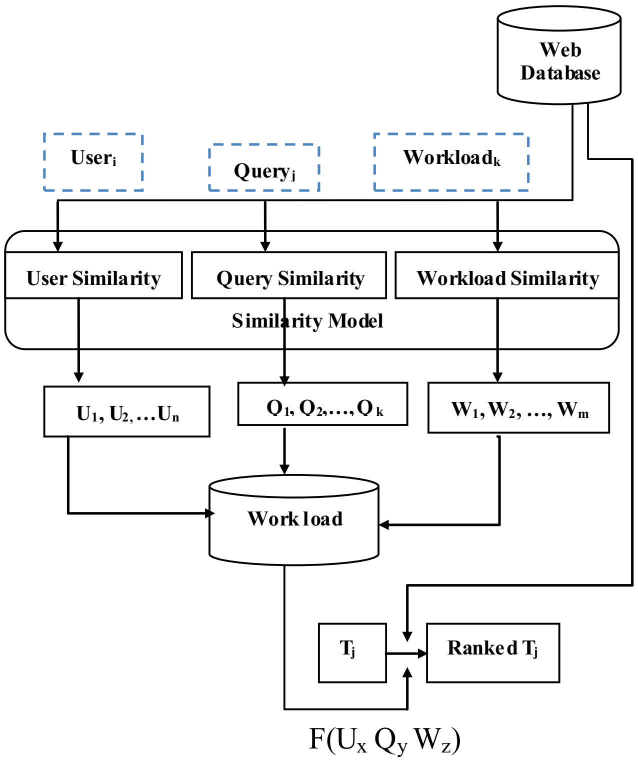

Figure 1. Similarity Model for Ranking

The core component of ranking framework is the similarity model as shown in Fig 1. The similarity model contains three models such as user, query and workload. When the user U_i enters the query Q_q the query similarity model determines the set of queries (Q_q, Q_1, Q_2...Q_p) most similar to Q_q. Similarly the user similarity model determines the set of users (U_p, U_1,U_2... U_v) most similar to U_i and the workload similarity model finds the set of workloads (W_r, W_1, W_2..... W_x) most similar to W_r. By using these three ordered set of similar user, queries and workload, it derives the ranking function $F(U_xQ_yW_z)$, such that the combination of U$_x$, Q$_y$ and W$_z$ is most similar to U_i, Q_q and W_r. This function is then used to rank the Q_j's results for U_i.

IV. PROBLEM DEFINITION

Given a web database table *D* over a set of *K* attributes *A={A1, A2, ... ,Ak}*, an user *U* can put-forth a query *Q* in the form *"SELECT * FROM D WHERE X1=V1 AND Xs=Vs"* where each *Xi € A* and Vi is value in its domain. Let *N={t1, t2, ..., tn}* be the set of result tuples for query *Q*. The query *Q* may generally result in zero tuples if the user *U* is too choosy in writing the query or may result in large number of tuples if the user *U* is not choosy. Hence a similarity based ranking of query results is proposed, whose objectives are:

- Faster retrieval of query results.
- Elimination of zero result set problem.
- Ranking of query results.

V. SIMILARITY MODEL GENERATION

In the proposed work similarity model is generated by considering the following different types of similarities.

A. User Similarity

The user similarity models aims to determine the similar users in a web database. This model is based on the attribute conditions such as location, profession and age. Two users U_1 and U_2 are said to be similar, if they have same values for the attributes such as location, profession and age. Some weightages will be assigned to the attributes such as for location, profession and age. The attributes profession and age are optional.

B. Query Similarity

The goal of this model is to determine the queries which are more similar to each other. The model is based on price, mileage, make and color. Two queries Q_1 and Q_2 are said to be similar if they have same price, mileage, make and color. The concept of weightage is used in order to determine the similar queries. Some values will be assigned to the attributes such as price, mileage, make and color. When two queries have the same weightages then those two queries are similar.

C. Ranking Function

By analyzing the user's browsing choices in a web database; the ranking function is derived which is a user-query pairs from the workload *W*. There should be at least one ranking function in the workload table which corresponds to the given input query entered by the user. The ranking framework derives the appropriate ranking function exists in the workload *W*.

D. Workload Generation

The workload *W* consists of the ranking function derived across several user-query pairs. The queries Q_1, Q_2, ... Q_n

entered by the user U_1, U_2, ... U_n are stored in the form of ranking function such as $f_{11}, f_{12}, \dots f_{mn}$. The users preference towards the specific attribute condition, specified in the query are generated as ranking function which is derived from the workload.

E. Workload Similarity

The objective of this work is to determine the workloads in terms of similarity with each other. In this model, two workloads are compared based on queries entered by the users. The workloads *W1* and *W2* are considered as similar if they have same location, make, price, colour status and so on.

F. Composite Similarity

When large number of users and queries are involved in a web database, then determining the similar users, queries may not be the best solution. In such situations the composite similarity model can be used, which is the combination of user and query similarity models.

G. Correlating Using Spearman's Rank Coefficient

The Spearman's rank coefficient is given by the following equation.

$$1- [6\sum d^2 / n\ (n^2-1)] \qquad (1)$$

Where, *d* is difference between the two numbers in each pair of ranks and *n* is the total number of pairs of data. The Spearman's coefficient can vary between -1 and 1. Interpretation of result is as follows.

- Close to -1 → Negative correlation
- Close to 0 → No linear correlation
- Close to 1 → Positive correlation

Correlation between different results and the quality of ranking is determined using the Spearman's rank coefficient given in equation (1).

VI. ALGORITHMS

The different algorithms proposed in this work are:

A. User and Query Similarity

TABLE I
Algorithm to find user and query similarity

```
Input:   1. User U with attributes location (L),
            Profession (P) and age (A).
         2. Query Q with attributes price (PR), make (MK),
            mileage (ML), color (C)
         3. Workload W containing ranking function.
Output:  U_kset: Top K similar users
         Q_kset: Top k similar queries
         /*User Similarity*/
         WEIGHT_USER[N]: Array of size N
         WEIGHT=0: Variable
Step1:   for i = 1 to N do /*N = Total number of users*/
             /* Determine the user similarity as U(L,P,A)=U_i' (L,P,A) */
             if(U(L)== U_i'(L))
                 WEIGHT=WEIGHT+80;
             if(U(P)== U_i'(P))
                 WEIGHT=WEIGHT+10;
             if(U(A)== U_i'(A))
                 WEIGHT=WEIGHT+10;
             WEIGHT_USER[i]=WEIGHT;
         end for
Step2:   Sort WEIGHT_USER[i] in descending order

Step3:   Ukset=Select top K users from WEIGHT_USER[ ]
         /*Query Similarity*/
         WEIGHT_QUERY[N]: Array of size N
         WEIGHT=0: Variable
Step4:   for i = 1 to N do /*N = number of queries*/
             /* Determine the query similarity as
             Q(PR,MK,ML,C)=Q_i' (PR,MK,ML,C) */
             if(Q(PR)== Q_i'(PR))
                 WEIGHT=WEIGHT+30;
             if(Q(MK)== Q_i'(MK))
                 WEIGHT=WEIGHT+30;
             if(Q(ML)== Q_i'(ML))
                 WEIGHT=WEIGHT+30;
             if(Q(C)== Q_i'(C))
                 WEIGHT=WEIGHT+10;
             WEIGHT_QUERY[i]=WEIGHT;
         end for
Step5:   Sort WEIGHT_QUERY[i] in descending order

Step6:   Q_kset=Select top K queries from
         WEIGHT_QUERY[ ]
```

The user and query similarity algorithm determines similar users by scrutinizing *N* users in a web database. The algorithm for finding similar user and query is as shown in TABLE I. The variables *L, P, A* denote the attributes location, profession and age of a user *U* entering the query *Q*. Values of these attributes are compared with the values of the users U_i' already exists in the database. *PR, MK, ML, C* are the variables used to denote the attributes price, make, mileage, color of the query *Q* entered by user *U.* The values of these attributes are compared with the values of the query Q_i' already existing in the database.

B. Ranking Function

TABLE II
Algorithm for deriving ranking function

```
Input:   U_kset and Q_kset
Output:  F_mn : Ranking function
Step 1:  for each U_m € U_kset do
Step 2: for each $Q_n$ € $Q_{kse}$t do
          rank ($U_m$€ $U_{kset}$) = Index of $U_m$ in $U_{kset;}$
          rank ($Q_n$ € $Q_{kset}$) = Index of $Q_n$ in $Q_{kset;}$

Step 3: Rank ($U_m$, $Q_n$) = rank ($U_m$€ $U_{kset}$) + rank
          ($Q_n$ € $Q_{kset)}$
Step 4: end for
Step 5: end for
Step 6: $F_{mn}$= Get_Ranking_Function
```

The algorithm for deriving the ranking function is as shown in TABLE II. It checks whether there are any user U_m and query Q_n in the U_{kset} and Q_{kset}. It combines the user and query pair and derives the ranking function F_{mn} to rank the query results.

C. *Workload Generation*

TABLE III
Algorithm for Generation of Workload

```
Input:  Users $U_m$ and query $Q_n$ from Database D
Output: Workload W Containing ranking function $F_{mn}$

Step 1: Determine $U_m$ and $Q_n$ in a Database D
Step 2: Determine user preferences towards the query Q
Step 3: Generate the user-query pair i.e $F_{mn}$
```

The workload can be generated using the algorithm as shown in TABLE III. All the *'m'* users and *'n'* queries are examined to generate the ranking function F_{mn} and stored in the workload *W*.

D. *Workload Similarity*

TABLE IV
Algorithm to find Workload Similarity

```
Input:  Ranking function $F_{mn}$
Output: Gives the most similar ranking functions in W
Step 1: for each $F_{mn}$ € W do
Step 2:  Determine the similarity W and W'
Step 3: End for
Step 4: Sort ($F_{11}$.....$F_{mn}$)
```

The workload similarities establish the ranking functions which are most similar to each other. To find the similar ranking function in a workload *W* scan all the existing ranking function and compare it with other. Suppose there are two ranking functions such as F_{11} and F_{12}, these are similar if they have same values for the attributes such as location, price, make, mileage, color etc. The algorithm for finding similar workload is as indicated in TABLE IV.

VII. IMPLEMEMTATION AND PERFORMANCE ANALYSIS

The quality of ranking using similarity models is obtained by observations performed on ten different workloads. The ranking quality of user similarity model is as shown in Figure 2.

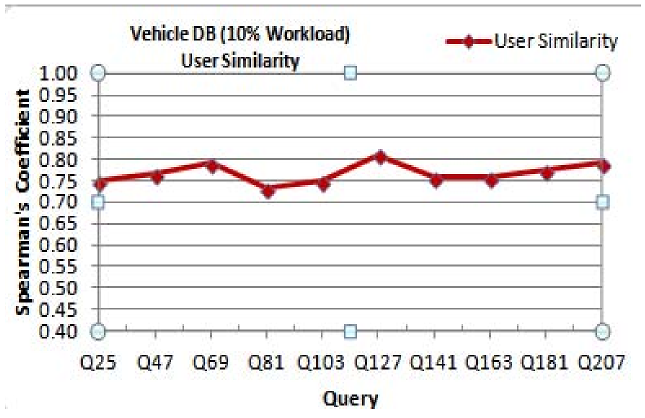

Figure 2. Ranking Quality of User Similarity

The ranking quality of query similarity is as shown in Figure 3. The user similarity model has the spearman's coefficient as 0.75 and the query similarity model has 0.70. Hence, the user similarity performs better than the query similarity model. This is because in web database there will be more number of similar users than the similar queries.

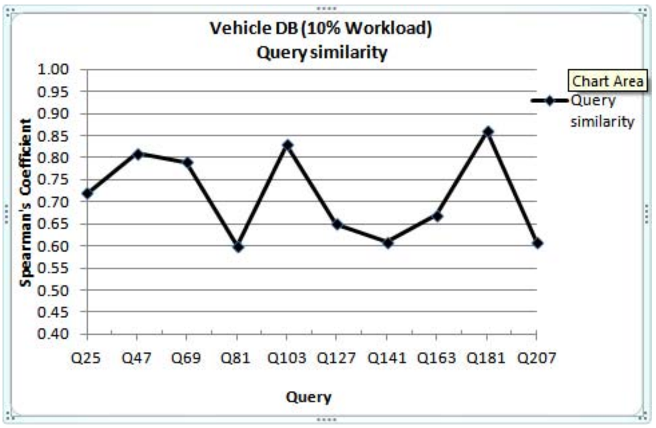

Figure 3. Ranking Quality of Query Similarity

The composite model performs better than the user and query similarity models which has the spearman's coefficient as 0.85 as shown in Figure 4.

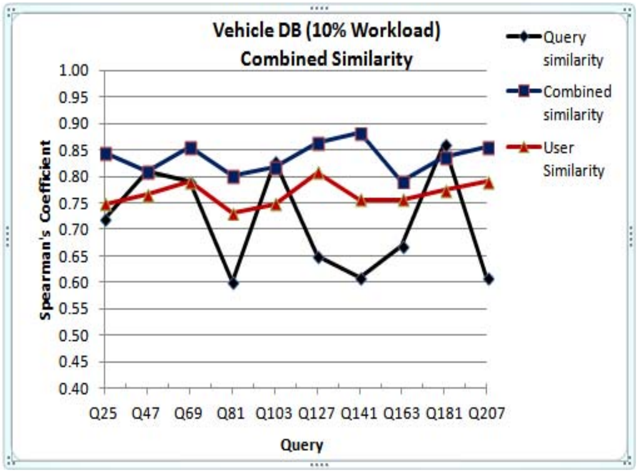

Figure 4. Ranking Quality of Composite Similarity

Comparing ranking quality achieved by the composite similarity model and the workload similarity model, the workload similarity model performs better than the composite similarity model. The workload similarity model has the spearman's coefficient 0.88 as shown in Figure 5.

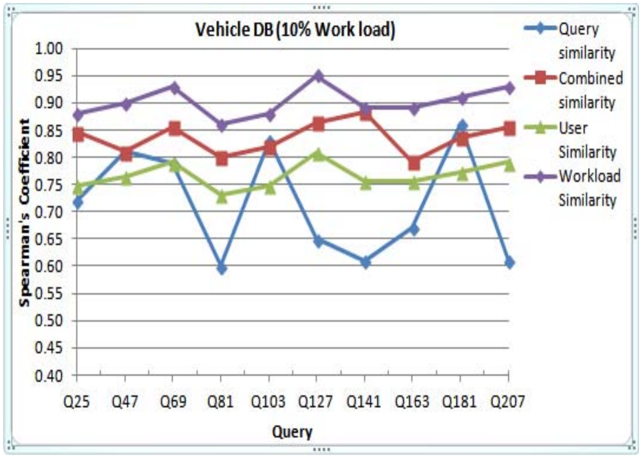

Figure 5. Ranking Quality of Workload Similarity

The goal of performance analysis is to determine whether the ranking framework can be incorporated into real world applications. The performance is evaluated by average time taken in seconds to perform ranking. The performance analysis is as shown in Figure 6. The composite similarity model takes 0.047 seconds to return the ranked results where as the workload similarity model takes 0.016 seconds which is less compared to the existing similarity models.

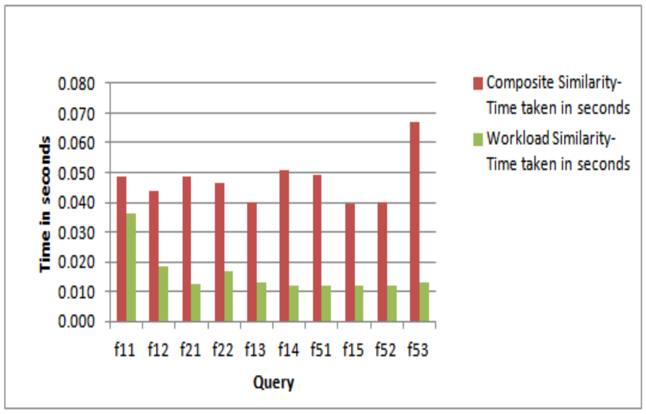

Figure 6. Performance Analysis

VIII. CONCLUSION

Similarity based ranking of the query results from real web databases provides the solution to the zero result set and many result set problems. Ranking of the query results is done in user and query dependent manner by analyzing the user, query, workload and composite similarity. The present work demonstrates the quality of ranking the query results by using the spearman's constant.

REFERENCES

[1] J. Basilico and T. Hofmann, “A Joint Framework for Collaborative and Content Filtering” *Proc. 27th Ann. International ACM SIGIR Conf. Research and Development in Information Retrieval,* pp. 550-551, 2004.

[2] M. K. Bergman, “The Deep Web: Surfacing Hidden Value”, J. Electronic Publishing, *vol. 7, no. 1,* pp. 41-50, 2001.

[3] 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.

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

[5] S. Chaudhuri, G. Das, V. Hristidis and G. Weikum, “Probabilistic Information Retrieval approach for Ranking of Database Query Results”, *TODS, vol. 31, no. 3,* pp. 1134–1168, 2006.

[6] S. Gauch and M. Speretta, “User Profiles for Personalized Information Access,” *Adaptive Web,* pp. 54-89, 2007.

[7] M. Balabanovic and Y. Shoham, “Content-Based Collaborative Recommendation”, *Comm. ACM, vol. 40, no.3* pp. 66-72, 2007.

[8] T. Hofmann, “Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis,” *Proc. 26th Ann. Int’l ACM SIGIR Conf. Research and Development in Information Retrieval,* pp. 259-266, 2003.

[9] K. Werner, “Foundations of Preferences in Database Systems,” *in VLDB. VLDB Endowment,* pp. 311–322, 2002

[10] C. Li, M. Soliman, K. C.-C. Chang, and I. Ilya, “Ranksql: Supporting ranking queries in relational database management systems,” *in VLDB*, pp. 1342–1345, 2005.

[11] G. Koutrika and Y. E. Ioannidis, “Constrained Optimalities in Query Personalization,”*in SIGMOD Conference*, pp. 73–84, 2005.

[12] Agrawal, S. Chaudhuri,G. Das, and A. Gionis, “Automated Ranking of Database Query Results,” *in CIDR,* 2003.

[13] W. Su, J. Wang, Q. Huang, and F. Lochovsky, “Query Result Ranking Over E-commerce Web Databases,” *in CIKM*, pp. 575–584, 2006.

[15] R. Agrawal , R. Rantzau , and E. Terzi , “Context-sensitive Ranking,” *in SIGMOD Conference. New York, NY, USA: ACM,* pp. 383–394, 2006.

[16] S.R.F.D. Retrieval, “Supporting Ranking for Data Retrieval,” *Ph.D.dissertation University of Illinois, Urbana Champaign,* 2005.

[17] G. Agarwal, N. Mallick, S. Turuvekere, and C. Zhai, “Ranking Database Queries with User Feedback: A neural network approach,” *in DASFAA*, pp. 424–431, 2008.

[18] M. Ortega-Binderberger, K. Chakrabarti, and S. Mehrotra, “An Approach to Integrating Query Refinement in Sql,” *in EDBT. Springer-Verlag*, pp. 15–33, 2002.

[19] K. Chakrabarti, K. Porkaew, and S. Mehrotra, “Efficient Query Refinement in Multimedia Databases*,” in ICDE*, p. 196, 2000.

[20] L. Wu, C. Faloutsos, K. P. Sycara, and T. R. Payne, “Falcon: Feedback Adaptive Loop for Content-Based Retrieval,” in *VLDB,* pp. 297–306, 2000.

[21] B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, and S. Sudarshan,“Banks: Browsing and Keyword Searching in Relational Databases,” *in VLDB, no.1083-1086,* 2002.

[22] G. Koutrika and Y. E. Ioannidis, “Personalization of Queries in Database Systems,” *in ICDE*, pp. 597–608, 2004.

[23] H. Yu, Y. Kim, and S. won Hwang, “Rv-svm: An efficient Method for Learning Ranking svm,” *in PAKDD*, pp. 426–438, 2009.

[24] X. Jiang, L.-H. Lim, Y. Yao and Y. Ye , “Learning to Rank with Combinatorial Hodge Theory,” *CoRR,* 2008.