User and Query-Dependent Ranking for Web Databases using K-D Tree

Jemish Patel[1], Shilpa Sherasiya[2]

[1]M.E., Computer Engineering, Kalol Institute of Technology & Research Center, Gujarat, India

[2]Asst. Prof., Computer Engineering, Kalol Institute of Technology & Research Center, Gujarat, India

ABSTRACT

Query dependent ranking has become a routine task as the requirement of searching the various values of web database has grown. Requirement of searching electronic product, car, real estate and other products has emerged with the hike of internet use. Earlier approach for addressing this problem have used frequencies for database values, query log and user profile. A common thread in most of this approach is that ranking is done in a user- and/or query– independent manner. Our research has focused on user and query dependent approach for getting the result from various datasets. Here, a query dependent ranking model is presented where functionalities related to workload depends on the several ranking strategies like K-D Tree, K-Mean or K-NN algorithm. We demonstrate a comparative studies of result with similar queries. We defined a similarity formally in alternative ways to discuss the effectiveness analytically and experimentally over two district database.

Keyword: *Web databases, user similarity, query similarity, workload, Automated ranking.*

1. INTRODUCTION:

Data mining is used to extract the information from any system by analysing the present in the form of data. In this paper author focuses on the problem of frequent pattern mining. It became a routine task, for searching web databases in domains such as vehicles, real estate, etc., one of the problems in this context is ranking the results of a user query. Earlier approaches for addressing this problem have used frequencies of database values, query logs, and user profiles. A common thread in most of these approaches is that ranking is done in a user and query independent manner. We present a ranking model, based on two complementary notions of user and query similarity, to derive a ranking function for a given user query. This function is acquired from a sparse workload comprising of several such ranking functions derived for various user-query pairs. To finding a number of databases on the Web for airline reservations, find a car, screening properties. Usually, these databases are obtained by blending the query criteria in the schema attribute. If the number of results returned is large, in order (s), further investigation is a more useful time to view the answer, you want to select. Currently, by displaying the sort query results by the value of a single attribute database on the Web, to simplify this task, for example, price, and mileage, however, Web most users, the obtained command using the values of multiple attributes will be close to its expected to prefer.

Example 1. Two users—a company executive (U1) and a student (U2), seek answers to the same query (Q1): "Make ¼ Honda AND Location ¼ Dallas; TX," for which more than 18,000 tuples are typically returned in response. Intuitively, U1 would typically search for new vehicles with specific color choices (e.g., only red colored vehicles), and hence would prefer vehicles with "Condition ¼ New AND Color ¼ Red" to be ranked and displayed higher than the others. In contrast, U2 would most likely search for old vehicles priced under a specific amount (e.g., "Price < 5; 000$"); hence, for U2, vehicles with "Condition ¼ Old AND Price < 5; 000$" should be displayed before the rest.

Example 2. The same student user (U2) moves to Google for an internship and asks a different query (say Q4): "Make ¼ Pontiac AND Location ¼ Mountain View; CA." We can presume (since he has procured an internship) that he may be willing to pay a slightly higher price for a lesser mileage vehicle (e.g., "Mileage < 100; 000"), and hence would prefer vehicles with "Condition ¼ Old AND Mileage < 100; 000" to be ranked higher than others.

Example 1 illustrates that different web users may have contrasting ranking preferences toward the results of the same query. Example 2 emphasizes that the same user may display different ranking preferences for the results of different queries. Thus, it is evident that in the context of web databases, where a large set of queries given by varied classes of users is involved, the corresponding results should be ranked in a user- and query-dependent manner.

Contributions. The contributions of this paper are:

1. We propose a user- and query-dependent approach for ranking query results of web databases.
2. We 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. We present experimental results over two web databases supported by Google Base to validate our approach in terms of efficiency as well as quality for real-world use.
4. We present a discussion on the approaches for acquiring/generating a workload, and propose a learning method for the same with experimental results.

2. PROBLEM DEFINITION AND ARCHITECTURE:

2.1 Problem Definition:

Consider a web database table D over a set of M attributes, A ¼ fA1;A2; . . .; Am. A user Ui asks a query Qj of the form: SELECT * FROM D WHERE A1 ¼ a1 AND …. AND As ¼ as, where each Ai 2 A and ai is a value in its domain. Let Nj ¼ ft1; t2; . . . ; tn be the set of result tuples for Qj, and W be a workload of ranking functions derived across several user query pairs (refer to Tables 1 and 2 for an example). The ranking problem can be stated as: "For the query Qj given by the user Ui, determine aranking function FUiQj from W." Given the scale of web users and the large number of queries that can be posted on D, We will not possess a function for every user-query pair; hence, the need for a similarity-based method to find an acceptable function (FUxQy) in place of the missing FUiQj. The ranking problem, thus, can be split into:

1. Identifying a ranking function using the similarity model: Given W, determine a user Ux similar to Ui and a Query Qy similar to Qj such that the function FUxQy exists in W.
2. Generating a workload of ranking functions: Given a user Ux asking query Qy, based on Ux's preferences toward Qi's results, determine, explicitly or implicitly, a ranking function FUxQy .W is then established as a collection of such ranking functions learned over different user-query pairs.

2.2 Ranking Architecture:

The Similarity model (shown in Fig. 1) forms the core component of our ranking framework. When the user U_i poses the query Q_j, the query-similarity model determines the set of queries ($fQ_j;Q_1;Q_2; . . .;Q_p$) most similar to Q_j.

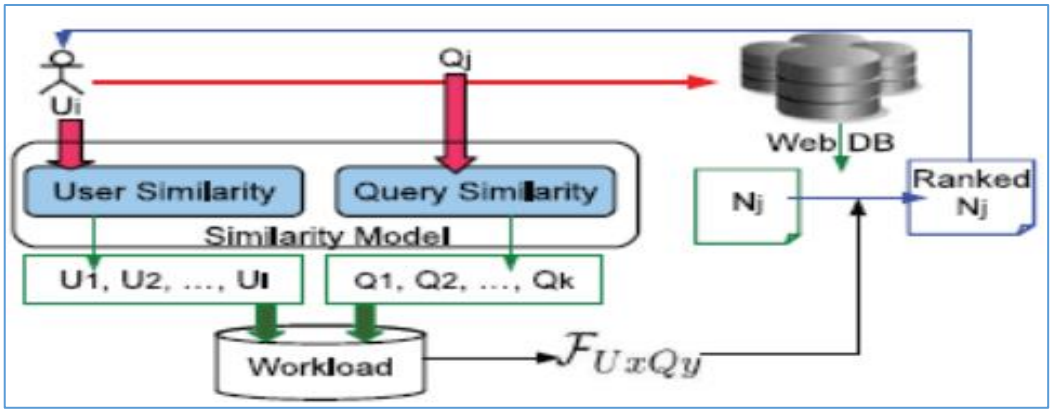


Fig. 1. Similarity model for ranking.

The above component of ranking framework is the similarity model. The set of users ({U_1, U_2 ...Ur}) most similar to U_i, determined by the user similarity model the query similarity model determines the set of queries ({Q_1, Q2... Qk}) most similar to Q_j. Using these similar queries and users, it searches the workload to identify the function FU_xQ_y. The ranking functions for several user-query pairs are formed from the workload used in this framework, the ranking function is of the linear weighted-sum type. The mechanism used for deriving this function captures the: i) significance associated by the user to each attribute i.e., an *attribute-weight* and ii) user's emphasis on individual values of an attribute i.e., a *value-weight*.

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) user's emphasis on individual values of an attribute, i.e., a value weight.

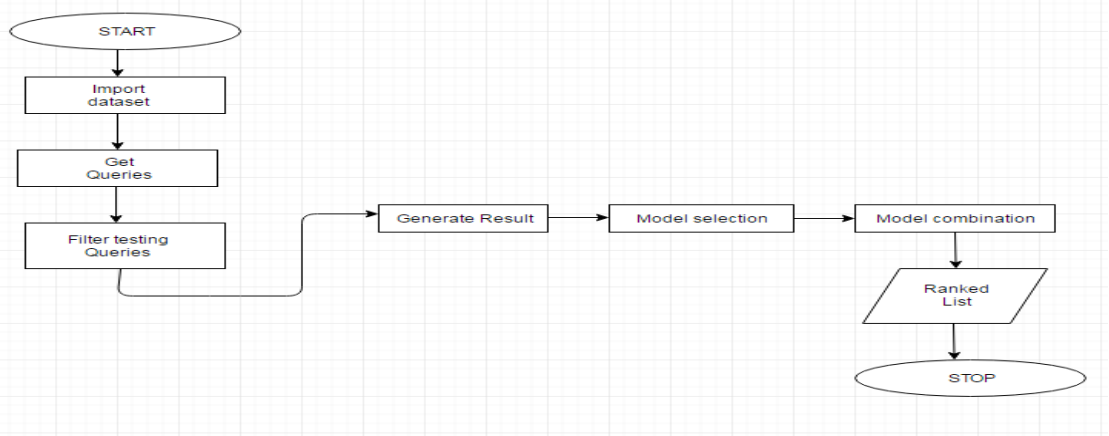
$$tuple\ score(t) = \sum_{i=1}^{m} w_i * v_i,$$

These weights can then be integrated into a ranking function Fxy to assign a tuple score to every tuple t inNy using tuple scoreðtÞ ¼ where we represents the attribute-weight of Ai and vi represents the value-weight for Ai's value in tuple t. The workload W is populated using such ranking functions. Tables 1 and 2 show two instances of the workload (represented in the form of a matrix of users and queries). Cell [x,y] in the workload, if defined, consists of the ranking function Fxy for the user-query pair Ux and Qy.

3. PROPOSED WORK:

3.1 Flow of Proposed work:

- We propose a user and query dependent approach for ranking query results of Web databases.
- Develop a ranking model based on two complementary measures of query similarity and user similarity and user-query pairs.
- To present experimental results over two Web databases supported by Google Base to validate our approach in terms of efficiency as well as quality for real-world use.
- The time complexity of online and offline processing will be reduced if KD-Tree for nearest neighbor search.
- Will try to implement it with Euclidean distance as the metric in KNN method because it is fastest method.
- Ranking will be compared for performance based on start time and end time for executing query.
- For finding the value of K for clustering following top-K similarity can be used.
  1. Strict top-K uses similarity.
  2. Uses based top-K similarity.
  3. Workload based top-K used similarity.



Steps for flow of proposed work:

1. Query represented by query taken from database.
2. Filtration for testing queries.
3. Call K-D & 2-D node class for model construction.
4. For model selection we use KD tree.Java
5. Combine KD-tree using step 2 and 4.
6. Display result of KD-tree with nearest neighbour and center neighbour.

3.2 Proposed Algorithm:

Algorithm [1]

Input: Attribute A_i and its value range Web database D with the total number of tuples | D | as n and Total number of tables T
Output: A query with ranking stored in an array H_r.
Method:
For each table T with n Tuples
If
A_i is a categorical attribute
For each category aij of A_i, probe D using a query with condition "$Ai=a_{ij}$"
Q_r = Kd_Tree (T,n,A_i); // to rank attributes for query get its occurrence count c
Add an element (a_{ij}, c) into H_r
If Ai is a numerical value attribute with value range (a_low, a_up)
t = |D| / n
low = a_low, up = a_up
Do
probe D with a query with condition "low $\leq A_i <$ a_up
Qr = KD _ Tree (T ,n , A_i);
get its occurrence count c
If c $\leq$ t
Add an element (low, up, c) into Hr
low =a_up, up = a_up
else
up = low + (up - low) / 2
While low < a_up
End For
End For
Return Hr

Algorithm [2]

INPUT: U_i, Q_j, Workload W (M queries, N users), T_s , T_e
OUTPUT: Ranking Fun F_{xy} to be used for U_i, Q_j, Response time
STEP 1:
T_s = calc_time (); //Start time (current time)
for p = 1 to M do
Calculate Query Condition Similarity (Q_j, Q_p) with functional dependencies
end for // Based on descending order of similarity with Q_j
Sort(Q_1, Q_2, …….Q_M)
Select Q_K set i.e, top-K queries from the above sorted set
STEP 2:
for r = 1 to N do // User Similarity (U_t, U_r) with User profile
Q_K set
end for // Based on descending order of similarity with U_t
Sort (U_1, U_2, …..U_N) to yield U set
STEP 3:
for Each Q_s in Q_K set do
for Each U_t in U_s set do
Rank (U_t, Q_s) = Rank (U_t £ U_s set) + Rank (Q_s £ Q_K set)
end for
F_{xy} = Get-Ranking Function ()
T_e = time (); //end time (current time)
Response time = T_e - T_s;

4. EXPERIMENTAL WORK AND RESULT:

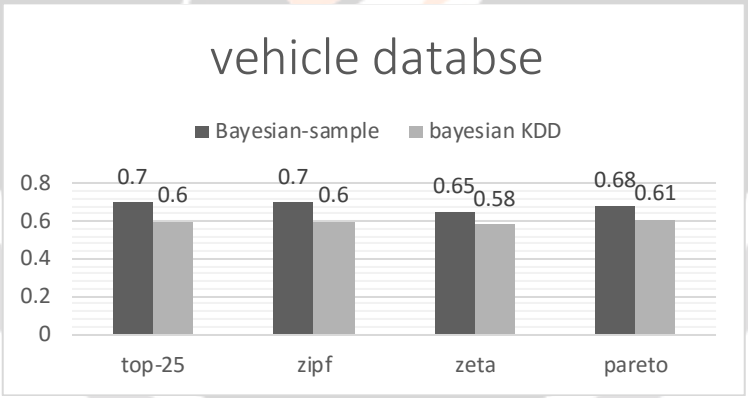


Fig.2 Ranking Quality for models

RESULTS:

This section describe the current status of implementation along with appropriate screen shot.

1. **Data analysis:**

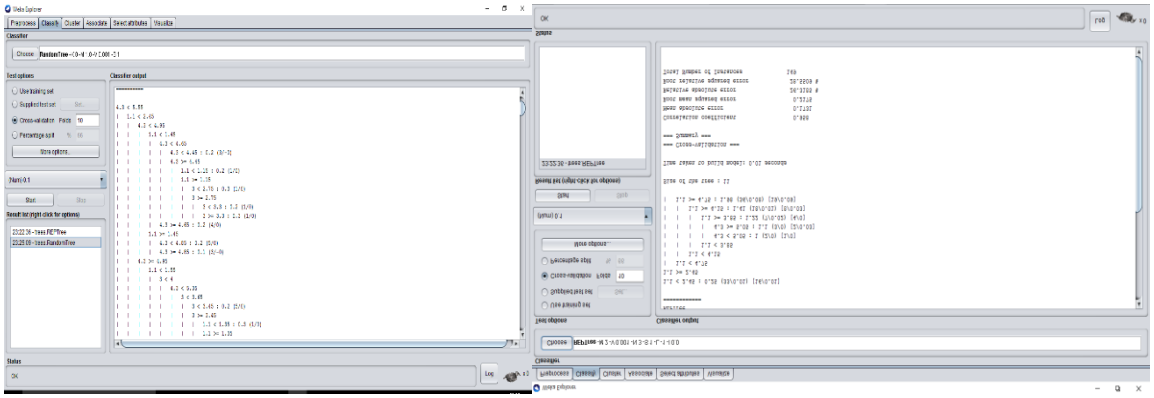


Fig.3 K-D Tree Datasets Sources

2. Visualization:

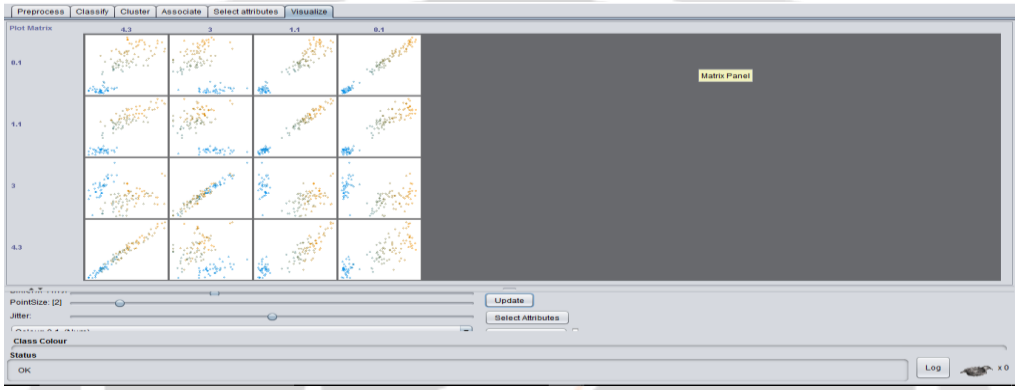


Fig.4 Visualization of K-D Tree formation.

5. CONCLUSIONS:

- The similarity models (user, query, and combined) and presented experimental results over two web databases to corroborate our analysis
- User- and query-dependent solution for ranking query results for web databases.
- K-Nearest Neighbor (KNN) approach to learning ranking functions along this direction.
- Experimental results show that the proposed approach outperforms both the single model approach and the query classification based approach.

6. ACKNOWLEDGEMENT:

The author would like to thank the reviewers for their precious comments and suggestions that contributed to the expansion of this work.

7. REFERENCES:

1. Aditya Telang, Chengkai Li, and Sharma Chakravarthy, One Size Does Not Fit All: Toward User- and Query-Dependent Ranking for Web Databases, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 24, NO. 9, EPTEMBER 2012.

2. Xiubo Geng, Tie-Yan Liu, Tao Qin, Andrew Arnold, Hang Li, Heung-Yeung Shum, Query Dependent Ranking Using K-Nearest Neighbor, SIGIR'08, July 20–24, 2008, Singapore.
3. Lian-Wang Lee, Jung-Yi Jiang, ChunDer Wu, Shie-Jue Lee, A Query-Dependent Ranking Approach for Search Engines, 2009 IEEE Second International Workshop on Computer Science and Engineering.
4. Ram Kiran Puttagunta, S Ravi Kishan, User and Query Based Ranking for Web Data Bases, International Journal of Innovative Research in Computer and Communication Engineering Vol. 2, Issue 11, November 2014.
5. P.Ayyadurai, S.Jayanthi, Automated Ranking for Web Databases using K-Means Algorithm and UQDR Approach, International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 6 , June – 2013.
6. J ung-Yi Jiang, Lian-Wang Lee, and Shie-J ue Lee, A Distributional Similarity Measure for Query-Dependent Ranking in Web Mining, Proceedings of the Ninth International Conference on Machine Learning and Cybernetics, Qingdao, 11-14 July 2010.
7. Soumya S., Vismi V., Jeena C.D., Nisha Oommachen, Employee Searching based on User and Query-Dependent Ranking, International Journal of Computer Applications (0975 – 8887) Volume 62– No.5, January 2013.
8. Minky Jindal, Nisha kharb, Data Mining in Web Search Engine Optimization and User Assisted Rank Results, InternationalJournal of Computer Applications (0975 – 8887) Volume 95– No.8,June 2014.
9. N. V. Pardakhe, Prof. R. R. Keole, Analysis of Various Web Page Ranking Algorithms in Web Structure Mining , International Journal of Advanced Research in Computer and Communication Engineering Vol.2, Issue 12, December 2013
10. S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum, "Probabilistic Information Retrieval Approach for Ranking of Database Query Results," ACM Trans. Database Systems, vol. 31, no. 3, pp. 1134- 1168, 2006.
11. C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank Aggregation Methods for the Web," Proc. Int'l Conf. World Wide Web (WWW), pp. 613-622, 2001.
12. S. Gauch and M. Speretta, "User Profiles for Personalized Information Access," Adaptive Web, pp. 54-89, 2007.
13. Google, Google Base, http://www.google.com/base, 2012.
14. B. He, "Relevance Feedback," Encyclopedia of Database Systems, pp. 2378-2379, Springer, 2009.
15. H. Yu, Y. Kim, and S. won Hwang, "Rv-svm: An Efficient Methodvfor Learning Ranking Svm," Proc. Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD), pp. 426-438, 2009.
16. T.C. Zhou, H. Ma, M.R. Lyu, and I. King, "Userrec: A User Recommendation Framework in Social Tagging Systems," Proc. 24th AAAI Conf. Artificial Intelligence, 2010.
17. A. Telang, S. Chakravarthy, and C. Li, "Establishing a Workload for Ranking in Web Databases," technical report, UT Arlington, http://cse.uta.edu/research/Publications/Downloads/CSE- 2010-3.pdf, 2010.
18. Y. Rui, T.S. Huang, and S. Mehrotra, "Content-Based Image Retrieval with Relevance Feedback in Mars," Proc. IEEE Int'l Conf. Image Processing, pp. 815-818, 1997.