Available online on http://www.rspublication.com/ijca/ijca_index.htm ISSN: 2250-1797

SIMULATION OF RANKING WEB DATABASES BASED ON USER AND QUERY DEPENDENT RANKS

Dilip Chakravarthy#1, Dr Nagaraj rao#2, Dr Murali Krishna#3

#1 Dept Of Computer Science Engineering, MITS/JNTUA,

#2 Dept Of Computer Science Engineering, MITS/JNTUA,

#3 Dept Of Computer Science Engineering, MITS/JNTUA,

ABSTRACT

The Ussage of internet in now a days is more and it became necessity for the people to do some applications such as searching web data bases in domains like vehicles,Realestates,etc.The problem with such search is ranking the results of query.Earlier we are using the frequencies of database values for Ranking.It is given based on either User dependent or Query dependent manner.This paper simulates the usage of ranking query results based on user and query Dependent ranks by taking user and query similarities as input including the workload. We define these similarities formally in alternative ways and discuss their effectiveness analytically and experimentally over two distinct Web databases.

Key words: User similarity, Query similarity, workload.

Corresponding Author: B.Dilip Chakravarthy

INTRODUCTION:

With the increase in usage of deep web which lead to some new applications such as airline reservations,vehicle search,the data bases are typically searched based on User similarity or query similarity.Earlier we are ranking based on tuple scores which leads to the result of more tuples which is a complicated process and time consuming process if the data base is more.The scenarios of User similarity and Query similarity are explained with examples.
Example1:Consider the data base of vehicle which consists of different bikes having the no of attributes such as make,color.If two persons consider professor and a student if they wants to search a vehicle to buy,the resultant search consists of no of tuples of bikes.Professor may be thinking his thought of bike in the first link where as student is thinking that his resultant should contain his thought of bike in first link this is solved by using user similarity i.e by taking user input.

Example2:Consider the same student who has chosen the link to be displayed first has changed his idea because of some internship given by the company.The resultant tuples will change considering the query.This is solved by by Query similarity.

The current sorting-based mechanisms used by Web databases do not perform such ranking. While some extensions to SQL allow *manual* specification of attribute weights [30] [21] [23]

Available online on http://www.rspublication.com/ijca/ijca_index.htm ISSN: 2250-1797

[33], this approach is cumbersome for **most** Web users. *Automated* ranking of database results has been studied in the context of relational databases, and although a number of techniques [10] [11][27] [34] perform *query-dependent ranking*, they do not differentiate between users and hence, provide a single ranking order for a given query across *all users*. In contrast, techniques for building extensive user profiles [20] as well as requiring users to order data tuples [18], proposed for *user-dependent* ranking, do not distinguish between queries and provide a single ranking order for any query given by the same user. Recommendation (i.e.,collaborative [17] [5] [8] and content filtering [4] [6] [14]) as well as information retrieval systems use the notions of user- and object/item-similarity for recommending objects to users. Although our work is inspired by this idea, there are differences that prevent its direct applicability to database ranking (elaborated in Section 2). In this paper, we propose a *user*- and *query-dependent* approach for ranking the results ofWeb 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 *similarity-based* ranking model proposed in the paper. The intuition behind our approach is: i) for the results of a given query, similar users display comparable ranking preferences, and ii) 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.

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 [16] used in document retrieval systems. However there exist several challenges in applying these techniques to Web databases directly. Although the focus of this paper is on the usage, instead of the acquisition of such workloads, we discuss and compare some potential approaches for establishing such workloads and elaborate on a learning method for deriving individual ranking functions.

2. RELATED WORK

Although there was no notion of ranking in traditional databases, it has existed in the context of information retrieval for quite some time. With the advent of the Web, ranking gained prominence due to the volume of information being searched/browsed. Currently, ranking has become ubiquitous and is used in document retrieval systems, recommender systems, Web search/browsing, and traditional databases as well. Below, we relate our effort to earlier work in these areas.

Ranking in Recommendation Systems: Given the notion of *user*- and *query*-similarity, it appears that our proposal is similar to the techniques of **collaborative** [17] [5] [8] and **content** filtering [4] [6] [14] used in recommendation systems.However, there are some important differences (between ranking tuples for database queries versus recommending items in a specific order) that distinguish our work. For instance, each cell in the *user-item* matrix of recommendation systems represents a single scalar value that indicates the rating/preference of a particular user towards a specific item. Similarly, in the context of recommendations for social tagging [2] [24] [35],each cell in the corresponding *user-*

URL/item-tag matrix indicates the presence or absence of a tag provided by a user for a given URL/item. In contrast, each cell in the *user-query* matrix (used for database ranking) contains an ordered set of tuples (represented by a ranking function). Further, although the rating/relevance given to each tuple (in the results of a given query) by a user can be considered to be similar to a rating given for an item in recommendation systems, if the same tuple occurs in the results of distinct queries, it may receive different ratings from the same user. This aspect of the same item receiving varied ratings by the same user in different contexts is not addressed by current recommendations systems to the best of our knowledge.Another important distinction that sets our work apart from recommendation systems is the notion of *similarity*. In content filtering, the similarity between items is established either using a domain expert, or user profiles [14], or by using a feature recognition algorithm [4] over the different features of an item (e.g., author and publisher of a book, director and actor in a movie, etc.). In contrast, since our framework requires establishing similarity between actual SQL queries (instead of simple keyword queries), the direct application of these techniques does not seem to be appropriate. To the best of our knowledge, a model for establishing similarity between database queries (expressed in SQL) has not received attention.

In addition, a user profile is unlikely to reveal the kind of queries a user might be interested in. Further, since we assume that the same user may have different preferences for different queries, capturing this information via profiles will not be a suitable alternative. The notion of user similarity used in our framework is identical to the one adopted in collaborative filtering; however, the technique used for determining this similarity is different. In collaborative filtering, users are compared based on the ratings given to individual items (i.e., if two users have given a positive/negative rating for the same items, then the two users are similar). In the context of database ranking, we propose a rigorous definition of user similarity based on the similarity between their respective ranking functions, and hence ranked orders. Furthermore, this work extends user-personalization using context information based on user and query similarity instead of static profiles and data analysis.

Ranking in Databases: Although ranking query results for relational and Web databases has received significant attention over the past years, simultaneous support for automated *user-* and *query-dependent* ranking has not been addressed in this context. For instance, [10] address the problem of *querydependent* ranking by adapting the vector model from information retrieval, where as [11][27] do the same by adapting the probabilistic model. However, for a given query, these techniques provide the same ordering of tuples across all users. Employing user personalizations by considering the context and profiles of users for *user-dependent* ranking in databases has been proposed in [20]. Similarly, the work proposed in [18] 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. The closest form of *query-* and *user-dependent* ranking in relational databases involves manual specification of the ranking function/preferences as part of SQL queries [30] [21] [23] [33]. However, this technique is unsuitable forWeb users who are not proficient with query languages and ranking functions. 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.

3. PROBLEM DEFINITION AND ARCHITECTURE

3.1Problem Definition

Consider a Web database table *D* over a set of *M* attributes, *A*= {A1,A2, ...,AM}. A user Ui asks a query Qj of the form:SELECT * FROM *D* WHERE A1 = a1 AND · · · AND As =as, where each Ai ∈ *A* and ai is a value in its domain. Let Nj = {t1, 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 Table1).The ranking problem can be stated as: "*For the query* Qj *given by the user* Ui, *determine a* ranking function FUiQj *from* **W**". Given the scale of Web users and the large number of queries that can be posed on *D*,**W**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 towards Qy's results, determine, explicitly or implicitly, a ranking function FUxQy . **W** is then established as a collection of such ranking functions learnt over different user-query pairs.The above description refers to *point* queries with conjunctive conditions. However, queries may contain range/IN conditions and several Boolean operators (AND, OR, NOT).However, our focus is on the problem of point queries over a single table. Extensions are being explored as future work.

	Q1	Q2	Q3	Q4
U1	F11	F12	F13	F14
U2	F21	F22	F23	F24
U3	F31	F32	F33	F34

Table1

3.2 Ranking Architecture

The Similarity model (shown in Figure 1) forms the core component of our ranking framework. When the user Ui poses the query Qj, the *query-similarity* model determines the set of queries ({Qj,Q1,Q2, ...,Qp}) most similar to Qj. Likewise, the *user-similarity* model determines the set of users ({Ui, U1, U2, ...Ur}) most similar to Ui. Using these two ordered sets of similar queries and users, it searches the workload to identify the function FUxQy such that the combination of Ux and Qy is most similar to Ui and Qj. FUxQy is then used to rank Qj's results for Ui.

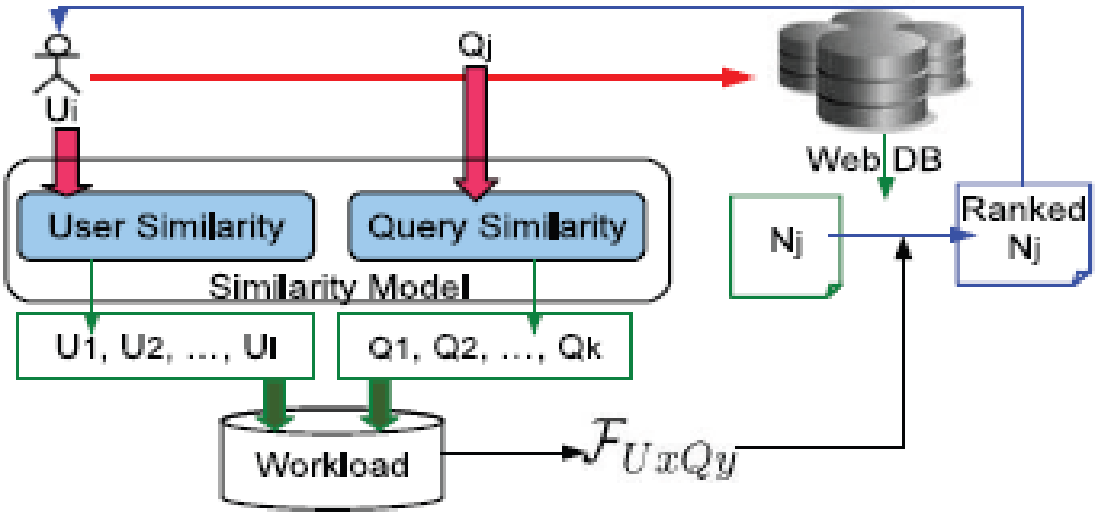

Fig. 1. Similarity Model for Ranking

4.SIMILARITY MODEL FOR RANKING

Similarity Model Explains about the finding of the related ranking function based on the given workload which comprises of ranking functions given for User-Query pairs.user similarity and query similarity identify the suitable user and suitable query.

4.1 Query Similarity:

For the given Query find the similar query in the workload this is given by the model query conditional similarity or by Query result similarity.

4.1.1 Query conditional similarity

In this model the similarity between two queries is determined by comparing the attribute value in the condition.

Definition Given two queries Q and Q1, each with the conjunctive selection conditions, respectively of the form "WHERE A1=a1 AND · · · AND Am=am" and "WHERE A1=a_1 AND · · · AND Am=a_m" (where ai or a_i is 'any'2 if Ai is not specified), the *query-condition* similarity between Q and Q1 is given as the conjunctive similarities between the values ai and a_i for every attribute Ai .

$$\text{similarity}(Q,Q1) = \prod_{I=1}^{m} \text{sim}(Q[Ai = ai], Q1[Ai = a\_i]) \quad (1)$$

4.1.2 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.

Definition Given two queries Q and Q1, let N and N1 be their query results. The *query-result* similarity between Q and Q1is then computed as the similarity between the result sets N and N1, given by Equation .

Available online on http://www.rspublication.com/ijca/ijca_index.htm ISSN: 2250-1797

similarity(Q,Q1) = sim(N,N1) (2)

4.2 User Similarity

User similarity can be determined by Considering the featuers of user.Let say if a user is a teenager with rank U1 then a similar rank is given for other teenagers also.

4.3 Composite Ranking

In order to derive a user's (Ui) ranking function for a query (Qj), we have proposed two independent approaches based on *user* and *query* similarity. However, given the scale of Web users and queries, and the sparseness of the workload, applying only one model may not be the best choice at all times.

The goal of this composite model is to determine a ranking function (Fxy) derived for the most similar query (Qy) to Qj given by the most similar user (Ux) to Ui to rank Qj 's results. The process for finding such an appropriate ranking function is given by the Algorithm

Algorithm 1 Deriving Ranking Functions from Workload
INPUT: Ui, Qj , Workload **W** (M queries, N users)
OUTPUT: Ranking Function Fxy to be used for Ui, Qj
STEP ONE:
for p = 1 to M **do**
%% Using Equation 1 or 2 %%
Calculate Query Condition Similarity (Qj , Qp)
end for
STEP TWO:
for r = 1 to N **do**
Calculate User Similarity (Ui, Ur)
end for
STEP THREE:
Rank(Ui,Qj) =F(Rank(Ur) Rank(Qp))
end for
end for
Fxy = Get-RankingFunction().

5.CONCLUSION

In this paper, we proposed a *user*- and *query*-dependent solution for ranking query results for Web databases. We formally defined the similarity models (*user*, *query* and *combined*) and presented experimental results over two Web databases to corroborate our 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 Web databases, an important challenge is the design and maintenance of an appropriate workload that satisfies properties of similarity-based ranking. Determining techniques for inferring ranking functions over Web databases is an interesting challenge as well. Another interesting problem would be to combine the notion of user similarity proposed in our work with existing user profiles to analyze 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.

ACKNOWLEDGEMENTS

We thank the anonymous referees (Dr Aditya Telang) for their extremely useful comments and suggestions on an earlier draft of this paper

REFERENCES

A. Telang, S. Chakravarthy, and C. Li, One Size does not fit all:Towards User and Query Dependent Ranking For Web Databases.

[1] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated ranking of database query results. In *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. In *SIGMOD Conference*, pages 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. *ACM Communications*, 40(3):66–72, 1997.
[5] J. Basilico and T. Hofmann. A joint framework for collaborative and content filtering. In *SIGIR*, pages 550–551, 2004.
[6] C. Basu, H. Hirsh, and W. W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI/IAAI*, pages 714–720, 1998.
[7] M. K. Bergman. The deep web: Surfacing hidden value. *Journal of Electronic Publishing*, 7(1), 2001.
[8] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *Intnl. Conf. on Machine Learning (ICML)*, pages 46–54, 1998.
[9] K. C.-C. Chang, B. He, C. Li, M. Patil, and Z. Zhang. Structured databases on the web: Observations and implications. *SIGMOD Record*, 33(3):61–70, 2004.
[10] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic ranking of database query results. In *VLDB*, pages 888–899, 2004.
[11] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic information retrieval approach for ranking of database query results. *TODS*, 31(3):1134–1168, 2006.
[12] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *International conference on World Wide Web (WWW)*, pages 613–622, New York, NY, USA, 2001. ACM.
[13] N. Fuhr. A probabilistic framework for vague queries and imprecise information in databases. In *VLDB*, pages 696–707, 1990.
[14] S. Gauch and M. S. et. al. User profiles for personalized information access. In *Adaptive Web*, pages 54–89, 2007.
[15] Google. Google base. http://www.google.com/base.
[16] B. He. Relevance feedback. In *Encyclopedia of Database Systems*, pages 2378–2379, 2009.
[17] T. Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR*, pages 259–266, 2003.
[18] S.-W. Hwang. *Supporting Ranking For Data Retrieval*. PhD thesis, University of Illinois, Urbana Champaign, 2005.
[19] T. Kanungo and D. Mount. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions of Pattern Analysis in Machine Intelligence*, 24(7):881–892, 2002.

[20] G. Koutrika and Y. E. Ioannidis. Constrained optimalities in query personalization. In *SIGMOD Conference*, pages 73–84, 2005.
[21] C. Li, K. C.-C. Chang, I. F. Ilyas, and S. Song. Ranksql: Query algebra and optimization for relational top-k queries. In *SIGMOD Conference*, pages 131–142, 2005.
[22] X. Luo, X. Wei, and J. Zhang. Guided game-based learning using fuzzy cognitive maps. *IEEE Transactions on Learning Technologies*, PP(99):1–1, 2010.
[23] A. Marian, N. Bruno, and L. Gravano. Evaluating top-k queries over web-accessible databases. *ACM Transactions of Database Systems*,29(2):319–362, 2004.
[24] A. Penev and R. K. Wong. Finding similar pages in a social tagging repository. In *WWW*, pages 1091–1092, 2008.
[25] Y. Rui, T. S. Huang, and S. Mehrotra. Content-based image retrieval with relevance feedback in mars. In *IEEE Internationa Conference on Image Processing*, pages 815–818, 1997.
[26] X. Shi and C. C. Yang. Mining related queries from web search engine query logs using an improved association rule mining model. *J. Am.Soc. Inf. Sci. Technol.*, 58:1871–1883, October 2007.
[27] W. Su, J. Wang, Q. Huang, and F. Lochovsky. Query result ranking over e-commerce web databases. In *Conference on Information and Knowledge Management (CIKM)*, pages 575–584, 2006.
[28] 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.
[29] A. Telang, C. Li, and S. Chakravarthy. One size does not fit all: Towards user- and query-dependent ranking for web databases. Technical report, UT Arlington, http://cse.uta.edu/research/Publications/Downloads/CSE-2009-6.pdf, 2009.
[30] K. Werner. Foundations of preferences in database systems. In *VLDB*, pages 311–322. VLDB Endowment, 2002.
[31] L. Wu and C. F. et. al. Falcon: Feedback adaptive loop for content-based retrieval. In *VLDB*, pages 297–306, 2000.
[32] Z. Xu, X. Luo, and W. Lu. Association link network: An incremental semantic data model on organizing web resources. In *Intl. Conf. on Parallel and Distributed Systems (ICPADS)*, pages 793–798, 2010.
[33] H. Yu, S.-w. Hwang, and K. C.-C. Chang. Enabling soft queries for data retrieval. *Information Systems*, 32(4):560–574, 2007.