Employee Searching based on User and Query-Dependent Ranking

Soumya S., Vismi V., Jeena C.D., Nisha Oommachen.

Department of Computer Science and Engineering, University of Kerala
Sree Buddha College of Engineering
Alappuzha district
Pattoor

ABSTRACT
The growth of the Web and the Internet leads to the development of an ever increasing number of interesting application classes. The most common method used now in companies is normal recruitment process. If a company wants an employee immediately, the only way for recruitment is advertising in any media. After receiving applications from the employees, they need to check the qualification, experience etc. It is a time required process.

This paper proposes a method for employee searching by using a user and query dependent ranking. Here present a ranking model based on user inputs. This ranking model is acquired from several other ranking functions derived for various user-query pairs. This is based on the intuition that similar users display comparable ranking preferences over the result of similar queries. This paper gives an idea about how the ranking can be used.

Keywords
User Similarity, Query Similarity, Automatic Ranking, Workload, Relational Queries.

1. INTRODUCTION
The success and growth of the Internet and Web leads to the development of a large number of Web databases for a variety of applications. Database systems support only a Boolean query model. If query is not selective then too many tuples may be in the answer. It is time consuming to select the most appropriate answer .Web databases simplify this task by sorting the query result. Currently this sorting is done on the values of a single attribute. The ordering based on multiple attribute values would be closer to the Web user's expectation.

Here depict two scenarios as running examples.

Example-1: *Two users – a software company executive (U_1) and a nonsoftware company executive, for example a data entry company (U_2), seek answers to the same query (Q_1): "Working area= computer AND Location = Dallas, TX", for which more than 18,000 tuples are typically returned in response. Intuitively, U_1 would typically search for employees with Programming skills in particular language, and hence would prefer employees with "Condition =programmer AND language = Java" to be ranked and displayed higher than the others. In contrast, U_2 would most likely search for data entry operators with minimum speed in data entry; hence, for U_2, employees with "Condition = Dataentry operator AND qualification=Plus Two" should be displayed before the rest.*

Example-2: *The same user (U_2) moves to do some medical transcription work and asks a different query (say Q_4): "Working area = Medical field AND Location = Mountain View". It can presume that he may want employees with slightly higher qualification for medical transcription, and hence would prefer employees with "Condition = Data Entry Operator AND Qualification=Degree" to be ranked higher than others.*

Example-1 shows that towards the results of the same query, different Web Users may have contrasting ranking preferences [2]. Example-2 shows that the same user may display different ranking preferences for the results of different queries [2]. Thus in the case of Web databases, where a large set of queries is involved, the corresponding results should be ranked in a user-and query-dependent manner.

The current sorting mechanism used by Web databases is an automated ranking of database results. Automated ranking provide a single ranking order for a given query across all users because they do not differentiate between users. In contrast, techniques for building extensive user profiles [3] as well as requiring users to order data tuples [4], proposed for *user-dependent* ranking, do not distinguish between queries and provide a single ranking order for any query given by the same user.

This paper proposes an application of user- and query-dependent approach for ranking the results of Web databases queries. The key goal of an information retrieval system is to retrieve information which might be useful or relevant to the user. Employees are recruited into the company by normal methods such campus placements, advertising in any media etc. But it is a time required process.

For filling a single vacancy the above method is not as efficient. Here present a method for employee searching by using a user and query dependent ranking. The employer can search in the site and can select employees with required qualification.

For a query Q_j given by a user U_i, a relevant ranking function is identified from a workload of ranking functions, to rank Q_j's results. **Query similarity** indicates that for the results of a given query, similar users display comparable ranking preferences. And the **user similarity** means a user displays analogous ranking preferences over results of similar queries. The ranking function here used is a function of attribute weights and value weights. The former denoting the significance of individual attributes and the latter representing the importance of attribute values. A minimal workload is important to make this approach practically useful. By

adapting relevant feedback mechanism, this proposed technique can acquire such a workload.

2. RELATED WORK

There was no concept of ranking in traditional databases. Currently ranking has become everywhere at once and is used in document retrieval systems, traditional data bases, Web searching/browsing as well.

2.1 Ranking done in database

This context proposes address the problem of query dependent ranking. But, for a given query, this technique provides the same ordering of tuples across all users. By considering the profiles of users for user-dependent ranking in databases has been proposed here. The drawbacks in all these works focus on the fact that they ignore the concept that the same user may have varied ranking preferences for different queries. The closest form of query- and user-dependent ranking in relational databases has been proposed here. This technique is also unsuitable for Web users who are not proficient with query languages and ranking functions. In contrast, this 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.

2.2 Relevance Feedback

Inferring a ranking function by analyzing the user's interaction with the query results originates from the concepts of relevance feedback [7] [8] [9] in the domain of document and image retrieval systems. The direct application of either explicit or implicit feedback mechanisms for inferring database ranking functions has several challenges.

3. PROBLEM DEFINITION AND ARCHITECTURE

The ranking problem can be stated as: "*For the query* Q_j *given by the user* U_i, *determine a* ranking function FU_iQ_j *from* **W**". The ranking problem can be split into:

1. *Identifying a ranking function using the similarity model*: Given **W**, determine a user U_x similar to U_i and a query Q_y similar to Q_j such that the function FU_xQ_{ys} exists in **W**.

2. *Generating a workload of ranking functions*: Given a user U_x asking query Q_y, based on U_x's preferences towards Q_y's results, determine, explicitly or implicitly, a ranking function FU_xQ_y. **W** is then established as a collection of such ranking functions learnt over different user-query pairs.

3.1 Ranking Architecture

The core component of ranking framework is the similarity model (Figure 1). The set of users ($\{U_i, U_1, U_2, ...U_r\}$) most similar to U_i, determined by the user similarity model

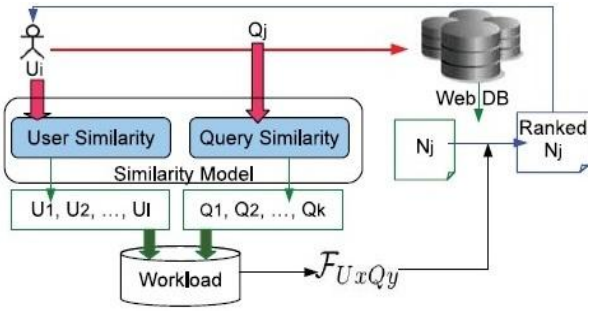

Fig 1: Similarity Ranking Model [2]

The query similarity model determines the set of queries ($\{Q_j,Q_1,Q_2, ...,Q_p\}$) 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.

In the proposed paper, 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*.

$$tuplescore(t) = \sum_{i=1}^{m} w_i * v_i \quad (1)$$

Where w_i represents the *attribute-weight* of A_i and v_i represents the *value-weight* for A_i's value in tuple t.

4. SIMILARITY MODEL FOR RANKING

When ranking functions are known for a small set of user-query pairs, then the concept of similarity-based ranking is aimed. 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.

4.1 Query Similarity

For the user U_1 from *Example-1*, a ranking function does not exist for ranking Q_1's results (N_1). However, from *Example-2*, it is known that a user is likely to have displayed different ranking preferences for different query results. Consequently, a randomly selected function from U_1's workload is not likely to give a desirable ranking order over N_1. On the other hand, the ranking functions are likely to be comparable for queries similar to each other [2].

The proposed paper advances the hypothesis that if Q_1 is most similar to query Q_y (in U_1's workload), U_1 would display similar ranking preferences over the results of both queries; thus, the ranking function (F_{1y}) derived for Q_y can be used to rank N_1. Similar to recommendation systems, this framework can utilize the aggregate function, composed from the functions corresponding to the *top-k* most similar queries to Q_1, to rank N_1 [2]. This proposal of *query similarity* into two alternative models: i) *query condition* similarity, and ii) *query-result* similarity.

4.1.1 Query-Condition Similarity

By comparing the attribute values in the query conditions, the similarity between two queries can be determined.

Given two queries Q and Q', each with the conjunctive selection conditions, respectively of the form "WHERE $A_1=a_1$ AND • • • AND $A_m=a_m$" and "WHERE $A_1=a_1$' AND • • • AND $A_m=a_m$' " , the *query-condition* similarity between Q and Q' is given as the conjunctive similarities between the values a_i and a_i' for every attribute A_i (Equation 1).

$$\text{Similarity}(Q,Q') = \prod_{i=1}^{m} \text{sim}(Q[A_i = a_i], Q'[A_i = a_i']) \quad (2)$$

4.1.2 Query-Result Similarity

If two queries are similar, the results are likely to greater similarity. Similarity between a pair of queries is evaluated as the similarity between the tuples in the respective query results. Given two queries Q and Q', let N and N' be their query results. The query-result similarity between Q and

Q' is then computed as the similarity between the result sets N and N', given by Equation 2.

$$\text{similarity}(Q,Q') = \text{sim}(N,N') \quad (3)$$

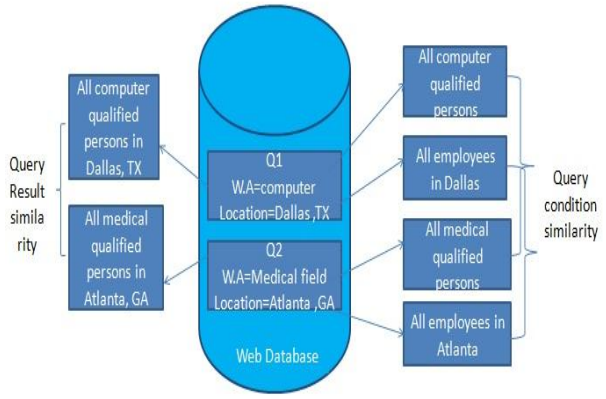

Fig 2: Query similarity model summarized view

The above figure shows the computation of similarity for the two models.

4.2 User Similarity

It is known from Example-1 that different users may display different ranking preferences towards the same query. Here put forward the hypothesis that if U_1 is similar to an existing user U_x, then, for the results of a given query (say Q_1), both users will show similar ranking preferences; therefore, U_x's ranking function (F_{x1}) can be used to rank Q_1's results for U_1 as well. Given two users U_i and U_j with the set of common queries – $\{Q_1, Q_2, ..., Q_r\}$, for which ranking functions ($\{F_{i1}, F_{i2}, ..., F_{ir}\}$ and $\{F_{j1}, F_{j2}, ..., F_{jr}\}$) exist in W, the user similarity between U_i and U_j is expressed as the average similarity between their individual ranking functions for each query Q_p (shown in Equation 3):

$$\text{Similarity}(U_i, U_j) = \frac{\sum_{r} sim(F_{ip}, F_{jp})}{r} \quad (4)$$

4.3 The Composite Similarity Model

The goal of this composite model is to determine a ranking function (F_{xy}). Finding such an appropriate ranking function is given by the Algorithm.

INPUT: U_i, Q_j , Workload W (M queries, N users)

OUTPUT: Ranking Function F_{xy} to be used for U_i, Q_j

STEP ONE:

for p = 1 to M do

%% Using Equation 2 %%

Calculate Query Condition Similarity (Q_j ,Q_p) **end for**

%% Based on descending order of similarity with Q_j %%

Sort(Q_1, Q_2, ... Q_M)

Select Q_{Kset} i.e., top-K queries from the above sorted set

STEP TWO:

for r = 1 to N do

%% Using Equation 4 %%

Calculate User Similarity (U_i, U_r) over Q_{Kset}

end for

%% Based on descending order of similarity with U_i %%

Sort(U_1, U_2, ... U_N) to yield U_{set}

STEP THREE:

for Each $Q_s \in Q_{Kset}$ do

for Each $U_t \in U_{set}$ do

Rank(U_t,Q_s) =Rank($U_t \in U_{set}$) + Rank($Q_s \in Q_{Kset}$)

end for

end for

F_{xy} = Get-RankingFunction()

The input to the algorithm is a user (U_i) and a query (Q_j) along with the workload matrix (W) containing ranking functions. The algorithm begins by determining the *querycondition similarity* (STEP ONE) between Q_j and every query in the workload. It then sorts all these queries (in descending order) based on their similarity with Q_j and selects the set (Q_{Kset}) of the top-K most similar queries to Q_j that satisfy the conditions for the *top-K user similarity* model. Based on these selected queries, the algorithm determines the *usersimilarity*(STEP TWO) between U_i and every user in the workload. All the users are then sorted (again, in descending order) based on their similarity to U_i. It then generate a list of all the user-query pairs (by combining the elements from the two sorted sets), and linearise these pairs by assigning a rank (which is the sum of query and user similarity ranks) to each pair (STEP THREE). For instance, if U_x and Q_y occur as the x^{th} and y^{th} elements in the respective ordering with the input pair, the pair (U_x, Q_y) is assigned an aggregate rank. In this case, a rank of "x + y" will be assigned. The "Get-Ranking Function" method then selects the pair (U_x, Q_y) that has the lowest combined rank and contains a ranking function (F_{xy}) in the workload. Then, in order to rank the results (N_j), the corresponding attribute weights and value weights obtained for F_{xy} will be individually applied to each tuple in N_j.

5. WORKLOAD OF RANKING FUNCTIONS

In this paper, the proposed model uses a workload of ranking functions. Obtaining such a ranking function is not a trivial task in the context of Web databases. Since obtaining ranking functions from users on the Web is difficult determining the exact set of ranking functions to be derived for establishing the workload is important.

6. CONCLUSION

This paper proposes a model for employee searching by using a user- and query-dependent ranking method. By using this method, it solves the Many-Answers Problem which leverages data and workload statistics and correlations. The design and maintenance of an appropriate workload that satisfies properties of similarity-based ranking is very challenging.

7. ACKNOWLEDGMENTS

We thank the anonymous referees for their extremely useful comments on an earlier draft of this article

8. REFERENCES

[1] Google. Google base. http://www.google.com/base.

[2] AdityaTelang, Chengkai Li, Sharma Chakravarthy, "One size Does Not Fit All: Towards User- and Query Dependent Rasnking For Web Databases".

[3] G. Koutrika and Y. E. Ioannidis. Constrained optimalities in query personalization. In SIGMOD Conference, pages 73–84, 2005.

[4] 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.

[5] A. Penev and R. K. Wong. Finding similar pages in a social tagging repository. In WWW, pages 1091–1092, 2008

[6] T. C. Zhou, H. Ma, M. R. Lyu, and I. King. Userrec: A user recommendation framework in social tagging systems. In *AAAI*, 2010.

[7] B. He. Relevance feedback. In Encyclopedia of Database Systems, pages 2378–2379, 2009.

[8] Y. Rui, T. S. Huang, and S. Mehrotra. Content-based image retrieval with relevance feedback in mars. In IEEE International Conference on Image Processing, pages 815–818, 1997.

[9] L. Wu and C. F. et. al. Falcon: Feedback adaptive loop for content-based retrieval. In VLDB, pages 297–306, 2000.

[10] Google. Google base. http://www.google.com/base.

[11] 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.

[12] S. Gauch and M. S. et. al. User profiles for personalized information access. In Adaptive Web, pages 54–89, 2007.

[13] A. Penev and R. K. Wong. Finding similar pages in a social tagging repository. In WWW, pages 1091–1092, 2008.

[14] T. C. Zhou, H. Ma, M. R. Lyu, and I. King.Userrec: A user recommendation framework in social tagging systems. In AAAI, 2010.

[15] H. Yu, S.-w.Hwang, and K. C.-C. Chang. Enabling soft queries for data retrieval. Information Systems, 32(4):560–574, 2007.

[16] A. Telang, C. Li, and S. Chakravarthy. One size does not fit all: Towardsuser- and query-dependent ranking for web databases. Technical report,UT Arlington, ttp://cse.uta.edu/research/Publications/Downloads/CSE-2009-6.pdf,2009.