Improving Web Database Search Incorporating Users Query Information

Rakesh Rawat
Faculty of Science & Technology
Queensland University of Technology
Brisbane, Australia
r.rawat@qut.edu.au

Richi Nayak
Faculty of Science & Technology
Queensland University of Technology
Brisbane, Australia
r.nayak@qut.edu.au

Yuefeng Li
Faculty of Science & Technology
Queensland University of Technology
Brisbane, Australia
Y2.li@qut.edu.au

ABSTRACT
The growing importance and need of data processing for information extraction is vital for Web databases. Due to the sheer size and volume of databases, retrieval of relevant information as needed by users has become a cumbersome process. Information seekers are faced by information overloading - too many result sets are returned for their queries. Moreover, too few or no results are returned if a specific query is asked. This paper proposes a ranking algorithm that gives higher preference to a user's current search and also utilizes profile information in order to obtain the relevant results for a user's query.

Categories and Subject Descriptors
H.2.3 [Database Management]: Languages--query languages; H.2.4 [Database Management]: Systems—query processing;

General Terms
Algorithms, Measurement, Performance.

Keywords
Query ranking, Feature Importance Technique based Ranking, Ranking SQL search results, Ranking personalized search results.

1. INTRODUCTION

Searching is one of the prominent and most widely utilized activity for finding the relevant information from the Web databases. A user query can be based on a single feature or a group of available features. At times when the user's query is specific, very few or no results are returned as all query conditions are not matched. On the other hand, when a generic query is asked, too many results (rows) are returned. The problem is that these results are not ranked in an order of relevancy to the user.

In order to return relevant results for users query, many methods are used like collaborative filtering techniques (which group similar users/query), [10], query refinement [9] and utilizing user profile [1], [10]. However, universal methods that cater to multiple problems and utilize all available information are still lacking. This paper proposes a ranking algorithm, the Features Importance Technique (FIT), that can rank results sets based on any or combination of available information sources like features searched, workload and profile information. This paper also extends the use of this method for returning personalized search results to visitors of a Website. Preliminary experiments show an improvement of about 23% in terms of F-score over the standard Web parameterized search.

2. PROBLEM STATEMENT

When a user directs a parameterized query to a search engine, a common problem faced is that either too few or too many results are returned. In such a scenario, finding right information becomes a cumbersome process. In both cases query reformulation is a normal behaviour of Web users [5]. To provide relevant information to users various methods like query reformulation [10] and query relaxation [8],[9] have been proposed, however, these methods have not lessened the burden of searching relevant information as needed by a user. Query reformulation may often distract from actual search needs, while query relaxation may again return too many answers. These methods may partially resolve such problems but considering the nature of search queries given by users a method which fits in all circumstances and is flexible is still lacking.

3. RELATED WORK

Ranking models motivated by the IR community are mostly workload dependant and widely use TF, IDF or a combination of these to rank and score database results. One of the biggest disadvantages of using TF-IDF as pointed out by [7], [13] is that query results may be biased to rank highly searched results higher. This can result in neglecting users actual search needs. Both the techniques QF and OFIDF [2] are workload dependant and the similarity function is inspired by the TF- IDF. To overcome this,

workload attributes are assigned importance however, using IDF to find similarity between tied tuples with missing attributes, again creates problems when tuples with high IDF and low IDF are used to score [2]. The combined metrics called as QFIDF [2] may work well in most of cases, however in cases when a query is entirely different or is un related to previous workload, QFIDF ranking may fail to give good rankings. The other drawback with QFIDF is that in case when searched features are relatively fewer compared to the available features, a non zero score may bias the results relative to a users query. In another significant work [3], the researchers have applied probabilistic information retrieval model based approach for ranking database query results. Effective solution to the problem of many- answers has been proposed. In another prominent work the researchers [4] have proposed a novel method of ranking top *k*-rows that match with the user's requirement. Their methodology revolves around a scoring variable which is used to retrieve top-*k* query results. If this variable is set high no rows are retrieved and if low too many rows are retrieved.

The other community using machine learning approaches to rank database tuples uses various data mining approaches from kNN (k Nearest Neighbour) [6] to classification [12] and various machine learning approaches like neural net [1] to rank database tuples. One such kNN approach as proposed by [12] uses a BM25 model to find top k ranked documents/results for a given user's query. However one major problem with kNN approach is that when two different users issue two similar queries, the ranking given by such model would have similar results for the two users, regardless of the intentions of the two users. The other drawback of kNN methodology is deducing the exact or optimal number of k for each query. The query classification (QC) [12], based approach may not work effectively when a query is un-categorized or have no classification details. The other drawback is that, a wrong categorization of queries may give contrasting results, than as expected by a user.

Another work [1] utilizes user profile information to increase efficiency of searched results. An ontology based ranking methodology [10], improves search by utilizing user profile information. Three sub techniques such as re-ranking, filtering and query expansion are utilized to collectively improve the performance of recommended documents. A recent approach [11], uses a probabilistic ranking model based on partial orders to rank result sets in case when lots of values are missing or are uncertain. In another ranking method [14], a database keyword based search is proposed by indexing the related information about tuples. In other work [9], methods of tackling empty answers have been proposed by rewriting the users query and relaxing constraints.

Unlike most of the previously discussed methods, which either use query relaxation or are too much workload dependant, FIT offers a lot of flexibility in terms of its applicability and implementation. In case where no workload or user profile information is available explicit categorization provided by a user or collaborative information (depicting important searched features) can be used to rank most relevant results as needed by a user.

4. THE PROPOSED METHODOLOGY

Consider the $(C_1 ... C_m)$ search features or parameters in a Website. As an example for a car sales website these features may be makes type, model and body types of a car. If m is the number of total columns (features), then assume v, a variable which depicts the proportionate representation of each feature, and its value is a representation of each feature in the overall search space. It is calculated as shown in equation (1).

$$v = \frac{1}{\text{Total No of Features (m)}}. \quad (1)$$

The value v is used to get a proportionate distribution of each feature in the overall search space. Let α is a preference variable used for normalization. In case when multiple parameters are searched, the weight of important features as searched by user can dominate the value of v, hence to normalize these values α is used. The value of preference variable α can be set depending upon the importance of each feature to a user. Usually α is set a fraction higher than the value of v. Ideally, the value of preference variable is represented as shown in equation (2)

$$\text{Preference Variable } \alpha = \left[v + \frac{v}{2} \right]. \quad (2)$$

The next important variable is the Importance Number of a feature and is denoted by I_n. This value is set by identifying the search behaviour of users from the Web log data. More the feature is used in past searches, higher the value it gets.

$$\text{Searched Ratio}(C_n) = \frac{\text{Total number of users used the feature in searches}}{\text{Total number of searches in the database}} \quad (3)$$

Features are ranked according to descending order of their searched ratio. The highly searched feature get an importance value (I_n) of 1, next highly searched gets 2 and so on. Table 1 shows the Importance Number (I_n) of few features derived from workload of a real car website.

Table 1: Example of Importance Number of Features for a car sales Website.

Imp. No. (I_n)	Feature's Name	Data Type	% Searched by all users (Web-Log Data)
1	Make	String	99.9%
2	Model	String	99.8%
3	Cost	Numeric	60.5%

For a textual and binary search feature (E.g. Make, Model, safety of a car), the score is represented as ${}_t\mu_r$ and is evaluated as shown in equation (4), when number of features searched $n \geq 2$.

$$ {}_t\mu_r = \left(\text{No. of Total Features Searched} - \text{Importance No. of Searched Feature}\right) \times \left(\text{Preference Variable} - \frac{1}{\text{Number of Features}}\right) \text{ or } \sum t\mu_r = \{(n - I_r) \times (\alpha - v)\} \quad (4) $$

However, when the number of features searched is 1, then equation (4) reduces to $\sum_t \mu_r = (\alpha - v)$. In case of Numeric Features (E.g. cost, engine size of car) the proportionate similarity between two compared numeric attributes is calculated. If ${}_uC_l$ is the numeric attribute searched by the user and ${}_dC_l$ is the corresponding database value that has to be compared with this, then score of numeric features ${}_n\mu_l$ is evaluated as shown in equation (5) as

$$ \text{if } {}_dC_l > {}_uC_l \text{ then } {}_n\mu_l = \left(\frac{{}_uC_l}{{}_dCl} \times v\right) \text{ else} $$

$$ \text{if } {}_uC_l > {}_dC_l \text{ then } {}_n\mu_l = \left(\frac{{}_dC_l}{{}_uC_l} \times v\right) \quad (5) $$

Thus, equation (5) compares the two numeric features relative to the other features. In case when two compared values are same the overall score of the tuples that exactly match these searched values would increase drastically. In such a scenario regardless of importance number of all categorical features, the ranking would be biased towards such numeric features. Hence to remove this biasness and to normalize the attribute score, such attributes are multiplied by the proportionate representation of each feature (v). The complete FIT based Ranking Algorithm is discussed in Figure 1.

Input: Query with parameters/features as searched by a user.
Output: Ranked result sets matching users query and interests.

Let m be the number of total features and n be number of features as searched by a user. Let k be the number of tuples to be compared with users search and let I_n is the importance number of each searched feature.

If ${}_uC_n$ be users search query and ${}_dC_n$ be result sets to be compared with users search query.

Begin

Step 1. Calculate variable $v = 1/m$.

Step 2. Evaluate Preference Variable $\alpha = (v + \frac{v}{2})$

Step 3. // Compare all features as searched by user.

for i=1 to k

for i=1 to n

// Retrieve I_r of each Search feature ${}_uC_r$.

// For text or binary feature compute score as

If ${}_uC_r = {}_dC_r$ && $n \geq 2$ then

$\sum_t \mu_r = \{(n - I_r) \times (\alpha - v)\}$;

Else

$\sum_t \mu_r = (\alpha - v)$;

// For numeric feature compute score as

If ${}_uC_l > {}_dC_l$ then

${}_n\mu_l = \{({}_dC_l / {}_uC_l) \times v\}$;

Else

${}_n\mu_l = \{({}_uC_l / {}_dC_l) \times v\}$;

End for

Step 4. // Compute total similarity score of tuple as

$\sum \mu_n = {}_t\mu_r + {}_n\mu_l$

End for

Return: Aggregate Score of all $\mu_{1..k}$ tuples.

End

Figure 1: Feature Importance Technique (FIT) based Ranking Algorithm.

Once the aggregate score of database attributes matching a users query are evaluated then these scores are added. Once all scores are calculated normalization is done as shown in equation (6),

$$ \mu_k = \frac{\mu_k}{\mu_p} \quad (6) $$

where μ_p is the highest score of all retrieved rows and μ_k is the score of k^{th} row.

5. EVALUATION

The FIT algorithm was tested on server log data provided by a popular car sale website of Australia. We conducted two type of evaluation experiments. First method tested the effectiveness of FIT in comparison to other widely adopted Web/ database ranking methodologies. In the other evaluation metrics we tested the effectiveness of personalized search over the normal web database search.

5.1.1 Experimental Design- Web Search Ranking

In the first evaluation method we checked the effectiveness of rankings by comparing SQL, FIT, IDF (Inverse Document Frequency), QF (Query Frequency) and QC (Query Classification). For methods like FIT, IDF, QF and QC Top n results in descending order of similarity score were taken. The table containing car information was 3.8 GB in size and had about 42 million rows and 63 attributes or features containing information about cars.

7 Students were involved in the testing phase of experiments. Each student was asked to submit 5 queries to the main cars database, containing 1,2,3,4 and 5 parameters. For each query with more than one searched feature the users were asked to give the order of preferences for individual features from high to low. The top 20 records for each query were retrieved using different methods like SQL, FIT, IDF, QF and QC. These top 20 results without the details of the method were given to each user for evaluation. Users were asked to rate these based on NDCG scheme where a score of 4 meant perfect, 3 meant excellent, 2 meant good, 1 meant fair and 0 meant bad. For the QC criteria the

feature cost was used as a query classification criteria, where each query was classified as a low cost car, if its cost was between 1-25000 $, medium cost cars if cost was between 25001-50000 $, high cost cars if cost was between 50001-100000 and premium cost car if cost was above 100000$. A total of 7 different test query sets as given by each user were used for evaluation. All search queries consisted of at least a single feature or a combination of features such as Make, Series, Cost, Body type, Engine Size and Drive type. The other remaining features (total 57 attributes) were all retrieved in the results but no query criterion was based on these 57 features.

5.1.2 Evaluation- Web Search ranking

We have used Normalized Discounted Cumulative Gain (NDCG) to measure the effectiveness of various ranking methods. NDCG is specifically suited for Web search evaluation [1]. The ranked results for a given query q are scored from top to bottom. NDCG for a given query q is given as

$$N_q = M_q j = \sum_{j=1}^{k} (2^{r(j)} - 1) / \log(1 + j) \qquad (7)$$

Where M_q is a normalization constant whose value is 1 for a perfect ordering and each $r(j)$ is value of rating (Eg. 0=Bad, 4=Perfect) at position j. NDCG ranks highly rated results higher and is thus well suited for such evaluation.

5.1.3 Results and Discussion- Web Search ranking

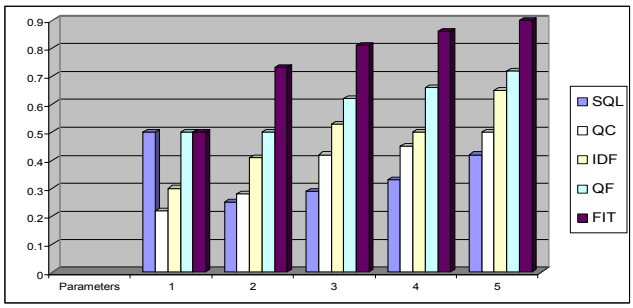

Figure 2: NDCG Comparative results of various ranking methods.

From the results in figure 2, it can be clearly seen that FIT based ranking performs consistently well. IDF extends TF-IDF concept of IR model to rank database tuples. IDF may not be very effective, as evaluating numerical similarity measures are not well defined [2]. IDF is biased towards documents with higher occurrences. One of the reasons for its average performance may be the unavailability of high number of similar occurrences in workload as searched by the users. The performance of IDF ranking is nearly the average of all other methods.

In case of QF it is purely workload dependant and has various disadvantages especially in case when workload information is unreliable [2]. QF based ranking may work well in most of the scenarios however in case when a users query is unrelated to his previous workload then QF will fail to retrieve good rankings. Example if a user had searched for sedan cars in the past, all workload based rankings methods would rank such cars higher, even though the user may have currently searched for an entirely new body type like a 'utility' vehicle.

QC performed worst in the absence of clear classification parameters (like no cost criteria was searched by users). Overall QC performed slightly better than normal SQL search. This happens because QC will retrieve results from other similar queries classified in the same category, where in SQL fails to retrieve results in case where query selection criteria are rigid.

In case of FIT as the number of searched parameters increase, range of records displayed to a user increases significantly. This happens because even a single match of attribute gets some score. FIT will always retrieve rows as long as there are some matching values to a users query in the database. Thus, it can handle empty answers problem effectively. The too many answers problem can be eliminated by considering only the highest scored records. Unlike the previous methods like QFIDF [2] in case of FIT, a no matching of attributes does not get any score, and thus unmatched attributes do not play any role and are ranked lower in the overall search results displayed to a user.

In case of insufficient workload information, FIT has the ability to score based on the current search preferences as specified by user. Good ranking of results as obtained from the test experiments in all such cases where users may have correctly classified their preferences for a query is a clear indicator of this approach.

5.2.1 Experimental Design- Personalized Search:

In the second evaluation method we checked for the effectiveness of personalized search over normal SQL search, which is the most common search adopted by many website service providers. In this evaluation method user profile information is used to give search results to a user.

Three days real server search log data of 20 test users was used to build user profiles. The user profile of each user had information about searched parameters like make, model, body type, search type and cost and what role each feature or parameter played in a user's search. The higher a particular parameter was searched, the higher importance it gets. A set of three search queries with highest TF-IDF were taken for each user

5.2.2 Evaluation- Personalized Search

To evaluate personalized search, searches with features as car make, model, body type, search type and cost were taken. These user searches were then expanded as per his profile. The user profiles contained information about user preferences, e.g. for make, body type and cost, user preferences are shown in table 2.

Table 2: Sample Profile of a user.

User Id	Make		Body type		Cost Range	
	Name	Score	Name	Score	Name	Score
1	Ford	1	Sedan	1	C1	1
1	GM	0.75	SUV	0.75	C2	0.78

From the server workload, the searches of the same 20 users for the next 3 days excluding the profile creation data were taken as benchmark. On an average there were eight searches per user for benchmarking. For each of the 3 test queries of each user, the top 10 rows with highest score were retrieved by the FIT algorithm and were given as search results to the users. Apart from this the normal search given by a SQL database query, were used as a

baseline measure. The results given by the FIT and baseline were compared with the benchmark. Precision and recall were based on these, and their values were identified as False Positive (F.P) when results returned by FIT (highest Score) or baseline did not match even a single search from the list of results in the benchmark. False Negative (F.N) when benchmark had records of a user but neither FIT (highest Score) nor baseline retrieved the same results as in the benchmark, and True Positive (T.P) when number of searches given by FIT (highest Score) or baseline matched to the benchmark search results. Thus PR was calculated as shown in equation (8).

$$\Pr ecision = \frac{T.P.}{T.P. + F.P.}. \ , \ \text{Re}call = \frac{T.P.}{T.P. + F.N.}. \qquad (8)$$

5.2.3 Results and Discussion- Personalized Search

These initial results (table 3) indicate that personalized search improves the quality of search results given to a user. Higher value of precision in case of personalized search clearly point out that given results are more likely to be searched by users, as it matches to some extent their search behaviour.

Table 3: **Results of Personalized search vs baseline search.**

No. of Users	Average Precision Baseline	Average Precision FIT	Average Recall Baseline	Average Recall FIT
20	0.24	0.36	0.40	0.43

Due to Web users unpredictable search behaviour, in both the case the recall is low. A user may query for certain items and then may not search any one of the retrieved items. A near similar recall value in both the cases confirms this. However in case of personalized search there is a marginal increase in the value of recall, which indicates that personalized search may give results which users are more likely to search.

6. CONCLUSION

FIT ranking is a flexible ranking method, which can optimally utilize available information for best ranking of results. In case when workload is used or user preferences are available, FIT gives the exact occurrence of these features an extra impetus relatively in consideration to other features and scores them higher. In case when no information is available it still manages to score based on number of features searched and matched. In Web databases with millions of values and multiple search features, the performance of search results can be improved by the proposed FIT algorithm. However, to give optimal ranked search results some workload information or user profile details are needed.

7. ACKNOWLEDGMENTS

This research is funded by CRC (Co-operative Research Centre), Australia and Queensland University of Technology, Brisbane under the CRC-Smart Service Project, 2009-10.

7. REFERENCES

[1] Agichtein, E., Brill, E., and Dumais, S., "Improving web search ranking by incorporating user behavior information," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*Seattle, Washington, USA, 2006, p. 26.

[2] Agrawal, S., Chaudhuri, S., Das, G., and Gionis, A., "Automated ranking of database query results," in *Proceedings of CIDR* 2003, pp. 171-183.

[3] Chaudhuri, S., Das, G., Hristidis, V., and Weikum, G., "Probabilistic ranking of database query results," in *Proceedings of the Thirtieth international conference on Very large data bases*, Toronto, Canada 2004, pp. 888 - 899

[4] Chaudhuri, S. and Gravano, L., "Evaluating top-k selection queries," in *Proceedings of the 25th VLDB Conference*, Edinburgh, Scotland, 1999, pp. 399-410.

[5] Jansen, B. J., Booth, D. L., and Spink, A., "Patterns of query reformulation during Web searching," *Journal of the American Society for Information Science and Technology,* vol. 60, pp. 1358-1371, 2009.

[6] Kriegel., T. S. a. H.-P., "Optimal multi-step k-nearest neighbor search.," in *Proceedings of the 1998 ACM International Conference on Management of Data (SIGMOD'98)*, June 1998.

[7] Li, G., Zhou, X., Feng, J., and Wang, J., "Progressive keyword search in relational databases," in *ICDE*, 2009, pp. 1183-1186.

[8] Li, X., Zhang, J., and Li, L., "Providing Relevant Answers for Queries over E-Commerce Web Databases," in *Active Media Technology*: Springer Berlin / Heidelberg, 2009, pp. 467-477.

[9] Meng, X., Ma, Z. M., and Yan, L., "Answering approximate queries over autonomous web databases," in *Proceedings of the 18th international conference on World wide web*, Madrid, Spain 2009, pp. 1021-1030.

[10] Pretschner, A. and Gauch, S., "Ontology based personalized search," in *11th IEEE International Conference on Tools with Artificial Intelligence*, Chicago, 1999, pp. 391-398.

[11] Soliman, M. A. and Ilyas, I. F., "Ranking with uncertain scores," in *25th International Conference on Data Engineering*, Shanghai, China, 2009, pp. 317–328.

[12] Xiubo, G., Tie-Yan, L., Tao, Q., Andrew, A., Hang, L., and Heung-Yeung, S., "Query dependent ranking using K-nearest neighbor," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* Singapore, Singapore: ACM, 2008.

[13] Xu, Y., Ishikawa, Y., and Guan, J., "Effective top-k keyword search in relational databases considering query semantics," in *Advances in Web and Network Technologies, and Information Management*. vol. 5731/2010: Springer Berlin / Heidelberg, 2010, pp. 172-184.

[14] Zhu, L., Shen-Da Ji, W., and Liu, C. N, "Keyword Search Based on Knowledge Base in Relational Databases," in *8th Intl. Conf. on Machine Learning and Cybernetics*, 2009.