RELATIVE QUERY RESULTS RANKING FOR ONLINE USERS IN WEB DATABASES

Pramod Kumar Ghadei

Research Scholar, Sathyabama University, Chennai, Tamil Nadu, India
pramod-ghadei@yahoo.com

Dr. S. Sridhar

Research Supervisor, Sathyabama University, Chennai, Tamil Nadu, India
drs.sridhar@yahoo.com

Abstract

To handle with the problem of so many-answers replied from an online Web database in response to a relative query, this paper proposes a unique approach to rank the similar query results. Depending upon the on the database workload, we hypothesize how much the user worries about each and every attribute and assigns a specific weight to it. Then, based on the relationship among of different attribute values, a user satisfaction degree ranking method develops, which ranks all the answer tuples according to their satisfaction degree level for the initial query asked. Then, for the tuples with the same satisfaction degree, each undefined attribute value is assigned with a score according to its "importance" to the user, which is used for differentiating these tuples. Finally, the results of preparatory experiment shows effectiveness of the ranking techniques used for better user satisfaction in web-database systems.

Keywords: Web database, user satisfaction, relative query, query results ranking.

1. Introduction

Due to the fast development of the WWW, online business is developed and changed out over the Web promptly. To display the product information by using the internet is usually supported by the web database has become an essential way for online business. Here the web database is accessible only via user interface is referred to as online Web databases. In real time applications, the online Web database query processing models have always shows that the user knows what he/she wants and is able to specify queries that exactly express his/her query requirements. Practically, real-time users do not have inadequate knowledge about database details and its structure, and their query motives are usually imprecise too. They would like to see more significant information with respect t to their queries. Therefore, the query submitted by user should be soft constraints for the receiving query results.

Example: *Consider a house rent online Web database W from sulekha.com web site consisting of a single table HouseDB with attributes: postedby,housetype, bedrooms, location,Year,rent, …. Each tuple in HouseDB represents a house for rent.*

Based on the database, the user may issue the following query:

Q: *HouseDB* (postedby =*owner* housetype = 2bedrooms rent≤8000)

By receiving this query, the query processing model used by *HouseDB* will provide a list of owner provided information that is rented below 8000. However, given that broker provides similar house, the user may also be interested in viewing all broker information *rented* around 8000. The user may also be interested in a owner or broker advertised house *rented* 8500. Unfortunately, in the example above, the query processing model used by *HouseDB* would not provide the broker information or the slightly higher rented for owner information as possible answers of interest as the user did not specifically ask for them in her query. For dealing with the problem mentioned above and providing more significant answers, Nambiar [9] and our previous paper [14] have proposed the query satisfaction approaches, the basic idea of which is to relax the initial query to form an relative(or relaxed) query in order to obtain the relative query results.

Anyhow, one frequent problem faced by Web users is that there are usually enormous answers returned for a relative query. Most of online Web databases rank their query results with respect to ascending or descending order of a single attribute (e.g., sorted by *Date*, sorted by rent, etc.). Whereas, many users thinks multiple attributes (including both the specified and unspecified attributes) simultaneously when judging the significance or interesting for a result. In this paper, we solves the many-answers problem for the relative queries over online Web databases by proposing an unique ranking method, SSR (Satisfaction & Significance Ranking),

which can rank the relative query results by considering the satisfaction degree of specified attribute values to the initial query and the significance degree of unspecified attribute values to the user's choices.

The remaining of paper is organized as follows. Section 2 reviews related work. Section 3 gives the definition of relative query and results ranking problem. Section 4 presents the attribute value assignment. Section 5 proposes the satisfaction degree ranking method while the significance degree ranking method is proposed in Section 6. The experiment is Noted in Section 7. The paper is concluded in Section 8.

2. Related Work

Ranking user Query has been inspected in information retrieval for a long days. The probabilistic ranking model [5] and [4] and the statistical language model [10] have been successfully used for ranking functions. In addition, [11] and [6] explore the combination of database and information retrieval techniques to rank tuples of text attributes. In [2] and [12], few keyword-query based on retrieval techniques for databases are suggested. Whereas, most of these techniques concentrates on text attributes so it is very difficult to apply these techniques to rank tuples with categorical and numerical attributes.

Likewise, various researches have been proposed to rank the database query results. In [5] and [1], the SQL query language is extended to allow the user to specify the ranking function according to their choice for the specified attributes. In [7] and [8], the higher rates of tuples in a relation are extracted automatically by help of analyzing the workloads, which shows what users are looking for and what they thinks as important. In [13] and [3], a quantitative and a qualitative choice model were proposed, respectively. In the first, choices are specified indirectly using rating functions that associate a numeric rate with every tuple of the answer query. While in the second, choices among tuples are specified directly using binary choice relations. However, it should be pointed out that these approaches mainly focus on the actual query results ranking and most likely hard to rank the relative query results.

3. Relatively Query and Results Ranking

Let us consider an autonomous online Web database W with categorical and numerical attributes $B = \{ B_1, B_2, \ldots, B_n\}$and a selection query Q over W with a conjunctive selection condition of the form $Q = {}_{i\{1,\ldots, m\}}Ci$, where Ci takes the form of $Bi = bi$, $m \leq n$ and each Bi in the query condition is an attribute from B and ai is a value in its domain. By softening Q, an relative (or relaxed) query Q' which is used to find all tuples of W that show similarity to Q above a threshold $\alpha \in (0, 1]$ is obtained. Specifically, $Q'(W) = \{t \mid t \in W, \text{Similarity}(Q, t) > \alpha\}$, where, the threshold α is given by users or system. Further, the set of attributes $Y = \{B1, \ldots, Bs\} \subseteq B$ is known as the set of attributes specified by the query , while the set $X = B–Y$ is known as the set of unspecified attributes. Let $R \subseteq \{r1,\ldots,rn\}$ be the set of result tuples returned by W for the relative query Q' .The too many-answers problem happens when the relative query Q' is not too selective resulting in a large R.

Therefore, the objective of this problem is to find an order over the set of tuples in R that satisfies as much as possible the basic query and user's choices.

4. Attribute Value Assignment

In real world scenario, each and every attributes of an online Web database may not be equally important for deciding the significance between the relative query results and basic query. Thus, it is necessary to evaluate the coefficients of attributes value.

4.1 Database workloads from log

Database workloads – log of past user queries, have been shown as being a valuable source for essentially estimating the users interest [8]. The workload information can help to decide the frequency with which database attributes were often specified by users and thus may be important to any new users. Therefore, we decide the attribute value by the frequency of development of the attribute specified in queries in the workload. In other word, the more frequency of the attribute specified in queries in the workload means it has been received more consideration from users, so which should be higher value.

4.2. Attribute value Assignment

Let $Fre(B_i)$ be the frequency of the attribute Bi specified by queries in the workload. Let m be the total number of queries in the workload and then we have the formula (1) for assigning

the value of each and every attribute as follows,

$$VImp(B_i) = (Fre(B_i)+1)/m \quad (1)$$

Apparently , the attribute value assignment method is reasonable for web uses. For example, in *HouseDB*, the attributes *rent* and *housetyp*e are specified by users evidently more than the

attributes *floo*r and *sqft*, which indicates that the user considers more about the *rents* and *housetypes* when plans for a rented house, and thus they should be assigned the higher value coefficient. Note that, in order to make the

VImp(B_i) gets a nonzero value even if an attribute is never referred in the workload, we define the frequency of the attribute B_i as *Fre*(B_i)+1 in formula (1).

5. Satisfaction Ranking

5.1Tuple's Satisfaction Degree to the basic query

Definition: Let Q′ be an relative query that is softened by an basic query *Q* over an online Web databases, r is an answer tuple for Q′, the set of attributes *Y* ={*B1*, …, *Bm*}⊆*B* is the set of attributes specified by the basic query conditions <*C1*,…,*Ck*> in *Q*. Then, the satisfaction degree of the answer tuple r to the basic query *Q* can be defined as,

$$Satisfaction(r,Q) = \sum_{i=1}^{m} V_{imp}(A_I) \times \begin{cases} Sim(Q.B_i, t.B_i) \\ if\ Dom(B_i) = categorical \\ 1 - \frac{Q.B_i - t.B_i}{Q.B_i} \\ if\ Dom(B_i) = numerical \end{cases} \quad (2)$$

where, *Bi* is the attribute specified by the query condition *Ci* in *Q*, *VImp*(B_i) is the value of attribute B_i, *Sim*(Q.B_i,*rt*.B_i) is the similarity between the value of categorical attribute *Bi* of tuple *r* to the value specified by the condition *Ci* of the basic query *Q*.

The main objective of satisfaction degree ranking method is that according to the tuple's satisfaction degree to the basic query to rank the relative query results, the higher the Satisfaction degree, the higher is the tuple's ranking rate.

5.2 Calculating the relevant of categorical values

We review an approach which is taken from our earlier paper [2] for developing the relevant coefficient among categorical attribute values by using database workload. The insight is that if certain pairs of values <*g*, *h*> often "occur together" in the query workload, they are similar. Let *f*(*g*, *h*) be the frequency of the values *u* and *v* of categorical attribute *B* occurring together in a *IN* clause in the workload. Also let *f*(*g*)be the frequency of happening of the value *u* of categorical attribute *B* in a *IN* clause in the workload, and *f*(*h*) be the frequency of happening of the value *v* of categorical attribute *B* in a *IN* clause in the workload. Then, the similarity coefficient between *g* and *h* can be evaluated by using the successive t formula,

$$Sim(g,h) = \frac{f(g,h)+1}{\max(f(g),f(h))} \quad (3)$$

The formula (3) illustrate that, the more frequently occurring together of the same pair of attribute values is, the higher their similarity coefficient is.

Later we can use the formula (2) to compute the ranking rate of each answer tuple of the relative query. For a large online Web database, however, it is not sufficient by only using the satisfaction degree to rank the answer tuples, since there may too many tuples which satisfy the relative query closely tie for the same satisfaction degree and thus get ordered promptly. So in the next section, we will examine the significance degree ranking approach, according to the significance degree of unspecified attribute values to the user's choices to differentiate the tuples that part with the same satisfaction degree.

6. Significance Evaluation Ranking

6.1 Fundamental Probabilistic information retrieval model

The fundamental formulas from probability theory are offered as follows: *p(a|b) = p(b|a)p(a)/p(b)* and *p*(*a*,*b*|*c*) = *p*(*a*|*c*)*p*(*b*|*a*,*c*).In the context of the information retrieval, consider *D* a collection document. For a (fixed) query *Q*, let *R* represent the set of significant documents, and *R* =*D*–*R* be the set of insignificant documents. In order to rank any document *r* in *D*, it's need to find the probability of the significance of *r* for the query given the text features of *t* (e.g., the word frequencies in *r*), i.e., *p*(*R*|*r*). More formally, in probabilistic information retrieval, documents are ranked according to the descending order of their odds of significance, defined as the following score rate:

$$Score\ rate(r) = \frac{p(R \quad r)}{p(R' \quad r)} \frac{\frac{p(r \quad R)p(R)}{p(r)}}{\frac{p(r \quad R')p(R')}{p(r)}} \propto \frac{p(r \quad R)}{p(r \quad R')} \quad (4)$$

6.2 Variation of PIR models for relational data

In the variation of PIR models for structured databases, each and every tuple in a single database table *D* is effectively treated as a "document." Recall the notation from Section 3, where *Y* is the set of specified attributes

asked in the query, and X is the remaining set of unspecified attributes. So any tuple r as partitioned into two parts, r(Y) and $r(X)$, where $r(Y)$ is the subset of values corresponding to the attributes in Y, and $t(X)$ is the remaining subset of values corresponding to the attributes in X. Replacing t with Y and X, we can get the Equation (5).

$$Score(r) \propto \frac{p(X \quad R)}{p(X \quad Y, D)} \tag{5}$$

Then, by using the limited independence assumptions discussed in [13], the Equation (5) can be transformed as,

$$Score(r) \propto \frac{p(X \quad R)}{p(X \quad D)} . \frac{1}{p(Y \quad X, D)} \tag{6}$$

6.3 Workload-based evaluation of ranking functions

Evaluating the quantities $p(x \mid R)$ requires knowledge of R, which is unknown during query time. In this case, earlier works [12, 14] suggested that leverages available workload information for estimating $p(x \mid R)$. As mentioned above, the workload L is represented as a set of "tuples", where each tuple represents a query and is a vector containing the matching values of the specified attributes. Consider an basic query Q which specifies a set Y of attribute values, assume that there are k attributes in Y, while the set X are the not mentioned attributes. Suppose R as all query records in L that also request for Y and then the user's choices can be obtained by analysing the subset of the workload that contains queries that also request for Y. Therefore, for query Q, with specified attribute set Y, $p(x|R)$ as $p(x|Y,L)$. According to [13], making this replacement in Equation (6), we get

This can be finally rewritten as:

$$Score\ rate(r) \propto \prod_{x \in X} \frac{p(x \quad L)}{p(x \quad D)} \prod_{x \in X} \frac{p(x \quad y, L)}{p(x \quad y, D)} \tag{7}$$

Therefore, we should adapt the Equation (7) to make it suitable for calculating the Significance of relative query results. After this, the relative query results ranking approach is fully proposed.

7. **Experiments**

7.1 Experimental arrangement

For evaluation we created a rented house database *HouseDB* (*Rent, house type, bedrooms, sqft, balcony, distance, landmark)* containing 20,500 tuples extracted from sulekha.com. For developing a workload, we requested 10 people, some of them are actual house renters, to provide us with queries that they would execute if they wanted to rent a house. We collected a total of 120 queries, each typically referencing 2~5 attributes.

Besides SSR described above, we implemented two other ranking methods to compare with SSR. The one is Satisfaction degree ranking method (shorted for SR) proposed in section 5, the other one is RANDOM ranking method, which ranks the similar query results randomly.

7.2 Ranking for quality experiments

Since getting users to rank the whole database for each query would have been extremely tedious, we used the following strategy. For the dataset, we generate 10 test queries. For each test query *Qi* we generated a set S*i* of 40 tuples likely to contain a good mix of significant and insignificant tuples to the query. Finally, we presented the queries along with their corresponding S_i's (with tuples randomly permuted) to each and every user in our study. Each user's responsibility was to mark the top 10 tuples as the significant tuples that they preferred most from the forty unique tuples collected for each asked query. During ranking, they were asked to behave like real time and experienced renters to mark the tuples according to their choices.

For formally comparing the ranking quality, we use the Precision/Recall metrics. Figure 1 shows the precision of the various ranking methods. It can be seen that SSR greatly outperforms both SR and RANDOM. The averaged ranking precision of SSR is 0.79, while the averaged ranking precision of SR is 0.58. The reason is that SSR considers the weights of specified attributes and the significance of not specified attribute values to the user choices.

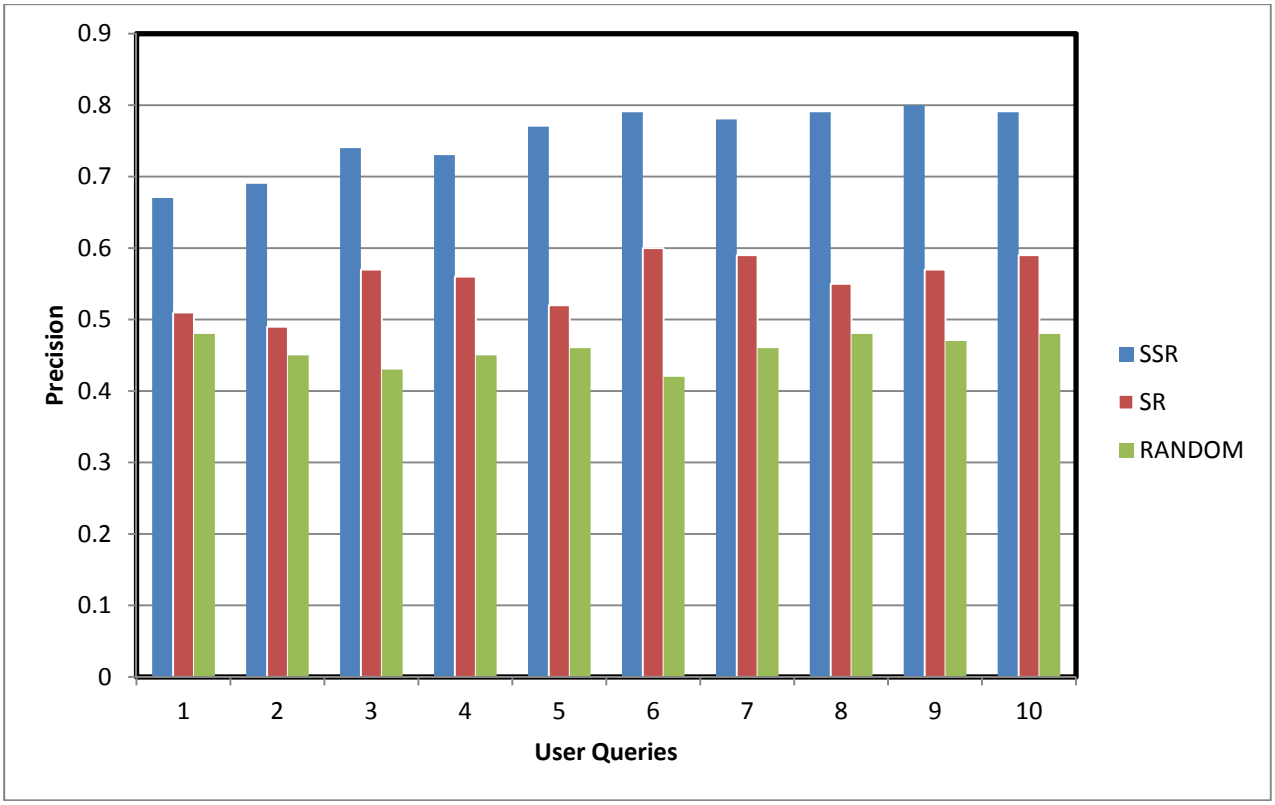

Figure 1. Precision of various ranking methods for user query results

8. Conclusions

This paper presented a comprehensively automated unique approach for ranking the relative query results for online Web database, which leverages the tuple's satisfaction degree level of Basic query and the significance degree to the user's choices. The results of preliminary experiments demonstrated the effectiveness of our ranking system. It is very interesting to inspect how to make the relative query results meets different type of user's choices.

References

[1] B. M. Ortega, K. Chakrabarti, and S. Mehrotra. "An approach to integrating query refinement in SQL," Proceedings of the EDBT Conference, 2002, pp. 15-33.
[2] G. Bhalotia, C. Nakhe, A. Hulgeri, S. Chakrabarti and S. Sudarshan. "Keyword searching and browsing in databases using BANKS," Proceedings of the ICDE Conference, 2002, pp. 431-440.
[3] J. Chomicki. "Preference formulas in relational queries,"ACM Trans. Database Syst., 2003, 28(4), pp. 427-466.
[4] K. Sparck Jones, S. Walker and S.E. Robertson. "A probabilistic model of information retrieval: development and comparative experiments-part 2," Inf. Process. Management, 2000, 36(6), pp. 809-840.
[5] K. Sparck Jones, S. Walker and S. E. Robertson. "A probabilistic model of information retrieval: development and comparative experiments-part 1," Inf. Process. Management, 2000, 36(6), pp. 779-808.
[6] N. Fuhr. "A probabilistic relational model for the integration of IR and databases," Proceedings of the ACM SIGIR Conference, 1993, pp. 309-317.
[7] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. "Automated ranking of database query results," ACM Trans. Database Syst., 2003, 28(2), pp. 140-174.
[8] S. Chaudhuri, G. Das, and V. Hristidis. "Probabilistic information retrieval approach for ranking of database query results," ACM Trans. Database Syst., 2006, 31(3),pp.1134-1168.
[9] U. Nambiar, S. Kambhampati, "Answering imprecise queries over web databases", Proceedings of the ICDE Conference, 2006, pp. 45-54.
[10] V. Hristidis and Y. Papakonstantinou. "Discover: keyword search in relational databases," Proceedings of the VLDB Conference, 2002, pp.670-681.
[11] W. B. Croft and J. Lafferty. "Language modeling for information retrieval," Kluwer Academic Publishers, 2003, pp. 11-56.
[12] W. Cohen. "Providing database-like access to the web using queries based on textual similarity," Proceedings of the ACM SIGMOD Conference, 1998, pp. 558-560.
[13] W. Kießling. "Foundations of preferences in database systems," Proceedings of the VLDB Conference, 2002,pp. 311-322.
[14] X. Li, J Zhang, and F. H. Li. "Providing relevant answers for queries over e-commerce web databases," Proceedings of the AMT Conference, 2009, pp. 467-477.