IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661, ISBN: 2278-8727Volume 8, Issue 2 (Nov. - Dec. 2012), PP 13-18
www.iosrjournals.org

RankingFor Web Databases Using SVM and K-Means algorithm

[1]P.Ayyadurai, [2]S.Jayanthi
[1]M.E-Student, [2]Assistant Professor,Srinivasan Engineering College, DhanalakshmiSrinivasan group, Perambalur, Tamilnadu, India

Abstract: *The Usage of internet in now a day is more and it became necessity for the people to do some applications such as searching web data bases in domains like Animation, vehicles, Movie, Real estates, etc. One of the problems in this context is ranking the results of a user query information. Earlier approaches problem have toused frequencies of database value regions, handling query logs, and user profiles information. A common thread in most of these approaches is that ranking is done in a usage page ranking 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. K- Means algorithm used for cluster and re ranking process, multiple database system used for clustering the data. Among rank learning methods, ranking SVM has been favorably applied to various applications, e.g., optimizing search engine for web information, improving data retrieval quality. We define these similarities formally in discuss their effectiveness analytically and experimentally over four distinct web databases.*
General Terms *-Clustering, Support vector machine, K-means algorithm, Web Databases.*
Keywords: *Automated Ranking, Animation Database, Vehicle and Movie Databases, User similarity, Query similarity, workload.*

I. Introduction

Data mining refers to extracting or "mining" knowledge from large amounts of data. The term is actually a misnomer. Remember that the mining of gold from rocksor sand is referred to as goldmining rather than rock or sand mining. Thus, data miningshould have been more appropriately named "knowledge mining from data," which isunfortunately somewhat long. "Knowledge mining," a shorter term, may not reflect theemphasis on mining from large amounts of data. Data mining is used in a wide range of industries - including retail, finance, health care, manufacturing transportation, and aerospaces [1].

The World Wide Web and its associated distributed information services, such as Yahoo!, Google, America Online, and AltaVista, provide rich, worldwide, on-line information services, where data objects are linked together to facilitate interactive access. Users seeking information of interest traverse from one object via links to another. Such systems provide ample opportunities and challenges for data mining. For example, understanding user access patterns will not only help improve system design but also leads to better marketing decisions (e.g., by placing advertisements in frequently visited documents, or by providing better customer/user classification and behavior analysis). Capturing user access patterns in such distributed information environments is called Web usage mining (or Weblog mining) [2], [3].

Theemergence of the deep web has led to the proliferation of a large number of web databases for a variety of applications (e.g., airline reservations, animation, vehicle search,movie, real estate scouting). These databases are typically searched by formulating query conditions on their schema attributes. When the number of results returned is large, it is time consuming to browse and choose the most useful answer(s) for further investigation. Currently, web databases simplify this task by displaying query results sorted on the values of a single attribute (e.g., Price, Mileage, etc.). However, most web users would prefer an orderingderived using multiple attribute values, which would be closer to their expectation. Consider Google Base's Vehicle database that comprises of a table with attributes Make, Price, Mileage, Location, Color, etc., where each tuple represents a vehicle for sale. We use the following two scenarios as our running examples. To 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 [1], [4], [5].

The ranking function used in our framework is a linear weighted-sum function comprising of: 1) attribute-weights denoting the significance of individual attributes and 2) value-weights representing the importance of attribute values. 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 technique used in document retrieval systems. However there exist several challenges in applying these techniques to Web databases directly [1], [6].

II. 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 [1].

A.Ranking in recommendation systems

Given the notion of user- and query-similarity, it appears that our proposal is similar to the techniques of **collaborative** and **content** filtering 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-itemmatrix of recommendation systems represents a single scalar value that indicates the rating/preference of a particular user.

B. Ranking in databases

Ranking query results for relational and Web databases has received significant attention over the past years, simultaneous support for automated user- and query-dependentranking has not been addressed in this context [1], [7], [8].

For instance, address the problem of query dependentranking by adapting the vector model from information retrieval, where as do the same by adapting the probabilistic model. However, for a given query, these techniques provide the same ordering of tuples across all users.

III. Problem Definition And Architecture

A. Problem Definition

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. 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, this approach is cumbersome for most web users. To provide a single ranking order for a given query across all users.In contrast techniques for building extensive user profiles as well as requiring users to order data tuples [2], [9].

TABLE 1
Sample Workload-A

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
U1	??	F12	-	-	F15	-	F17	-
U2	F21	F22	-	F24	-	F26	F27	-
U3	F31	F32	F33	F34	-	-	F37	-

TABLE 2
Sample Workload-B

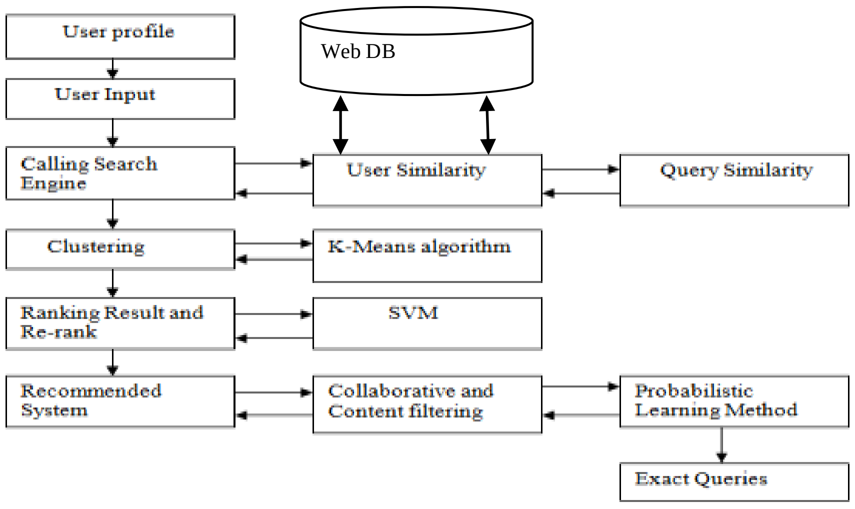
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
U1	??	-	F13	-	F15	F16	F17	F18
U2	F21	F22	-	F24	-	-	-	F28
U3	F31	F32	-	F34	-	-	F37	F38

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 towardQy's results, determine, explicitly or implicitly [1], [10].

B.Functional and Ranking Architecture



C. Functional procedure

Step 1: Setup web and created new user profile.
Step 2: Searching user query on any search engine.
Step 3: Clusters the specific queries using K-Means algorithm.
Step 4: Re-ranking for different users arrived, rerank automatically to support SVM.
Step 5: Filters and Test the exact query displayed.

IV. K-Means And Similarity Model

In order to meaningfully restrict the number of queries that are similar to each other, one alternative is to cluster queries in the workload based on query similarity. This can be done using a simple K-means clustering method . Using K-means, we cluster m queries into K clusters based on a predefined K and number of iterations [11], [12].

A. Query similarity

For a value of K=2, the simple K-means algorithm will generate two clusters—C1 containing Q1 and Q2 (along with other similar queries), and C2 containing Q7 (in addition to other queries not similar to Q1). Then estimate the similarity between U1 and every other user only for the cluster C1 (since it contains queries most similar to the input query) [1], [13].

B. User based similarity

TABLE 3
Sample Workload-A

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
U1	??	F12	-	-	F15	-	F17	-
U2	F21	F22	-	F24	-	F26	F27	-
U3	F31	F32	F33	F34	-	-	F37	-

TABLE 4
Sample Workload-B

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
U1	??	-	F13	-	F15	F16	F17	F18
U2	F21	F22	-	F24	-	-	-	F28
U3	F31	F32	-	F34	-	-	F37	F38

In this model, calculate user similarity for a given query Q1 by U1, by selecting top-K most similar queries to Q1, each of which has a ranking function for U1. Consequently for workload-A, using k=3, the queries Q2, Q5, and Q7 would be selected. Likewise, in the case of workload-B, this measure would select Q3, Q5, and Q6 using the “top-3” selection. Two different workloads have been proposed to setup queries and their performed effective analysis on the web.

C. workload based similarity

In order to address the problems in previous two models, propose a workload-based top-K model that provides the stability of the query-independent model (in terms of ensuring that ranking is always possible, assuming there is at least one nonempty cell in the workload for that user) and ensures that similarity between users can be computed in a query dependent manner [1], [14].

V. Experimental Evaluation

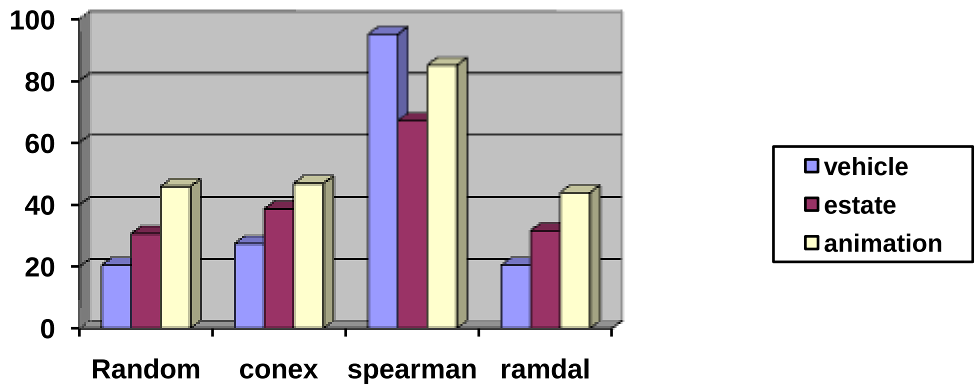
We have evaluated each proposed model (query-similarity and user-similarity) in isolation, and then compared both these models with the combined model for quality/accuracy. We also evaluated the efficiency of our ranking framework.

Ideally, we would have preferred to compare our approach against existing ranking schemes in databases. However, what has been addressed in literature is the use of exclusive profiles for user-based ranking (the techniques for the same do not distinguish between queries) or the analysis of the database in terms of frequencies of attribute values for query-dependent ranking (which does not differentiate between users). In the context of web databases like Google Base, the data are obtained on-the-fly from a collection of data sources [15]; thus, obtaining the entire database for determining the individual ranking functions, for comparing with query-dependent ranking techniques, is difficult. Even, if we obtain ranking functions for different queries, all users will see the same ranking order for a given query. Thus, comparing such static ordering of tuples against our approach (that determines distinct ranking of tuples for each user and query separately) would not be a meaningful/fair comparison. Similarly, we felt that the comparing static user profiles (that ignore the different preferences of the same user for different queries) to our proposed definition of user similarity, for user-dependentranking will not be fair. Hence, we have tried to compare the proposed user, query, and combined similarities to indicate the effectiveness of each model with respect to the other two models.

Quality Evaluation

Query similarity. Based on the two proposed models of query similarity in the absence of a function Fij for a user-query pair (Ui, Qj), the most similar query (Qc and Qr using the query-condition and the query result model, respectively) asked by Ui, for which a function (Fic and Fir, respectively) exists in the workload, is selected and the corresponding function is used to rank Qj’s results. We test the quality of both query similarity models as follows: We rank Qj’s results (Nj) using Fic and Fir, respectively, and obtain two sets of ranked results (R0 and R00). We then use the original (masked) function Fij to rank Nj and obtain the set (R). Since R represents the true ranking order provided by Ui for Qj, we determine the quality of this model by computing the Spearman rank correlation coefficient (8) between R and R0, and between R and R00. If the coefficients obtained are high (nearing 1.0), it validates our hypothesis (that for similar queries, the same user displays similar ranking preferences). Furthermore, if the coefficient between R and R0 is greater than the one between R and R00, our understanding that query-condition model performs better than the query-result model is validated. We performed the above process for each user asking every query. The Analysis graph shows, over both the domains, the query-condition model outperforms the query-result model. The graphs indicate that the comparative loss of quality (highest value of Spearman coefficient being 0.95 for query 5) is due to the restricted number of queries in the workload. Although finding a similar query (for which a ranking function isavailable) for a workload comprising of 20 queries and only 10 percent of ranking functions is difficult, the results are very encouraging. Based on the results of these experiments, we believe that the query similarity model would perform at an acceptable level of quality even for large, sparse workloads.

Graph 1. Quality evaluation



Efficiency Evaluation

The goal of this study was to determine whether our framework can be incorporated into a real-world application. We generated a workload comprising of 1 million queries and 1 million users, and randomly masked out ranking functions such that only 0.001percent of the workload was filled. We then generated 20 additional queries and selected 50 random users from the workload.

We measure the efficiency of our system in terms of the average time, taken across all users, to perform ranking over the results of these queries (using K-Means Algorithm).

If we use main memory for storing the workload and notuse any precomputation and indexing for retrieval, determining similarities (STEPS ONE & TWO) are computational bottlenecks as compared to the latter. In order to reduce the time for estimating query similarities, we can precompute pairwise similarities between all values of every attribute in the schema. Furthermore, in order to reduce the time to lookup every query in the workload and then evaluate its similarity with the input query, we use a value-based hashing technique to store all the queries in the workload. Likewise, all users are stored using a similar technique where the values corresponding to a user refer to various properties of the user profile.

VI. Conclusion

In this Paper, proposed a user- and query-dependent solution for ranking query results for web databases. 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, 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 a similarity-based ranking.

Acknowledgement

I take this chance to express my deep sense and knowledge of gratitude to our **Management**, our principal**Dr. B. Karthikeyan ,** for providing an excellent infrastructure and support to pursue this research work at our college.I express my profound thanks to Head of the department **Prof. J. Mercy Geraldine** for his administration and keen interest, which motivated me along the course as well as research work, and also thank to all staff members.

The authors thank the anonymous referees for their extremely useful comments and suggestions on an earlier draft of this paper.

References

[1] AdityaTelang, 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, September 2012.

[2] Alex Penev, Raymond K. Wong; Finding Similar Pages in a Social Tagging Repository;WWW 2008,April 21–25, 2008, Beijing, China. ACM 978-1-60558-085-2/08/04.

[3] Cynthia Dwork, Ravi Kumar, MoniNaor, D. Sivakumar; Rank Aggregation Methods for the Web;WWW10,May 1-5, 2001, Hong Kong. ACM 1-58113-348-0/01/0005.

[4] Hwanjo Yu, Youngdae Kim, and SeungwonHwang.RV-SVM: An Efficient Method for Learning Ranking SVM;Korea Research Foundation Grant funded by the Korean Government(KRF-2008-314-D00483)2009.

[5] KaushikChakrabarti, SurajitChaudhuri, Seung-won Hwang; Automatic Categorization of Query Results;SIGMOD 2004, June 13–18, 2004, Paris, France.

[6] Marko balabanovic, yoavshoham. Content based, collaborative Recommendation; March 1997/Vol. 40, No. 3 Communication Of The ACM.

[7] SurajitChadhuri, Gautam Das, VagelisHristidis; Probabilistic Information Retrieval Approach for Ranking of Database Query Results; ACM Transactions on Database Systems, Vol. 31, No. 3, September 2006, Pages 1134–1168.
[8] SubbaraoKambhampati, Garrett Wolf, Yi Chen, HemalKhatriBhaumikChokshi, Jianchun Fan, UllasNambiar; QUIC: Handling Query Imprecision & Data Incompleteness; in Autonomous Databases;(http://creativecommons.org/licenses/by/2.5/) CIDR 2007.
[9] Tapas Kanungo, Senior Member, IEEE, David M. Mount, Member, IEEE, Nathan S. Netanyahu, Member, IEEE, Christine D. Piatko, Ruth Silverman, and Angela; An Efficient C-Means Clustering Algorithm: Analysis and Implementation; IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. 24, No. 7, July 2002.
[10] Weifeng Su, Jiying Wang, Qiong Huang, Fred Lochovsky; Query Result Ranking over E-commerce Web Databases; CIKM'06, November 6–11, 2006, Arlington, VA Virginia, USA.
[11] Wei YAN, Li YAN, ZongminMA.;Automated Ranking of XML Fuzzy Query Results;Journal of Computational Information Systems 8: 6 (2012) 2567–2574 Available at http://www.Jofcis.com.
[12] Xiaodong Shi and Christopher C. Yang; Mining Related Queries from Web Search Engine Query Logs Using an Improved Association Rule Mining Model; Journal Of The American Society For Information Science And Technology, 58(12):1871–1883, 2007.
[13] N. Fuhr, "*A Probabilistic Framework for Vague Queries and Imprecise Information in Databases,*" Proc. 16th Int'l Conf. VeryLarge Data Bases (VLDB), pp. 696-707, 1990.
[14] S. Gauch and M. Speretta, *"User Profiles for Personalized Information Access,"* Adaptive Web, pp. 54-89, 2007.
[15] Google, Google Base, http://www.google.com/base, 2012.

AUTHORS PROFILE

P.Ayyaduraireceived the B.Tech Degree Information Technology and now he is an M.E student in the Department Of Computer Science & Engineering, Srinivasan Engineering College – Dhanalakshmi Srinivasan Group of Institutions, Perambalur,TN, India.
His research interest includes Web Databases, Data mining, Wireless Networks and Image Processing.

S.Jayanthi is working as Assistant Professor at Srinivasan Engineering College – Dhanalakshmi Srinivasan Group of Institutions, Perambalur,TN, India.
His main research interest includes Data Mining and Neural Networks.