www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume - 3 Issue -8 August, 2014 Page No. 7415-7419

Dynamic Query Forms for Non-Relational Database Queries

S.BhaskaraNaik, B.VijayaBhaskar Reddy

Mr. BhaskaraNaik, Department of Computer Science and Engineering, ShriShirdiSai Institute of Science and Engineering,Anantapur, bhaskaranaik999@gmail.com
Mr. B.VijayaBhaskar Reddy, M.Tech, Assistant Professor , Department of Computer Science and Engineering ,ShriShirdiSai Institute of Science and Engineering,Anantapur, bvijay.br@gmail.com

Abstract:

Modern scientific databases and web databases maintain large and heterogeneous data. These real-world databasescontain over hundreds or even thousands of relations and attributes. Traditional predefined query forms are not able to satisfyvarious ad-hoc queries from users on those databases. This paper proposes DQF, a novel database query form interface, which isable to dynamically generate query forms. The essence of DQF is to capture a user's preference and rank query form components,assisting him/her to make decisions. The generation of a query form is an iterative process and is guided by the user. Ateach iteration, the system automatically generates ranking lists of form components and the user then adds the desired formcomponents into the query form. The ranking of form components is based on the captured user preference. A user can alsofill the query form and submit queries to view the query result at each iteration. In this way, a query form could be dynamicallyrefined till the user satisfies with the query results. We utilize the expected F-measure for measuring the goodness of a queryform. A probabilistic model is developed for estimating the goodness of a query form in DQF. Our experimental evaluation anduser study demonstrate the effectiveness and efficiency of the system.

Keywords:non-relational, Query, Performance

1. INTRODUCTION

Database systems support a simple Boolean queryretrieval model, where a selection query on a SQLdatabase returns all tuples that satisfy the conditions in thequery. This often leads to the *Many-Answers Problem*:when the query is not very selective, too many tuples maybe in the answer.In this section, we formally define the Many-AnswersProblem in ranking database query results, and alsooutline a general architecture of our solution.

The Many-Answers Problem has been investigatedoutside the database area, especially in InformationRetrieval (IR), where many documents often satisfy agiven keyword-based query. Approaches to overcome thisproblem range from *query reformulation* techniques (e.g.,the user is prompted to refine the query to make it moreselective), to *automatic ranking* of the query results bytheir degree of "relevance" to the query (though the usermay not have explicitly specified how) and returning onlythe top-*K* subset.

In this paper we propose an automated rankingapproach for the Many-Answers Problem for databasequeries. Our solution is principled, comprehensive, andefficient. We summarize our contributions below.

Any ranking function for the Many-Answers Problemhas to look beyond the attributes specified in the query,because all answer tuples satisfy the specified conditions.

However, investigating unspecified attributes isparticularly tricky since we need to determine what theuser's preferences for these unspecified attributes are. Inthis paper we propose that the ranking function of a tupledepends on two factors:

(a) a *global score* which capturesthe global importance of unspecified attribute values, and(b) a *conditional score* which captures the strengths ofdependencies (or correlations) between specified andunspecified attribute values. For example, for the query"City = Seattle and View = Waterfront", a home that isalso located in a "SchoolDistrict = Excellent" gets highrank because good school districts are globally desirable.A home with also "BoatDock = Yes" gets high rankbecause people desiring a waterfront are likely to want aboat dock. While these scores may be estimated by thehelp of domain expertise or through user feedback, wepropose an automatic estimation of these scores via*workload as well as data* analysis. For example, pastworkload may reveal that a large fraction of users seekinghomes with a waterfront view have also requested forboat docks.

The next challenge is how do we translate these basicintuitions into principled andquantitatively describableranking functions? To achieve this, we develop rankingfunctions that are based on *Probabilistic InformationRetrieval (PIR)* ranking models. We chose PIR modelsbecause we could extend them to model datadependencies and correlations (the critical ingredients ofour approach) in a more principled manner than if we hadworked with alternate IR ranking models such as theVector-Space model. We note that correlations are oftenignored in IR because they are very difficult to capture inthe very high-dimensional and sparsely populated featurespaces of text data, whereas there are often strongcorrelations between attribute values in relational data(with functional dependencies being extreme cases),which is a much lower-dimensional, more explicitlystructured and densely populated space that our rankingfunctions can effectively work on.

2. Problem Definition and Architecture

In this section, we formally define the Many-Answers Problem in ranking database query results, and alsooutline a general architecture of our solution.

2.1 Problem Definition

We start by defining the simplest problem instance.Consider a database table D with n tuples $\{t1, \dots, tn\}$ overa set of m categorical attributes $A = \{A1, \dots, Am\}$.Consider a "SELECT * FROM D" query Q with aconjunctive selection condition of the form "WHERE$X1=x1$ AND … AND $Xs=xs$", where each Xi is an attributefrom A and xi is a value in its domain. The set of attributes$X = \{X1, \dots, Xs\}$Í A is known as the set of attributes*specified* by the query, while the set $Y = A - X$ is knownas the set of *unspecified* attributes. Let S Í $\{t1, \dots, tn\}$ bethe answer set of Q. The *Many-Answers Problem* occurswhen the query is not too selective, resulting in a large S.

2.2 General Architecture

Figure shows the architecture of systemfor enabling ranking of database query results. Asmentioned in the introduction, the main components arethe preprocessing component, an intermediate knowledgerepresentation layer in which the ranking functions areencoded and materialized, and a query processingcomponent. The modular and generic nature of oursystem allows for easy customization of the rankingfunctions for different applications.

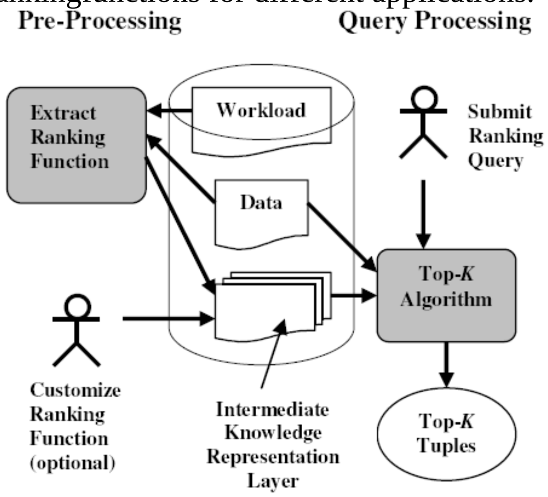


Figure 1: Architecture of Ranking System

3. RANKING ALGORITHM

In this paper the ranking model is based on two notions such as user similarity and query similarity.User similarity indicates that different users can have samepreferences. Query similarity indicates that different users can have identical queries. In order to accomplish this ranking of users and queries are to be maintained. We have developed a workload file that contains the user and query ranking functions. When newrecord is entered into database, obviously that is disposed by a user. There possibly many users who delivered that query previously and there might be same queries delivered earlier. The workload file is in tabular form and it gets updated with ranking functions as per the proposed algorithm as and when new queries are made. The proposed model has two patterns mixed. They are known as user addictive ranking model and query addictive ranking model.

However, we prefer applying both of them for more excellent results. The recommended ranking model in this paper is a linear weighted sum function. It includes attribute heaviness and value heaviness. Attribute weights indicate the importance of attributes while the value weight indicates the importance of values of attributes. Relevance feedback techniques are utilized for making the workloadminimal. The main contributions of this paper are

- User and query dependent accession for ranking web databases.
- Ranking model based on user correspondence and query correspondence notions.
- Two synthetic databases such as college and hospital used for experiments. However, the model can be tested with web databases.
- We used a new workload approach for keeping up the updated ranking of users and queries.

```
INPUT: Ui, Qj, Workload W (M queries, N users)
OUTPUT: Ranking Function Fxy to be used for Ui, Qj
STEP ONE:
For P ¼ 1 to M do
%% Using Equation 2 %%
Calculate Query Condition Similarity (Qj, Qp)
End for
%% Based on descending order of similarity (Qj, Qp)
Sort(Qi, Q2,......QM)
Select QKset i.e., top-K queries from the above sorted set
STEP TWO:
For r ¼ ! to N do
%% Using Equation 7 %%
Calculate User Similarity(Ui, Ur) over QKset
End for
%% Based on descending order of similarity with Ui %%
Sort(U1, U2,......UN) to yield Uset
STEP THREE:
For Each Qs 2 QKset do
For Each Ut 2 Uset do
Rank (Ut:Qsp ¼
```

Figure 3: Ranking Algorithm

This algorithm is implemented in theprototype application which shows both user and query dependent rankings for query results of web databases.

4. SAMPLE WORKLOAD FILE

The sample workload file is given in fig. which shows queries, users and the ranking functionscalculated as per the algorithm given in listing 1.

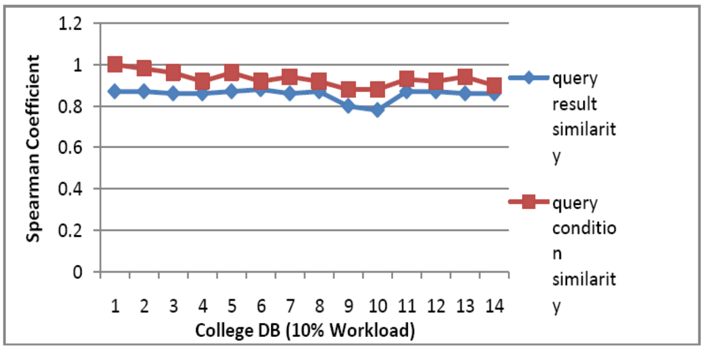
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
U1	??	F12	--	--	F15	--	F17	
U2	F21	F22	--	F24	--	F26	F27	--
U3	F31	F32	F33	F34	--	--	F37	--

Figure 4: Sample database

5. EXPERMENTAL EVALUATION

The experiments are made using a prototype application with two synthetic web databases such as college and hospital. The experimental results are evaluated by visualizing the results in the form of graphs. Figures show the ranking quality of query similarity models for both databases with 10% work load.

Figure 5: Ranking quality of query similarity (College DB)



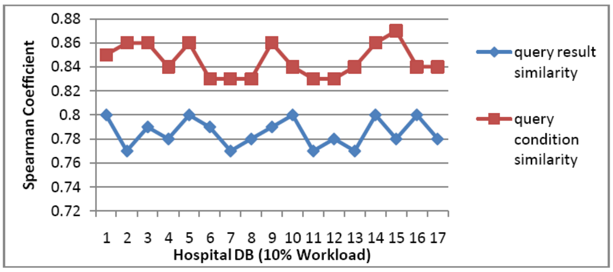


Figure 6 – Ranking quality of query similarity (Hospital DB)

As can be seen in fig. 5 and 6, query condition similarity average is found across all queries. The Xaxis shows queries while the Y axisshows spearman coefficient. As it is obvious in the graphs, the querycondition modeloutperforms query result model. The lostof quality is due the limited workload that is 10%.When workload increases, the quality also increases.

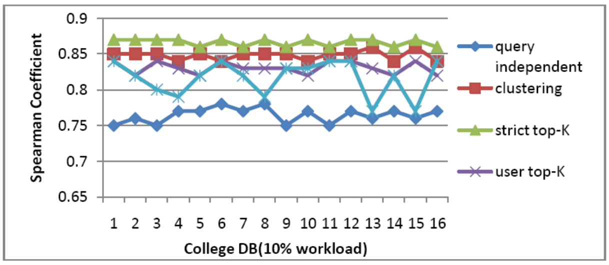


Figure 7 – Ranking quality of user similarity model (College DB)

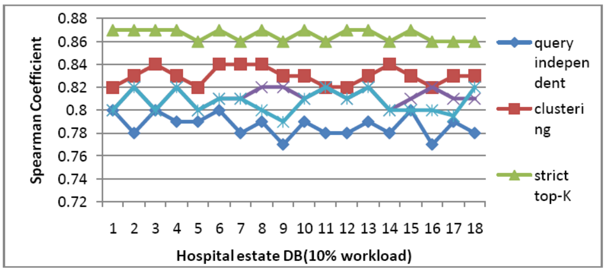


Figure 8 – Ranking quality of user similarity model (Hospital D)

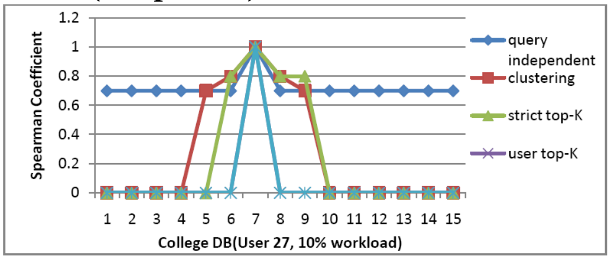


Figure 9 – Ranking quality of user similarity model (College DB)

Fig. 7, 8, and 9 show the average ranking quality achieved from both college and hospital database acrossall queries for all users. The results disclose that authoritarian top-K model performs bittern than other models. However,the strict top-K has no ranking functions for many queries.

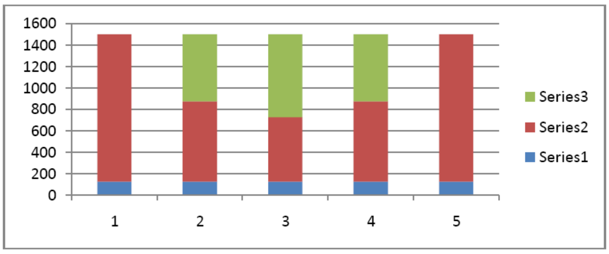


Figure10 – Ranking functions derived for user similarity (College DB)

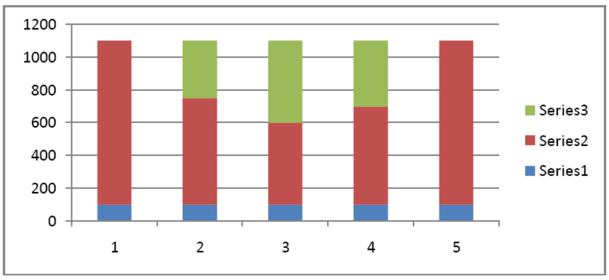


Figure 11 – Ranking functions derived for user similarity (Hospital DB)

Fig. 10 and 11 confirm the fact that different models have different abilities for determining rankingfunctions across the workload. However, the strict top-K is precise and superior to all other models fromthe perspective of ranking function.

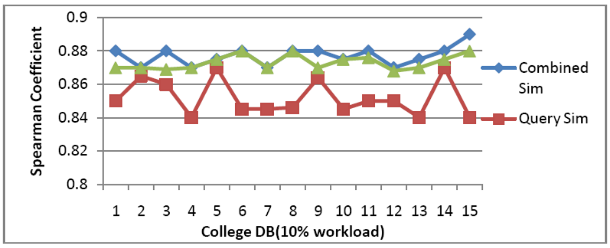


Figure 12 – Ranking quality of combined similarity model

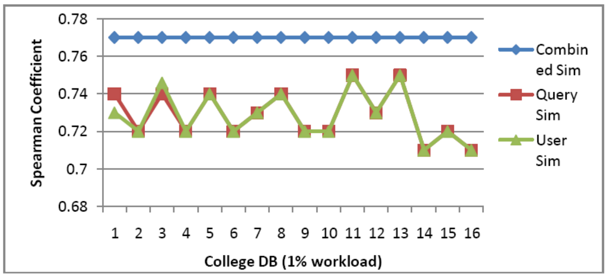


Figure 13 – Ranking quality of combined similarity model

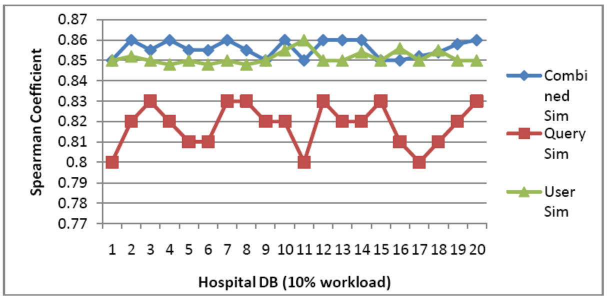


Figure 14 – Ranking quality of combined similarity model

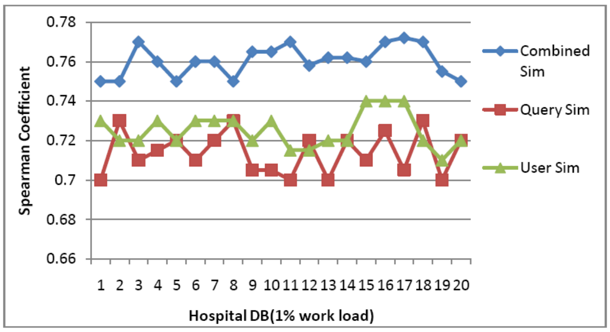


Figure 15 – Ranking quality of combined similarity model

Fig. 12, 13, 14 and 15 show the quality of combined models for both databases with 1% and 10% workload. Theimportant observation is that the composite model is performing better than other individual models. Anotherfact established here is that with more ranking functions in workload better similarity and quality of results isachieved.

6. CONCLUSION

This paper proposed a new ranking model for ranking query results of databases. We used two synthetic databases for examinations. They are college database and hospital database. The model is basedon both query similarity and user correspondence. We have also created a prototype web based application thatdemonstrates the efficiency of the proposed ranking pattern. A workload file is kept up that continuallystores updated ranking functions for both user similarity and query similarity. When a new query is made, thisworkload file is used for giving ranking to the query results. Designing and keeping up a workload ischallenging in the context of web databases. We have implemented an algorithm for estimating user and querysimilarities and update workload persistently. The examination results disclose that our new ranking pattern works well and it can be explored for real world web databases.

7.REFERENCES

[1] S. Agrawal, S. Chaudhuri and G. Das.DBXplorer: A Systemfor Keyword Based Search over Relational Databases. ICDE2002.

[2] S. Agrawal, S. Chaudhuri, G. Das and A. Gionis.AutomatedRanking of Database Query Results.TechnicalReport,Microsoft Research, in preparation.

[3] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A. I.Verkamo.Fast Discovery of Association Rules.AdvancesinKnowledge Discovery and Data Mining, 1995.

[4] R. Baeza-Yates and B. Ribeiro-Neto.ModernInformationRetrieval.ACM Press, 1999.

[5] J. Breese, D. Heckerman and C. Kadie.EmpiricalAnalysisof Predictive Algorithms for Collaborative Filtering.14th Conference on Uncertainty in Artificial Intelligence, 1998.

[6] N. Bruno, L. Gravano, and S. Chaudhuri. Top-K Selection Queries over Relational Databases: MappingStrategies and Performance Evaluation. ACM Transactions on Database Systems (TODS), vol. 27, no. 2, June 2002.

[7] N. Bruno, L. Gravano, A. Marian.Evaluating Top-K Queriesover Web-Accessible Databases.ICDE 2002.

[8] M. J. Carey and D. Kossmann. On Saying "EnoughAlready!" in SQL. SIGMOD 1997, 219-230.

[9] M. J. Carey and D. Kossmann. Reducing the BrakingDistance of an SQL Query Engine.VLDB 1998.158-169.

[10] K. Chakrabarti, K. Porkaew and S. Mehrotra.EfficientQuery Ref. in Multimedia Databases.ICDE 2000.

[11] S. Chaudhuri and V. Narasayya.Program for TPC-D DataGeneration with Skew. http://research.microsoft.com/dmx/AutoAdmin.

[12] W. Cohen. Integration of Heterogeneous DatabasesWithout Common Domains Using Queries Based on TextualSimilarity. SIGMOD, 1998.

[13] W. Cohen. Providing Database-like Access to the WebUsing Queries Based on Textual Similarity.SIGMOD 1998.

[14] R. Fagin. Fuzzy Queries in Multimedia Database Systems.PODS 1998.

[15] R. Fagin, A. Lotem and M. Naor.OptimalAggregationAlgorithms for Middleware.PODS 2001.

[16] C. Faloutsos and K-I. Lin. Fastmap: A Fast Algorithm forIndexing, Data mining and Visualization of Traditional andMultimedia Datasets. SIGMOD 1995.