ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 8, August 2015

Creation of Dynamic Query Forms and Ranking of its Components based on User's Preference

Greeshma Radhakrishnan, Dr. S. Sasidhar Babu

M.Tech Student, Department of CSE, SNGCE, Kadayiruppu, Kerala, India

Professor, Department of CSE, SNGCE, Kadayiruppu, Kerala, India

ABSTRACT: At present real world databases maintain large and heterogeneous data which contain thousands of attributes and relations. Accessing the information from this corpulent database is a nontrivial task. The current system uses a static approach to query these databases.These static query forms are created manually and can express only a very limited set of queries. It is very difficult to create and design a set of static question forms to answer various ad-hoc database queries on the modern complex databases. So there is need of such a system that dynamically generates a Query Forms according to the user's desire at run time. In this paper, we propose a mechanism to let a user modify the generated query form to express the interested query. The main idea is to get user preferences through user interactions. DQF aims to select appropriate query components. The system also adapts a question type iteratively. At each iteration DQF provides a ranked list of query form component to user. A user can fill the query form by selecting the desired form components and can submit it to view the query result.. User checks the result and if satisfied with the result then the system will give ranking of the query by providing some suggestions. The user can click the suggestion and obtains a ranked list of the components. For finding the goodness of the query form components, some probabilistic measures are used.

KEYWORDS: Dynamic Query Form, ranking metric, precision, recall, Fscore

I. INTRODUCTION

One of the most widely used user interfaces for querying databases is the query form. Traditional query forms such as static query forms are designed and developed by the developers manually. Due to the development of the web information and scientific databases, the databases become very large and more complex, so that the developer can't predict all the queries while developing the query form. If a user is unable to convey to the database what he or she wants from it, even the richest data store provides little or no value. Writing well-structured queries, in languages such as SQL and XQuery, can be challenging. This can be due to user's lack of familiarity with the query language and the underlying schema.

In natural sciences, such as genomics and diseases, the databases have over hundreds of entities for chemical and biological data resources. Many web databases, such as Freebase and DBPedia, typically have thousands of structured web entities.Therefore, designing such a static inquiry form to satisfy various adhoc queries on those complex databases is more difficult. There are various existing systems like Static query forms (SQF) and customized query forms (CQF) that have lot of drawbacks as it does not have query refinement and no dynamic nature to solve adhoc queries. Many existing database management and development tools, such as Easy Query, Cold Fusion, SAP and Microsoft Access, provide several mechanisms to let users create customized queries on databases. However, the creation of customized queries totally depends on user's manual editing. The manual form generation is no longer easy as queries and schema become more complex.

The most common form of query is a static query. A static query is one that will be passed to the database. Only Select queries may be static. Only very simple queries should use the static query mechanism. If we need more complex queries, dynamic query forms can be used. So a dynamic query form addresses these challenges by dynamically

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 8, August 2015

generating the query forms. Here the proposed system is a Dynamic Query Form system: DQF. It is a query interface which is capable of dynamically generating query forms for users. Users are often willing to perform many rounds of actions before identifying the final candidates which is different from many information retrieval systems. The essence of DQF is to capture user's preferences during user interactions and also to adapt the query forms iteratively. Each iteration consists of two types of user interactions: Query Form Enrichment and Query Execution. The basic query form is then enriched iteratively via the interactions between the user and our system. It very user friendly which means we can easily select tables. Then it will automatically show the attributes present in that table. The users have the facility for filling up the query form and submit it to the system. The system will generate the query results along with some suggestions. The suggestions provided will be the ranked list of attributes. By refining the form components in this way the user can choose the desired attributes according to his/her preference for it. This is done until the user gets satisfied with the query results. Here in this paper we mainly study the ranking and the dynamic generation of query forms based on user's preference. This dynamic approach will leads to a higher success rate and it also generates simpler query forms when compared to the static approach. The ranking of form components makes it easier for users to customize query forms.

The goodness of a query form is estimated by applying F-measure. F-measure is a typical metric to evaluate query results. This metric is also appropriate for query forms because query forms are designed to help users query the database. The goodness of a query form is determined by the query results generated from the query form. Based on this, the potential query form components are ranked and recommended so that users can refine the query form easily. Based on the metric, develops efficient algorithms to estimate the goodness of the projection and selection form components. Here efficiency is important because DQF is an online system where users often expect quick response.

II. RELATED WORK

A lot of research works focus on database interfaces which assist users to query the relational database without SQL. The two most widely used database querying interfaces are QBE (Query-By- Example) [21] and Query Form. At present, query forms have been utilized in most real-world business or scientific information systems. Current studies and works mainly focus on how to generate the query forms. In [12] proposed a system which allows end-users to customize the existing query form at run time. Recently, [11], [5] proposed automatic approaches to generate the database query forms without user participation. In [3], [15], novel user interfaces have been developed to assist the user to type the database queries based on the query workload. Query refinement is a common practical technique used by most information retrieval systems [20] [2] develops an adaptive forms system for data entry, which can be dynamically changed according to the previous data input by the user.

Existing database clients and tools make great efforts to help developers design and generate the query forms, such as EasyQuery [17], Cold Fusion [19], SAP, Microsoft Access and so on. They provide visual interfaces for developers to create or customize query forms. The problem of those tools is that, they are provided for the professional developers who are familiar with their databases, not for end-users [11]. [12] Proposed a system which allows end-users to customize the existing query form at run time. However, an end-user may not be familiar with the database. If the database schema is very large, it is difficult for them to find appropriate database entities and attributes and to create desired query forms.

M. Jayapandian *et al.* [11] [5] proposed automatic approaches to generate the database query forms without user participation. [11] Presented a data driven method. It first finds a set of data attributes, which are most likely queried based on the database schema and data instances. Then, the query forms are generated based on the selected attributes. [5] is a workload-driven method. It applies clustering algorithm on historical queries to find the representative queries. The query forms are then generated based on those representative queries. One problem of the aforementioned approaches is that, if the database schema is large and complex, user queries could be quite diverse. In that case, even if we generate lots of query forms in advance, there are still user queries that cannot be satisfied by any one of query forms. Another problem is that, when we generate a large number of query forms, how to let users find an appropriate and desired query form would be challenging. A solution that combines keyword search with query form generation is proposed in [9]. It automatically generates a lot of query forms in advance. The user inputs several keywords to find relevant query forms from a large number of pre-generated query forms. It works well in the databases which have rich textual information in data tuples and schemas. However, it is not appropriate when the user does not have concrete keywords to describe thequeries at the beginning, especially for the numeric attributes.

N. Khoussainova *et al*.[3], [15], novel user interfaces have been developed to assist the user to type the database queries based on the query workload, the data distribution and the database schema. Different from our work which focuses on query forms, the queries in their work are in the forms of SQL and keywords.

W. B. Frakes *et al*.[20] proposed a query refinement for the query forms. Query refinement is a common practical technique used by most information retrieval systems. It recommends new terms related to the query or modifies the terms according to the navigation path of the user in the search engine. But for the database query form, a database query is a structured relational query, not just a set of terms.

H. Wang *et al*.[6],[4] proposed Dynamic faceted search which is a type of search engines where relevant facets are presented for the users according to their navigation paths. Dynamicfaceted search engines are similar to our dynamic query forms if we only consider *Selection* components in a query. However, besides *Selections*, a database query form has other important components, such as *Projection* components. *Projection* components control the output of the query form and cannot be ignored. Moreover, designs of *Selection* and *Projection* have inherent influences to each other.

III. QUERY FORMS

Query form is one of the majority used user interfaces for querying databases. Form-based interfaces are used frequently, but usually each form is designed in an adhoc way and its applicability is restricted to a small set offixed queries.The traditional approach to querying is to use a form fill-in interface, but such an approach leads to user frustration when the query returns either zero hits or a very large number of hits. Often, users cannot even estimate the total number of hits their query would have returned as the system only returns the first 25 - 50 hits. It is difficult to estimate how much data is available on a given topic and how to increase or reduce result set sizes. Dynamic Queries involve the interactive control by a user of visual query parameters that generate a rapid and visual display of database search results. It allows users to perform rapid and dynamic elimination of undesired data.

IV. DYNAMIC QUERY FORM

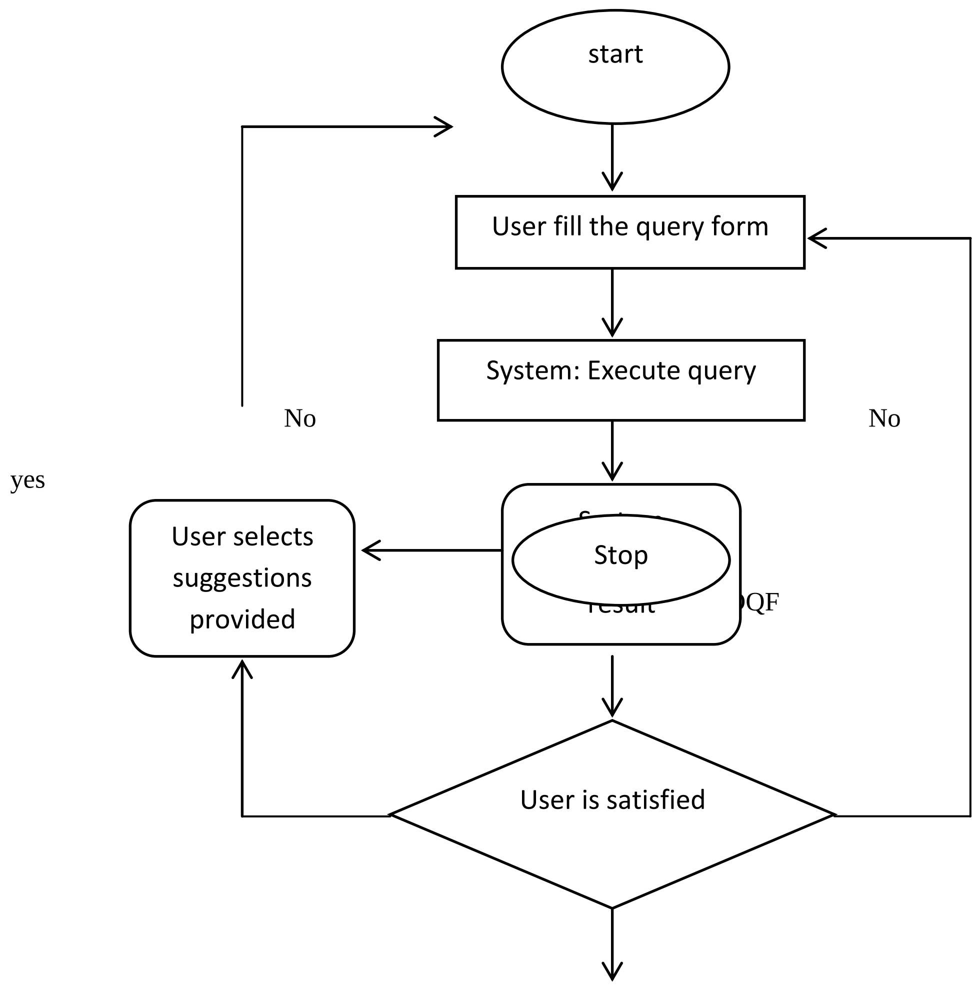
In this paper, the main focus is to capture user's preference while generating a query form.Dynamic query from allows user to generates the query form dynamically according to his/her preference. We will allow users to generate a dynamic query form and execute the query results until he gets required query result. We will allow user to rank the query Form. F-Measure will be used here for ranking a query form. DQF will actually work in two phases one is Query Execution and Query enrichment. In Query execution user will fill the query form with desired form components, query form will be executed by the system and result will be displayed. If user gets desired output of that query form then he will rank the query form for future use. Depending on the ranking, query form will be restored for further use. And then he can exit from the systems else he will enriched the query form by selecting other form components of his desire to get expected query result. After that he will execute the query form and the process will repeat until user will get desired output.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 8, August 2015



A. Query Processing

This system supports to captures user interests during user interactions. User interests can be iteratively to form desired result of user. Each iteration has two stages to interact with user. In the first stage, user can interact to enrich query form. Second stage user can interact to execute query. Dynamic query form generates a ranked list of query form components to the user. Then user selects the desired form components from that list to find desired result. Then user can submit the query by submitting the current query form. This displays the desired query results and based on selected components. On this displayed results user can provide feedback to the system about the query results retrieved. Thus the system provides a double filtering mechanism in order to refine the query forms.

B. Ranking score estimation

For evaluating query results there are two measures available such as precision and recall. Based on different inputs provided to query forms can give different output in query results. To achieve expected query result we will be using expected recall and expected precision. Expected precision is the proportion of the query results which is interested by current user and expected recall is the proportion of users expected interested data instances which are returned by current query form. For ranking score estimation two components will be ranked one is projection and another is selection [17]. In projection components ranking, entities (Tables) and their respective attributes (Columns) will be ranked. Here attribute with maximum F-score will be selected. And in raking selection form components first important attribute will be selected and ranked, here the F-score would be computed incrementally on desired attributes.

C. Quality Metric

The quality of a particular query result can be depending on precision and recall results. Using query form one can produce different input and different inputs can produce different outputs of query results, it means query form can achieve different precision and recall. So using expected recall and expected precision expected performance of query result can be calculated. Above probabilistic models are used to find expected recall and expected precision. Using both recall and precision metrics F-measure can be calculated; ultimately performance of query form can be evaluated.

V. QUERY RESULT

To check the query form is useful or not, user does not have time to go through every result .To avoid the many answer problem [16] in database here uses a clustering technique. Then, the user can click through interested clusters to view the detailed data instances. There are many one-pass clustering algorithms for generating the compressed view efficiently [18]. In our implementation, we choose the incremental data clustering framework [18] because of the efficiency issue certainly, different data clustering methods [1].

Another important usage of the compressed view is to collect the user feedback. Using the collected feed-back, we can estimate the goodness of a query form so that we could recommend appropriate query form components. In real world, end-users are reluctant to provide explicit feedback [14]. The click-through on the compressed view table is an implicit feedback to tell our system which cluster (or subset) of data instances is desired by the user. But it can help our system generate recommended form components that help users discover more desired data instances. In some recommendation systems and search engines, the end-users are also allowed to provide the negative feedback. The negative feedback is a collection of the data instances that are not desired by the users. In the query form results, we assume most of the queried data instances are not desired by the users because if they are already desired, then the query form generation is almost done. Therefore, the positive feedback is more informative than the negative feedback in the query form generation. Our proposed model can be easily extended for incorporating the negative feedback.

VI. SIMULATION RESULTS

We implemented the dynamic query forms as a web based system using JDK 1.6 with Java Server Page. The dynamic web interface for the query forms used open source JavaScript library jQuery 1.4. We used MySQL 5.1.39 as the database engine. All experiments were run using a machine with Intel Core 2 CPU @2.83GHz, 3.5G main memory.
We compare two approaches to generate query form for the given datasets.
• DQF: The dynamic query form system proposed in this paper.
• SQF: The static query form generation approach uses query workload. Queries in the workload are first divided into clusters. Each cluster is converted into a query form.

The run-time cost of ranking projection and selection components for DQF depends on the current form components and the query result size. Thus we selected 4 complex queries with large result size for each data set. The running times of ranking projection are all lessthan 1 millisecond, since DQF only computes the schema distance and conditional probabilities of attributes. Fig. 2shows the time for DQF to rank selection components forqueries on the data sets. The results show that the execution time grows approximately linearly with respect to the query result size. The execution time is between 1 to 3 seconds for one dataset when the results contain 4000 records, less than 0.11 second for anotherwhen the results contain 1600 records. So DQF can be used in an interactive environment.

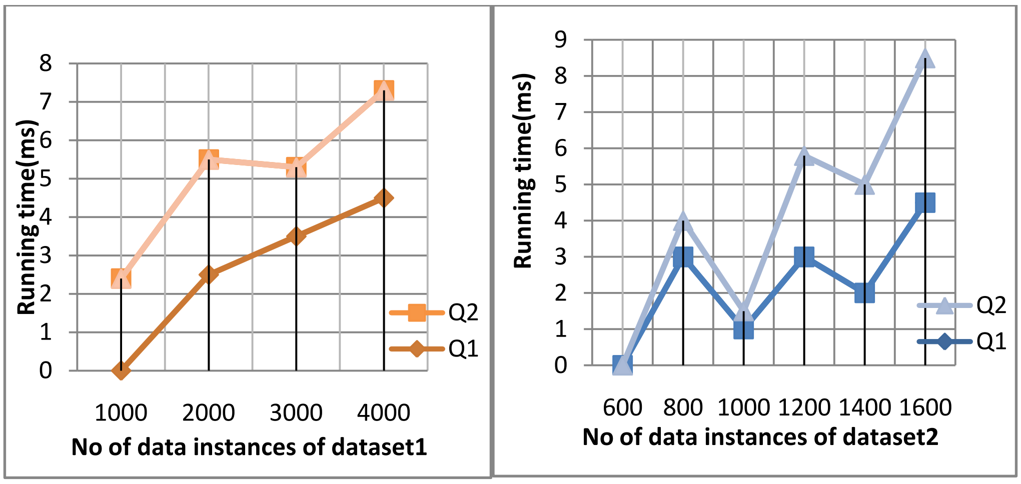


Fig.2.Execution time graph for different datasets

VII. CONCLUSION AND FUTURE WORK

In this paper we propose a dynamic query form generation approach which helps users dynamically generate query forms has been proposed. The key idea is to use a probabilistic model to rank form components based on user preferences. The user preference is captured using both historical queries and runtime feedback such as click-through. The dynamic approach leads to higher success rate and simpler query forms compared with a static approach. The ranking of form components also makes it easier for users to customize query forms. As future work, this approach can be extended to non-relational data and also develop multiple methods to capture the user's interest for the queries besides the click feedback. For instance, can add a text-box for users to input some keywords queries. The relevance score between the keywords and the query form can be incorporated into the ranking of form components at each step.

ACKNOWLEDGEMENT

First of all I would like to thank almighty god. I would like to thank my guide Dr. S.Sasidhar Babu, Professor in Computer Science and Engineering department in Sree Narayana Gurukulam College of Engineering, Kolenchery, Kerala. Also to my friends and family.

REFERENCES

1. Liang Tang, Tao Li, Yexi Jiang, and Zhiyuan Chen Dynamic Query Forms for Database Queries, in proceedings of TKDE.2013.62,pages 1041-4347,U.S.A,June 2013
2. K. Chen, H. Chen, N. Conway, J. M. Hellerstein, and T. S. Parikh, "Usher: Improving data quality with dynamic forms," in *Proc. ICDE*, Long Beach, CA, USA, Mar. 2010, pp. 321–332.
3. N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu, "Snipsuggest: Context-aware auto completion for SQL," *Proc. VLDB*, vol. 4, no. 1, pp. 22–33, 2010.
4. C. Li, N. Yan, S. B. Roy, L. Lisham, and G. Das, "Facetedpedia: Dynamic generation of query-dependent faceted interfaces for wikipedia," in *Proc. WWW*, Raleigh, NC, USA, Apr. 2010, pp. 651–660.
5. M. Jayapandian and H. V. Jagadish, "Automating the design and construction of query forms," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 10, pp. 1389–1402, Oct. 2009.
6. S. B. Roy, H. Wang, U. Nambiar, G. Das, and M. K. Mohania, "Dynacet: Building dynamic faceted search systems over databases," in *Proc. ICDE*, Shanghai, China, Mar. 2009, pp. 1463–1466.
7. Q. T. Tran, C.-Y. Chan, and S. Parthasarathy, "Query by output," in *Proc. SIGMOD*, Providence, RI, USA, Sept. 2009, pp. 535–548.
8. G. Chatzopoulou, M. Eirinaki, and N. Polyzotis, "Query recommendations for interactive database exploration," in *Proc. SSDBM*, New Orleans, LA, USA, Jun. 2009, pp. 3–18.
9. E. Chu, A. Baid, X. Chai, A. Doan, and J. F. Naughton, "Combining keyword search and forms for ad hoc querying of databases," in *Proc. ACM SIGMOD*, Providence, RI, USA, Jun. 2009, pp. 349–360.
10. N. Khoussainova, M. Balazinska, W. Gatterbauer, Y. Kwon, and D. Suciu, "A case for a collaborative query management system," in *Proc. CIDR*, Asilomar, CA, USA, Jan. 2009.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 8, August 2015

11. M. Jayapandian and H. V. Jagadish, "Automated creation of a forms-based database query interface," *Proc. VLDB*, vol. 1, no. 1, pp. 695–709, Aug. 2008.
12. M. Jayapandian and H. V. Jagadish, "Expressive query specification through form customization," in *Proc. Int. Conf. EDBT*, Nantes, France, Mar. 2008, pp. 416–427.
13. S. Zhu, T. Li, Z. Chen, D. Wang, and Y. Gong, "Dynamic active probing of helpdesk databases," *Proc. VLDB*, vol. 1, no. 1, pp. 748–760, Aug. 2008.
14. T. Joachims and F. Radlinski. Search engines that learn from implicit feedback. IEEE Computer (COMPUTER), 40(8):34–40, 2007.
15. A. Nandi and H. V. Jagadish, "Assisted querying using instant response interfaces," in *Proc. ACM SIGMOD*, Beijing, China, 2007, pp. 1156–1158.
16. S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic information retrieval approach for ranking of database query results. ACM Trans. Database Syst. (TODS), 31(3):1134–1168, 2006.
17. Korzh.com. (2005). *EasyQuery* [Online]. Available: http://devtools.korzh.com/eq/dotnet/
18. C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In Proceedings of VLDB, pages 81–92, Berlin, Germany, September 2003.
19. Adobe. (1995). *Cold Fusion* [Online]. Available: http://www.adobe.com/products/coldfusion/
20. W. B. Frakes and R. A. Baeza-Yates, *Information Retrieval: Data Structures and Algorithms*. Englewood Cliffs, NJ, USA: Prentice- Hall, 1992.
21. M. M. Zloof, "Query-by-example: The invocation and definition of tables and forms," in *Proc. VLDB*, Framingham, MA, USA, Sept. 1975, pp. 1–14.