Volume 4, Issue 10, October 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

An Integrated Mining Framework using Dynamic Query Forms

B. Ramaganesh
Associate professor (H.O.D),
Department of Computer Science Engineering
VEMU Institute of Technology, Chittoor Dist, A.P, India

Lokanath J C
IV SEM Master of Technology,
Department of Computer Science Engineering,
VEMU Institute of Technology, Chittoor Dist, A.P, India

Abstract*— Modern applications are based on huge very huge data. Working with such a vast data requires more efforts in forming queries for analyzing the data, this being the major issue in the existing systems implemented based on static predefined queries in the present day. Even though there are fixed number of queries being in a finite loop, identifying the best query among the listed queries will again be a big challenge to the users as it evidences large number of queries. Apart from these, one more major problem is of refining the query for the accurate results based on the user inputs. The current proposal of approach drives a solution to these problems prioritizing more user friendly format by allowing user to perform all the operations on the data. The system representing Dynamic Query Form will allow the user to interact with the data source with a dynamic interface by giving them a freedom to form the customized queries for analyzing the data. The next key approach is of ranking method based on the user preferences which will make the frequent queries available on the user profile. The resultant approach leads in more reliable and efficient way of handling large and complex database schemas with great quality by prioritizing user satisfaction.

Keywords*— Ranking, DQF-Dynamic Query Form, Query Form, User Interaction, Query Form Generation.

I. INTRODUCTION

Query structure is a standout amongst the most generally utilized client between faces for questioning databases. Customary question structures are outlined and predefined by designers or DBA in different data administration frameworks. With the quick improvement of web data and scientific databases, present day databases get to be extensive and complex. In common sciences, for example, genomics and infections, the databases have over many elements for substance and organic information assets. Numerous web databases, for example, Freebase and DBPedia, ordinarily have a great many organized web elements. In this way, it is hard to outline a set of static Query structures to fulfil different specially appointed database questions on those complex databases. Numerous existing database administration and improvement devices, for example, Oracle BI, Microsoft BI, EasyQuery, Cold Fusion, SAP, Siebel and Microsoft Access, give a few systems to let clients make altered questions on databases. Be that as it may, the making of modified questions completely relies on upon clients' manual altering. In the event that a client is not acquainted with the database composition ahead of time, those hundreds or a huge number of information properties would befuddle him/her.

II. OUR APPROACH

In this paper, we propose a Dynamic Query Form framework: DQF, a question interface which is equipped for alertly producing Query structures for clients. Not the same as customary record recovery, clients in database recovery are regularly ready to perform numerous rounds of activities (i.e., refining question conditions) before recognizing the final applicants. The embodiment of DQF is to catch client diversions amid client collaborations and to adjust the Query structure iteratively. Every cycle comprises of two sorts of client connections: Query Form Enrichment and Query Execution (see Table 1). Figure 1 demonstrates the workflow of DQF. It starts with a basic query form

Table 1: Interactions between Users and DQF

Query Form Enrichment	1) DQF recommends a ranked list of query form components to the user. 2) The user selects the desired form components into the current query form.
Query Execution	1) The user fills out the current query form and submit a query. 2) DQF executes the query and shows the results. 3) The user provides the feedback about the query results.

This contains very few primary attributes of the database. The basic query form is then enriched iteratively via the interactions between the user and our system until the user is satisfied with the query results. In this paper, we mainly study the ranking of query form components and the dynamic generation of query forms.

III. CONTRIBUTIONS

Our contributions can be summarized as follows:

• We propose a dynamic query form system which generates the query forms according to the user's desire at run time. The system provides a solution for the query interface in large and complex databases.

• We apply F-measure to estimate the goodness of a query form. 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, we rank and recommend the potential query form components so that users can refine the query form easily.

• Based on the proposed metric, we develop 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.

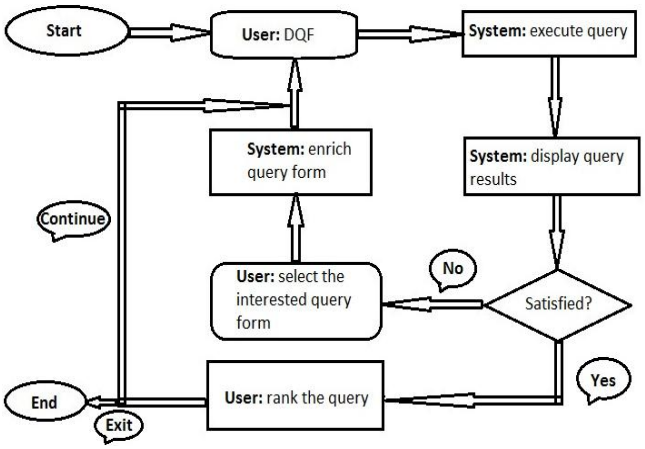

Fig. 1. Flowchart of Dynamic Query Form in Mining framework

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 defines the query form and introduces how users interact with our dynamic query form. Section 4 defines a probabilistic model to rank query form components. Section 7 concludes the paper.

IV. RELATED WORKS

How to let non-expert users make use of the relational database is a challenging topic. A lot of research works focus on database interfaces which assist users to query the relational database without SQL. QBE (Query-By-Example) and Query Form are two most widely used database querying interfaces. 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.

Customized Query Form: Existing database customers and apparatuses endeavour incredible deliberations to help designers outline and create the question structures, for example, Oracle BI, Microsoft BI, Siebel, EasyQuery, Cold Fusion, SAP, Microsoft Access and so on.

They give visual interfaces to engineers to make or modify question structures. The issue of those devices is that, they are accommodated the expert designers who are acquainted with their databases, not for end-clients proposed a framework which permits end-clients to modify the current question structure at run time. Then again, an end-client may not be acquainted with the database. On the off chance that the database diagram is extensive, it is troublesome for them to find suitable database elements and credits and to make coveted Query structures.

Automatic Static Query Form: Recently, expert postured programmed methodologies to produce the database question structures without client support. Introduced information driven strategy. It first finds a set of information qualities, which are probably questioned focused around the database mapping and information occurrences. At that point, the Query structures are produced focused around the chose qualities is a workload-driven system. It applies grouping calculation on chronicled inquiries to find the agent questions. The Query structures are then produced focused around those agent questions. One issue of the previously stated methodologies is that, if the database mapping is extensive and complex, client questions could be very different. All things considered, regardless of the possibility that we produce heaps of Query structures ahead of time, there are still client questions that can't be satisfied by any of question structures. An alternate issue is that, when we produce an extensive number of Query structures, how to let client's find a fitting and sought question structure would be testing. An answer that joins watchword look with Query structure era is proposed in. It naturally produces a great deal of Query structures ahead of time. The client inputs a few magic words to find important question structures from countless created Query structures. It functions admirably in the databases which have rich printed data in information tuples and constructions. Nonetheless, it is not proper when the client does not have solid watchwords to depict the inquiries at the starting, particularly for the numeric traits.

Autocompletion for Database Queries: In novel client interfaces have been produced to support the client to sort the database questions focused around the question workload, the information dispersion and the database pattern. Not the same as our work which concentrates on Query structures, the questions in their work are in the manifestations of SQL and essential words.

Query Refinement: Query refinement is a typical down to earth method utilized by most data recovery frameworks. It proposes new terms identified with the Query or modifies the terms as indicated by the route way of the client in the web search tool. At the same time for the database Query structure, a database question is an organized social Query, not only a set of terms.

Dynamic Faceted Search: Dynamic faceted pursuit is a sort of web indexes where important features are exhibited for the clients as indicated by their route ways. Element faceted web search tool are like our element Query structures if we consider Selection segments in a question. In any case, other than Selections, a database Query structure has other paramount parts, for example, Projection segments. Projection segments control the yield of the question structure and can't be overlooked. Also, outlines of Selection and Projection have inborn influences to one another.

Database Query Recommendation: Recent studies acquaint community oriented methodologies with prescribe database question segments for database investigation. They treat SQL inquiries as things in the collective filtering approach, and propose comparable questions to related clients. In any case, they don't consider the integrity of the question results proposes a system to prescribe an option database Query focused around consequences of a Query. The distinction from our work is that, their proposal is a complete question and our suggestion is a Query part for every emphasis.

Dynamic Data Entry Form: creates a versatile structure framework for information passage, which can be progressively changed as indicated by the past information include by the client. Our work is different as we are dealing with database query forms instead of data-entry forms.

V. QUERY FORM INTERFACE

Query Form: In this section we formally defines the query form. Each query form corresponds to an SQL query template.

Definition 1: A query form F is defined as a tuple (AF, RF, σF, (RF)), which represents a database query template as follows:

$$F = (\text{SELECT } A1,A2,...,Ak \\ \text{FROM } \triangleleft\ \triangleright\ (RF) \text{ WHERE } \sigma F),$$

Where AF = {A1, A2,..., Ak} are k attributes for projection, k > 0. RF = {R1, R2,..., Rn} is the set of n relations (or entities) involved in this query, n > 0. Each attribute in AF belongs to one relation in RF. σF is a conjunction of expressions for selections (or conditions) on relations in RF. $\triangleleft\ \triangleright$ (RF) is a join function to generate a conjunction of expressions for joining relations of RF.

In the user interface of a query form F, AF is the set of columns of the result table. σF is the set of input components for users to fill. Query forms allow users to fill parameters to generate different queries. RF and $\triangleleft\ \triangleright$ (RF) are not visible in the user interface, which are usually generated by the system according to the database schema. For a query form F, $\triangleleft\ \triangleright$ (RF) is automatically constructed according to the foreign keys among relations in RF. Meanwhile, RF is determined by AF and σF. RF is the union set of relations which contains at least one attribute of AF or σF. Hence, the components of query form F are actually determined byAF and σF. As we mentioned, only AF and σF are visible to the user in the user interface. In this paper, we focus on the projection and selection components of a query form. Ad-hoc join is not handled by our dynamic query form because join is not a part of the query form and is invisible for users. As for ”Aggregation” and ”Order by” in SQL, there are limited options for users. For example, ”Aggregation” can only be MAX, MIN, AVG, and so on; and ”Order by” can only be ”increasing order” and ”decreasing order”. Our dynamic query form can be easily extended to include those options by implementing them as dropdown boxes in the user interface of the query form.

Query Results: To decide whether a query form is desired or not, a user does not have time to go over every data instance in the query results. In addition, many database queries output a huge amount of data instances. In order to avoid this “Many-Answer” problem, we only output a compressed result table to show a high- level view of the query results first. Each instance in the compressed table represents a cluster of actual data instances. Then, the user can click through interested clusters to view the detailed data instances. Figure 2 shows the flow of user actions. The com- pressed high-level view of query results is proposed in. There are many one-pass clustering algorithms for generating the compressed view efficiently. In our implementation, we choose the incremental data clustering framework because of the efficiency issue. Certainly, different data clustering methods would have different compressed views for the users. Also, different clustering methods are preferable to different data types. In this paper, clustering is just to provide a better view of the query results for the user. The system developers can select a different clustering algorithm if needed.

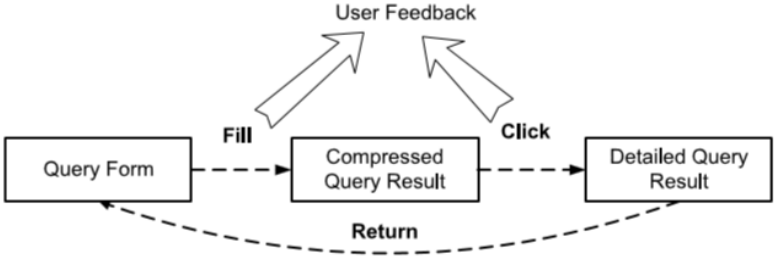

Fig. 2. User Actions

Another important usage of the compressed view is to collect the user feedback. Using the collected feedback, 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. 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. The clicked subset is denoted by Duf. Note that Duf is only a subset of all user desired data instances in the database. 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. RANKING METRIC

Query forms are designed to return the user's de- sired result. There are two traditional measures to evaluate the quality of the query results: precision and recall. Query forms are able to produce different queries by different inputs, and different queries can output different query results and achieve different precisions and recalls, so we use expected precision and expected recall to evaluate the expected performance of the query form. Intuitively, expected precision is the expected proportion of the query results which are interested by the current user. Expected recall is the expected proportion of user interested data instances which are returned by the current query form. The user interest is estimated based on the user's click- through on query results displayed by the query form. For example, if some data instances are clicked by the user, these data instances must have high user interests. Then, the query form components which can capture these data instances should be ranked higher than other components.

Algorithm Scope: Query Construction

Algorithm Name: Ranking Algorithm
Input Arguments: Query Id, Date of Ranking, Rank, Usage Frequency
Output Arguments: List of Queries based on the criteria's
Algorithm:

```
Do
{
While (Query Id==Null)
{
If (Rank ==Maximum && Usage Frequency==Very High)
{
Add the query to the wish list;
}
Query Id--;
}
Usage Frequency--;
Rank--;
}
While (Rank ==least && Usage Frequency==low);
```

VII. CONCLUSIONS

In this paper we propose a dynamic query form generation approach which helps users dynamically generate query forms. The key idea is to use a probabilistic model to rank form components based on user preferences. We capture user preference using both historical queries and run-time feedback such as click- through. Experimental results show that the dynamic approach often 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, we will study how our approach can be extended to non relational data. As for the future work, we plan to develop multiple methods to capture the user's interest for the queries besides the click feedback. For instance, we 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.

ACKNOWLEDGMENT

B Ramaganesh: Ramaganesh is currently working as an associate professor and HOD in the department of computer science and engineering, Vemu Institute of technology, Kothakota. His research interests are Data Mining, Software Engineering and Information Retrieval.

Lokanath J C: Lokanath received the Bachelor degree in Information Technology from the Department of Information Technology, JNTU Anantapur, Anantapur in 2011. He is currently a Master Degree student in the Computer Science and Engineering, Vemu Institute of technology affiliated to JNTU Anantapur, Ananatapur. His research interests are data mining, data warehouse, information retrieval and web designing.

REFERENCES

[1] EasyQuery. http://devtools.korzh.com/eq/dotnet/.
[2] Cold Fusion. http://www.adobe.com/products/coldfusion/.
[3] DBPedia. http://DBPedia.org.
[4] Freebase. http://www.freebase.com.
[5] 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.
[6] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diver- sifying search results. In Proceedings of WSDM, pages 5–14, Barcelona, Spain, February 2009.
[7] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated ranking of database query results. In CIDR, 2003.
[8] S. Boriah, V. Chandola, and V. Kumar. Similarity measures for categorical data: A comparative evaluation. In Proceedings of SIAM International Conference on Data Mining (SDM 2008), pages 243–254, Atlanta, Georgia, USA, April 2008.
[9] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. Query recom- mendations for interactive database exploration. In Proceedings of SSDBM, pages 3–18, New Orleans, LA, USA, June 2009.
[10] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Proba- bilistic information retrieval approach for ranking of database query results. ACM Trans. Database Syst. (TODS), 31(3):1134– 1168, 2006.
[11] K. Chen, H. Chen, N. Conway, J. M. Hellerstein, and T. S. Parikh. Usher: Improving data quality with dynamic forms. In Proceedings of ICDE conference, pages 321–332, Long Beach, California, USA, March 2010.
[12] E. Chu, A. Baid, X. Chai, A. Doan, and J. F. Naughton. Combining keyword search and forms for ad hoc querying of databases. In Proceedings of ACM SIGMOD Conference, pages 349–360, Providence, Rhode Island, USA, June 2009.
[13] S. Cohen-Boulakia, O. Biton, S. Davidson, and C. Froidevaux. Bioguidesrs: querying multiple sources with a user-centric perspective. Bioinformatics, 23(10):1301–1303, 2007.

