



International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Random Query Formulation for Database Queries

Gopi Krishna Lakkasani

M.Tech, CSE Department, Madanapalle Institute of Technology & Science, JNTUA, Madanapalle, AP, India

Balakrishna Nayudori

Asst. Professor, CSE Department, Madanapalle Institute of Technology & Science, JNTUA, Madanapalle, AP, India

Abstract --Query form is one of the most widely used user interfaces for querying databases. Traditional query forms are designed and pre-defined by developers or DBA in various information management systems. With the fast improvement of web information and scientific databases, recent databases become very large and composite. Therefore, it is difficult to design a set of static query forms to assure various ad-hoc database questions lying on those complex databases. This paper proposes a Random Query Formulation (RQF) for database queries, a novel database query form interface, which is able to dynamically produce query forms, for both relational and non-relational (unstructured) data.

Keywords: Query form, database, ad-hoc, relational, random query formulation, CAPTCHA, refinement, enrichment.

I. INTRODUCTION

A database is only as functional as its query interface allows it to be. If a user is not capable to communicate to the database what he or she wishes from it, even the richest data store provides petite or no value. Writing well-structured queries, in languages such as SQL and XQuery, can be challenging due to a number of reasons, including the user's lack of familiarity with the query language and the user's ignorance of the underlying schema. A form-based query interface, which only requires filling blanks to identify query parameters, is precious since it helps make data users with no knowledge of official query languages or the database schema. In practice, 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 of fixed queries.

Query form is one of the majority used user interfaces for querying databases. Traditional query forms are designed and predefined by developers or DBA in various information management systems. With the rapid development of web information and scientific databases, modern databases become very large and complex.

Dynamic Query Form framework (DQF), an inquiry outskirt which is fit for eagerly creating Query forms for clients. Not quite the same as routine report rescue, clients in database recuperation are as often as possible eager to do part of rounds of activities (that is refining nature's domain) before distinguishing a definitive competitors. The heart of DQF is to bind client profits amid client correspondences and to adjust the Query form over and over.

II. RELATED WORK

A. Static forms:

The existing solution is using loading of static forms from the stored forms list. If any updates are made to the database they are not reflected in the forms which provide the user old forms which are not modified as in the database. If any data is added and user wants to retrieve it, the static forms will not reflect the added data in the database.

B. Dynamic Query Form (DQF):

Dynamic Query Form system: DQF, a query interface which is capable of dynamically generating query forms for users. Different from traditional document retrieval, users in database retrieval are often willing to perform many rounds of actions (i.e., refining query environment) before identifying the final candidates. The heart of DQF is to capture user interests through user communications and to adapt the query form repeatedly.

C. Disadvantages

- 1. The existing dynamic query-forms does not deal with unstructured data that is stored in text files.
- 2. The existing solution are less secure, they are susceptible to web bots.
- 3. It can be used for finding the percentage of data accessed but can't find the agent who accessed the data.
- 4. It is more time consuming process.
- 5. It is not applicable for group based and multilevel data.

III. PROPOSED WORK

RQF(Random Query Formulation): RQF generates query form dynamically and adds a secure CAPTCHA to the generated query form to make it more secure. The heart of RQF is to capture user interests during user interactions and to adjust the query form repeatedly. Each iteration consists of two types of user interactions: Query Form Enrichment and

Lakkasani et al., International Journal of Advanced Research in Computer Science and Software Engineering 4(8), August - 2014, pp. 262-265

Query Execution. Query Form Enrichment is used to enrich the generated query form based on the user feedback. The feedback is used to rank the generated query forms.

A. Advantages of proposed system

- 1. The RQF system works for unstructured data and structured data.
- 2. RQF provides two fold security by using a Security question and CAPTCHA.
- 3. RQF uses a new CAPTCHA called "Object based CAPTCHA" which is more secure that ordinary text based CAPTCHA.
- 4. The techniques used in these are simple and practical.

The RQF involves four basic sub-systems namely Query Generation, CAPTACHA generation, Query Refinement and Query Ranking. The architecture of the proposed work is illustrated in figure-1

- 1. Query Generation
- 2. CAPTACHA generation:
- 3. Query Refinement Module
- 4. Query Form Ranking Module

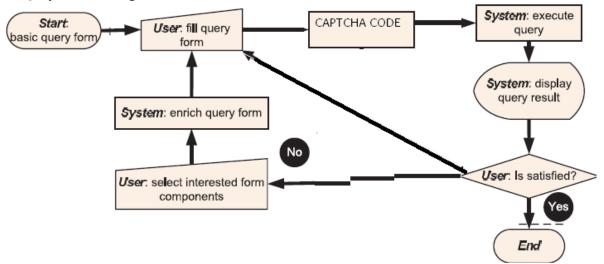


Fig-1: Architecture of RQF

B. Ouery Generation:

Query form is generated based on the user selection of the database and the fields in the database. DSQF supports multiple databases, hence this module must be able to allow the user to select the database and the fields in the database.

C. CAPTACHA generation:

This modules involves the generation of CAPTCHA image, which involves many sub systems as follows

CAPTCHA Engine: it is responsible for generating the text required for the captcha image.

The different algorithms available are

Dictionary words: Dictionary available online.

Random String: Challenge string composed of random character and words.

Markov text: The algorithm starts with an

- (i) Arbitrary letter pair or diagram. At each state it chooses the next character according to probabilities derived form the input text. CAPTCHA Producer: This module create the image of the text generated.
- (ii) Gimpy engine: This subsystem is responsible for modify text from the Text Generator and change the curvature of the text.
- (iii) Word rendering: This subsystem will deliver or place the ultimate image CAPTCHA into the web form.

D. Query Refinement Module:

Query refinement is a common practical technique used by most information retrieval systems. It recommends new terms related to the query or modifies the vocabulary according to the steering path of the user in the search engine. But for the database query form, a data base query is a structured relational query, not just a set of terms. In RQF the input for refinement is taken by using multiple ways like entering text in a textbox and by clicking a check box or radio button.

E. Query Form Ranking Module:

The goodness of a query form is determined by the query results generated from the query form. Based on this, I rank and recommend the potential query form components so that users can refine the query form easily.

F. Semantic Algorithm:

In RQF a new algorithm is used for getting data from unstructured text documents. The main purpose of the algorithm is to get the relevant information for a query from unstructured text documents. The algorithm involves the following steps.

Input: List of documents $D[] = d_1, d_2, \dots, d_n$ **Output:** Query result documents $q[] = \{d_i, \dots\}$

- 1. For i=1 to n do
- 2. From the given list of documents, a document is selected, say di.
- 3. The selected document contents are splitted into tokens(word).
- 4. The stop words are removed from these tokens and token list is generated as shown below $T_i[] = \{t_i, t_2, \dots\}$
- 5. The search keyword if compared with the list of tokens.
- 6. The matching count namely match count is calculated. Match count is the no of matching tokens with the keyword.
- 7. The match count for each document is stored in a separate array say match[]. A document match count is zero when no single token in the document matches with the keyword.
- 8. End for
- 9. From the match[] array remove the documents corresponding to a match count of zero.
- 10. Based a threshold, the documents are taken from the match count and placed in the Query result array q[]. The threshold can be any integer other than zero. For example if the threshold is 2, all documents in which two tokens are matching will be placed in the resultant query.
- 11. Finally the document list in the query list is displayed.

G. Example:

For example consider 6 documents and let the search keyword be="java". The algorithm works as follows

Step1: Doc[]={ doc1,doc2,doc3,doc4,doc5,doc,6}

 $Suppose\ doc1,\ doc2\ are\ related\ to\ C-tutorial,\ doc\ 4,\ doc\ 6\ are\ related\ with\ java\ tutorial\ and\ doc3,\ doc5\ are\ related\ with\ C++\ tutorial.$

Step 2: Each documents is split into a list of tokens after removing stop words

The token list is generated for each document

Step 3: The search keyword java is matched with the list of tokens.

Step 4: The match count for each document is generated.

Obviously doc 4, doc 6 have greater match count than rest of the documents,

For example take the following match counts

Match count for doc1

Document	Match cour	nt)
Doc1	0	
Doc2	10	
Doc3	0	
Doc4	58	
Doc 5	0	
Doc6	72	
Fig: 2. Q	Query Results	

Step 5: The documents with zero match count are discarded. So from the above example doc1, doc3 and doc5 are discarded. **Step 6:** Resed on threshold value the query result documents are selected.

Step 6: Based on threshold value the query result documents are selected.

For example if the threshold=30, then documents with match count>=30 are selected and displayed as a query result. So for the above example doc4 and doc6 are displayed as query results.

H. Screen shot of Secure Random Query form



IV. CONCLUSION

In this paper, we implemented a random query generation technique. The query forms facilitate the user to send a query to the database without knowing the total structure of the database. In this project, I secured the query form by using security question and CAPTCHA code. The security question is not an ordinary question but it is a time locked puzzle. The user needs to answer this puzzle within a stipulated time otherwise the users query form will not be submitted. The implemented technique also deals with non-relational data or unstructured data. It gets the results from unstructured data that is in the form text documents and presents the results to the user, when user submits a query by giving a keyword. The main advantage of the implemented techniques is it more secure and dynamic.

Scope of future enhancement:

The implemented RQF system deals with unstructured or non-relational data which is stored in only text files, this can be further extended to get query results from other sources of unstructured data such as graphs etc.,

REFERENCES

- [1] "Dynamic Query Forms for Database Queries",Liang Tang, Tao Li, Yexi Jiang, and Zhiyuan Chen, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING,2013.
- [2] "Automating the Design and Construction of Query Forms", Magesh Jayapandian and H.V. Jagadish,IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 21, NO. 10, OCTOBER 2009,. [3].Text-based CAPTCHA Strengths and Weaknesses, Elie Bursztein, Matthieu Martin, and John C. Mitchell Stanford University.
- [4] "Automated creation of a forms-based database query interface", M. Jayapandian and H. V. Jagadish. In Proceedings of the VLDB Endowment, pages 695–709, August 2008.
- [5] "Expressive query specification through form customization", M. Jayapandian and H. V. Jagadish. In Proceedings of International Conference on Extending Database Technology (EDBT), pages 416–427, Nantes, France, March 2008.
- [6] "Automating the design and construction of query forms", M. Jayapandian and H. V. Jagadish. IEEE TKDE, 21(10):1389– 1402, 2009.