Relational Keyword Search System Using Fuzzy Type-Ahead Search

Miss Kate Surekha B.
Department of computer engineering of JSCOE
Handewadi Road, Hadapsar,
Pune-411028, India
Surekha.kate@gmail.com

Prof.Ingle Madhav.D.
Department of computer engineering of JSCOE
Handewadi Road, Hadapsar,
Pune-411028, India
ingle.madhav@gmail.com

Abstract— The search performance is one of major concern in any of search query method presented by different researchers. There are many methods already presented and for improving the search results performances still in this area continue working is going. One of the most commonly used method is *autocomplete*, which predicts a word or phrase that the user may type in based on the partial string the user has typed in. There is one limitation of traditional autocomplete method is that the system treats a query with multiple keywords as a *single string*, thus it does not do a full-text search on the data. In this paper we are using the extension of fuzzy type-ahead search in XML method which overcome the limitations of previous methods. We have identified following features of this method: keywords. It allows users to explore data as they type, even in the presence of minor errors of their keywords. We offer effective index structures and to achieve a highly interactive speed top-k algorithms. We will check ranking functions and early termination techniques to identify relevant top-k answers progressively. We've implemented our actual data set method, and experimental results show that our method achieves efficiency and high search result quality.

Keywords— keyword search; fuzzy type-ahead search; optimization.

I. INTRODUCTION

With the growth of the Web, there has been a rapid increase in the number of users who need to access information without having a detailed knowledge of the query languages; even relatively simple query languages are too complicated for non-experts that are designed for them. In this paper, for overcome the limitations of previous methods we are presenting the extension of fuzzy type-ahead search in XML method which is recently presented. This method can find high-quality answers that have keywords matching query keywords approximately. This method also access information in XML data in the query keywords searches as user system. It those types as data users, even the presence of minor errors of your keywords to find out. Fuzzy search further improves user search experiences by finding relevant answers with keywords similar to query keywords.

Our proposed method has the following features:
1) Search as user type: It extends Auto complete method by supporting queries with multiple keywords in XML data.
2) Fuzzy: Fuzzy method can find high-quality answers that have keywords matching query keywords approximately.
3) Fuzzy method is efficient in terms of search time. However, there are chances to further improve this search results by using the existing forward-index structure method with aim of improving the search efficiency and result quality.

II. LITERATURE SURVEY

A. Keyword Searching and Browsing in Databases using BANKS

Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe, Soumen Chakrabarti, S. Sudarshan describe BANKS system which enables keyword-based search on relational databases, with data and schema browsing. BANKS enables users to extract information in a simple manner without any knowledge of the language or any need for writing complex queries.[2] Simple query languages designed for non-experts are even too complicated for such users, who don't have knowledge of query language. Query languages for semi-structured/XML data are more complex, increasing the impedance mismatch further.

B. Keyword Querying and Ranking in Databases

There are two types of challenges, ranking Challenges and Query Processing Challenges described in this paper in databases leverage information retrieval, traditional relational query processing, as well as more recent innovations in database algorithms[6].

C. An Empirical Performance Evaluation of Relational Keyword Search Systems

Today all Internet users use a search engine daily, performing number of searches for accessing information. The success of keyword search stems from what it does not require namely, a specialized query language or knowledge of the underlying data structure. Internet users increasingly demand keyword search interfaces for accessing information, so that it is natural to extend this paradigm to relational data[1].

Overall, existing relational keyword search systems performance is somewhat disappointing, particularly with regard to the number of queries completed successfully in our query workload. In this paper the number of timeout and memory exceptions are witnessed. Because our larger execution times might only reflect our choice to use larger datasets, we focus on two concerns that we have related to memory utilization.

D. Keyword Search on Structured and Semi-Structured Data

In this paper, Yi Chen, Wei Wang, Ziyang Liu, Xuemin Lin given an overview of the state-of-the-art techniques for supporting keyword search on structured and semi-structured data, including query result denition, rank- ing functions, result generation and top-k query processing, snippet generation, result clustering, query cleaning, perfor- mance optimization, and search quality evaluation [7]. There is majority of data on the Web which is still unstructured.

E. Keyword Proximity Search in Complex Data Graphs

In keyword search over data graphs, an answer is a nonredundant subtree in which includes the given keywords. Algorithm for enumerating answers used within an architecture that has two main components: an engine that generates a set of candidate answers and a ranker that evaluates their score. To be effective, the engine must have three fundamental properties. It should not miss relevant answers, that has to be efficient and must generate the answers in an order that is highly correlated with the desired ranking. But efficiency of keyword search on graphs is very costly to process [3].

III. RELATIONAL KEYWORD SEARCH SYSTEM

Keyword search (KWS) over relational databases has recently received significant attention. Many solutions and many prototypes have been developed in this area for improving the search results performances. The search textbox has transformed the way people interact with information. Despite the wide-ranging success of Internet search engines in making information accessible. We search query in dataset and show the result. In our project calculated execution time for particular Search. Applied existing techniques that given in paper[1]. We calculate Rank score for particular search and will Store result in Database. In proposed system we are using the extension of fuzzy type-ahead search in XML method which giving high ranking score with easily retrieve of information. By using fuzzy type-ahead search technique we calculate rank score for proposed system. Then will compare existing and proposed system and Show result in graph.

In existing system dblp dataset is used in XML format. This dataset is an input of system. The work of existing system is divided into three parts.

A. Keyword Search

keyword search module search a keyword in dblp.xml file . User can enter a keyword through keyword search method which he wants to search. If keyword is already exist in system then it sent to keyword matching module and then return the similar documents that matched with keyword to the user. If keyword is not existed in system then keyword is extracted from database and compared this with subset of relevant keywords in keyword matching component. Finally, similar documents returned to the user.

B. Approximation Score Calculate

Aim of this module is to calculate approximate score. Query processing work with top-k scoring function. DISCOVER system calculate the score of rank results.

C. Top-k based Return

Aim of *Top-k based Return* module is to return top-k based results which are in specific documents. DISOVER system using scoring function for the return of approximate top-k based results.

IV. PROPOSED SYSTEM

A. Problem Definition

Traditional autocomplete method that is used in existing system has one limitation that treats a query with multiple keywords as a single string, so that it does not do a full-text search on the database. To address this problem, we are using the extension of fuzzy type-ahead search in XML method. This method treats the query as a set of keywords, and search answers by matching keywords in documents with these keywords.

B. System Architecture

There is an limitation of traditional autocomplete method that is used in system treats a query with multiple keywords as a single string, so that it does not do a full-text search on the database. To address this problem, we are using the extension of fuzzy type-ahead search in XML method. This fuzzy type-ahead search method treats the query as a set of keywords, and search answers by matching keywords in documents with these keywords. It does a full-text search on the underlying data as the user types in query keywords letter by letter. In this way, this method will provide better optimized results, also it will help to the user, in getting instant feedback even typing a partial query, thus obtain more knowledge about the underlying data. The work of this system is divided into following four parts.

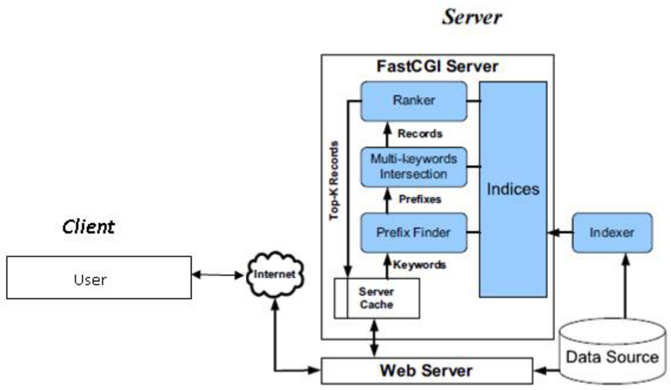

Fig. 1. Fuzzy type search architecture

A. Indexer

It is a process that reads data from specified sources, tokenizes the data, and creates the following structures:

1. create a radix trie structure.

2. Create a forward index that stores the sorted list of keyword IDs for each record.

3. Then creates a data itself.

For improving performance of system, we can also maintain a forward index, which keeps the sorted keyword IDs for each record.

B. Incremental Fuzzy Prefix Finder

For each query keyword, the Incremental Fuzzy Prefix Finder incrementally computes its *similar keywords* in the dataset and retrieve their corresponding complete keywords as the *similar keywords*. We are using prefix filtering idea in our method. We use this prefix filtering property to incrementally compute the similar prefixes of a new query. For a new query, the fuzzy prefix finder find out first similar prefixes of previous queries from the server cache, then computes similar prefixes for the current query, and for future computation stores the results in the server cache.

C. Multi-keyword Intersection

Fuzzy prefix finder produce the sets of similar keywords as an input (for multiple keywords) to the Multi-keyword Intersection module and computes the *relevant answers*, which contain a matched similar keyword from each set. First construct the union list for every keyword for to identify the relevant answers and then compute the union lists intersection. For improving the performance of computing the intersection we can use forward lists. For checking whether each candidate record on the shortest union list contains similar keywords of every other query keyword we use the forward index. For checking the keyword range of each similar keyword for other keywords, for example, [*s*, _], we check whether the candidate record on the shortest union list contains keywords in the range. For finding the keyword ID on the corresponding forward list, first we use a binary-search method. Then will get the smallest keyword ID on the list that is larger than or equal to *s*. After we check whether the keyword ID is smaller than _. If so, this candidate record contains a keyword in the range.

D. *Ranker*

The Ranker module is used to ranks the answers and to identify the top-*k* best answers for a constant *k*. For to quantify the similarity between two words *wi* and *wj* , that is denoted by ed(*wi*, *wj*) we using edit distance function. If the edit distance between an input keyword and its similar prefixes in documents dominates the other parameters, first we want to compute the answer with the smallest edit distance.. If there are not enough top answers with edit distance τ , then we will compute the answers with an edit distance $\tau + 1$, and so on.

V. ANALYSIS AND RESULT

In this paper comparison between existing system and proposed system is done. In existing system, evaluation of different systems is computed. Number of experiments are performed on different systems and datasets. Number of search query terms and collection frequency of terms are also calculated. In keyword search system, we performed number of searches (see in table I) . We calculated execution time and ranking score for particular keyword as shown in table I.

Keyword	Execution_Time	Rank
and	0.054	76.262
Bibliography	0.101	528.173
Capabilities	0.089	493.697
DB&LP	0.136	159.8
Derivability, Redundancy and Consistency	0.115	4000.355
Interactive	0.177	568.089
Large	0.056	153.272
Logic	0.074	164.288
Redundancy	0.08	323
Redundancy and Consistency	0.055	1157.615
relational	0.128	404.6
Stored in Large Data Banks	0.114	1769.088
the	0.061	77.33299
Theoretical Background	0.084	1163.412

Fig. 2. Search keyword details

In our proposed system we calculating optimization on evaluation of system performances, so it give better results than existing system. We calculated execution time and ranking score for particular keyword as shown in table II. It reduced execution time at better level than existing system. It also reduce ranking score than keyword search system.

Keyword	Execution_Time	Rank
Large	0.027	135.524
Logic	0.04	143.48
Redundancy and Consistency	0.01	599.93
Relational	0.006	197.2
Systems	0.006	140.998

Fig. 3. Fuzzy search keyword details

We have done comparison between existing system and proposed system. Following graph shows comparison between both systems. For 'Large' keyword in keyword search execution time- 0.056 and rank score- 153.272 is required. But in Fuzzy search, execution time- 0.027 and rank score- 135.524 is required.

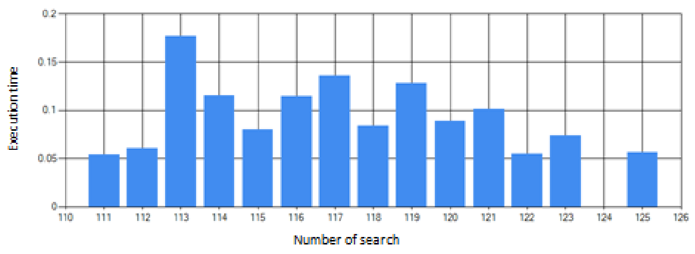

Fig. 4. Keyword search system

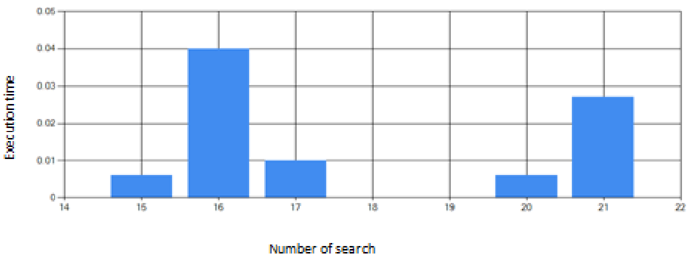

Fig. 5. Fuzzy search system

In keyword search system, 125th search is done for 'Large' keyword. In Fuzzy search system 21th search is done for same keyword 'Large'. Here for keywords in proposed system requires less execution time and rank score. Hence fuzzy search method provide better optimized results. In this way, our proposed system easily retrieve data with high ranking score by using fuzzy technique.

VI .CONCLUSION

It will conclude from the comparison between existing methods and our new fuzzy type-ahead search method that fuzzy method provides better optimized results than existing system. In fuzzy search system less execution time and ranking score is required for specific search than keyword search system. So that fuzzy search method give high ranking score with easily retrieve data. Fuzzy search technique do full-text search on the underlying data as the user types in query keywords letter by letter. Hence, this system helps to user for getting instant feedback after typing a partial query, thus he can obtain more knowledge about the underlying data, which helps the user prepare complete queries.

ACKNOWLWDGEMENT

I would like thanks to Prof. M. D. Ingle professor of Computer Engineering at J.S.P.M .Hadapsar ,who guided through this paper.

REFERENCES

1. Joel Coffman, Alfred C. Weaver (2014). An Empirical Performance Evaluation of Relational Keyword Search Systems. Technical Report, University of Virginia Charlottesville, VA, USA.

2. G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword Searching and Browsing in Databases using BANKS," in Proceedings of the 18th International

Conference on Data Engineering, ser. ICDE ’02, February 2002, pp. 431–440.

3. K. Golenberg, B. Kimelfeld, and Y. Sagiv, “Keyword Proximity Search in Complex Data Graphs,” in Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD ’08, June 2008, pp. 927–940.

4. Guoliang Li, Shengyue Ji, Chen Li, Jiannan Wang, Jianhua FengEfficient, “Fuzzy Type-Ahead Search in TASTIER” *Tsinghua University, Beijing 100084, China. UC Irvine, CA 92697-3435, USA.*

5. H. Bast and I. Weber, “Type less, find more: fast autocompletion search with a succinct index,” in *SIGIR*, 2006, pp. 364–371.

6. S. Chaudhuri and G. Das, “Keyword Querying and Ranking in Databases,” Proceedings of the VLDB Endowment, vol. 2, pp. 1658–1659, August 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1687553. 1687622

7. Y. Chen, W. Wang, Z. Liu, and X. Lin, “Keyword Search on Structured and Semi-Structured Data,” in Proceedings of the 35th SIGMOD International Conference on Management of Data, ser. SIGMOD ’09, June 2009, pp. 1005–1010.

8. J. Coffman and A. C. Weaver, “A Framework for Evaluating Database Keyword Search Strategies,” in Proceedings of the 19th ACM International Conference on Information and Knowledge Management, ser. CIKM ’10, October 2010, pp. 729–738.

9. G. Li, S. Ji, C. Li, and J. Feng, “Efficient type-ahead search on relational data: a tastier approach,” in *SIGMOD*, 2009, pp. 695–706.

10. S. Ji, G. Li, C. Li, and J. Feng, “Interative fuzzy keyword search,” in *WWW 2009*, 2009, pp. 371–380.

11. V. Hristidis, L. Gravano, and Y. Papakonstantinou, “Efficient IR-style Keyword Search over Relational Databases,” in Proceedings of the 29th International Conference on Very Large Data Bases, ser. VLDB ’03, September 2003, pp. 850–861.