Type-Ahead Search in XML data based on Improved Forward Index Structure: ATASK

Ms. Sonali S. Kadam
Computer dept.
JSPM'S BSIOTR (W)
Pune, India
sonali.kadam4@gmail.com

Prof. Sanchika A. Bhajpai
Computer dept.
JSPM'S BSIOTR (W)
Pune, India
sanchi.scriet@gmail.com

***Abstract*—** Recently, keyword based search system is most widely used in many real time applications for getting the required information from a large amount of dataset in less time. There are many keyword search based systems and a method presented by various authors already, these methods becomes inefficient in different ways like time to retrieve data, fetch that data. The all previous methods did not work for search XML data in a type-ahead search, and hence it is not trivial to extend existing techniques to support fuzzy type-ahead search in XML data. Previous methods are not purely based on XML data and as XML data is consisting of parent and child nodes, it is complex to understand such format to read for existing methods. Existing methods directly works on single document. Thus to overcome the limitations of existing methods, we need to have efficient XML based type-ahead search using forward indexing method. Recently, we have studied one method, which is called as TASX (pronounced "task") which is fuzzy type-ahead search method in XML data. This method searches the XML data during the typing of keyword from user end and it searches XML data even if it's misspelled. Experimentally this method showing efficient performance as compared to existing methods, but there are still suggestions over this method for improvement. Here, we are presenting extended approach for XML based type-ahead search method ATASX (pronounced "a task"). In this method we are proposing to use improved forward-index structure method with aim of improving the search efficiency it reduces searching time and provides result quality.

Keywords- *Keyword Search System, Query, XML, TASX, Fuzzy, Forward index Structure.*

*****

I. INTRODUCTION

Keyword search methods are recently have a great attention in data mining and knowledge discovery. It is become apparent most effective paradigm for discovering information on web. The advantage of keyword search is its simplicity-that user do not have to learn about complex query language and can issue query without having any knowledge about structure of xml document. Ranking the results of query is the most important requirement for the keyword search so that the most relevant results will be appeared. Keyword search provides simple and user friendly query interface to access xml data in web. Keyword search over xml is not always the entire document but deeply nested xml[1]. Xml was designed to transport and store data. It does not do anything, it is created to structure, store, and transport information.xml document contains text with some tags which is organized in hierarchy with open and close tag.xml model addresses the limitation of html search engine i.e. Google which returns full text document but the xml captures additional semantics such as in a full text titles, references and subsections are explicitly captured using xml tags. For querying xml data keyword search is proposed as an alternative method. In traditional methods like xpath and xquery, to query over xml data it requires query languages which are very hard to comprehend for non database users. It can only understand by professionals.

In order to reduce user's typing effort ATASX (pronounced "a task"), a fuzzy type-ahead search method in XML data used. ATASX searches the XML data, as user's type in query keywords; even it has minor errors in their keywords. ATASX provides a user friendly interface for users to explore XML data, and can save users typing effort. In this work we have studied the research challenges that came naturally in this computing paradigm. The main challenge is search efficiency. Each query with multiple keywords needs to be answered efficiently[4]. To make search really interactive, for each keystroke on the client browser, from the time the user presses the key to the time the results computed from the server are displayed on the browser, the delay should be as small as possible. Interactive speed requires this delay should be within milliseconds so that user get the answer immediately. This time includes the network transfer time, execution time, and the time for the browser to execute its Java-Script. Low Running time is challenging task especially when the backend repository has a large amount of data. To achieve the high efficiency, we propose effective index structures and algorithms to answer keyword queries in XML data. We examine effective ranking functions and early termination techniques to progressively identify top-k answers[14]. To the best of our knowledge, this is the first paper to study fuzzy type-ahead search in XML data based on improved forward index structuring method.

II. MINIMAL-COST TREE

In this section, new framework to find relevant answers to a keyword query over an XML document. In the framework, each node on the XML tree is potentially relevant to the query with different scores. For each node, we define its corresponding answer to the query as its subtree with paths to nodes that include the query keywords. This subtree is called the "minimal-cost tree" for this node. Different nodes correspond to different answers to the query, and we will study how to quantify the relevance of each answer to the query for ranking.

III. FUZZY TYPE AHEAD SEARCH

Existing methods cannot search XML data in a type-ahead search manner, and it is not trivial to extend existing techniques to support fuzzy type-ahead search in XML data. This is because XML contains parent-child relationships, and we need to identify relevant XML subtrees that capture such structural relationships from XML data to answer keyword queries, instead of single documents. In this, we propose ATASX (pronounced "task"), a fuzzy type-ahead search method in XML data. ATASX searches the XML data on the fly as user's type in query keywords, even in the presence of minor errors of their keywords. ATASX provides a friendly interface for users to explore XML data, and can significantly save users typing effort. In this, we study research challenges that arise naturally in this computing paradigm. The main challenge is search efficiency. Each query with multiple keywords needs to be answered efficiently. To make search really interactive, for each keystroke on the client browser, from the time the user presses the key to the time the results computed from the server are displayed on the browser, the delay should be as small as possible. An interactive speed requires this delay should be within milliseconds. Notice that this time includes the network transfer delay, execution time on the server, and the time for the browser to execute its Java-Script. This low-running-time requirement is especially challenging when the backend repository has a large amount of data. To achieve our goal, we propose effective index structures and algorithms to answer keyword queries in XML data. We examine effective ranking functions and early termination techniques to progressively identify top-k answers. To the best of our knowledge, this is the first paper to study fuzzy type-ahead search in XML data. To summarize, we make the following contributions:. We formalize the problem of fuzzy type-ahead search in XML data. . We propose effective index structures and efficient algorithms to achieve a high interactive speed for fuzzy type-ahead search in XML data. We develop ranking functions and early termination techniques to progressively and efficiently identify the top-k relevant answers. We have conducted an extensive experimental study. The results show that our method achieves high search efficiency and result quality.

Recently fuzzy type ahead search is studied which allows minor mistakes in query. Type ahead search is a user interface interaction method to progressively search for filter through text. As the user types text, one or possible matches for text are found and immediately present to user. The fuzzy type ahead search in xml data returns the approximate results. The best similar prefixes are matched and returned. For this edit distance is used. Edit distance is defined as number of operations (delete, insert, substitute) required to make the two words equal. For example user typed the query "mices" but the mices is not in the xml document it contains miches ed(mices, miches) is 1 so therefore the best similar prefix is miches it is displayed.

Fuzzy Type ahead Search using improved Forward index structure Algorithm Steps –

1. Compute Ranking of Sub Tree

There are mainly two ranking function to compute rank or score between node *n* and keyword *ki.*

- The case I shows that *n* contains *ki.*
- The case II shows that *n* does not contain *ki* but has a descendant containing *ki.*

Case I: *n* contains keyword *ki* the relevance or score of node *n* and keyword *ki* is calculated by:

$$\mathrm{SCORE}_1(n, k_i) = \frac{ln(1 + tf(k_i, n)) * ln(idf(k_i))}{(1 - s) + s * ntl(n)}.$$

Where,

tf (ki, n) – number of occurrences of *ki* in sub tree rooted n

idf (ki) - ratio of number of nodes in xml to number of nodes that contain keyword

ki ntl (n) - length of *n /nmax* length, *nmax* = node with max terms

s - Constant set to 0.2 (Assumption)

Assume user composed a query containing keyword "db"

SCORE (13, db) = ln (1+1)*ln (27/2)

(1- 0.2)+ (0.2*1)

= 1.52

Case II: node *n* does not contain keyword *ki* whereas its descendant has *ki* Second ranking function to calculate the score between *n* and *ki* is given by:

$$\mathrm{SCORE}_2(n, k_j) = \sum_{p \in P} \alpha^{\delta(n,p)} * \mathrm{SCORE}_1(p, k_j),$$

Where,

P - Set of pivotal nodes

α - constant set to 0.8(Assumption)

$\delta(n,p)$ - Distance between *n* and *p*

Assume the user entered query "db"

SCORE2 (12, db) = (0.8)*score1 (13, db)

= 0.8 *1.52

=1.21

 ISSN: 2321-8169
3406 – 3410

2. Ranking for Fuzzy Search Algorithm

Given a user keyword query *Q= {k1, k2 ...kl}* in terms of fuzzy search, a MCT may not contain the exact input keywords, but contain predicted words for each keyword. Let predicted words be *{w1, w2... wl}* the best similar prefix of *wi* could be considered to be most similar to *ki*. The function to quantify the similarity between *ki* and *wi* is given by

$$sim(k_i, w_i) = \gamma * \frac{1}{1 + \mathrm{ed}(k_i, a_i)^2} + (1 - \gamma) * \frac{|a_i|}{|w_i|},$$

Where,
ed – edit distance
ai – prefix,
wi – predicted word
γ – constant

3. Improvement Using Forward Index Structure Approach

Improved forward index is used to improve search performance. We can utilize "random access" based on the forward index to do an early termination in the algorithms. That is, given an XML element and an input keyword, we can get the corresponding score of the keyword and the element using the forward index, without accessing inverted lists. Fagin et al. have proved that the threshold-based algorithm using random access is optimal over all algorithms that correctly find the top k answers [12]. Thus, in this we have improved forward index to implement random access.

Procedure

- Construct a trie structure to maintain the keyword contained in the element.
- Each leaf node in the forward index keeps score of element e to the corresponding keyword of the leaf node.
- Given partial keyword we can efficiently check element e contains a word having similar prefixes.

The time complexity of sorted access *O (1)* and for random access is *O(ed * AN)*, where ed is edit distance threshold and AN is active number of nodes [27]. Suppose *ed* A > I*, we will not maintain forward index, where I is average inverted list length. The main advantage of forward index avoids unnecessary element access compare to extended trie structure.

IV. EXPERIMENTAL SETUP

The Type-Ahead Search in XML Data Based on Forward Index Structure for XML data is implemented in Java, JSP and downloadable package available on Internet is used. For tomcat server the apache tomcat software package available on Internet is used. It is written in Java
and runs on almost any platform. The model can be applied directly to a dataset.

1. RESULTS AND DISCUSSION

In this section the performance of Type-Ahead Search In XML Data Based On Forward Index Structure XML data with other methods discussed. The results provided are based on comparisons between proposed model and other models. Table III shows elapsed time in milliseconds to execute query.

Table 1: Comparison Results between Type-Ahead Search in XML Data Based on Forward Index Structure and Other methods.

Query(Input)	Method	Time-in ms
Keyword	Exact Search	280
Keyword	Fuzzy Search	210
Keyword	Effective Top-K Search	172
Keyword	Effective MCT with Improved Forward index Structure	17

2. RESULT SNAPSHOTS

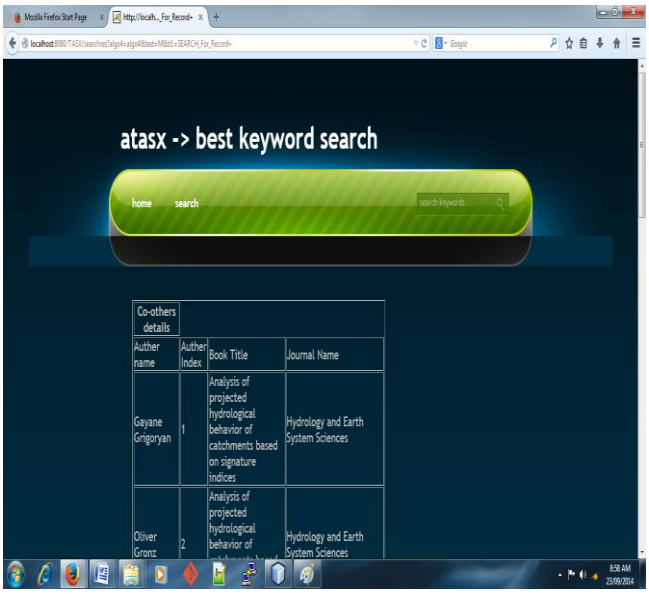

Fig.1. Effective MCT with improved forward index structure Input shown

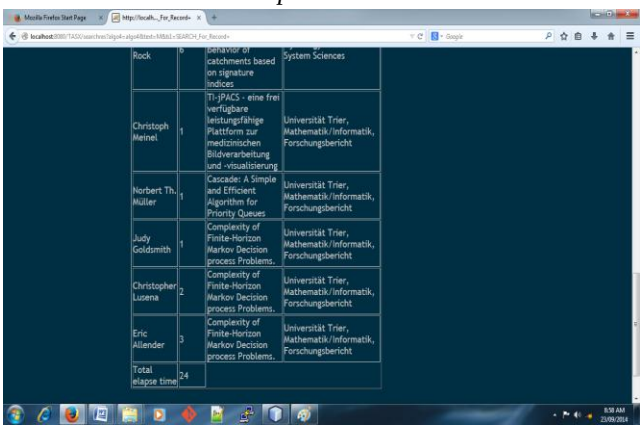

Fig.2. Effective MCT with improved forward index structure Elapsed time

 ISSN: 2321-8169
3406 – 3410

Here, Fig.2 we can take the input keyword for search and total elapsed time required to search that keyword is calculated. Elapsed time is time required between start time and end time. From this we analyzed our system performance.

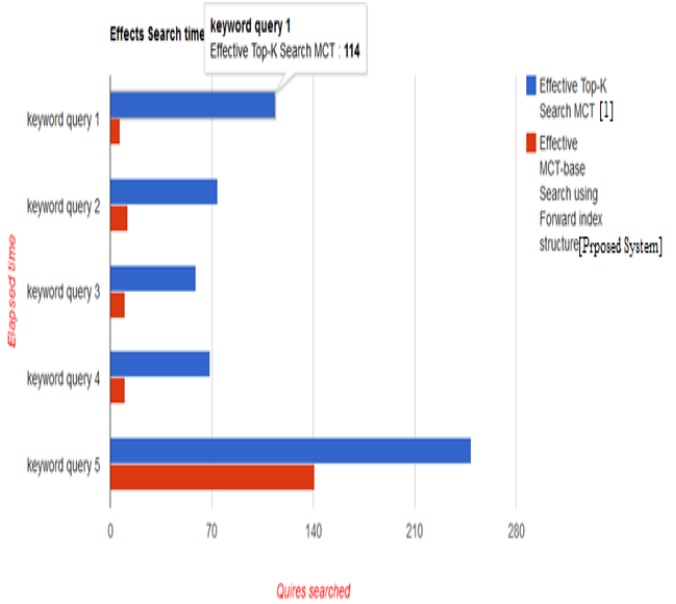

Fig.3. Comparative Graph of Effective Top-K search MCT and Improved forward index structure

When user select result then it shows comparative analysis between different systems, Comparative graph shows how our system gives better performance than existing system. As mentioned in figure 2 effective top-K MCT [1] is existing system and Effective MCT with improved forward index structure method is the proposed system.

Table 2: Difference between Mct and Mct with improved forwared index

id	mct	mct with forward index
1	114	7
2	74	12
3	59	10
4	69	10
5	249	141
(NULL)	*(NULL)*	*(NULL)*

Table 8.2 shows comparative analysis between MCT and MCT with improved forward index structure.

Elapsed time required for existing system and proposed system is stored into table 2. Time is calculated in milliseconds. Here, we can see that time required for existing is more than proposed one.

V. CONCLUSION

We studied and implemented proposed method "Type-Ahead Search in XML Data based on improved forward index structure".

We have identified following features of this method:

1. Keywords.-It allows users to explore data as they type, even in the presence of minor errors of their keywords.
2. Fuzzy type-ahead search method in xml data using improved forward index structure, this method searches the xml data during the typing of keyword from user end and it searches xml data even if it's misspelled.
3. Improving the search efficiency it reduces searching time and provides result quality

As discussed in table 1.Experimentally this method showing efficient performance as compared to existing methods in order to elapsed time that is time required to execute that string/keyword. We are presenting extended approach for xml based type-ahead search method. In this method we are proposing to use improved forward-index structure method with aim of improving the search efficiency it reduces searching time and provides result quality. our proposed method has the following features: it extends auto complete by supporting queries with multiple keywords in xml data, fuzzy: it can find high-quality answers that have keywords matching query keywords approximately ,this 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. In this project we are adding the method improved forward index structure.

ACKNOWLEDGMENT

Authors thanks BSIOTR(W), Pune for every support to write this paper. Authors also thanks Prof. G. M. Bhandari, Head, Department of Computer Engineering, Jspm's Bhivarabai Sawant Institute of Technology & Research (W), Pune who guided & encouraged me in completing the this work.

REFERENCES

[1] Jianhua Feng, Senior Member, IEEE, and Guoliang Li, Member, IEEE "Efficient Fuzzy Type-Ahead Search in XML Data", ieee transactions on knowledge and data engineering, vol. 24, no. 5, may 2012.

[2] S. Agrawal, S. Chaudhuri, and G. Das, "Dbxplorer: A System for Keyword-Based Search over Relational Databases," Proc. Int'l Conf. Data Eng. (ICDE), pp. 5-16, 2002.

[3] S. Amer-Yahia, D. Hiemstra, T. Roelleke, D. Srivastava, and G.Weikum, "Db&ir Integration: Report on the Dagstuhl Seminar 'Ranked Xml Querying'," SIGMOD Record, vol. 37, no. 3, pp. 46-49, 2008.

[4] A. Balmin, V. Hristidis, and Y. Papakonstantinou, "Objectrank: Authority-Based Keyword Search in Databases," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 564-575, 2004.

[5] Z. Bao, T.W. Ling, B. Chen, and J. Lu, "Effective XML Keyword Search with Relevance Oriented Ranking," Proc. Int'l Conf. Data Eng. (ICDE), 2009.

[6] H. Bast and I. Weber, "Type Less, Find More: Fast Autocompletion Search with a Succinct Index," Proc. Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR), pp. 364-371, 2006.

[7] H. Bast and I. Weber, "The Completesearch Engine: Interactive, Efficient, and towards Ir&db Integration," Proc. Biennial Conf. Innovative Data Systems Research (CIDR), pp. 88-95, 2007.

[8] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword Searching and Browsing in Databases Using Banks," Proc. Int'l Conf. Data Eng. (ICDE), pp. 431-440, 2002.

[9] Y. Chen, W. Wang, Z. Liu, and X. Lin,"Keyword Search on Structured and Semi-Structured Data," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 1005-1010, 2009.

[10] E. Chu, A. Baid, X. Chai, A. Doan, and J.F. Naughton, "Combining Keyword Search and Forms for Ad Hoc Querying of Databases," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 349-360,2009.

[11] S. Cohen, Y. Kanza, B. Kimelfeld, and Y. Sagiv, "Interconnection Semantics for Keyword Search in Xml," Proc. Int'l Conf. Information and Knowledge Management (CIKM), pp. 389-396, 2005.

[12] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv, "Xsearch: A Semantic Search Engine for Xml," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 45-56, 2003.

[13] B.B. Dalvi, M. Kshirsagar, and S. Sudarshan, "Keyword Search on External Memory Data Graphs," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 1189-1204, 2008.

[14] B. Ding, J.X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin, "Finding Top-k Min-Cost Connected Trees in Databases," Proc. Int'l Conf. Data Eng. (ICDE), pp. 836-845, 2007.

[15] R. Fagin, A. Lotem, and M. Naor, "Optimal Aggregation Algorithms for Middleware," Proc. ACM SIGMOD-SIGACTSIGART Symp. Principles of Database Systems (PODS), 2001.

[16] I.D. Felipe, V. Hristidis, and N. Rishe, "Keyword Search on Spatial Databases," Proc. Int'l Conf. Data Eng. (ICDE), pp. 656-665, 2008.

[17] K. Golenberg, B. Kimelfeld, and Y. Sagiv, "Keyword Proximity Search in Complex Data Graphs," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 927-940, 2008.

[18] L. Guo, J. Shanmugasundaram, and G. Yona, "Topology Search over Biological Databases," Proc. Int'l Conf. Data Eng. (ICDE),pp. 556-565, 2007.

[19] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram, "Xrank: Ranked Keyword Search over Xml Documents," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 16-27, 2003.

[20] D. Harel and R.E. Tarjan, "Fast Algorithms for Finding Nearest Common Ancestors," SIAM J. Computing, vol. 13, no. 2, pp. 338-355, 1984.