ISSN: 2320-1363

Effective Fuzzy Characteristics of Forward Search in Xml Information

K.V.Ajay Kumar*1, P Ramesh Babu*2

M.Tech (CSE) Student Department of CSE, Priyadarshini Institute of Technology & Science, Chintalapudi, Guntur(Dist), Ap, India.

Associate Professor, Department of CSE in Priyadarshini Institute of Technology & Science, Chintalapudi, Guntur(Dist), Ap, India

kvajaykumar45@gmail.com, rameshbabup81@gmail.com

Abstract

Keywords are suitable for query XML streams without schema information. In current forms of keywords search on XML streams and rank functions do not always represent user's intensions. In this paper the skyline Top-K keyword queries, a novel kind of keyword queries on XML streams are presented. This paper studies the problem of XML message brokering with user subscribed profiles of keyword queries and presents a Keyword based XML Message Broker (KEMB) to address this problem. In case where the user has limited knowledge about the data often the user feels left in the dark when issuing queries, and has to use a try and see approach for finding information. In this paper we proposed fuzzy type forward search in XML data, a new information access paradigm in which the system searches XML data on the fly as the user types in query keywords. It allows users to explore data as they type even in the presence of minor errors of their keywords. Our proposed method has the following features: 1) search as you type: it extends auto complete by supporting queries with multiple keywords in XML data. 2) Fuzzy: it can find high quality answers that have keywords matching query keywords approximately. 3) Efficient: our effective index structures and searching algorithms can achieve a high interactive speed. We have implemented our methods on real data sets and the experimental results show that our method achieves high search efficiency and result quality.

Keywords: XML, Keyword Search, Characteristics of Forward Search, Fuzzy Search.

1. Introduction

We propose XIR, a novel method for processing partial match queries on heterogeneous XML documents using information retrieval (IR) techniques. A partial match query is defined as the one having the descendent-or-self axis in its path expression. In its general form, a partial match query has branch predicates forming branching paths. The objective of XIR is to efficiently support this type of queries for large-scale documents of heterogeneous schemas. XIR has its basis on the conventional schema-level methods using relational tables and significantly improves their efficiency using two techniques: an

ISSN: 2320-1363

inverted index technique and a novel prefix match join. The former indexes the labels in label paths as keywords in texts, and allows for finding the label paths matching the queries more efficiently than string match used in the conventional methods. The latter supports branching path expressions, and allows for finding the result nodes more efficiently than containment joins used in the conventional methods. We compare the efficiency of XIR with those of XRel and XParent using XML documents crawled from the Internet. The results show that XIR is more efficient than both XRel and XParent by several orders of magnitude for linear path expressions, and by several factors for branching path expressions. Current data sharing and integration among various organizations require a central and trusted authority to first collect data from all data sources and then integrate the collected data. In this paper we propose TASX a fuzzy type-forward search method in XML data. TASX searches the XML data on the fly as user's type in query keywords even in the presence of minor errors of their keywords. TASX provides a friendly interface for users to explore XML data, and can significantly save users typing effort. In this article, 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- forward search in XML data. To summarize, we make the following contributions:

- We formalize the problem of fuzzy type- forward search in XML data.
- We propose effective index structure and efficient algorithms to achieve a high interactive speed for fuzzy type-forward search in XML data.
- We developed 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 methos achieves high search efficiency and result quality.

ISSN: 2320-1363

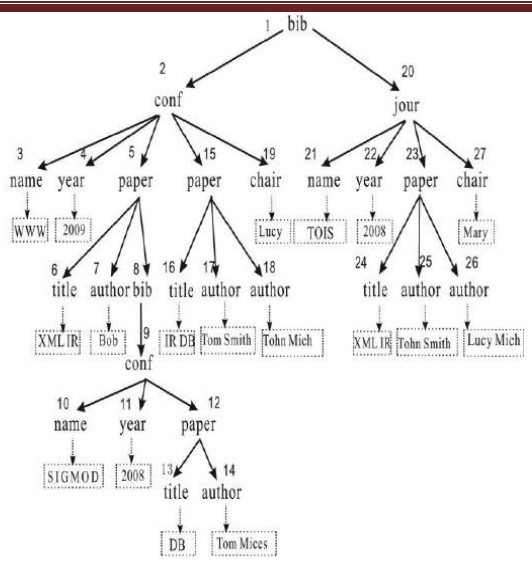

Fig1.1 An XML Document.

- In an XML tree, every two nodes are connected through their LCA.
- Not all connected trees are relevant even if the size is small.
- The focus is defining query results to prune irrelevant sub tree.

2. Related Work

Keyword search in XML data has attracted great attention recently. Xu and Papakonstantinou proposed Smallest Lowest Common Ancestor (SLCA) to improve search efficiency. Sun et al. Studied multi way SLCA-based keyword search to enhance search performance. Schema free X Query employed the idea of meaningful LCA, and proposed a stack based sort-merge algorithm by considering XML structures and incorporating a new function Mlcas into XQuery. XSEarch focuses on the semantics and the ranking of the results, and extends keyword search. It employs the semantics of meaningful relation between XML nodes to answer keyword queries, and two nodes are meaningfully related if they are in same set, which can be given by administrators or users.

3. Problem Formulation of Fuzzy Characteristics of Forward Search In XML Data

We first introduce how TASX works for queries with multiple keywords in XML data, by allowing minor errors of query keywords and inconsistencies in the data itself. Assume there us an underling XML document that resides on a server. Each keystroke that the user types invoke a query which includes the current string the user has typed. The browser sends the query to the server, which computes and returns to the user the best answer ranked by their relevancy to the query. The server first tokenizes the query string into several keywords using delimiters such as the space character. For the partial keywords we would like to know the possible words the user intends to type. However given the limited information we can only identify a set of complete words I the data set which have similar prefixes with the partial keywords. This set of complete words is called the predicted words. We use edit distance to qualify the similarity between two words. The edit distance between two words S_1 and S_2 denoted by $ed(S_1,S_2)$ is the minimum number of edit operations of single characters needed to transform the first one to the second.

4. LCA-Based Fuzzy Characteristics of Forward Search

In this section we propose an LCA-based fuzzy type- forward search method. We use

ISSN: 2320-1363

the semantics of ELCA to identify relevant answer on top of predicted words.

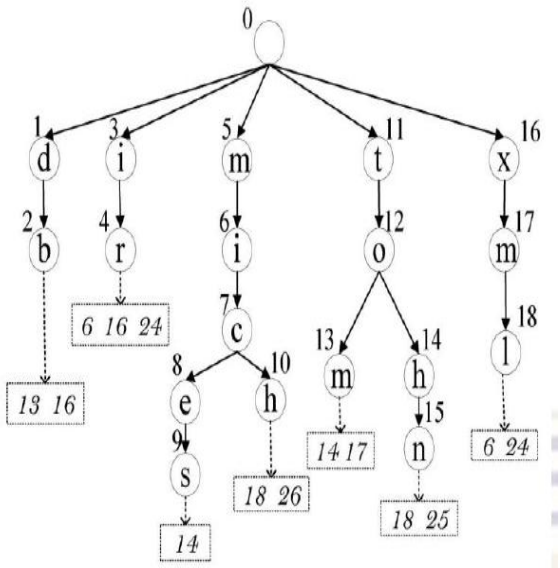

Figure 4.1 The trie on top of words in figure 1.1

4.1 Index Structure

We use a trie structure to index the words in the underlying XML data. Each word w corresponds to a unique path from the root of the trie to a leaf node. Each node on the path has a label of a character in w. for each node we store an inverted list of the leaf node. For instance consider the XML document in Fig1.1. The trie structure for the tokenized words is shown in Fig.4.1. the word "mich" has a node ID of 10. Its inverted list includes XML elements 18 and 26.

4.2 Answering Queries with a Single Keyword:

Firstly study how to answer a query with a single keyword using the trie structure. Each keystroke that a user types invokes a query of the current string and the client browser sends the query string to the server. We first consider the case of exact search. On native way to process such a query on the server is to answer the query from scratch as follows: we first find the trie node corresponding to this keyword by traversing the trie from the root. For example suppose a user types in the character "mich" letter by letter. When the user types in the character "m" te client sends the query "m" to the server. The server finds the trie node corresponding to this keyword (node 5). Then it locates the leaf descendants of node 5 and retrieves the corresponding predicted words and the predicted XML elements. When the user types in the character "I" the client sends a query string "mi" to the server. The server answers the query from scratch as follows: it first finds node 6 for this string then locates the leaf descendants of node 6. It retrieves the corresponding predicted words ("mich"). In general the user may modify the previous query string arbitrarily, or copy and paste a completely different string. In this case for the new query string among all the keywords typed by the user, we identify the cached keyword that has the longest prefix with the new query. Then we use this prefix to incrementally answer the new query, by inserting the characters after the longest prefix of the new query one by one.

4.3 Answering Queries with Multiple Keywords:

Now we consider how to do fuzzy type-forward search in the case of a query with multiple keywords. For a keystroke that invokes a query, we first tokenize the query string into keywords k_1, k_2 .... k_l . For each keyword k_i ($l \leq i \leq l$), we compute its

ISSN: 2320-1363

corresponding active nodes, and for each such active node, we retrieve its leaf descendants and corresponding inverted lists. Then, we compute union list Uk_i for every k_i. Finally, we compute the predicted answers on top of lists Uk_1, Uk_2 . . . Uk_l .We use the semantics of ELCA to compute the corresponding answers. We use the binary-search-based method to compute ELCAs.

5. Progressive And Effective Top –K Fuzzy Characteristics of Forward Search

The LCA-based Characteristics of forward search algorithm in XML data has two main limitations. First, they use the "AND" semantics between input keywords of a query, and ignore the answers that contain some of the query keywords (but not all the keywords). For example, suppose a user types in a keyword query "DB IR Tom" on the XML document in Fig. 1.1. The ELCAs to the query are nodes 15 and 5. Although node 12 does not have leaf nodes corresponding to all the three keywords, it might still be more relevant than node 5 that contains many irrelevant papers. Second, in order to compute the best results to a query, existing methods need find candidates first before ranking them, and this approach is not efficient for computing the best answers. A more efficient algorithm might be able to find the best answers without generating all candidates.

In our approach each node on the XML tree could be potentially relevant to a keyword query, and we use a ranking function to decide the best answer to the query. For instance consider the XML document in Fig1.1 for the keyword "DB", we index nodes 13, 16, 12, 15, 9, 2, 8, 1, and 5 for this keyword as shown below fig 5.1.

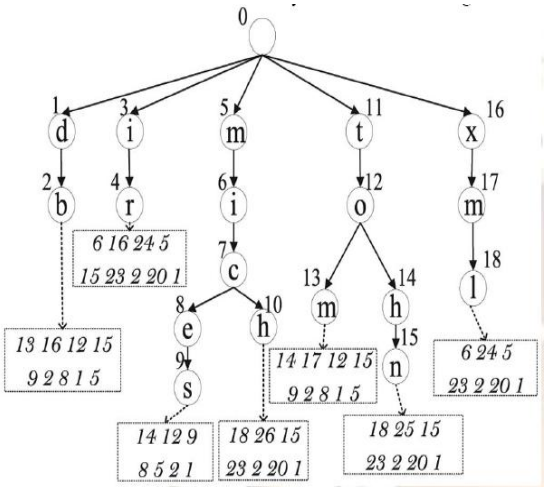

Figure 5.1. The extended trie on top of words in figure.1.1.

For the keyword "IR", we index nodes 6, 16, 24, 5, 15, 23, 2, 20, and 1. For the keyword "Tom", we index nodes 14, 17, 12, 15, 9, 2, 8, 1, and 5. The nodes are sorted by their relevance to the keyword. Fig. 5.1 gives the extended trie structure. For instance, assume a user types in a keyword query "DB IR Tom". We use the extended trie structure to find nodes 15 and 12 as the top-2 relevant nodes. We propose Minimal-Cost Trees (MCTs) to construct the answers rooted at nodes 15 and 12. We develop effective ranking techniques to rank XML elements on the inverted lists in the extended trie structure. We can employ threshold-based algorithms to progressively and efficiently identify the Top-K relevant answers. Moreover, our approach automatically supports the "OR" semantics.

5.1. Minimal Cost Tree:

ISSN: 2320-1363

This section we introduce a 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. This sub-tree 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. Given a keyword query each node n in the XML document is potentially relevant to the query. We introduce the notion of minimal cost tree rooted at node n to define the answer to the query.

5.2 **Ranking Minimal Cost Trees:**

In this section, we discuss how to rank a minimal cost tree. We first introduce a ranking function for exact search and then extend the ranking function to support fuzzy search.

5.2.1 **Ranking for Exact Search.**

To rank a Minimal-Cost Tree, we first evaluate the relevance between the root node and each input keyword, and then combine these relevance scores for every input keyword as the overall score of the Minimal-Cost Tree. We propose two ranking functions to compute the relevance score between the root note n to an input keyword k_i. The first one considers the case that n contains k_i. The second one considers the case that n does not contain ki but has a descendant containing k_i. However, if n does not contain k_i, the first ranking function cannot quantify the relevancy between node n and keyword k_i. To address this issue, we extend the first ranking function and propose the second ranking function. Given a keyword k_j, a quasi-content node n for k_j, suppose p is the pivotal node for n and k_j. The distance between n and p can indicate how relevant the node n is to keyword k_j. The smaller the distance between n and p, the larger relevancy score between n and k_j should be. Based on this observation, we proposed the second ranking function to compute the relevance between n and k_j as follows:

$SCORE_2(n,k_j)= \sum_{p\in P} \alpha\delta(n,p) * SCORE1p,kj.$

Where P is the set of pivotal nodes for n and k_j, is a damping factor between 0 and 1, and Δ(n, p) denotes the distance between node n and node p. As the distance between n and p increases, n becomes less relevant to k_j. As a trade off, our experiments suggested that a good value for is 0.8, and our method achieves the best performance at this point. This is because it will degrade the importance of ancestor nodes for a smaller α and thus may miss meaningful and relevant results; on the contrary, it will involve some duplicates and less important results for a larger .

6. Conclusion

In this article, we studied the problem of fuzzy type- forward search in XML data. We are proposed effective index structures, efficient algorithms, and novel optimization techniques to progressively and efficiently identify the Top-K answers. We examined the LCA-based method to interactively identify the predicted answers. We have

ISSN: 2320-1363

developed a Minimal-Cost-Tree-based search method to efficiently and progressively identify the most relevant answers. We proposed a heap-based method to avoid constructing union lists on the fly. We devised a forward-index structure to further improve search performance. We have implemented our method, and the experimental results show that our method achieves high search efficiency and result quality.

References

[1] 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, 2010.

[2] 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.

[3] M.D. Atkinson, J.-R. Sack, N. Santoro, and T. Strothotte, "Min-max Heaps and Generalized Priority Queues," Comm. ACM, vol. 29, no. 10, pp. 996-1000, 2011.

[4] A. Balmin, V. Hristidis, and Y. Papakonstantinou, "Object rank: Authority-Based Keyword Search in Databases," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 564-575, 2011.

[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, 2010.

[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, 2011.

[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, 2011.

[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, 2011.

[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, 2011.