International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Smart Type-Ahead Search in XML

Supriya. N. Chaudhari[1], Vaishali M. Deshmukh[2]

[1]Sant Gadage Baba University, Prof. Ram Meghe Inst. of Tech. & Res, Badnera, Amravati, Maharashtra India

[2]Sant Gadage Baba University, HOD Information Technology, Prof. Ram Meghe Inst. of Tech. & Res, Badnera, Amravati, Maharashtra India

Abstract: ***Now a day in this digital world, internet search keyword paradigm are much popularized. However the search engine that uses html based model does not capture more semantics. But the xml model captures more semantics and navigates into document and displays more relevant information. The keyword search is alternative method to search in xml data, which is user friendly, user no need to know about the knowledge of xml data. This paper focuses on the survey of techniques used to retrieve the top k results from the xml document more efficiently. In addition to this, focus is given how to improve search performance by using data view. Data view is maintained after every successful search which will increase search performance as other searches will first begin with data view. 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) Intelligent: Our effective index structures, searching algorithms and data view can achieve a very high interactive speed. Answering queries using data views has shown significant performance benefits.***

Keywords: XML, keyword search, type-ahead search, fuzzy search, data views.

1. Introduction

Now a day's an internet search engines are much popularized, where keyword search paradigm has become very crucial. However the search engine that uses html based model does not capture more semantics. But the xml model captures more semantics and navigates into document and displays more relevant information. The keyword search is alternative method to search in xml data, which is user friendly, user no need to know about the knowledge of xml data. This paper focuses on the survey of techniques used to retrieve the top k results from the xml document more efficiently and how to speed up the information retrieval process.

Over Word Wide Web millions of data is stored. From where it is very difficult to find exactly what is intended. Therefore keyword search becomes very important paradigm to solve the purpose. A keyword search looks for words anywhere in the record. It is emerged as most effective paradigm for discovering information on web. The advantage of keyword search is its simplicity-users do not have to learn complex query language and can issue query without any knowledge about structure of xml document. The most important requirement for the keyword search is to rank the results of query so that the most relevant results appear. 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. 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 approach 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. However the traditional approaches are not user friendly. To solve this problem many systems introduced various features. One method id Autocomplete which predicts the words the user had typed in. More and more websites support these features example Google, yahoo. One limitation of this approach is it treats multiple key words as single key word and do not allow them to appear in different places. To address this problem, Bast and Weber [19] proposed complete search in textual documents which allows multiple keywords to appear in different places but it does not allow minor mistakes in query.

Recently, Ji. Feng and G.Li, [1], [2] studied fuzzy type ahead search [1],[2] 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.

But every time it is not possible to store the data as relational data. For example in court case many more documents are whose details can not be stored as a relational data. So if we have these documents in the form of XML document, we can easily apply XML search to retrieve documents vey easily and effectively. So document retrieval is also very important paradigm. When documents are available electronically and you need a hard copy, you can get the documents you need very easily and quickly. For example in case of Documents filed in litigation in federal, state, and local courts, including bankruptcy, tax, and administrative courts and Public court file information etc. In this case you cannot store all the

information as relational database where just buy using any structured query language user can fire the query and get the result. To solve this type of purpose we are trying to import new method which will allow you to search your document by searching simple keyword over XML data. All documents are stored as XML database and keyword searching results are stored in separate document. So that next search will begin with that document first where recently used keywords search results are stored and if it does not found the keyword search will begin with original data. This paradigm will decrease the time required for searching and indirectly improve the performance.

In this paper, we propose a fuzzy type-ahead search method in XML data. Method searches the XML data on the fly as user type in query keywords, even in the presence of minor errors of their keywords. This method provides a friendly interface for users to explore XML data, and can significantly save users typing effort. In this paper, 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 for its execution. 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. We will maintain one data view which stores the results of previous search with its title and document too. As search keyword increase data view size will also increases which will become the real challenge, to maintain this data view. 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 develop a data view to store the successful search results as title and document too.
- We have conducted an extensive experimental study. The results show that our method achieves high search efficiency and result quality.

The remainder of this paper is organized as follows: Section 2 gives the preliminaries. We formalize the problem of fuzzy type-ahead search in XML data, propose a lowest common ancestor (LCA)-based method and introduces a progressive search method in Section 3. Extensive experimental evaluations are provided in Section 4. We review related work in Section 5 and conclude in Section 6.

2. Preliminaries

2.1 Notations

An XML document can be modeled as a rooted and labeled tree. A node v in the tree corresponds to an element in the XML document and has a label. For two nodes u and v, we use "u $\prec$ v" ("u $\succ$ v," respectively) to denote that node u is an ancestor (descendant, respectively) of node v. We use "u $\preceq$ v" to denote that u $\preceq$ v or u = v. For example, consider the XML document in Fig. 1, we have paper (node 5) $\preceq$ author (node 7) and paper (node 12) $\succ$ conf (node 2).

A keyword query consists of a set of keywords $\{k_1; k_2; \ldots; k_l\}$. For each keyword k_i, we call the nodes in the tree that contain the keyword the content nodes for k_i. The ancestor nodes1 of the content nodes are called the quasi-content nodes of the keyword. For example, consider the XML document in Fig. 1, title (node 16) is a content node for keyword "DB," and conf (node 2) is a quasicontent node of keyword "DB."

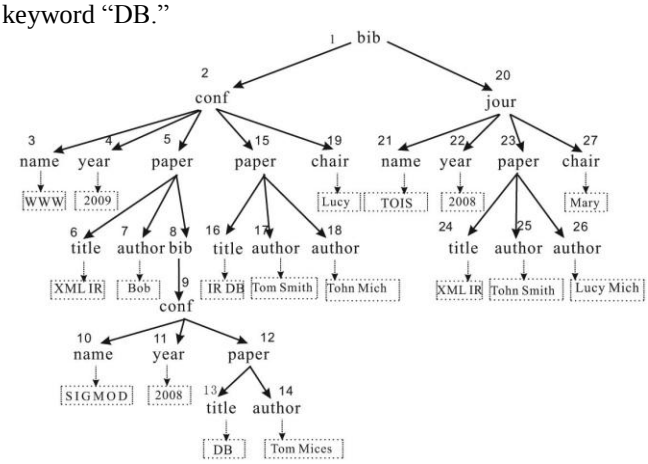

Figure 1: An XML document.

2.2 Information Retrieval

Information retrieval might be regarded as an extension to document retrieval where the documents that are returned are processed to condense or extract the particular information sought by the user. Thus document retrieval could be followed by a text summarization stage that focuses on the query posed by the user, or an information extraction technique.

Here we are trying to discover the previously unknown information by automatically extracting information from a usually large amount of different unstructured textual resources, which is known as Text Mining, by making them structured resources like XML document. Text mining is the combination of different processes like; Data Mining, Information Retrieval, Statistics, Web Mining, Computational Linguistics & Natural Language Processing as shown in below fig 2.

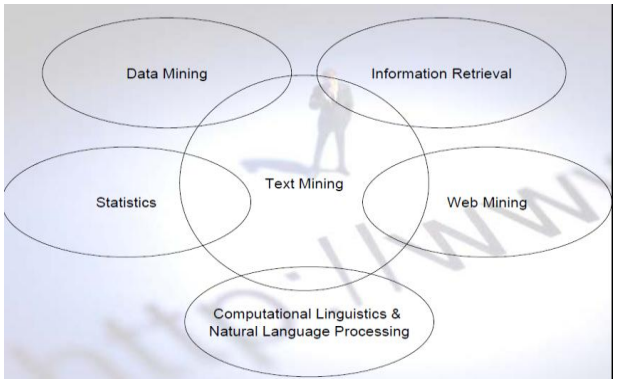

Figure 2: Text Mining

2.3 Keyword Search in XML data

In the literature, there are different ways to define the answers to a keyword query on an XML document .A commonly used one is based on the notion of lowest common ancestor [20]. Given an XML document D and its XML nodes v_1; v_2; . . . ; v_m, we say a node u in the document is the lowest common ancestor of these nodes if $\forall\ 1 < i < m,\ u \prec v_i$ and there does not exist another node u' such that $u \prec u'$ and $u' \preceq v_i$.

Intuitively, each LCA of the keyword query is the LCA of a set of content nodes corresponding to all the keywords in the query. Many algorithms for XML keyword search use the notion of LCA or its variants [19], [6], [5],[4], [7], [28]. For a keyword query, the LCA-based algorithm first retrieves content nodes in XML data that contain theinput keywords using inverted indices. It then identifies the LCAs of the content nodes, and takes the subtrees rooted at the LCAs as the answer to the query. For example, a bibliography XML document is shown in Fig. 1. Suppose a user issues a keyword query "DB Tom." The content nodes of "DB" and "Tom" are {13,16} and {14,17}, respectively. Nodes 2, 12, and 15 are LCAs of the keyword query. Notice that node 2 is the LCA of nodes 13 and 17. Evidently, node 2 is less relevant to the query than nodes 12 and 15, as nodes 13 and 17 correspond to values of different papers.

To address this limitation of using LCAs as query answers, many methods have been proposed [6], [8], [5],[9], [32] to improve search efficiency and result quality. Papakonstantinou [33] proposed exclusive lowest common ancestor (ELCA). Given a keyword query Q = {k_1; k_2; . . . ; k_l} and an XML document D, u 2 D is called an ELCA of Q, if and only if there exists nodes v1 Ɛ Ik_1; v2 Ɛ Ik_2 ; . . . ; v'Ɛ Ik' such that u is the LCA of v_1; v_2; . . . ; v_l, and for every vi, the descendants of u on the path from u to vi are not LCAs of Q nor ancestors of anyLCA of Q.

An LCA is an ELCA if it is still an LCA after excluding its LCA descendants. For example, the ELCAs to the keyword query "DB Tom" on the data in Fig. 1 are nodes 12 and 15. Node 2 is not an ELCA as it is not an LCA after excluding nodes 12 and 15. Xu and Papakonstantinou [33] proposed a binary-search-based method to efficiently identify ELCAs.

2.4 Problem Definition

We formalize the problem of fuzzy type-ahead search in XML data as follows:

Definition 1 (FUZZY TYPE-AHEAD SEARCH IN XML DATA). Given an XML document D, a keyword query Q ={k_1,k_2,….k_l} and an edit-distance threshold $_T$. Let the predicted-word set be W_k={w|w is a tokenized word in D and there exists a prefix of w, k_i', ed(ki,k_i') <= $_T$.} Let the predicted answer set be R_Q={r|r is a keyword-search result of query {$w_1 \in W_{k1}$, w2 $\in W_{k2}$, . . . ,$W_l \in W_{kl}$}}. For the keystroke that invokes Q, we return the top-k answers in RQ for a given value k, ranked by their relevancy to Q.

Let treat the data and query string as lowercase strings. Now focus on how to efficiently find the predicted answers, among which we can find the best top-k relevant answers using a ranking function. There are two challenges to support fuzzy type-ahead search in XML data. The first one is how to interactively and efficiently identify the predicted words that have prefixes similar to the input partial keyword after each keystroke from the user. The second one is how to progressively and effectively compute the top-k predicted answers of a query with multiple keywords, especially when there are many predicted words.

2.5 Method for Keyword search

1. LCA Based Method

The **l**owest common ancestor (LCA) is a concept in graph theory and computer science. Let *T* be a rooted tree with *n* nodes. The lowest common ancestor between two nodes *v* and *w* is defined as the lowest node in *T* that has both *v* and *w* as descendants.

The LCA of *v* and *w* in *T* is the shared ancestor of *v* and *w* that is located farthest from the root. There are different ways to answer the query on an xml document, one commonly used method is LCA based method [3]. Many algorithms that use query over xml uses this method. Content nodes are the parent node of the keyword. For example consider keyword db in fig1 then content node of db is node 13 and node16.The server contains index structure of xml document which each node is letter in keyword and leaf node contain all nodes that contain the keyword this leaf node is called inverted list.

Index Structures

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 leaf node, we store an inverted list of IDs of XML elements that contain the word of the leaf node. For instance, consider the XML document in Fig. 1. The trie structure for the tokenized words is shown in Fig. 3 The word "mich" has a node ID of 10. Its inverted list includes XML elements 18 and 26.

Procedure

- For keyword query the LCA based method retrieves content nodes in xml that are in inverted lists.
- Identify the LCAs of content nodes in inverted list.

- Takes the sub tree rooted at LCAs as answer to the query.

For example suppose the user typed the query "www db" then the content nodes of db are{13,16} and for www are3 ,the LCAs of these content nodes are nodes ,12,15,2,1.here the nodes 3,13,12,15 are more relevant answers but nodes 2 and 1 are not relevant answers.

Limitation

- It gives irrelevant answers.
- The results are not of high quality.

2. ELCA Based Methods

To address the limitation of LCA based method exclusive LCA (ELCA)[4] is proposed. It states that an LCA is ELCA if it is still an LCA after excluding its LCA descendents. for example suppose the user typed the query "db tom" then the content nodes of db are{13,16} and for tom are{14.17} ,the LCAs of these content nodes are nodes2,12,15,1.here the ELCAs are 12,15.the subtree rooted with these nodes is displayed which are relevant answers Node 2 is not an ELCA as it is not an LCA after excluding nodes 12 and 15.

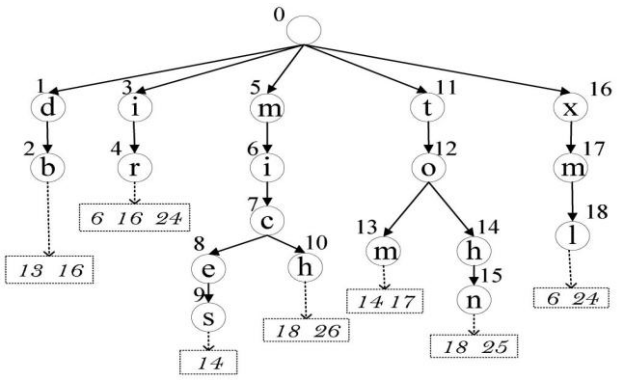

Figure 3: The trie on top of words in Fig. 1 (a part of words).

The LCA-based fuzzy type-ahead 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. 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.

To address these limitations, we develop novel ranking techniques and efficient search algorithms. 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 answers to the query. For each leaf node in the trie. The leaf node inverted list contains the content nodes and quasi contend nodes, scores of the keyword. For computing top k results heap based method [6] is used which uses the partial virtual inverted lists which contain the higher score nodes so to avoid the union of lists which is expensive. Fig. 4 gives the extended trie structure.

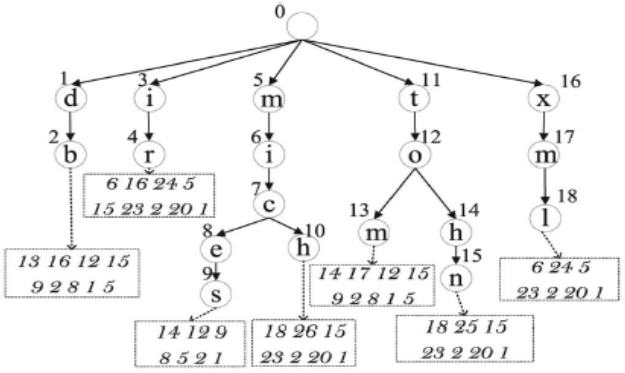

Figure 4: Extended tier structure

3. Progressively Searching the Keyword

3.1 Working

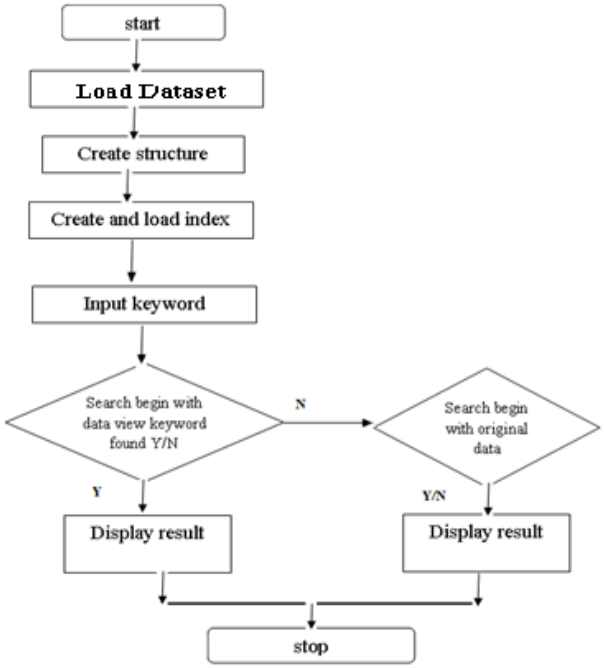

Figure 5: working of keyword search with data view

Above figure 6 shows the working. Here for experimental study purpose we have taken one document which is containing more than 50 xml document data. After creating its structure and index, when we will start keyword searching it will start search first with data view and then it will start searching original data if it doesn't find keyword in data view

3.2 Ranking the Sub tree

There are two ranking function to compute rank/score between node n and keyword ki
1) The case that n contains ki.
2) The case that n does not contain ki but has a descendant containing ki.

Case 1: n contains keyword ki

The relevance/score of node n and keyword ki is computed by

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

Where tf(ki,n) - no:of occurences of ki in subtree rooted n
idf(ki) - ratio of no:of nodes in xml to no:of nodes that contain keyword ki
ntl(n) - length of n /nmax length, nmax=node with max terms
s - Constant set to 0.2
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 2: node n does not contain keyword ki but its descendant has ki
Second ranking function to compute the score between n and kj is

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

Where
P - Set of pivotal nodes
α - constant set to 0.8
$\delta(n, p)$ - Distance between n and p
Assume the user composed query "db"
Score2 (12, db) = (0.8)*score1 (13, db)
= 0.8 *1.52
=1.21

3.3 Ranking Fuzzy Search

Given a keyword query Q={k1,k2,…..kl} in terms of fuzzy search, a minimal-cost tree 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

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

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

4. Experimental Study

We have implemented our method on real applications using our proposed techniques. We used some Reuters 21578 dataset. The sizes of dataset is about 100 in MB. We randomly selected 50 queries for each data set and Table 1 gives some sample queries. We implemented the hybrid algorithm of XRANK [19] for the LCA-based method. We used the Dewey inverted list and hash index. We implemented XRANK's ranking functions. We used the cache for incremental computation. Program implemented in JAVA. We conducted the evaluation on a PC running Windows operating system with an Intel(R) @ 2.5 GHz CPU and 4 GB RAM.

Table 1: Sample Keyword Query Used

Sr. No.	Queries	Typed Queries
1	Company	compa
2	German	germ
3	International	intern
4	Parliament	parl
5	Newyork	newy
6	Derivatives	deriv

4.1 Result Quality

This section evaluates result quality of the LCA-based method and MCT-based method. We generated 50 keyword queries. Answer relevance of the selected queries was judged from discussions of in our group. As users are usually interested in the top-k answers, we employed the top-k precision, i.e., the ratio of the number of answers deemed to be relevant in the first k results to k, to compare the LCA-based method and the MCT-based method. Table 2 shows the average top-k precision of the selected 50 queries. We see that our data view based search method achieves much higher result quality with less response time. This is attributed to our effective ranking functions that rank both content nodes and quasiconten nodes and incorporate structural information into our ranking functions.

Table 2: Precision

Precision %	Top 1	Top 10	Top 50
LCA	50	68	60
MCT	77	81	78
MCT-Data View	80	85	82

4.2 Scalability

This section evaluates the scalability of our algorithms. As an example, we used the Reuters 21578 dataset. We varied the number of XML documents in the data set from 100, 200. Fig. 6 shows the elapsed time of building the index structure, the sizes of indexes, and the average search time for 100 queries. We observe that our method scales very well with the increase of the data. In particular, the size of the trie is sublinear with the number of records. With the increase of the data sizes, the average search time also increased sublinearly. This is because of two main reasons. First, the time of finding the predicted words depends on the number of nodes on the trie, which increases sublinearly as the data size increases. Second, our method to incrementally compute the predicted words and progressively identify the predicted answers can save a lot of computation. Third, very important by using data view, we can store the search keywords and related document to view, data view elapsed time for search is less as compared to search which uses original large volume of data. Figure 6 gives the analysis for total time spend for different queries.

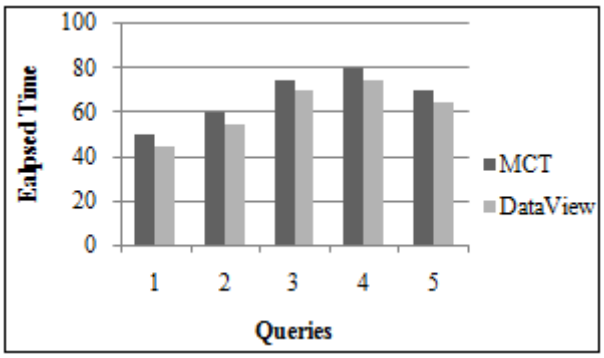

Figure 6: Search Time

5. Related Work

Keyword search in XML data has attracted great attention recently. Xu and Papakonstantinou [3] proposed smallest lowest common ancestor (SLCA) to improve search efficiency. Sun et al. [4] studied multiway SLCA-based keyword search to enhance search performance. Schema free XQuery [5] 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 [6] 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 a same set, which can be given by administrators or users. Li et al. [7] proposed valuable LCA (VLCA) to improve the meaningfulness and completeness of answers and devised a new efficient algorithm to identify the answers based on a stack-based algorithm. XKeyword [8] is proposed to offer keyword proximity search over XML documents, which models XML documents as graphs by considering IDREFs between XML elements. Hristidis et al. [9] proposed grouped distance minimum connecting tree (GDMCT) to answer keyword queries, which groups the relevant subtrees to answer keyword queries. It first identifies the minimum connected tree, which is a subtree with minimum number of edges, and then groups such trees to answer keyword queries. Shao et al. [32] studied the problem of keyword search on XML views. XSeek studied how to infer the most relevant return nodes without elicitation of user preferences. Liu and Chen proposed to reason and identify the most relevant answers. Huang et al. discussed how to generate snippets of XML keyword queries. Bao et al. [18] proposed to address the ambiguous problem of XML keyword search through studying search for and search via nodes.

In addition, the database research community has recently studied the problem of keyword search in relational databases [16], [20], [25],[26] graph databases [13], [23], [12], and heterogeneous data sources. DISCOVER-II [25], BANKS-I [26], BANKS-II [20], and DBXplorer [13] are recent systems to answer keyword queries in relational databases. DISCOVER and DBXplorer return the trees of tuples connected by primary-foreign-key relationships that contain all query keywords. DISCOVER-II extended DISCOVER to support keyword proximity search in terms of disjunctive (OR) semantics, different from DISCOVER which only considers the conjunctive (AND) semantics. BANKS proposed to use Steiner trees to answer keyword queries. It first modeled relational data as a graph where nodes are tuples and edges are foreign keys, and then found Steiner trees in the graph as answers using an approximation to the Steiner tree problem, which is proven to be an NP-hard problem. BANKS-II improved BANKS-I by using bidirectional expansion on graphs to find answers. He et al. [12] proposed a partition based method to efficiently find Steiner trees using the BLINKS index. Ding et al. [14] proposed to use dynamic programming for identifying Steiner trees. Dalvi et al. [Z] studied disk-based algorithms for keyword search on large graphs, using a new concept of "supernode graph."

More recently, Kimelfeld and Sagiv [14] discussed keyword proximity search in relational databases from theory viewpoint. They showed that the answer of keyword proximity search can be enumerated in ranked order with polynomial delay under data complexity. Golenberg et al. presented an incremental algorithm for enumerating subtrees in an approximate order which runs with polynomial delay and can find all top-k answers. Guo et al. [24] studied the problem of data topology search on biological databases. Sayyadian et al. [39] incorporated schema mapping into keyword search and proposed a new method to answer keyword search across heterogenous databases. Liu et al incorporated [27] IR ranking techniques to rank answers on relational data. They employed the techniques of phrase-based and concept-based models to improve result quality. Luo et al. [30] proposed a newranking method that adapts state-of-the-art IR ranking functions and principles into ranking tree-structured results composed of joined database tuples. They incorporated the idea of skyline to rank answers. Balmin et al. proposed Object-Rank [17] to improve results quality by extending hub-and-authority ranking-based method. This method is effective in ranking objects, pages, and entities, but it may cannot effectively rank tree-structured results (e.g., Steiner trees), since it does not consider structure compactness of an answer in its ranking function. Richardson and Domingos proposed to combine page content and link structure to answer queries.
Tao and Yu [36] proposed to find co-occurring terms of query keywords in addition to the answers, in order to provide users relevant information to refine the answers. Koutrika et al [15] proposed data clouds over structured data to summarize the results of keyword searches over structured data and use them to guide users to refine searches. Zhang et al. [35] and Felipe et al. [29] studied keyword search on spatial databases by combining inverted lists and R-tree indexes. Tran et al. [37] studied top-k keyword search on RDF data using summarized RDF graph. Qin et al.[39] studied three different semantics of m-keyword queries, namely, connect-tree semantics, distinct core semantics, and distinct root semantics, to answer keyword queries in relation databases. The search efficiency is achieved by new tuple reduction approaches that prune unnecessary tuples in relations effectively followed by processing the final results over the reduced relations. Chu et al. [22] proposed to combine forms and keyword search, and studied effective summary techniques to design forms. Yu et al. [34] and Vu et al. [38] studied keyword search over multiple databases in P2P environment. They emphasized on how to select relevant database sources in P2P environments. Chen et al.

[21] gave an excellent tutorial of keyword search in XML data and relational databases.

Type-ahead search is a new topic to query relational databases. Li et al. [31] studied type-ahead search in relational databases, which allows searching on the underlying relational databases on the fly as users type in query keywords. Ji et al. [11] studied fuzzy type-ahead search on a set of tuples/documents, which can on the fly find relevant answers by allowing minor errors between input keywords and the underlying data. A straightforward method for type ahead search in XML data is to first find all predicted words, and then use existing search semantics, e.g., LCA and ELCA, to compute relevant answers based on the predicted words. However, this method is very time consuming for finding top-k answers. To address this problem, we propose to progressively find the most relevant answers. For exact search, we propose to incrementally compute predicted words. For fuzzy search, we use existing techniques [11] to compute predicted words of query keywords. We extend the ranking functions in [31] to support fuzzy search, and propose new index structures and efficient algorithms to progressively find the most relevant answers. Text mining which is the combination of document retrieval, information retrieval, web mining etc, is also added to create a data view, which will make keyword searching faster as compared to ordinary searching where every search begins with the original data.

6. Conclusion

In this paper, we studied the problem of fuzzy type-ahead search in XML data. We 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 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 method with data view, and the experimental results show that our method achieves high search efficiency and result quality.

References

[1] G. Li and J. Feng, "Efficient Fuzzy Type-Ahead Search in XML Data," MAY 2012
[2] S. Ji, G. Li, C. Li, and J. Feng, "Efficient Interactive Fuzzy Keyword Search," Proc. Int'l Conf. World Wide Web (WWW), pp. 371-380,2009.
[3] Y. Xu and Y. Papakonstantinou, "Efficient Keyword Search for Smallest Lcas in XML Databases," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 537-538, 2005.
[4] C. Sun, C.Y. Chan, and A.K. Goenka, "Multiway Slca-Based Keyword Search in Xml Data," Proc. Int'l Conf. World Wide Web (WWW), pp. 1043-1052, 2007.
[5] Y. Li, C. Yu, and H.V. Jagadish, "Schema-Free Xquery," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 72-83, 2004.
[6] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv, "Xsearch: A SemanticSearch Engine for Xml," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 45-56, 2003.
[7] G. Li, J. Feng, J. Wang, and L. Zhou, "Effective Keyword Search for Valuable lcas over XML Documents," Proc. Conf. Information and Knowledge Management (CIKM), pp. 31-40, 2007.
[8] V. Hristidis, Y. Papakonstantinou, and A. Balmin, "KeywordProximity Search on XML Graphs," Proc. Int'l Conf. Data Eng. (ICDE), pp. 367-378, 2003.
[9] V. Hristidis, N. Koudas, Y. Papakonstantinou, and D. Srivastava, "Keyword Proximity Search in Xml Trees," IEEE Trans. Knowledge and Data Eng., vol. 18, no. 4, pp. 525-539, Apr. 2006
[10] Y. Huang, Z. Liu, and Y. Chen, "Query Biased Snippet Generation in Xml Search," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 315-326, 2008
[11] S. Ji, G. Li, C. Li, and J. Feng, "Efficient Interactive Fuzzy Keyword Search," Proc. Int'l Conf. World Wide Web (WWW), pp. 371-380, 2009\
[12] H. He, H. Wang, J. Yang, and P.S. Yu, "Blinks: Ranked Keyword Searches on Graphs," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 305-316, 2007
[13] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar, "Bidirectional Expansion for Keyword Search on Graph Databases," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 505-516, 2005
[14] B. Kimelfeld and Y. Sagiv, "Finding and Approximating Top-k Answers in Keyword Proximity Search," Proc. ACM SIGMODSIGACT- SIGART Symp. Principles of Database Systems (PODS), pp. 173-182, 2006
[15] G. Koutrika, Z.M. Zadeh, and H. Garcia-Molina, "Data Clouds: Summarizing Keyword Search Results over Structured Data," Proc. Int'l Conf. Extending Database Technology: Advances in Database Technology (EDBT), pp. 391-402, 2009.
[16] 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.
[17] 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.
[18] 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.
[19] 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.
[20] 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.
[21] 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.

[22] 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.
[23] 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.
[24] L. Guo, J. Shanmugasundaram, and G. Yona, "Topology Search over Biological Databases," Proc. Int'l Conf. Data Eng. (ICDE), pp. 556-565, 2007.
[25] V. Hristidis, L. Gravano, and Y. Papakonstantinou, "Efficient Ir- Style Keyword Search over Relational Databases," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 850-861, 2003.
[26] V. Hristidis and Y. Papakonstantinou, "Discover: Keyword Search in Relational Databases," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 670-681, 2002.
[27] F. Liu, C.T. Yu, W. Meng, and A. Chowdhury, "Effective Keyword Search in Relational Databases," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 563-574, 2006.
[28] Multiobjective Optimization: NSGA II," KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000. (technical report style)
[29] J. Geralds, "Sega Ends Production of Dreamcast," vnunet.com, para. 2, Jan. 31, 2001. [Online]. Available: http://nl1.vnunet.com/news/1116995. [Accessed: Sept. 12, 2004]. (General Internet site)
[30] Z. Liu and Y. Chen, "Identifying Meaningful Return Information for Xml Keyword Search," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 329-340, 2007.
[31] Z. Liu and Y. Chen, "Reasoning and Identifying Relevant Matches for Xml Keyword Search," Proc. VLDB Endowment, vol. 1, no. 1, pp. 921-932, 2008.
[32] Y. Luo, X. Lin, W. Wang, and X. Zhou, "Spark: Top-k Keyword Query in Relational Databases," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 115-126, 2007.
[33] G. Li, S. Ji, C. Li, and J. Feng, "Efficient Type-Ahead Search on Relational Data: A Tastier Approach," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 695-706, 2009.
[34] F. Shao, L. Guo, C. Botev, A. Bhaskar, M.M.M. Chettiar, F.Y. 0002, and J. Shanmugasundaram, "Efficient Keyword Search over Virtual XML Views," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 1057-1068, 2007.
[35] Y. Xu and Y. Papakonstantinou, "Efficient LCA Based Keyword Search in XML Data," Proc. Int'l Conf. Extending Database Technology: Advances in Database Technology (EDBT), pp. 535-546, 2008.
[36] B. Yu, G. Li, K.R. Sollins, and A.K.H. Tung, "Effective Keyword- Based Selection of Relational Databases," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 139-150, 2007.
[37] D. Zhang, Y.M. Chee, A. Mondal, A.K.H. Tung, and M. Kitsuregawa, "Keyword Search in Spatial Databases: Towards Searching by Document," Proc. Int'l Conf. Data Eng. (ICDE), pp. 688-699, 2009.
[38] Y. Tao and J.X. Yu, "Finding Frequent Co-Occurring Terms in Relational Keyword Search," Proc. Int'l Conf. Extending Database Technology: Advances in Database Technology (EDBT), pp. 839-850, 2009.
[39] T. Tran, H. Wang, S. Rudolph, and P. Cimiano, "Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data," Proc. Int'l Conf. Data Eng. (ICDE), pp. 405-416, 2009.
[40] Q.H. Vu, B.C. Ooi, D. Papadias, and A.K.H. Tung, "A Graph Method for Keyword-Based Selection of the Top-k Databases," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 915-926, 2008.
[41] L. Qin, J.X. Yu, and L. Chang, "Keyword Search in Databases: The Power of Rdbms," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 681-694, 2009

Author Profile

Supriya. N. Chaudhari received the B.E. in Computer Science & Engineering from Jawaharlal darda Institute of Engineering and Technology, Yavatmal Maharashtra, India in 2006 and pursuing M.E. degrees in Computer Science & Engineering from Prof. Ram Meghe Inst. Of Tech. & Res, Badnera, Amravati, Maharashtra India.