User-Friendly Keyword Based Search on XML Data

Jeetendra G. Kapase[1], **Sharmila M. Shinde**[2]

[1]Computer Engineering Department, JSCOE, Pune University, India

[2]Head of Computer Engineering Department, JSCOE, Pune University, India

Abstract: ***XML data is independent of platform. Therefore it's mainly used across system in order to exchange information. So searching XML data based on keyword is an important task. Considering current or existing system to find XML data based on given keyword, a user need to know XML query language to create keyword query, submit the query for processing and get the results. In this approach user having limited knowledge about XML data and XML query language, often feels complex and difficult to find require keyword in XML data and sometimes they has to use try and see approach to get relevant information. To overcome problem with existing system, Feng and Li proposed system to search XML data based fuzzy type-ahead searching methodology. This methodology has advantage as high quality and fast searching using effective index structure and top-k algorithm. In this paper we studied existing system and we proposed new framework for keyword based search over XML data by using forward index structure methods to enhance fuzzy type ahead search. The proposed system thus makes XML search user friendly and efficient. This proposed prototype will be effective and feasible to design and develop real world based XML data search application.***

Keywords: fuzzy search, forward index structure, keyword search, LCT, MCT, Top-k algorithm, user-friendly, XML Data.

1. Introduction

XML data form is widely used and accepted due to its platform independent feature. Existing system uses XML query languages such as XQuery and XPath to find XML data. This is powerful tool but it's complex and less friendly to non-database users. For example, XQuery *doc ("books.xml")/bookstore/book / [price<30]*, such XML queries are difficult to write and need good knowledge of XML query languages. There are many researcher who contributed to above challenges pertaining to search XML data [1], [2], [3], [4], [5], [6], [7], [8], [9],[14] with keyword oriented application which are user friendly and efficient to use. Where user just needs to think and type the desire keyword to be searched over XML data and the result is in the form of XML data based files containing data in XML elements to be searched. The proposed keyword based search approach is quite simple and does not require database query language expert users. Keyword search is widely accepted methodology to access information. In a traditional keyword-search system over XML data, a user creates a XML query, submits it to the system for processing, and retrieves relevant answers from XML data. This approach requires knowledge about XML data structure and data content. In the case where user who has limited knowledge, feels difficult to adopt this approach. Sometimes user will require trying few possible keywords; this could be tedious and time consuming activity.

Keyword based search becomes challenging task on XML data as there is hierarchical data relationship of XML data elements. In this paper, we study, a Fuzzy type-ahead search method in XML data [1]. It searches the XML data on the fly as user's type in query keywords, even in the presence of minor errors of their keywords. It provides a friendly interface for users to explore XML data, and can significantly save users typing effort. In this Paper, we study important research challenges that arise naturally in computing system. The main challenge is with addressing search efficiency. Each user query with multiple keywords searched needs to be answered fast and efficiently. To achieve our goal, we study effective index structures and top-*k* algorithms to answer keyword queries in XML data. The fuzzy type ahead search in xml data returns the approximate results.

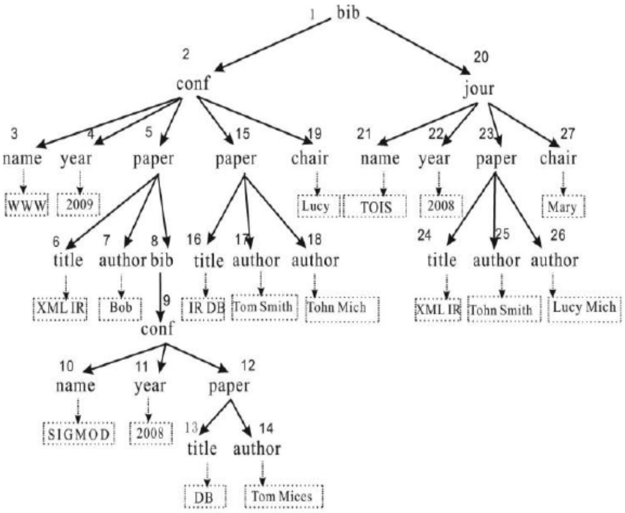


Figure 1: Parent-Child based XML tree structure

We study the effective ranking function and early termination techniques. In this paper we proposed technique to improve the keyword based fuzzy type ahead search on XML data by using forward index structure based method. The proposed system thus makes XML search user friendly and efficient. This proposed prototype will be effective and feasible to design and develop real world based XML data search application.

2. Related Work

Bast and Weber [10][11] proposed complete search in textual document based on type less, find more, which document, which can find relevant answer by allowing query keyword appear at any place in the document. However, this does not support approximate search, which can't allow minor error between query keyword and result. S Ji, G. Li and Feng proposed efficient interactive fuzzy search on textual document [12]. It allows user to search data on the fly, this framework allows user to explore data even in presence of minor errors. We also studied type-ahead search on relational databases [13]. *Lowest common ancestor* (LCA) of keyword query in the LCA of set of *content node* corresponding to all the keyword in the query. Many algorithms for XML keyword search use the notation of LCA [7]. To improve search efficiency and result quality, Xu and papkonstantiou [7] proposed *Exclusive Lowest Common Ancestor* (ELCA). Type-ahead search is also main part to specify matching relevant keywords into result statement even in the presence of minor error, output is approximate results [15]. The limitation of XML query is hard to understand to user and require query expert users [1]. To solve the minor error keyword search problem and matching particular word into query type-ahead search [1]. [1] *Minimal cost tree* (MCT) is for each node, sub tree is defined for corresponding answer to the query with paths to the node of tree that include searching keyword. J Chen, Lyad A. Kanjb showed how top-*k* algorithm works on XML database and how ranking of keyword helps in effective manner [16]. G Li, Chen Li, J Feng and L Zhou define how to retrieve particular keyword present in XML file and how to retrieve accurately if particular keyword is not perfectly matching [17]. Hristidis, Koudas, Srivastava proposed framework to identify the most specific LCA (i.e., context elements) along with *compact description minimum cost tree* (i.e., GDMCTs) [18].

We mainly focused and studied system proposed by Feng, Li [1], type-ahead search on XML content based on fuzzy approach, novel raking algorithm, extended trie structure, *minimal-cost tree* (MCTs) to construct answers rooted, Ranking MCT based on scores, finding top-*k* results. MCT technique constructs minimal-cost tree for node in the XML tree based predicted word, and return results having highest score, it has limitation that Top-Bottom, Left-Right node traversing require much time. LCA technique based algorithm first retrieves *content nodes* in XML data that contains input keyword using inverted indices. Then LCA of *content node* is identified and takes sub trees as the answer of the keyword. Limitation of LCA is that it may select less relevant sub tree based on the keyword, to address this limitation, many method have been proposed. Guo et al. [19] and Xu and Papakonstantinou [20] proposed *exclusive lowest common ancestor* (ELCA). Limitation LCA method is they use the "AND" semantic between keywords and ignore answers that contains some partial part of keywords and also need to find candidates first before ranking them, and this approach not efficient for computing the results. In fuzzy based type ahead technique computation time is at higher end in multi keyword search scenario.

3. Implementation

3.1 Proposed Framework

The proposed system framework is an enhancement to techniques introduced in [1].The main motive of proposed system is to improve XML data search efficiency. This helps user to make efficient processing of XML keyword search without knowledge of data content in XML data sources and non-database expert can easily retrieve results based given inputs. The main advantage of proposed system is its user-friendly interface design. New framework makes use of forward index structure approach in order to obtain top-*k* results with less processing time and populate the results. Additional feature in new system is that it supports approximate keywords search even presence of minor error. Keywords *'India', 'ndia', 'Idia', 'Indi'* will return same result.

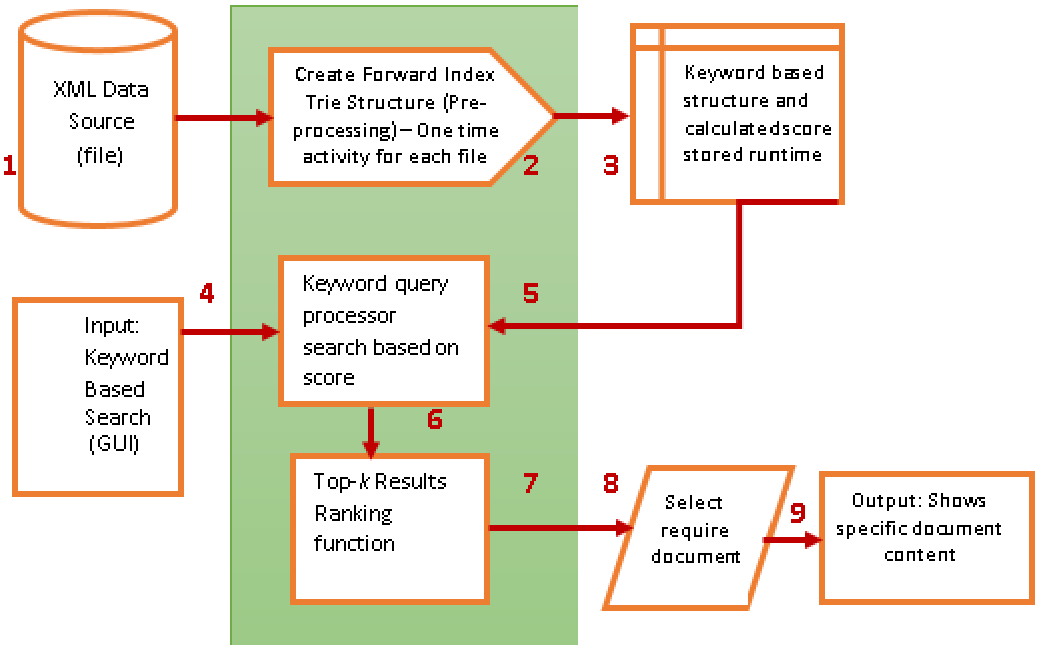

XML Data Source (file)
1
Create Forward Index Trie Structure (Pre-processing) – One time activity for each file
2
3
Keyword based structure and calculated score stored runtime
4
Input: Keyword Based Search (GUI)
Keyword query processor search based on score
5
6
Top-k Results Ranking function
7
8
Select require document
9
Output: Shows specific document content

Figure 2: Proposed System Model (Framework).

The proposed system uses forward indexing concept that creates indexing in the pre-processing step (step 2). This forward index helps to improve keyword based query processing. System take input from the user interface in the form single or multiple keywords as input (step 4). Here data source are XML data files (i.e. text files containing XML data) (step 1).Pre-processing step generates multiple sub trees based keyword and calculated score (step 3). Ranking function assigns numeric values to obtain results and generates top-*k* ranking (step 7). Based on ranking score highest or equal or zero results are filtered and displayed (step 8).User will select require document node and result is populated (step 9). The proposed system falls under NP-hard class category.

3.2 Details of System Model

In this section we showed details of proposed methods or Techniques. Major problem associated with standard XML query processing tool with their syntax (XPath, XQuery). LCA and ELCA based search method [7] [20]. MCT references are better and efficient [1]. Effective index structure and top-*k* algorithm propose to achieve high interaction speed [1].

1) *LCA based method:* The *Lowest common ancestor* (LCA) is concept in data structure graph theory. Let *T* be rooted tree with n nodes. The LCA between to two nodes *u* and *v* is defined as the lowest node in *T* that has both *u* and *v* as descendants. The LCA of *u* and *v* in *T* is the shared ancestor between *u* and *v* that is located farthest from root. This is most commonly used method to search XML data. *Contents nodes* are parent (sub rooted tree) node of the keyword. For e.g. consider keyword *"db"* to be searched in XML file shown in fig1.Then the content node are 13 and 16.Here server contains index structure of XML file where each node is letter of keyword and leaf node contains all node that contains keyword such leaf node are also called as *inverted list..*

   Procedure:

   a) LCA based method retrieves *content node* and there inverted list from XML data based on given keyword.
   b) Identify lowest common ancestor of *content node* in inverted list.
   c) Takes the rooted sub tree at LCA as answer of given keyword.

For example: user gave "*www db*" as input query then *content nodes* of *"db"* are {13, 16} and for *"www"* are {3}. The LCA identified of this *content node* are {12,15,2,1}.Here if we see that nodes {3,13,12,15} are more appropriate answers compared to nodes {2,1} which are not relevant, same is the limitation of LCA due to less relevant and not high quality results.

2) *ELCA based method:* To address limitation with LCA method, *exclusive lowest common ancestor* (ELCA) [21] method is proposed. It states that an LCA method is ELCA if it is still LCA after excluding its descendants LCA. For example. User typed keyword as *"db, tom"* then the *content node* for *"db"* are {13, 16} and for *"tom"* {14, 17}, identified LCA of the *content node* are {2, 12, 15, 1}. Here ELCA are {12, 15}. Rooted sub tree is displayed which are appropriate answer of given query. Node 2 is not an ELCA as it's not LCA after excluding nodes {12, 15}. To improve performance Xu and Papakonstantinou [7] proposed binary search based ELCA method.

3) *MCT based method:* To search appropriate answer based on keyword query over XML data. For each node we define its corresponding answer to query as its sub tree with paths to nodes that include keywords. Such tree is called as *minimal cost tree* (MCT).Different node corresponds to different answer to keyword.

Definition (MINIMAL-COST TREES) Given XML document D, n nodes in D, and keyword query Q = {k_1,k_2,...k_l}, a MCT of query Q and n node is sub tree rooted at n ,and for each keyword k_i Ɛ Q, if node n is a quasi-content node ok k_i, the sub tree includes the pivotal paths for k_i and node n.

Proposed parameterized top-*k* result algorithm executes in two phases. First one is structure algorithms that on a problem instance construct a trie structure of feasible size, and the second stage is an enumerating algorithm that produces the top-k best solutions to the keyword based on the structure. We studied and proposed new techniques that support efficient enumerating algorithm. For example: Consider the parent-child relationship tree based XML document in Fig. 1 and user keyword query *Q = {"db"," tom"," www"}.* Nodes {3, 13, 14, 16, and 17} are *content nodes* of the three keywords; nodes {1, 2, 5, 8, 9, 12, and 15} are their *quasi-content nodes*. Node 3 is pivotal *node* for node 2 and keyword *"www".* Node 16 is *pivotal node* for node 2 and keyword *"db".* Node 17 is *pivotal node* for node 2 and keyword "tom". The MCT of node 2 is the sub tree rooted at node 2, which contains the paths*:* $n_2 \rightarrow n_3$*,* $n_2 \rightarrow n_{15} \rightarrow n_{16}$*,* and $n_2 \rightarrow n_{15} \rightarrow n_{17}$*.* The main advantage of this approach is that, even if node doesn't have descendant nodes that it includes all the keywords of query and this node could still be considered as appropriate answer.

4) *Ranking Function:* There are mainly two ranking function to compute rank or score between node *n* and keyword k_i [22].

   a) The case that node *n* contains keyword k_i.
   b) The case that node *n* doesn't contain keyword k_i but has descendant containing k_i.

First case: n contains keyword k_i the relevance/score of node *n* and keyword k_i is computed by

$$SCORE_1(n,k_i) = \frac{\ln(1+tf(k_i,n)) \cdot \ln(idf(k_i))}{(1-s)+s \cdot ntl(n)}$$

Where,
tf (k_i, n) – number of occurrences of k_i in sub tree rooted *n.*
idf (k_i) – It is the ratio of number nodes *n* in XML data to number of nodes contain keyword k_i
ntl (n) – length of node *n /nmax* length, *nmax*=node with max number of terms
s - Constant set to 0.2 (assumption based)
Assume user entered a query containing keyword "*db",* so ranking function will return,

SCORE$_1$ *(13, db) = ln (1+1)*ln (27/2)*
--------------------- = 1.52
*(1- 0.2)+ (0.2*1)*

Second case: node *n* doesn't contain keyword k_i but its descendant has k_i; ranking function to compute the score between *n* and k_i is derived by,

$$SCORE_2(n,k_i) - \sum_{p\in P} \alpha^{\delta(n,p)} * SCORE_1(p,k_i)$$

Where,
p - Set of pivotal nodes
α - constant set to 0.8 (assumption based)
$\delta(n,p)$- Distance between node n and pivotal node p
Assume the user entered query *"db"*, so ranking function will return,
SCORE$_2$ *(12, db) = (0.8)*score1 (13, db)*
*= 0.8 *1.52 = 1.21*

5) *Ranking for fuzzy search:* Given a query $Q= \{k_1, k_2 ... k_l\}$ with keyword 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 $\{w_1, w_2... w_l\}$ the best similar prefix of w_i could be considered to be most similar to k_i .The function to identify or quantify the similarity between k_i and w_i is given by below

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

Where *ed* – edit distance, a_i – similar prefix, w_i – predicted word, γ – constant (assumption based)

6) *Compute top-k results progressively:* The trie based index structure is used to compute the results. The leaf node inverted list contains the content nodes and quasi contend nodes, scores/rank of the keyword. For computing top-*k* answers heap based method [1] is used which uses the Partial virtual inverted lists which contain the higher score nodes so to avoid the union of lists on the fly which is expensive provision.

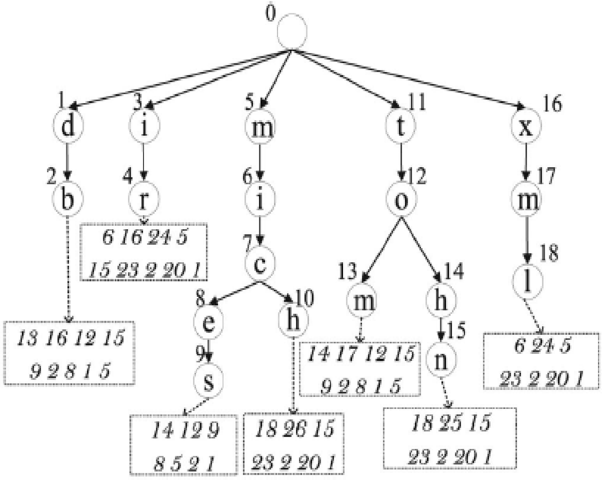


Figure 3: Extended tier structure

Procedure:
a) Sort scores in the inverted lists.
b) Inverted list is too long then partial virtual list considered heap based.
c) Construct MAX heap, such that node contain <*node, score*> form.
d) The top node of MAX heap is the highest score of node and is deleted with heap adjusted.
e) Deleted node with score value <= *T* (threshold) are taken into result set and return result set if top-*k* answers are retrieved.

For example: assume user entered the query *"db".* The inverted list of *"db"* contains the nodes {13, 16, 12,15,9, 2, 8, 1, and 5}.The scores of all these nodes computed by above two ranking functions are {1.52, 1. 52, 1.21, 1.21,0. 9728, 0, 495, 0.77, 0, 396, 0.6225 } respectively. These scores need to be sorted and MAX heap is constructed and a threshold is fixed be 10(assumption based) so the top elements of MAX heap are < (13, 1.52>, <16, 1.52>, <12, 1.21>, <15, 1.21> the top-*k* results are retrieved. This technique is more effective and efficient.

7) Improvement Using Forward Index structure method:
We proposed and implemented the forward index structure method to improve search performance. We made use of "random access" based on this method to do an early termination of 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 structure, without accessing inverted lists. Fagin et al. have proved that threshold-based algorithm using random access is optimal and efficient over all algorithms that correctly find the top-*k* results [1]. Thus, we implemented a forward index structure to implement random access.
Procedure
a) Construct a trie structure to maintain the keyword contained in the element on XML file.
b) Each leaf node in the forward index keeps score of element e to the corresponding keyword of the leaf node.
c) Given partial keyword we can efficiently check element e contains a word having similar prefixes

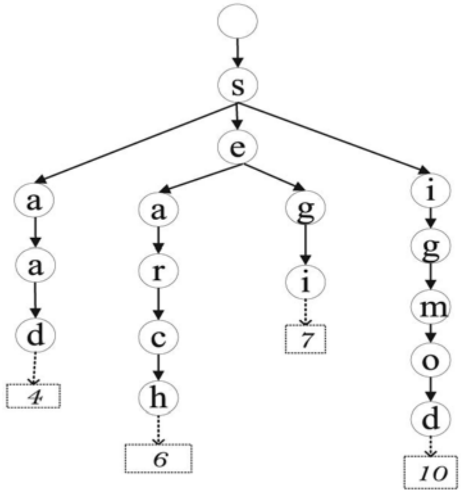
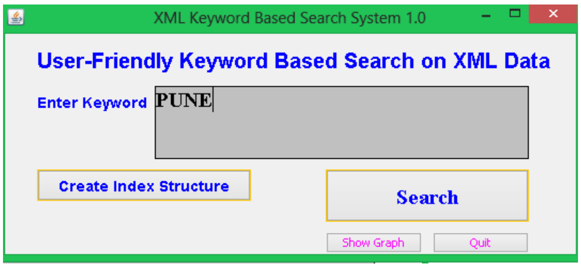


Figure 4: The forward index approach for element, Element contains keywords *{"saad," "search," "segi," and "sigmod."}*

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. Similarly, we can use forward index approach to improve search efficiency and performance.

4. Results

User interface (UI) a system implemented using java (JDK 6.0) language using swing and applets concept. As shown in Fig 5. Use initially need to 'Create Index Structure' pre-processing step of creating forward index. Single or multiple keyword input is accepted.



XML Keyword Based Search System 1.0

User-Friendly Keyword Based Search on XML Data

Enter Keyword PUNE

Figure 5: User Interface of implemented system

As experimental result we have compared search time of existing method extended index with our implemented system based on forward index method and found that new method has shown good performance improvement, blue bars indicates existing system search time and light brown bar indicates new system. XML file having 50 records where two keyword search time comparison is shown in Fig. 6. And XML file having 1000 records where single keyword search time comparison is shown in Fig. 7.

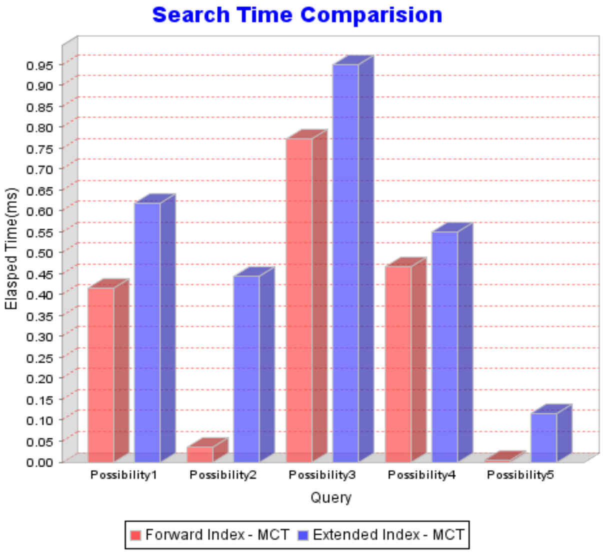


Figure 6: Search time comparison, XML file with 50 records.

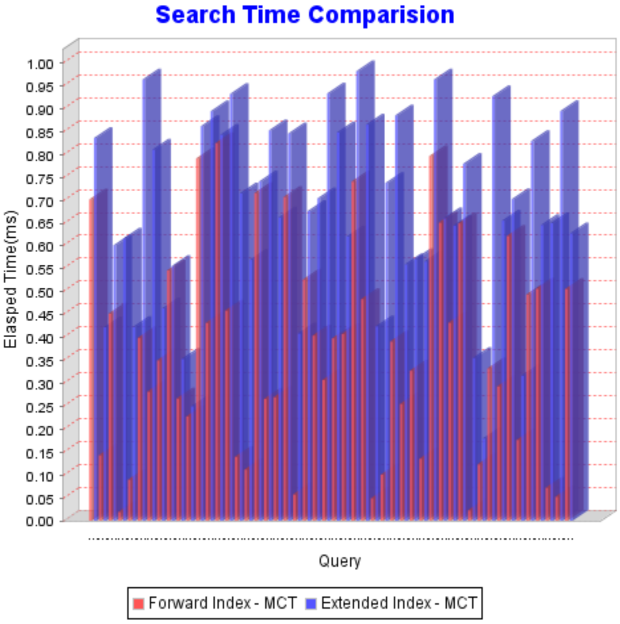


Figure 7: Search time comparison, XML file with 1000 records.

We can see that with implemented new system do not have extra memory cost on system, it has similar memory requirement as compared to existing system, table 1 shows the experimental results and required memory cost.

Table 1: Data sets and Index memory cost

Dataset (Input file documents)	Index cost (Memory)	Index structure
1000	<5 KB	Forward Index
1000	<5 KB	ExtendedIndex
500	<2 KB	Forward Index
500	<2 KB	ExtendedIndex

Forward index strcture based method showed better search time and effciency with no change in result quality and approximate keyword search aswell takes less search time. Experiment shown that this new proposed techniquie can be applied to real world XML search application. Comparative result can be verified from table 2.

Table 2: Data sets and Search cost (ms)

Dataset (Input file documents)	Search cost-single keyword(ms)	Search cost-multiple keyword(ms)	Index structure
1000	0.67	0.79	Forward Index
1000	0.88	1.00	ExtendedIndex
500	0.45	0.59	Forward Index
500	0.55	0.71	ExtendedIndex
50	0.31	0.46	Forward Index
50	0.51	0.59	ExtendedIndex

5. Conclusion and Future Scope

In this paper we studied the problems associated with searching XML data. We enhanced framework proposed Feng and Li [1], [17], fuzzy type-ahead search mechanism. We proposed user-friendly and efficient keyword based XML searching system without prior knowledge of XML data. We studied and implemented LCA, ELCA and MCT based combined approach to improve searching using forward index tree formation technique. We studied and implemented ranking function for two different cases. The indexing and forward indexing approach structure we used improves the performance of search process. The proposed system framework is useful and feasible to be used with real world search systems that operate on XML data sources.

6. Acknowledgment

I express true sense of gratitude towards my project guide Prof. S.M. Shinde, head of computer department for his invaluable co-operation and guidance that she gave me throughout my project. I like to specially thank our P.G coordinator Prof. M.D.Ingle for inspiring me and providing me all the lab facilities, which made project work very convenient and easy. I would also like to express my appreciation and thanks to JSCOE principal Dr. M.G. Jadhav and all my friends who knowingly or unknowingly have assisted me throughout my hard work.

References

[1] Jianhua Feng and Guoliang Li, “Efficient Fuzzy Type-Ahead Search in XML Data”. IEEE Transactions on Knowledge and Data Engineering, Vol. 24, NO. 5, MAY 2012.

[2] 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.

[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,“MultiwaySlca-Based Keyword Search in Xml Data,” Proc. Int’l Conf. World Wide Web (WWW), pp. 1043-1052, 2007.

[5] 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.

[6] Y. Li, C. Yu, and H.V. Jagadish, “Schema-Free XQuery,” Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 72-83, 2004.

[7] 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.

[8] G. Li, C. Li, J. Feng, and L. Zhou, “Sail Structure-Aware Indexing for Effective and Progressive Top-k Keyword Search over XML Documents,” Information Sciences, vol. 179, no. 21,pp. 3745-3762, 2009.

[9] 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.

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

[11] H. Bast and I. Weber, “Type Less, Find More: Fast Auto completion Search with a Succinct Index,” Proc. Ann. Int’l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR), pp.364-371, 2006.

[12] 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.

[13] 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.

[14] 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. L.Li, H. wang, J. LI, H.Gao” Efficient algorithm for skyline top-k keyword queries on XML streams” Harbin Institute of Technology.

[15] 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,2009

[16] G. Li, J. Feng, and L. Zhou, “Interactive Search in Xml Data,” Proc. Int’l Conf. World Wide Web (WWW), pp. 1063-1064, 2009.

[17] 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.

[18] 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.

[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] Z. Bao, T.W.Chen and J. Lu,” Effective XML Keyword search with relevance oriented Ranking”, proc. Int’l conf. Data Eng. (ICDE)2009

[21] 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.

Author Profile

Jeetendra G. Kapase is Student of ME Computer Engineering, Jayawantrao Sawant College of Engineering, Hadapsar, Pune University, India.

Prof. Sharmila M. Shinde is serving as Asssistant Professor, Head of Computer Engineering Departement, Jayawantrao Sawant College of Engineering, Hadapsar, Pune University, India.