XML Keyword Search Based On Probabilistic Approach

Dr. Dayananda. P[1], Dilip M Dodamani[2]

[1]*Assistant Professor, Dept. of ISE, MSRIT, Bengaluru-54, India*

[2]*M.Tech (Software Engineering), Dept. of ISE, MSRIT, Bengaluru-54, India*

***Abstract*- Keyword search is an easy and friendly approach to retrieve data from XML datasets. Since an XML document can have a small to relatively large amount of size and contain lots of user necessary data. An XML search output should be a part of the XML document which is formatted dynamically at search time, which is formed due to the structuredness of XML. Applying keyword searches over XML documents has number of problems, such as, what are the relevant elements that are matched in the document, way to generate the meaningful results, way of presenting the found results in simple and understanding form and so on.**

Here, keywords searching technique based on probabilistic approach has been discussed. A probabilistic based XML documents differs from traditional XML documents. The nodes are considered either as ordinary node or distributional nodes. ELM (Extreme Learning Machine) provides venomous advantages such as fast learning speed, ease of implementation. In this paper, ELM has been chosen to classify the nodes and calculate the probability. When keyword is queried, SLCA node results will be searched and probability of SLCA node rooted subtree is calculated by classifying the nodes into ordinary nodes and distributional nodes. The result is compared with existing approaches like Prstack and SVM. The proposed technique results in higher precision with lower execution time.

***Index Terms* - XML, keyword search, Probabilistic document, ELM, SLCA, Searching techniques.**

I. INTRODUCTION

Keyword search is the popular and most widely applied on XML data. It is considered to be effective and user friendly in retrieving the information required from XML data through querying. User doesn't need to have the knowledge of underlying document structure and complex query languages. Keyword search is powerful method for an ordinary user to retrieve information from XML documents. Searching over XML document differs from searching on text data. Searching over XML documents results in subtree rooted at common ancestor node. There are various choices in defining the common ancestor nodes, such as LCA (lowest common ancestor), ELCA (exclusive LCA), SLCA (smallest LCA) and so. These methods are used to extract subtree rooted at common ancestor. In this paper, SLCA concept has been used, which is considered to be the smallest set according to the definition.

Traditional databases were managing only the deterministic information which is considered to be a drawback. Nowadays many applications use data which are uncertain. Because of flexibility, it will allow normal way of representing the uncertain kind of data. Out of many, a probabilistic XML model, in which a probabilistic XML document (also called as p-document) is considered as tree which has labelled different types of nodes such as ordinary nodes and distributional nodes. Ordinary nodes are used to show the actual data while the distributional nodes are used to show probability distribution of children nodes. There are two types of distributional nodes. They are independent node (IND node) and mutually exclusive node. (MUX node).If a node is IND node then children nodes have no relation among them. If a node is MUX node, at most one child can exist in the possible world (at a particular instance of document).

ELM [1, 2] has better performance on classification kind of applications which is used to classify and analyse the nodes before searching over XML datasets. There are some dominant technologies like neural networks and SVM (support vector mechanism) which were facing issues like slow learning speed, poor computational scalability and so. An XML data tree can be seen as a group of nodes with one root node, connected nodes, and leaves nodes. A connected node has only a father node and more no. of children nodes. The keyword usually found in the leaves or in its father node of a leaf node.

II. RELATED WORK

Recently, keyword search domain has been studied extensively in traditional XML data. For searching a keyword in a XML dataset, most of previous work adopted LCA methods such as SLCA and ELCA. Lin and Feng [3] proposed XRANK with stack-based algorithm to generate SLCA. It has considered the problem of generating the query result efficiently. The method has considered both HTML and XML documents. Lin and Feng mentioned Challenges. First, XML queries do not always return back the entire documents, but can return elements that are deeply nested which contain the required keywords. Second, the nested structure of XML means that the representation of ranking is no longer at the granularity of a document, but at the granularity of an XML element. According to the authors, XRANK has considered the hierarchical and hyperlinked structure of XML documents,

and a two-dimensional notation of keyword proximity and it can be queried over the combined XML and HTML documents.

Yu Xu and Yannis [4] have proposed search method that outputs the smallest set of tree that has all the keywords. This paper introduced two algorithms; they are the Indexed-Lookup Eager algorithm when the keywords appear with different frequencies and the Scan Eager algorithm when the keywords have similar amount frequencies. Authors have presented an efficient keyword search algorithm, named Indexed Stack that returns nodes that contain all instances of all keywords in the query, after excluding the keyword instances that appear under nodes whose children already contain all keyword instances based on the query semantics. Also proposed the Indexed Stack algorithm to find the keyword queries based the ELCA semantics explained in XRANK method. The algorithm show the result nodes of the keyword queries which contains at least one set of all the keywords may be in their labels or labels of descending nodes.

Chong, Amit and Chee-Yong [5] designed Multi-way Search approach to compute SLCA for keyword. In this paper, a method has been proposed to help the previous keyword search method which was supporting only the AND semantics. The improved method is considering both AND & OR semantics to handle the keyword search. Authors analysed the problem of processing slca based query search and proposed a method called multiway-slca approach and extended their work to process more general keyword search queries that go beyond the traditional AND semantics to support any combination of AND and OR Boolean operators.

Wang, Chen and Lin [6] have focused on computing the query results efficiently by taking both SLCA and ELCA method together in to account. Algorithm makes use of set intersection operation of both SLCA and ELCA semantics together to generate results. Authors propose to assign each node a unique ID which is compatible with the document order. We also propose a new kind of inverted index, where for each keyword k_i the corresponding inverted list consists of all nodes that contain k_i in its subtree. They have also explained various optimization techniques to improve the performance.

Sara, Jonathan, Yaron and Yehoshua [7] have proposed advanced indexing techniques that facilitate efficient XSEarch implementation. XSEarch has a simple query language, suitable for a naive user. It returns semantically related document fragments that satisfy the user's query. Query answers are ranked using extended information-retrieval techniques and are generated in an order similar to the ranking. The main contribution of their work is in laying the foundations for a semantic search engine over XML documents. XSEarch returns semantically related fragments, ranked by estimated relevance. They have shown that it is possible to combine these qualities with an efficient, scalable and modular system. Thus, XSEarch can be seen as a general framework for semantic searching in XML documents.

Benny, Yuri and Sagiv [8] summarized and extended the probabilistic XML keyword search models that are previously proposed; the expressiveness and tractability of queries on different models are discussed by considering IND and MUX types of nodes in to account. Also addressed the problem of keyword search in probabilistic way of keyword search and computed SLCA node results by scanning the XML tree. Various known models of probabilistic XML can be represented as instantiations of abstract p-documents. Such documents have, in addition to ordinary nodes, distributional nodes that specify the probabilistic process of generating a random document. Within this abstraction, families of p-documents, which are natural extensions and combinations of previous models, are considered. The focus is on efficiency of applying twig queries (with projection) to p-documents. A closely related issue is the ability to (efficiently) translate a given document of one family into another family. Furthermore, both of these tasks have two variants that correspond to the value-based and object-based semantics.

Li, Feng, Wang and Zhou [9] have proposed Valuable Lowest Common Ancestor (VLCA) to accurately and effectively answer keyword queries over XML documents. Authors then proposed the concept of Compact VLCA (CVLCA) and compute the meaningful compact connected trees rooted as CVLCAs as the answers of keyword queries. To obtain more meaningful results of keyword queries, the notions of Valuable LCA and Compact VLCA too accurately and efficiently answer XML keyword queries. Based on the two concepts, they proposed the compact connected trees rooted CVLCAs as the answers of keyword queries. Moreover, we present an optimization technique for accelerating the computation of CVLCAs and devise an efficient stack-based algorithm to identify the meaningful compact connected trees.

Ziyang and Chen [10] have proposed MaxMatch, a novel semantics for identifying relevant matches. An efficient algorithm is designed for realizing this semantics. We take an axiomatic approach and have identified the properties that an XML keyword search algorithm should ideally possess in identifying relevant matches to keywords. Monotonicity states that data insertion (query keyword insertion) causes the number of query results to non-strictly monotonically increase (decrease).

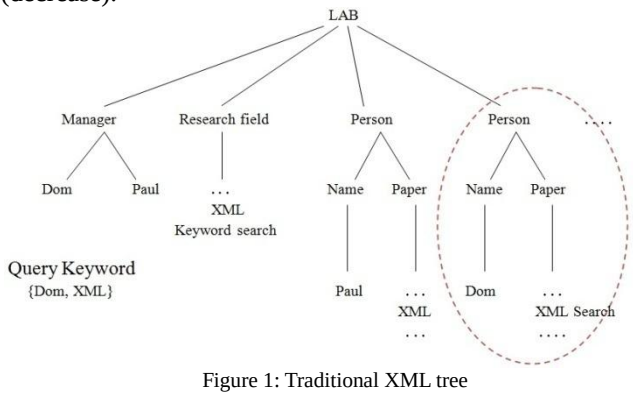


Figure 1: Traditional XML tree

III. PROBLEM DEFINITIONS

A. Searching the Keyword

An XML tree is usually modelled as labelled tree, in which elements are represented by the nodes and relation between the nodes are represented by edges. Given an XML document and keyword to be searched over that document, there are various works like LCA, SLCA, ELCA and so which can be taken as the resulting nodes. Most of the works has considered LCA and SLCA as the results of the matching nodes. SLCA is the most commonly used keyword searching method over the XML documents which is defined as follows;

***Definition 1*:** A node *u* is said to be a SLCA [11] node of a keyword search if (1) there is at least single match to each queried keyword in the part of tree rooted at *u*, and (2) there is no successor of *u* that satisfies condition (1).

For example, Consider a XML tree shown in the figure1 and a query Q = {Dom, XML}. The search result for this particular query is marked with the dotted oval in the figure 1. From the figures, nodes *lab* and *person* are LCA nodes and the node *person* is considered as the SLCA node. According to the definition of SLCA, the node *lab* has matching query in its descendant but the node *person* also has matching query in its descendant. Hence, SLCA result for the given query is taken to be the node *person.*

SLCA search method has been considered as the result for the keyword search on the XML documents. Since SLCA result set is the smallest set, every SLCA node should be considered as suitable result nodes for keyword search.

B. Probabilistic concept on XML data

A probability applied on XML document (is also called as p-document) can be visualized as a set of many deterministic XML documents. A probabilistic XML document is represented as a labeled tree which has two kinds of nodes, distributional nodes and Ordinary nodes. The Ordinary data nodes represent the actual information and distributional nodes represent probability distribution of the child nodes. Ordinary nodes are the main nodes that appear in both deterministic XML data and probabilistic XML data. Distributional nodes are used to generate the deterministic documents based on the probability, while these nodes do not occur in the deterministic data documents. IND nodes are shown with circle, for example, IND1, IND2 and IND3. MUX nodes are shown with rectangle with rounded corners, for example, MUX1. (Figure 2)

A p-document can generate all possible worlds (deterministic documents). During traversing, when a distributional node is occurred, there are two situations. First, if the node is IND type node which has m different children nodes then 2^m number of copies can be generated. For each copy, the probability is multiplication of probability of selecting particular child node and probability of not selecting other children nodes (i.e. 1- probability of each node). Second, if the node is MUX type node with m number of children nodes, m copies or m+1 copies can be generated. Probability of each copy can be probability of current selected child node and if no children nodes are selected then probability value will be probability of not selecting the children nodes (1- probability of selected nodes).

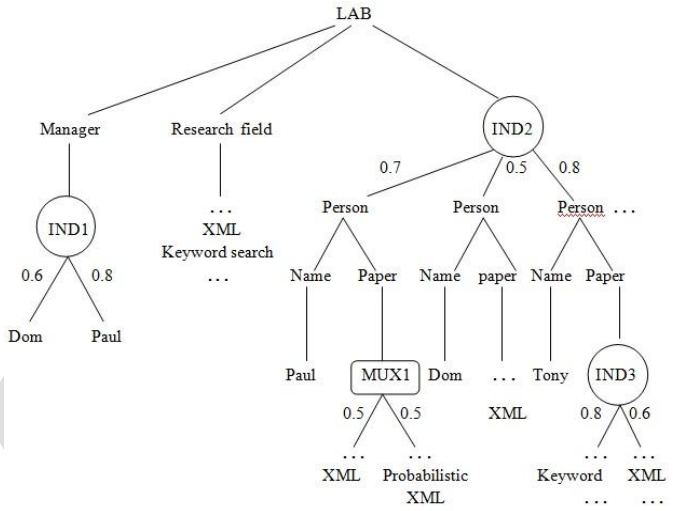


Figure 2: A probabilistic XML tree

Let us consider a probabilistic subtree and copies of p-documents are shown in the figure 3. If figure 3(1) selects B node as the child node of A node, then the probability will be 0.7 * (1-0.6) = 0.28. If node A does not select any of its children nodes then the probability of node A will be (1-0.7)*(1-0.6) = 0.12 (figure 3(4)). If the node B and node C are selected by node A, and node D selected node C as its child node (Figure3 (6)) then the probability is (0.7) * (0.6) * (0.5) = 0.21. Similarly the probabilities of selecting child nodes in different possible worlds (other copies) are shown in the figure 3.

A keyword search on probabilistic documents contains a p-document T, a keyword query Q = {k_1, k_2, ….., k_n}. The result for the query over T document will be an ordinary node from T which will be SLCA node generated by T. The probability value of a node v which has to be SLCA node in the possible worlds is represented as $Pr^T_{slca}(v)$ given by equation (1).

$$Pr^T_{slca}(v) = \sum_{i=1}^{m} \{Pr(w_i) \mid slca(v, w_i) = 1\} \qquad (1)$$

Where $Pr(w_i)$ is the probability value of selected node in all possible world. w_i = {w_1, w_2, …., w_n} which are generated by T. slca (v, w_i) = 1 represents that v is slca node that is present in the possible world w_i.

Definition 2: *SLCA on probabilistic XML data: Given a keyword query for search over probabilistic value based XML tree, given keyword query tries to find all the SLCA nodes v in all copies (possible worlds) with the probability of the*

probability values of all the possible worlds in which node v is considered to be SLCA node.

Definition 3: *Threshold SLCA on probabilistic XML data: Given a keyword query for searching over the probabilistic based XML tree, a keyword query searches for the possible SLCA nodes v in all the copies (possible worlds) with the probability value of all the probabilities of the possible copies in which node v is an SLCA node. And threshold probability should be less than probability value*.

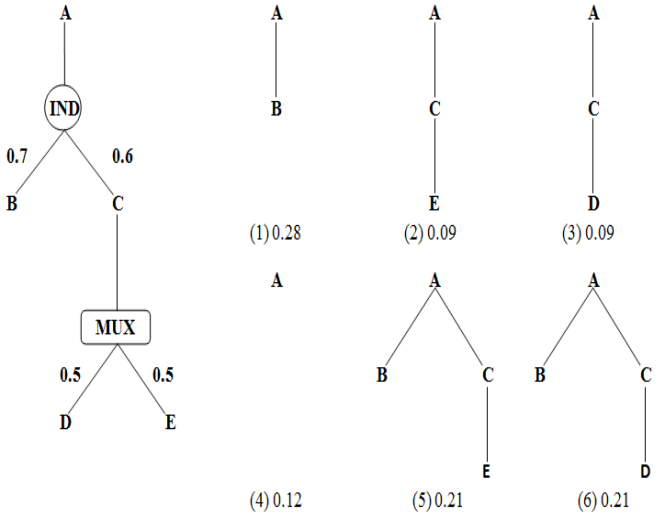


Figure 3: A probabilistic subtree and probability of all possible worlds (copies)

IV. NODE CLASSIFICATION

As mentioned earlier, a probability based document contains two kinds of nodes, ordinary and distributional nodes. The ordinary node shows actual data and the distributional node only represents the probability distributed along the nodes. Now, how the nodes are classified as ordinary or distributional nodes, is explained in the following section.

A. Classifying Ordinary nodes

When keyword node containing sub trees are extracted, the tree merging operation should be applied, this will result in SLCA nodes. Figures 4(1) & 4(2) are the keyword nodes subtree. Merging operation has been applied to generate tree as shown in Figure 4(3). So, the important part is to generate such kind of the subtree.

To generate such subtree, Dummy nodes are added randomly for actual tree nodes which have one or more keywords. If the subtree with root node V contains two keywords then add one dummy node as the sibling node of the node V. In figure 4(3), nodes {lab} and {person} have their dummy nodes. These dummy nodes are not part of actual tree. Adding random dummy nodes helps node classifications effectively.

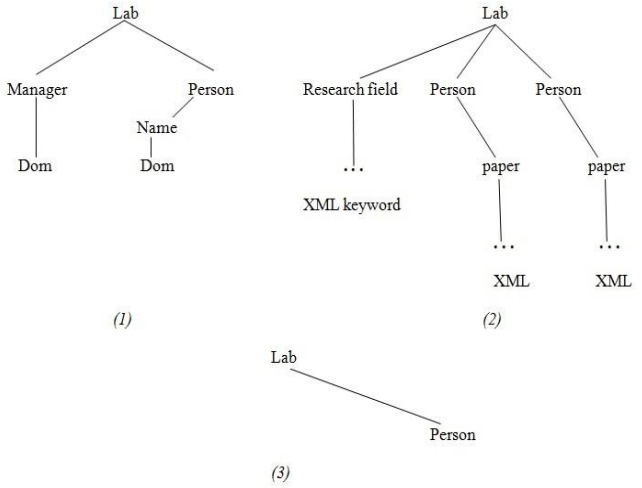


Figure 4: Sub trees for the search (1) Keyword {Dom}, (2) Keyword {XML} and (3) Common ancestor node for the keyword {Dom, XML}

B. Classifying Distributional nodes

As Discussed earlier, these nodes represent the probability values of children nodes. Based on types of distributional nodes, a p-document can have varying number of copies (possible world).

Case 1 (If node is an IND node): If a node *v* has *n* number of children nodes, then the no of possible copies are 2^n or if the node has one child node v_1 that contains keyword k_1 then Pv_1 is the probability of $v_1$that means the probability of tree with node v_1 for the keyword k_1 found is Pv_1. If number of children nodes containing the keyword k_1 is *n* then the probability will be calculated by considering all the possible situations or all the possible worlds which has the keyword. The sum of the probability will be the probability of subtree which contains the keyword k_1. (Figure 5)

Case 2 (If MUX node): The number of child nodes is n or n+1. Each will have its own probability value. Some of the world will contain the keyword, and such possible copies are important for our probabilistic keyword search which is shown (Figure 6).

Figure5&6 are shown as example for IND node and MUX node respectively. For the keyword {Dom} from figure5, Nodes {Dom} & {Paul} have IND1 as their father node.

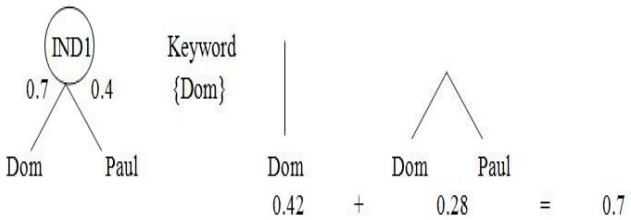


Figure 5: Example for case 1

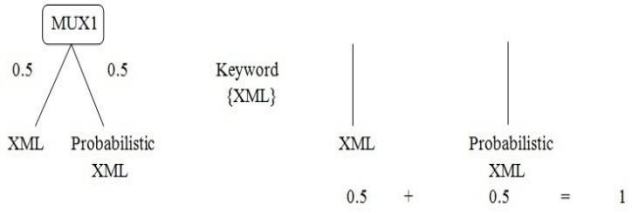


Figure 6: Example for case 2

The subtree with the keyword {Dom} has two possibilities. The probabilities for both situations are calculated as {0.7 * (1 - 0.4) = 0.42} and {0.7 * 0.4 = 0.28}. The probability of part of tree at node IND1 with the query keyword {Dom} is (0.42 + 0.28 = 0.7). For the query keyword {XML} (in figure6), nodes {XML} and {Probabilistic XML} have MUX1 as the father node. The subtree having {XML} brings has two entries; the probability of both nodes is 0.5 which means the probability at node MUX1 with the keyword query {XML} is {0.5 + 0.5 = 1}.

For each keyword, the tree is constructed by all the ancestors and the node itself along with distributional and ordinary nodes. The probability distribution of a particular keyword is presented by deleting all the distributional dummy nodes and connecting the node to its father node. After this merging all the keyword found subtree together to form a probabilistic tree. Then calculate the SLCA nodes over the probabilistic tree generated and separate the subtree at the SLCA node. Continue to calculate SLCA nodes for all the keywords.

V. PROBABILITY CALCULATION

A distributional node can appear in almost all the part of the tree except the root node and leaf nodes. From the figure7, it shows a Root node, node v as the SLCA node and keywords $\{k_1, k_2\}$. The path from Root node to node v is represented by p_1 and the path from node v to both keywords $\{k_1,k_2\}$ are represented by p_2 and p_3 respectively.

Case 3 (p1, p2 and p3 paths doesn't contain the distributional nodes) If v is an SLCA and none paths p1, p2 and p3 have distributional nodes then it is similar to the deterministic XML data. Hence the probability will be 1.

Case 4 (p1contain the distributional node) If p1 contains the distributional type node which implies node v will not come in all the worlds (copies). The distributional nodes in p1 will affect the probability of other children nodes.

Case 5 (p2 and p3has distributional type nodes) If any distributional nodes are present in the path from node V to any matched keyword then this node won't appear in any possible copies. In this condition, record of probability of all matched node has to be made.

Case 6 (p1, p2 and p3 has distributional type nodes) In this situation a decision has to be taken depending on which node has to be calculated first. SLCA nodes will be the children nodes in all the common ancestor nodes. So calculation is done from bottoms to up. From figure7, r2 and r3 paths have the priority. Hence first the situations r2 and r3 are to be calculated followed by the situation for r1 path as second step.

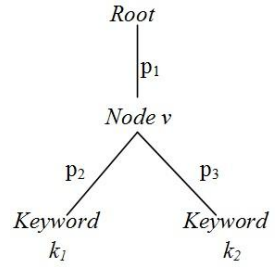


Figure 7: A subtree with SLCA node v and keywords $\{k_1, k_2\}$

VI. KEYWORD SEARCH OVER PROBABILISTIC XML TREE

Each node in an XML document has two parts of information such as code part and keyword part. Code part is used to analyse the relationship between other nodes which is helpful in finding the common ancestor for a particular document. Keyword is nothing but the actual data or main key for keyword search at each node which is usually assumed to be present at leaf nodes. If the node is distributional node then such nodes will be having an extra information i.e. probability value. For each query all the set of copies for each keyword will be generated and then perform merging of sets to get keyword nodes subtree.

Keyword search over the probabilistic XML documents mainly includes 4 steps as follows: (1) including dummy nodes randomly depending on the count of keywords, (2) grouping according to the queried keywords and kind of nodes, (3) employing the set merging function over the subtree to generate the common ancestor node probability tree, and (4) deleting and calculating the SLCA value.

First, adding dummy nodes. In the figure2, probability assigned XML tree contains the keywords {Dom, XML}. So as of first step, add the dummy nodes such as node {lab}, node {IND1} and node {person}. Second, grouping the nodes based on type of node. In the example query, one set is for keyword Dom and another set is for keyword XML. So every distributional type node has probability value implying the probability of subtree at the particular distributional node. The node Dom at node IND1 (figure2) has probability 0.6 which means the probability of having Dom as child node of IND1 is 0.6. Third remove the assumed nodes (distributional nodes) and reconnect them to father node. In the example, IND1 is deleted, the Dom is connected to its father node and the distributional probability is attached to the children nodes. Hence the probability 0.6 of the node IND1 is appended to the node Dom such that the probability of the node Dom will be 0.6. Lastly, repeat the procedure of calculating the SLCA value and deletion of subtree to get the final required subtree.

The algorithm 1 explains the procedure of searching the SLCA result nodes. The concept of probability will be applied to these search results. In order to calculate the SLCA node, an empty result array is initialised which will be storing the resulting nodes. Root value *u* is initialised to zero. Given a keyword for XML tree T, the descendant of keywords found will be stored in *x*. If node at *x* value is below the root value then the node is added to the result array. Once whole listed is generated then the process of random allocation of nodes is executed. Based on the type of node allotted, the probability will be calculated as explained in sections 4.1 and 4.2. As mentioned in the algorithm 1, node probability, probability value and threshold values are calculated. The results are analysed as higher the probability value higher is the user preference.

Input: A keyword query and set of XML documents.
Output: A list of files, keywords and their respective probabilities in descending orders.

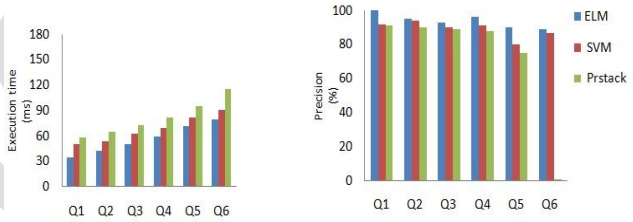
Begin // calculating SLCA
Initialise result = {} //resulting SLCA nodes
u = 0 // initial root value
for *each node v ∈ T*
Calculate x = descendant (lca (v, T))
if *(pre (u) <= pre(x)*
result = result ∪{u}
u = x;
return *result ∪ {u}*

continue...
// calculating the probability value
for *each result nodes, randomly assign probability values and randomly add MUX or IND node*
//Calculate Node probability (NP) based on the type of node.
If *(node = IND)*
*NP = probability of selected node * (1- probability of other nodes);*
Else *(node = MUX)*
NP= probability value of the selected node;
//probability of other nodes is Probability value (PV)
PV = 1 – NP;
//Threshold value (TV)
*TV = NP * PV;*
If *(NP <= TV)* then,
The node is not SLCA threshold node;
Else
The node is resulting SLCA threshold node;
End.

Algorithm 1: Calculating SLCA results and probability

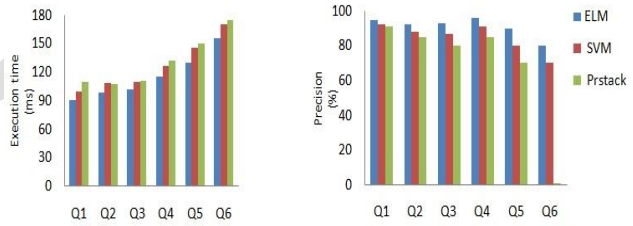
VII. PERFORMANCE EVALUATION

The analysis of the probability based keyword search method over different datasets has been done. The application is run in Microsoft visual studio environment running on windows 7. Various datasets such as Yahoo, UVW, and DBLP are downloaded from the UW repositories. Bigger size datasets have been splitted in to smaller size datasets in order to avoid overburden on the application. The probabilistic tree will be created by randomly appending the distributional nodes and independent nodes while traversing the tree. Then the original children nodes are reassigned to the randomly appended distributional nodes. As discussed earlier, for each MUX node the sum of probability of children nodes should be less than or equal to 1 and the no of keywords are restricted to 2 or 3.
Comparison of the execution time and precision value by the algorithms used in ELM with the Prstack and SVM with respect to yahoo and DBLP dataset is shown in the figure 7 and figure 8 respectively.



(A) (B)

Figure 7: (A) Execution time for Yahoo dataset. (B) Precision for yahoo dataset



(A) (B)

Figure 8: (A) Execution time for DBLP dataset. (B) Precision for DBLP dataset

As part of testing, Yahoo dataset has been used for querying through various types of keywords. The list of keywords used for querying the yahoo dataset along with their query notations has been recorded in the table 1 below. Six different keywords have been queried and their execution time has been recorded. Figure7 is showing the histogram of execution time taken by different methods. It shows that the execution time taken by ELM based search method is less as compared with other methods. Figure 8 is showing the histogram of precision comparison of ELM based method along with other methods. It can be concluded that the ELM based searching method has taken comparatively less time and provided a better precision for different keyword search.

The yahoo dataset is of the size 34KB. It has the total of 342 elements and the maximum depth of Yahoo tree is 5. Similarly,

the DBLP dataset is of the size 127MB. It has the total of 3332130 elements and the maximum depth of DBLP tree is 6.

Table (A) and Table (B) showing query notation and their respective keywords for yahoo and DBLP datasets respectively

(A)

Query Notations	Keywords from *YAHOO* dataset
QY1	Pentium CPU
QY2	Dell Dimensions
QY3	Standard Protection
QY4	Keyboard and mouse
QY5	Compaq Presario
QY6	Manufacturer

(B)

Query Notations	Keywords from *DBLP* dataset
QD1	Artificial Intelligence
QD2	Computer Science
QD3	Embedded Systems
QD4	Multi-Dimensional
QD5	Mathematical models
QD6	Neuro-computing

VIII. CONCLUSION

In this paper, the analysis of XML keyword searching method based on probabilistic concept over XML documents in both deterministic and non-deterministic data is done. The probabilistic XML model known as $PrXML^{\{IND,\ MUX\}}$ (INDividual nodes and Mutual eXclusive nodes)has been analyzed by using the concept of probability at random nodes in XML documents for given keywords queries. It is also seen that SLCA method of searching keywords gives the smallest set of result nodes. Hence, SLCA method of searching is considered to be more appropriate than other searching techniques such as ELCA, XSEarch, CVLCA and Max Match. SVM and Prstack methods compute node probabilities of all ancestor nodes and then record the results based on the nodes which contain keyword. Whereas ELM first classifies the nodes based on the keyword and then applies the probability calculation algorithm which will boost the execution time of ELM based method compared to SVM and Prstack. So, ELM based procedure with node classification has taken less execution time and more precision. Overall, Keyword searching topic over the probabilistic data (both deterministic and non deterministic data) has improved in returning better results.

REFERENCES

[1]. J. W. Cao, T. Chen, and J. Fan, “Fast online learning algorithm for landmark recognition based on BoW framework,” in Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications, Hangzhou , China, June 2014

[2]. J. W. Cao, Z. Lin, G.B. Huang, and N. Liu, “Voting based extreme learning machine,” Information Sciences, vol. 185, pp. 66–77, 2012

[3]. L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram, “XRANK: ranked keyword search over XML documents,” inProceedings of the ACMSIGMOD International Conference onManagement of Data (SIGMOD ’03), pp. 16–27, 2003

[4]. Y. Xu and Y. Papakonstantinou, “Efficient keyword search for smallest LCAs in XML databases,” in Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD ’05), pp. 527–538, June 2005

[5]. C. Sun, C.-Y. Chan, and A. K. Goenka, “Multiway SLCAbased keyword search in XML data,” in Proceedings of the 16th International World Wide Web Conference (WWW ’07), pp. 1043–1052, Alberta, Canada, May 2007

[6]. J. Zhou, Z. Bao, W. Wang et al., “Fast SLCA and ELCA computation for XML keyword queries based on set intersection,” in Proceedings of the 28th International Conference on Data Engineering (ICDE ’12), pp. 905–916, IEEE, Washington, DC, USA, April 2012

[7]. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: XSEarch: a semantic search engine for XML In: VLDB, pp. 45–56 (2003)

[8]. B. Kimelfeld, Y. Kosharovsky, and Y. Sagiv, “Query efficiency in probabilistic XMLmodels,” in Proceedings of the ACMSIGMOD International Conference on Management of Data (SIGMOD ’08), pp. 701–714, June 2008

[9]. Li, G., Feng, J., Wang, J., Zhou, L.: Effective keyword search for valuable LCAs over XML documents. In: CIKM, pp. 31–40 (2007)

[10]. Liu, Z., Chen, Y.: Reasoning and identifying relevant matches for XML keyword search. PVLDB 1(1), 921–932 (2008)

[11]. Liu, Ziyang, and Yi Chen. "Processing keyword search on XML: a survey." World Wide Web 14.5-6 (2011): 671-707

[12]. Dayananda, P., and Rajashree Shettar. "Survey on Information Retrieval in Semi Structured Data." International Journal of Computer Applications 32.8 (2011): 1-5

[13]. Zhao, Yue, Ye Yuan, and Guoren Wang. "Keyword Search over Probabilistic XML Documents Based on Node Classification." *Mathematical Problems in Engineering* 2015 (2015).