Keyword Search on Probabilistic XML Data Based on ELM

Yue Zhao*, Guoren Wang, and Ye Yuan

College of Information Science and Engineering, Northeastern University, Liaoning, Shenyang 110819
zhaoy0927@163.com

Abstract. This paper describe a keyword search measure on probabilistic XML data based on ELM (Extreme Learning Machine). We use this method to carry out keyword search on probabilistic XML data. A probabilistic XML document differs from a traditional XML document to realize keyword search in the consideration of possible world semantics. A probabilistic XML data can be seen as a set of nodes consisting of ordinary nodes and distributional nodes. ELM has good performance in text classification applications. As the typical semi-structured data, the label of XML data possesses the function of definition self. Label and keyword which has been contained in the node can be seen as the text data of the node. ELM offers significant advantages such as fast learning speed, ease of implementation and classification nodes effectively. Keyword search on the set after it classified by using ELM can pick up the speed of query. This paper uses ELM to classify nodes and carry keyword search on the set which has been classified. The experiments can show that the speed of query can receive significant improvement.

Keywords: Extreme learning machine, Node classification, Probabilistic XML data.

1 Introduction

Traditional databases only manage deterministic information, but many applications that use databases to involve uncertain data such as information extraction, information integration, web data mining, etc. Because of the flexibility of XML data model, it can easily allow a natural representation of uncertain data. Now, many probabilistic XML models are designed and analyzed[1-4]. This paper select a popular probabilistic XML model $PrXML^{\{ind,mux\}}$[5], which is discussed in [6]. In this model, a probabilistic XML document (called a p-document) is considered as a labeled tree which has two types of nodes, *ordinary* nodes and

* Yue Zhao, Ye Yuan and Guoren Wang were supported by the NSFC (Grant No.61025007, 61328202 and 61100024), National Basic Research Program of China (973, Grant No.2011CB302200-G), National High Technology Research and Development 863 Program of China (Grant No.2012AA011004), and the Fundamental Research Funds for the Central Universities (Grant No. N130504006).

J. Cao et al. (eds.), *Proceedings of ELM-2014 Volume 2,*
Proceedings in Adaptation, Learning and Optimization 4, DOI: 10.1007/978-3-319-14066-7_14

distributional nodes. Ordinary node is used to represent the actual data and distributional node is used to represent the probability distribution on the child nodes. There are two types of nodes in distributional nodes, IND and MUX. If a node is an IND node, its children nodes are *independent* of each other, while the children of a MUX node are $mutually-exclusive$. A real number from (0,1] is attached on each edge in the XML tree, indicating the conditional probability that the child node will appear under the parent node given the existence of the parent node. Keyword search has been widely applied on XML data. Users don't need know the knowledge of the underlying data structures and complex query language beforehand. So, keyword search is an easy method for ordinary users. In the past years, the definition of common ancestor node has several choices, such as LCA (Lowest Common Ancestor), SLCA (Smallest LCA) and so on. This paper select SLCA as the root node of result subtree.

ELM[7-10] has good performance on classification applications, and can be used to classify nodes before query XML data. Classification is considered as an important cognitive computation task[11-14]. A probabilistic XML data tree can be seen as a set of all the nodes including root node (only one), connected nodes, leaves nodes and distributional nodes. So, the classification need to consider two kinds of information, and they are keyword information and probability distributional information.

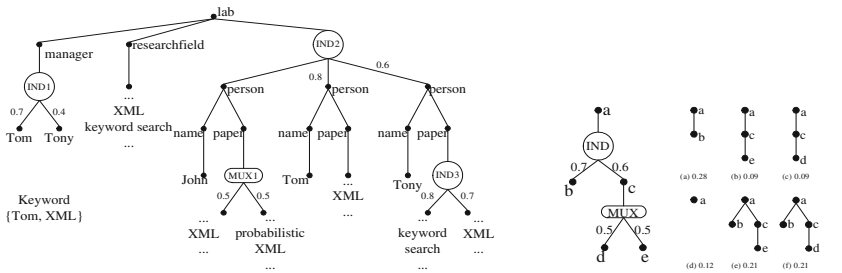
This paper is organized as follows: Section 2 introduces the probabilistic XML model and the formal semantics of keyword search result on probabilistic XML data. Section 3 shows that how to classify nodes. In section 4, we propose an algorithm to query keyword on probabilistic XML data by using ELM to classify nodes. The experimental and performance evaluation are presented in section 5. Section 6 gives the conclusion and future works.

2 Problem Definitions

2.1 Probabilistic XML Data

A probabilistic XML document (p-document) can be seen as a set of many deterministic XML documents. Each deterministic document is called a possible world. Ordinary nodes are prime XML nodes and they are appearing on both deterministic XML data and probabilistic XML data. Distributional nodes are only used to define the probabilistic process of generating deterministic documents, while those nodes do not occur on deterministic XML data. This paper adopts $PrXML^{\{ind,mux\}}$ as the probabilistic XML model. For example, figure 1(a) shows a p-document T.

Given a p-document T, we can traverse T in a top-down fashion. When we visit a distributional node, there are two situations according to the different typies. One situation is that if a node is an IND node with m children nodes, we generate 2^m copies. Another situation is that if a node is a MUX node with m children nodes, we generate m or $m+1$ copies. For example, figure 1(b) shows the copies of a p-document with their probabilities. Figure 1(b) select node b as the only child node of node a, and the probability is $0.7 * (1 - 0.6) = 0.28$.



(a) A probabilistic XML document (b) A probabilistic subtree

Fig. 1. Probabilistic XML data

If there is not any node selected as the children nodes of a, the probability of this copy is $(1 - 0.7) * (1 - 0.6) = 0.12$. Node a selects nodes b and c as its children nodes, and node c selects node d as its child node. The probability is $0.7 * 0.6 * 0.5 = 0.21$. The probabilities of the other copies (possible worlds) are easy to calculate from the above procedure.

2.2 Keyword Query

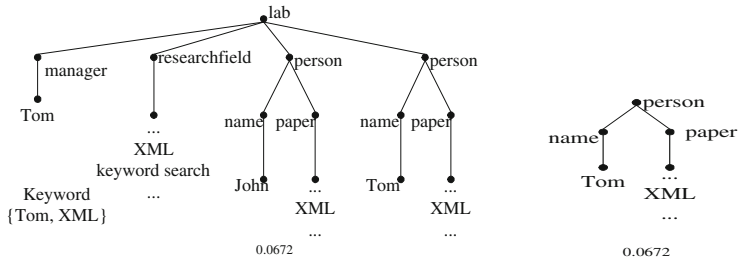
Usually, we model an XML tree as a labeled ordered tree, where nodes represent elements, and edges represent direct nesting relationship between nodes. Recently, keyword search has been studied in XML documents more and more. Given a set of keywords and a XML document, most work took LCA and SLCA of the matched nodes as the results. The function $lca(v_1, v_2, ..., v_k)$ computes the Lowest Common Ancestor of nodes $v_1, v_2, ..., v_k$. Given k keywords and the inverted lists $\{S_1, S_2, ..., S_k\}$ of them. The LCA of these keywords on T is defined as:

$$\begin{aligned} lca(v_1, v_2, ..., v_k) &= lca(S_1, S_2, ..., S_k) \\ &= \{lca(n_1, n_2, ..., n_k) \mid n_1 \in S_1, ..., n_k \in S_k\} \end{aligned} \tag{1}$$

$child(v, n_i)$ denote the children nodes of node v on the path from v to n_i. The SLCA is defined as follows:

$$\begin{aligned} &slca(\{v_1\}, S_2, ..., S_k) = \\ &\quad \{v \mid v \in lca(\{v_1\}, S_2, ..., S_k), \forall v' \in lca(\{v_1\}, S_2, ..., S_k)(v \not<_a v')\} \end{aligned} \tag{2}$$

For example, figure 2(a) give a traditional XML tree which is generated by figure 1(a) and a query $Q = \{Tom, XML\}$. The result of query is shown in figure 2(b). This paper selects SLCA as the result for the keyword search on probabilistic XML data. Because SLCA is the smallest set, every SLCA node should be seen as the suitable for the users.



(a) A traditional XML document (b) SLCA nodes

Fig. 2. SLCA nodes on XML data

A keyword search on p-document consists of a p-document T, a query $Q = \{k_1, k_2, ..., k_n\}$. We define the answers for a keyword search on T as ordinary nodes on T to be SLCAs in the possible worlds generated by T. The probability of a node v being an SLCA in the possible worlds is denoted as $Pr^T_{slca}(v)$. The formal definition is shown as follows:

$$Pr^T_{slca}(v) = \sum_{i=1}^{m} \{Pr(w_i) \mid slca(v, w_i) = true\} \tag{3}$$

where $Pr(w_i)$ is the existence probability of the possible world w_i. $\{w_1, w_2, ..., w_n\}$ denotes the possible worlds generated by T. $slca(v, w_i) = true$ indicates that v is an SLCA in the possible world w_i.

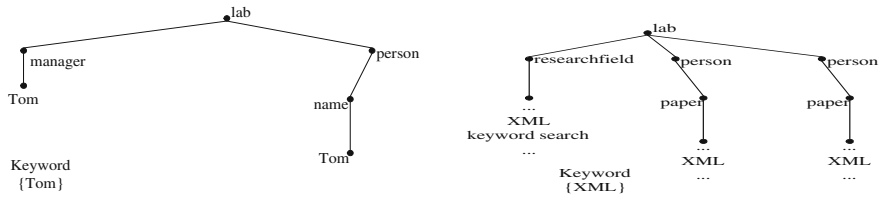
Definition 1: (SLCA on probabilistic XML data) *Given a query Q in a probabilistic XML tree T, an SLCA query finds the SLCA nodes v in all possible worlds with the probability of all the probabilities of the possible worlds in which the node v is an SLCA node.*

3 Classification of Nodes

3.1 Classification of Ordinary Nodes

From section 2, we can see that if we can find keyword nodes tree, the set intersection operation for keyword nodes tree should achieve SLCA nodes quickly. Figure 3(a) and 3(b) shows the keyword nodes tree. When we use set intersection operation to obtain the common ancestor nodes tree such as shown in figure 4(c). So, the important section is how to receive the keyword nodes tree.

To receive the keyword nodes tree, we need add dummy node for actual node which contains more than one keyword. If the subtree rooted at the node v contains two keywords, we should add one dummy node as the sibling node of node v. For example, node $\{lab\}$ and $\{person\}$ in figure 4 has its dummy node. These dummy nodes don't exist in the actual tree. The aim of adding dummy nodes is to classify nodes effectively.



(a) Keyword $\{Tom\}$ nodes tree (b) Keyword $\{XML\}$ nodes tree

Fig. 3. Keyword nodes tree

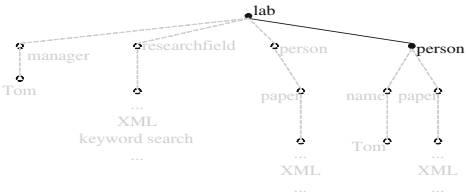
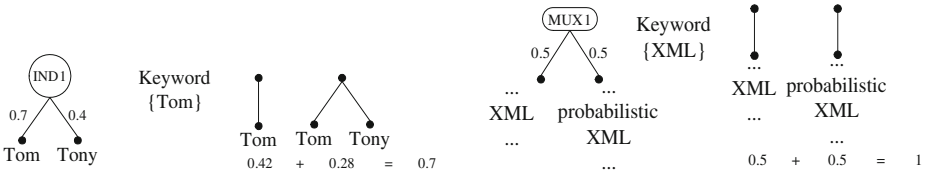


Fig. 4. A common nodes tree

3.2 Classification of Distributional Nodes

Distributional nodes can represent the probability distribution of the children nodes. A p-document defines a probability distribution over a space of deterministic XML documents. According to the different types of the distributional node, the number of copies is different. If a node is an IND node, and it has n children nodes, the number of copies is 2^n. Otherwise, if a node is a MUX node, the number is n or $n+1$. Each copy has its probability value. Some of the copies will contain the keyword, and the copies which contain the keyword are important for our probabilistic keyword search.

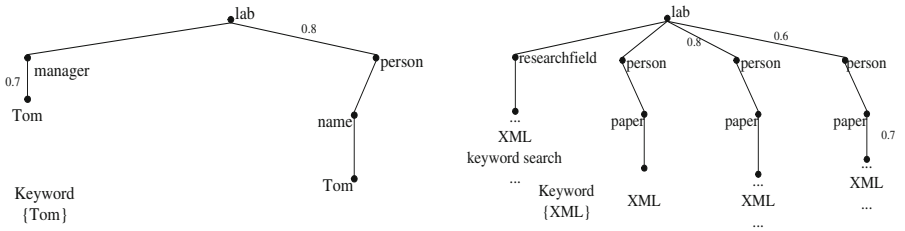


(a) An IND node (b) A MUX node

Fig. 5. Distributional nodes

Figure 5(a) shows an example of an IND node. For the keyword $\{Tom\}$, IND1 is a parent node of nodes $\{Tom\}$ and $\{Tony\}$. The copy with the existing of

$\{Tom\}$ has two situations, and their probabilities are $0.7 \times (1 - 0.4) = 0.42$ and $0.7 \times 0.4 = 0.28$. It means that the probability of the subtree rooted at node IND1 contains the keyword $\{Tom\}$ is $0.42 + 0.28 = 0.7$. Figure 5(b) shows an example of a MUX node. For the keyword $\{XML\}$, MUX1 is a parent node of node $\{XML\}$ and $\{ProbabilisticXML\}$. The copy with the existing of $\{XML\}$ has two situations, the probabilities are all 0.5. It means that the probability of the subtree rooted at node MUX1 contains the keyword $\{XML\}$ is $0.5+0.5 = 1$.



(a) Keyword $\{Tom\}$ probabilistic tree (b) Keyword $\{XML\}$ probabilistic tree

Fig. 6. Keyword nodes probabilistic tree

For each keyword, all its ancestor nodes and itself nodes will constitute a tree. This tree contains ordinary nodes and distributional nodes. To present the probability contribution situation of this tree which contains keyword, we will delete distributional nodes and connect its children nodes to its parent node with the existence probability of containing the keyword of the subtree rooted at its parent node according to the type of distributional node. For example, figure 6(a) shows a tree contained keyword $\{Tom\}$. Node $\{manager\}$ is a parent node of node $\{IND1\}$. The probability of a subtree rooted at node $\{IND1\}$ which contains keyword $\{Tom\}$ is 0.7. As shown in figure 6(b), it is the situation of the probabilistic tree which contains keyword $\{XML\}$.

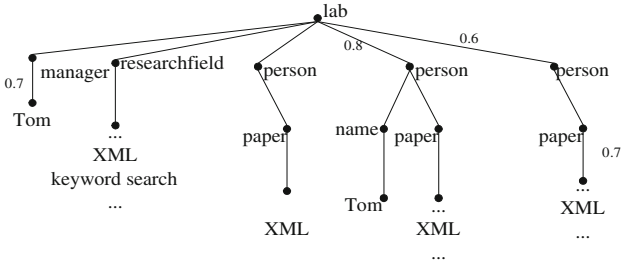


Fig. 7. A keyword nodes probabilistic tree

We merge all the keywords probabilistic trees together. A keyword nodes probabilistic tree can be generated. We need calculate SLCA nodes on this tree with the probability and delete the subtree rooted at SLCA nodes. Next, we need continue to calculate SLCA results on remaining nodes tree. So, if we repeat such operation, all the SLCA results will generate. For example, figure 7 is a keyword nodes probabilistic tree. This tree retains all the probabilities of the subtree which contains keyword. If we calculate SLCA results on this tree, the node $\{person\}$ is the only result. So, we need delete the subtree which is rooted at node $\{person\}$, and the remaining other nodes tree. To repeat calculated operation on the remaining nodes tree, we can see that the node $\{lab\}$ is another result. The calculation of the probabilities of all the SLCA nodes can be shown in next section.

4 Keyword Search on Probabilistic XML Data Based on ELM

Keyword search on probabilistic XML data based on classification mainly include four steps, they are shown as following: 1) Adding dummy nodes according to the number of keywords. 2) To classify nodes with ELM based on keyword according to the type of the nodes. 3) Use the set merging operation to structure the common ancestor nodes probabilistic tree. 4) Repeat the operation of calculating SLCA and deleting the subtree, all the SLCA results will generate. The key of the keyword search on p-document is how to calculate the probability of the SLCA results. Step 2 and step 4 all contain the computation of the probabilities.

Each node contains two kinds of information, they are code and keyword it contained. If a node is a distributional node, there are the third information in this node, that is probability. Code is used to judge the relationships between nodes, such as finding the common ancestor nodes. The keyword which is contained in a node is the key of keyword search. When we use ELM to classify nodes, keyword can be used as the label of the classification set. Every set represent one keyword. For given the query, we will find all the sets of keywords which is given by users, and operate set merging to obtain a keyword nodes tree.

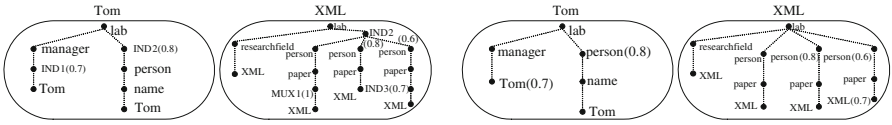
Next, we introduce four steps of the keyword search algorithm used ELM to classify nodes on probabilistic XML data one by one.

First, adding dummy nodes according to the number of keywords. We can see that the probabilistic XML tree in figure 1(a) contains two keywords $\{Tom, XML\}$. The algorithm use Dewey code to encode the XML tree. So, the first step is adding the dummy nodes for the node v which contains keywords in the subtree rooted at the node v. If the subtree which is rooted at the node v has n keywords, we will add $n-1$ dummy nodes.

Second, to classify nodes with ELM based on keyword according to the type of the nodes. From the dummy nodes tree, all the nodes and the distributional nodes are consisted of the classified nodes. ELM can classify the nodes to two sets such as shown in figure 8(a). The first set represent keyword Tom, and the second set represent keyword XML. Each distributional node has a probability

which represents the keyword probability of the subtree which is rooted at the distributional node. For example, the node $IND1$ has the probability with 0.7, that means the probability of containing keyword $\{Tom\}$ of the subtree which is rooted at $IND1$ is 0.7.

Then, we need delete all the distributional nodes and connect all their children nodes to their parent node. The probability of distributional node will be moved to its child node. For example, in figure 8(b), the node $\{Tom\}$ accept the probability 0.7 from its parent node $\{IND1\}$.



(a) A keyword p-tree (b) P-tree without distributional nodes

Fig. 8. Probabilistic tree

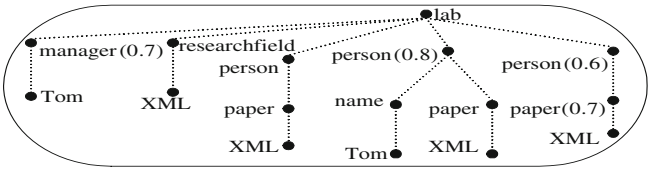


Fig. 9. A common nodes p-tree

Third, use the set merging operation to structure the common ancestor nodes probabilistic tree. The intersection of the two sets is the set which includes node $\{lab\}$ and $\{person\}$. Figure 9 shows the union set of two keyword sets.

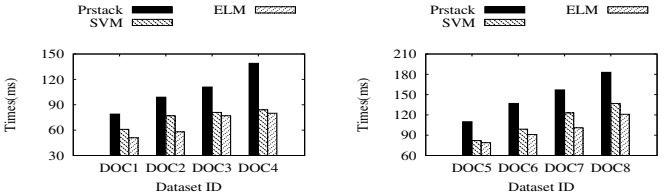
Finally, repeat the operation of calculating SLCA and deleting the subtree, all the SLCA results will generate. Let's calculate SLCA result on the tree which is shown in figure 9. The node $\{person\}$ with the SLCA probability of 0.8 is generated. So, the subtree which is rooted at $\{person\}$ will be deleted, and the probability $1 - 0.8 = 0.2$ will be leaved to the other nodes tree. Next, the node $\{lab\}$ is another result. The probability of this result is $0.2 \times 0.7 = 0.14$. Because, the extensive probability of the node Tom is 0.7, and the extensive probability of node $\{XML\}$ is 1, the SLCA probability of $\{lab\}$ is 0.14.

5 Performance Verification

The dataset we used is shown in table 1. In this paper, the algorithm select two datasets XMARK and DBLP. For each XML dataset used, we generate the

Table 1. properties of probabilistic XML data

ID	name	size	Ordinary	IND	MUX
DOC 1	XMARK 1	10M	159,307	14,630	15,471
DOC 2	XMARK 2	20M	364,199	41,251	38,100
DOC 3	XMARK 3	40M	689,470	77,228	61,535
DOC 4	XMARK 4	80M	1,497,433	161,277	159,495
DOC 5	DBLP 1	20M	361,370	68,345	70,790
DOC 6	DBLP 2	40M	731,561	238,450	227,540
DOC 7	DBLP 3	80M	1,477,345	440,007	405,857
DOC 8	DBLP 4	160M	3,260,109	788,367	771,320



(a) The time of DOC1 to DOC4 (b) The time of DOC5 to DOC8

Fig. 10. Vary query over DOC1 to DOC8

corresponding probabilistic XML tree, using the same method as used in [5]. We visit the nodes in the original XML tree in pre-order way. For each node v visited, we randomly generate some distributional nodes as children of v. For the original children of v, we select them as the children of the new generated distributional nodes and assign them random probability distributions. We need restrict that the sum of children nodes for a MUX node is no more than 1. The keyword has 8 situations. The number of keywords is 2 to 3.

We compare the query times of three situations about keyword search in probabilistic XML data. The first situation is using the method in [15] to retrieval the SLCA nodes, and the second situation is using SVM to classify nodes for the keyword search on p-document. The third situation classify nodes by using ELM. The speed of classification is shown in figure 10.

From the figure 10, we can see that ELM has advantages of speed compared with Prstack and SVM. Prstack will compute all the nodes probabilities of all the ancestor nodes of the keyword node and it will record all the situation of the node which contains the keyword. ELM can classify nodes according to the code and keywords by retrieve all the nodes once. So, the algorithm has high speed by using ELM to classify.

6 Conclusions

This paper use ELM to classify for keyword search on probabilistic XML data. Keyword search on probabilistic XML data has been received much attention in the literature. Finding efficient query processing method for keyword search on

probabilistic XML data is an important topic in this area. In this paper, SLCA is selected as the results. Classification for nodes is important among all the operations. ELM can increase retrieval speed for the classification. So, ELM can support keyword search on probabilistic XML data.

References

1. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: Xrank: Ranked keyword search over xml documents. In: SIGMOD (2003)
2. Xu, Y., Papakonstantinou, Y.: Efficient keyword search for smallest lcas in xml databases. In: SIGMOD (2005)
3. Sun, C., Chan, C.Y., Goenka, A.K.: Multiway slca-based keyword search in xml data. In: WWW (2007)
4. Zhou, J., Bao, Z., Wang, W., Ling, T.W., Chen, Z., Lin, X., Guo, J.: Fast SLCA and ELCA Computation for XML Keyword Queries based on Set Intersection. In: ICDE (2012)
5. Kimelfeld, B., Kosharovsky, Y., Sagiv, Y.: Query efficiency in probabilistic xml models. In: SIGMOD (2008)
6. Nierman, A., Jagadish, H.V.: ProTDB: Probabilistic data in xml. In: VLDB (2002)
7. Huang, G.-B.: Learning capability and storage capacity of two-hidden-layer feedforward networks. IEEE Transactions on Neural Networks (2003)
8. Huang, G.-B., Siew, C.-K.: Extreme learning machine with randomly assigned RBF kernels. International Journal of Information Technology (2005)
9. Huang, G.-B., Chen, L.: Enhanced random search based incremental extreme learning machine. Neurocomputing (2008)
10. Huang, G.-B., Chen, L.: Convex incremental extreme learning machine. Neurocomputing (2007)
11. Taylor, J.G.: Cognitive computation. Cognitive Computation (2009)
12. Wöllmer, M., Eyben, F., Graves, A., Schuller, B., Rigoll, G.: Bidirectional lstm networks for context-sensitive keyword detection in a cognitive virtual agent framework. Cognitive Computation (2010)
13. Mital, P.K., Smith, T.J., Hill, R.L., Henderson, J.M.: Clustering of gaze during dynamic scene viewing is predicted by motion. Cognitive Computation (2011)
14. Cambria, E., Hussain, A.: Sentic computing: Techniques, tools, and applications (2012)
15. Li, J., Liu, C., Zhou, R., Wang, W.: Top-k Keyword Search over Probabilistic XML Data. In: CICDE (2011)