Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 210961, 11 pages
http://dx.doi.org/10.1155/2015/210961

Research Article

Keyword Search over Probabilistic XML Documents Based on Node Classification

Yue Zhao, Ye Yuan, and Guoren Wang

College of Information Science and Engineering, Northeastern University, Liaoning, Shenyang 110819, China

Correspondence should be addressed to Yue Zhao; zhaoy0927@163.com

Received 22 August 2014; Revised 31 October 2014; Accepted 31 October 2014

Academic Editor: Amaury Lendasse

This paper describes 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 document 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 semistructured data; the label of XML data possesses the function of definition itself. Label and context of the node can be seen as the text data of this node. ELM offers significant advantages such as fast learning speed, ease of implementation, and effective node classification. Set intersection can compute SLCA quickly in the node sets which is classified by using ELM. In this paper, we adopt ELM to classify nodes and compute probability. We propose two algorithms that are based on ELM and probability threshold to improve the overall performance. The experimental results verify the benefits of our methods according to various evaluation metrics.

1. Introduction

Traditional databases only manage deterministic information, but many applications use databases to involve uncertain data such as information extraction, information integration, and web data mining. 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 selects 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 *distributional* nodes. Ordinary node is used to represent the actual data and distributional node is used to represent the probability distribution of the child nodes. There are two types of 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 *mutuallyexclusive*; that means, at most, one child can exist in a random instance document (a *possible world*). A real number from $(0, 1]$ is attached on each edge in an XML tree, indicating the conditional probability that the child node will appear under the parent node given the existence of its father node. From the attribute of a MUX node, we can see that the sum of all the existence probabilities of children nodes is 1 or less than 1.

Keyword search has been widely applied on XML data. It is considered to be an effective information discovery method to query XML data. Users do not need know the knowledge of the underlying data structures and complex query language beforehand. So, keyword search is an easy method for ordinary users to retrieve information. Keyword search on XML data is different from the query on text data. As a result, a subtree rooted at a common ancestor node will replace the whole text data. In the past years, the definition of a common ancestor node has several choices, such as LCA (lowest common ancestor), SLCA (smallest LCA), and ELCA (exclusive LCA). These definitions are used to determine the users' query intentions. SLCA and ELCA are the subset of LCA by adding some restrictive factor. In many cases, the size of a set determines the accuracy of the query. This paper selects SLCA as the root node of result subtree because that SLCA nodes set is the smallest set in all the definitions based on LCA.

It is known that both neural networks and SVM (*support vector machines*) have been playing the dominant roles out of numerous computational intelligence techniques.

But they face three challenging issues such as (1) slow learning speed, (2) trivial human intervene, and (3) poor computational scalability. ELM [7, 8] as emergent technology works for generalized single-hidden layer feedforward networks (SLFNs). ELM [9–12] 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 [13–16]. An XML data tree can be seen as a set of all the nodes including root node (only one), connected nodes, and leaves nodes. A connected node has only one father node and one or more children nodes. The keyword usually appears in the leaves nodes or its father node of a leaf node. So, the classification needs to consider two kinds of information, and they are keyword information and structural information. XML contains some structural information such as the element-subelement relationships and the element-value relationships. The element-subelement includes ancestor-descendant relationship, and father-child relationship. In addition, sibling relationship is an important relationship. If the number of keywords is more than one, the relationship between nodes plays a crucial role in the keyword search on XML data. So, the classification needs to think out structural information and keyword information on an XML data tree. The classification method is presented in Section 3.

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 how to classify nodes and the calculation method of probability. In Section 4, we propose an algorithm to query keyword on probabilistic XML data by using ELM to classify nodes. Section 5 introduces the method which descript the impact of the probability threshold. The experimental and performance evaluations are presented in Section 6. Sections 7 and 8 give the related work and the conclusion.

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. A probabilistic XML document represented as a labeled tree has *ordinary* nodes representing actual data and *distributional* nodes representing the probability distribution of the child nodes. Ordinary nodes are prime XML nodes and they always appeared on 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 $\text{PrXML}^{\{\text{ind,mux}\}}$ as the probabilistic XML model. For example, Figure 1 shows a p-document T. Ordinary nodes are shown as a black solid point, for example, {lab}, {manager}, {Tom} and {Tony}. IND nodes are depicted as rectangular boxes with rounded corners, for example, IND1, IND2, and IND3. MUX nodes are displayed as circles, for example, MUX1.

A p-document can generate all possible worlds (deterministic documents). 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 types. One situation is that if a node is an IND node with m children nodes, we generate 2^m copies. We randomly select children nodes of the IND node into a copy. A copy is a subset of all children nodes. For each copy, the probability is the product of all existence probabilities of the children nodes in the subset and the absence probabilities (1 − existence probability) of the children nodes not in the subset. Another situation is that if a node is a MUX node with m children nodes, we generate m or $m + 1$ copies. If the sum of all the existence probabilities of the children nodes is 1, there are m copies, otherwise the number of copies is $m + 1$. For each copy, the probability is the existence probability of the selected children node. If none of the children nodes has been selected, the probability is the absence probability. For example, Figure 2 shows the copies of a p-document with their probabilities. Figure 2(a) select node b as the only child node of node a, and the probability is $0.7 * (1 - 0.6) = 0.28$. If there is not any node selected as the children nodes of a, Figure 2(d) shows the probability of this copy is $(1 - 0.7) * (1 - 0.6) = 0.12$. As shown in Figure 2(e), 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, in which 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 an XML document, most work took LCA and SLCA of the matched nodes as the results. The function $\text{lca}(v_1, v_2, \ldots, v_k)$ computes the Lowest Common Ancestor of nodes $v_1, v_2, \ldots, v_k$. Given k keywords and the inverted lists $\{S_1, S_2, \ldots, S_k\}$ of them. The LCA of these keywords on T is defined as

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

$\text{child}(v, n_i)$ denote the child nodes of node v on the path from v to n_i.

Definition 1 (SLCA on XML data). Given a query Q in an XML tree T, an SLCA query finds the SLCA nodes v which is the child node of other LCA nodes.

The SLCA is defined as follows:

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

For example, Figure 3(a) gives a traditional XML tree which is generated by Figure 1 and a query $Q = \{\text{Tom}, \text{XML}\}$.

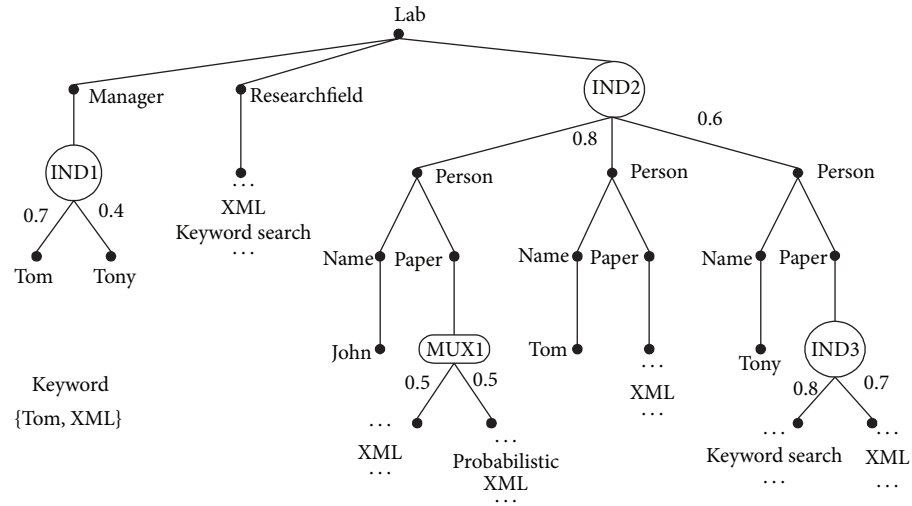


Figure 1: A probabilistic XML document.

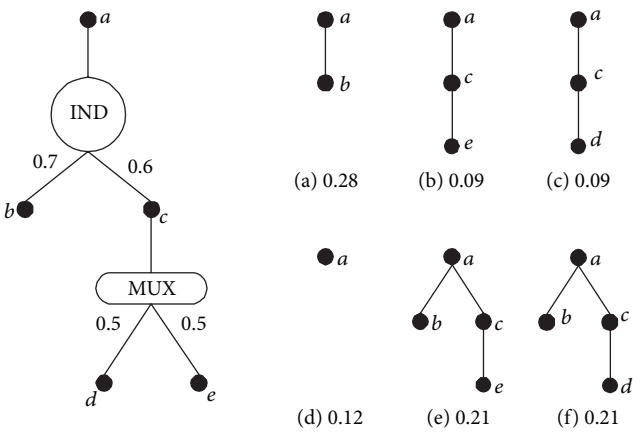


Figure 2: A probabilistic subtree.

The result of this query is shown in Figure 3(b). The node lab and person are LCA nodes. The SLCA node is person. According to the concept of SLCA, the node lab is an ancestor node of person. So, the SLCA node is person. From the definition of LCA and SLCA, we can see that SLCA is a subset of LCA.

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 a suitable result for users. SLCA node is the smallest common node of the nodes which contain keywords. As the result, we return the subtree rooted at SLCA node. So, when classification algorithm classify the nodes, it need put the keyword node and its all child nodes into one set.

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

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

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

$\Pr^T_{\text{slca}}(v)$ can also be computed with (4). Here, $\Pr(\text{path}_{r \to v})$ indicates the existence probability of v in the possible worlds. We can compute the existence probability by multiplying the conditional probabilities along the path from the root node r to node v. $\Pr^{\text{sub}}_{\text{slca}}(v)$ is the local probability for v being an SLCA node in $T_{\text{sub}}(v)$. $T_{\text{sub}}(v)$ denotes a subtree of T rooted at v. Consider

$$\Pr^T_{\text{slca}}(v) = \Pr(\text{path}_{r \to v}) \times \Pr^{\text{sub}}_{\text{slca}}(v). \quad (4)$$

From (3), we can obtain the following equation to compute $\Pr^{\text{sub}}_{\text{slca}}(v)$:

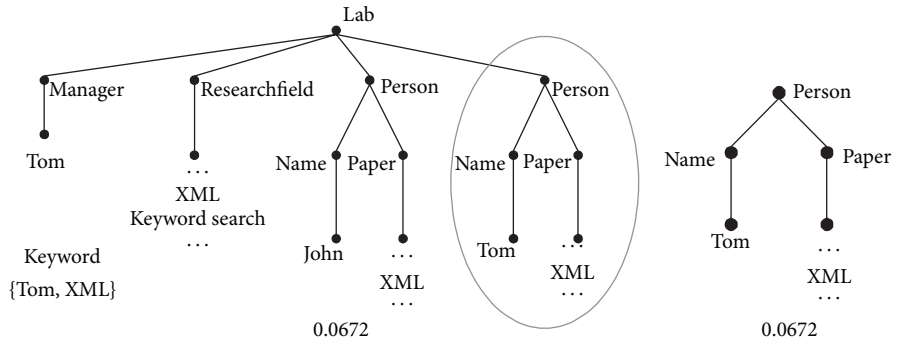
$$\Pr^{\text{sub}}_{\text{slca}}(v) = \sum_{i=1}^{m'} \{\Pr(w'_i) \mid \text{slca}(v, w'_i) = \text{true}\}, \quad (5)$$

where $\{w'_i, w'_2, \ldots, w'_{m'}\}$ denotes the local possible worlds generated from $T_{\text{sub}}(v)$. $\text{slca}(v, w'_i) = \text{true}$ means that v is an SLCA in the possible world w'_i rooted at v; namely, the root node v is the only LCA node. The probability of this subtree rooted at node v can be shown as

$$\Pr^{\text{sub}}_{\text{slca}}(v) = \Pr(\text{path}_{v \to k_1}) \times \Pr(\text{path}_{v \to k_2}). \quad (6)$$

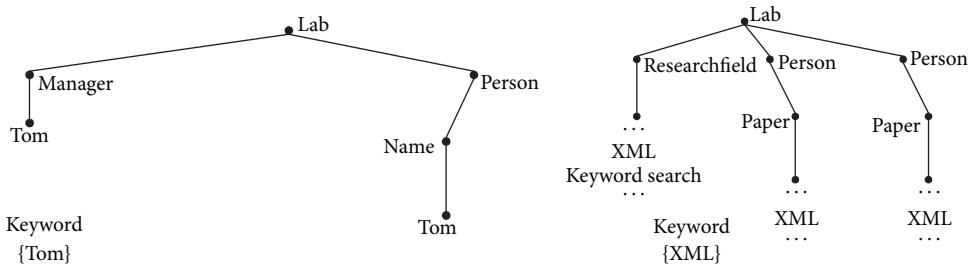
As an SLCA result of keyword search, the probability of node v is

$$\begin{aligned} \Pr^T_{\text{slca}}(v) &= \Pr(\text{path}_{r \to v}) \times \Pr^{\text{sub}}_{\text{slca}}(v) \\ &= \Pr(\text{path}_{r \to v}) \times \Pr(\text{path}_{v \to k_1}) \times \Pr(\text{path}_{v \to k_2}). \end{aligned} \quad (7)$$



(a) A traditional XML document (b) SLCA

Figure 3: SLCA nodes on XML data.



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

Figure 4: Keyword nodes tree.

For example, we give an example to compute $\Pr^{T}_{\text{slca}}(\text{person})$ (we select the node person with the existence probability of 0.8) in Figure 1. We can see that $\Pr(\text{path}_{\text{lab}\to\text{person}}) = 0.8$. $\Pr^{\text{sub}}_{\text{slca}}(\text{person}) = 1$. So, the result is $\Pr^{T}_{\text{slca}} = 0.8 \times 1 = 0.8$.

Definition 2 (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.

Definition 3 (Threshold 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. And the probability must be more than threshold θ.

If we do not consider the distributional nodes of the p-document, we can see that every SLCA node is an LCA node. For a given keyword query Q and a XML tree, node v is a common ancestor of Q on an XML tree if the subtree rooted at v contains each keyword of Q at least once. For example, for query Q = {Tom, XML}, the common ancestor nodes of Q are lab and person. If we built a tree contains common ancestor nodes according to the relationship on a XML tree, the leaves nodes are the SLCA nodes.

3. Classification of Nodes and Calculation of Probability

A p-document contains two types of nodes, ordinary, nodes and distributional nodes. An ordinary node can represent actual data on probabilistic XML data, but a distributional can only represent the probability distribution of a node.

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. Figures 4(a) and 4(b) shows the keyword nodes tree. When we use set intersection operation to obtain the common ancestor nodes tree such as shown in Figure 5. So, the important section is how to receive the keyword nodes tree.

To receive the keyword nodes tree, we need to 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 5 has its dummy node. These dummy nodes do not exist in the actual tree. The aim of adding dummy nodes is to classify nodes effectively.

3.2. Classification of Distributional Nodes. Distributional nodes can represent the probability distribution of the children nodes. A p-document defines a probability distribution

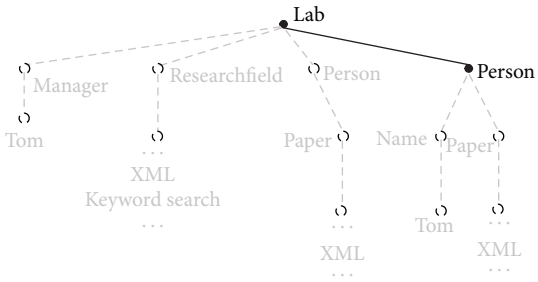


Figure 5: A common nodes tree.

over a space of deterministic XML documents. According to the different types of the distributional node, the number of copies is different.

Case 1 (If a node is an IND node). It has n children nodes, the number of copies is 2^n. If there is only one child node v_1 contains keyword k_1 and the probability of v_1 is p_{v_1}, the probability of the subtree rooted as v_1 which contains k_1 is p_{v_1}. If the number of children nodes which contains keyword k_1 is n, the probability need consider all the situations which contains keyword k_1. The sum of these probabilities is the probability of the subtree which contains keyword k_1. Considering all the situations is a very complicated calculation process. So, we can first calculate the probability of the subtree which do not contain the keyword k_1. So, the probability of an IND node is shown as (8).

$$p_{k_1}(v) = \begin{cases} p_{v_1} & \{\text{one } k_1 \text{ matched node}\} \\ 1 - \prod_{i=1}^{n} \left(1 - p_{v_i}\right) & \{n\ k_1 \text{ matched nodes}\} \end{cases} \quad (8)$$

Case 2 (If a node is a MUX node). The number of child nodes 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. So, the probability of a MUX node is shown as (9).

$$p_{k_1}(v) = \sum_{i=1}^{n} p_{v_i}. \quad (9)$$

Figure 6(a) shows an example of an IND node. For the keyword {Tom}, IND1 is a father 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 6(b) shows an example of a MUX node. For the keyword {XML}, MUX1 is a parent node of node {XML} and {Probabilistic XML}. 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 containing the keyword {XML} is $0.5 + 0.5 = 1$.

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 keywords, we will delete distributional nodes and connect its children nodes to its father node with the existence probability of containing the keyword of the subtree rooted at its father node according to the type of distributional node. For example, Figure 7(a) shows a tree contained keyword Tom. Node {manager} is a father node of node {IND1}. The probability of a subtree rooted at node {IND1} which contains keyword Tom is 0.7. As shown in Figure 7(b), it is the situation of the probabilistic tree which contains keyword XML.

We merge all the keywords probabilistic trees together. It will generate a keyword nodes probabilistic tree. We need calculate SLCA nodes on this tree with the probability and delete the subtree rooted at SLCA nodes. Next, we need to continue to calculate SLCA results on remaining nodes tree. So, if we repeat such operation, all the SLCA results will be generated. For example, Figure 8 is a keyword nodes probabilistic tree. This tree retains all the probabilities of the subtree which contains keyword.

3.3. Probability Calculation. If we calculate SLCA results on the tree in Figure 8, the node {person} which is shown in Figure 9(a) is the only result. So, we need to delete the subtree which is rooted at node {person}, and the remaining nodes tree is shown in Figure 9(b). To repeat calculated operation on the remaining nodes tree, we can see that the node {lab} is another result. In this section, we introduce how to calculate the probabilities of all the SLCA nodes.

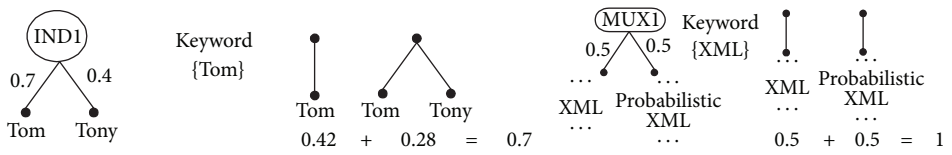
A distributional node can appear in all the positions in a tree excepted root node and leaf nodes. If node v is an SLCA node and there are two keywords $\{k_1, k_2\}$, the probability of node v needs to compute 3 values from (7); they are $p(\text{path}_{r' \to v})$ $p(\text{path}_{v \to k_1})$ and $p(\text{path}_{v \to k_2})$. As shown in Figure 10, r_1 represents the path from root node to node v and r_2 r_3 express the path from node v to k_1 k_2, respectively. So, if there are n keywords, we need to divide this tree into $n + 1$ parts.

Next, we discuss how to compute probability according to 4 situations.

Case 3 (*r1 r2* and *r3* do not contain distributional nodes). If node v is an SLCA node and there are no distributional nodes in r_1 r_2 and r_3, that means this situation is the same as deterministic XML data. The probability is 1.

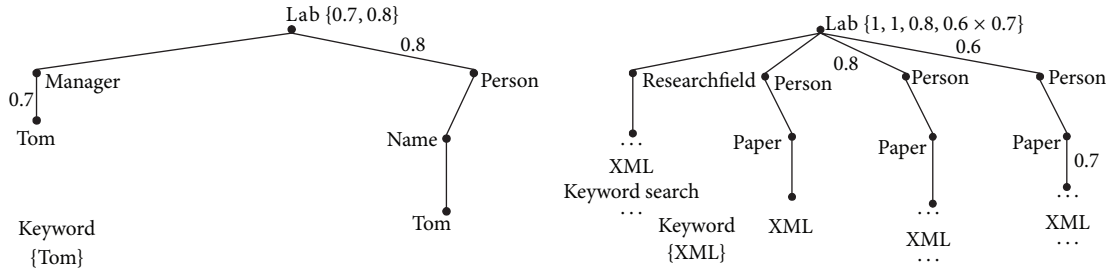
Case 4 (*r1* contains distributional nodes). If r_1 contains distributional nodes, that means node v will not appear in some possible worlds. These possible worlds do not contain node v. All the distributional nodes in r_1 will influence other nodes' probability in a keyword nodes probabilistic tree (just as in Figure 8). Figure 11 shows the processing of calculation in a keyword nodes probabilistic tree. The path from node {lab} to node {person} contains a distributional node, and the probability of the path is 0.8. So, when we finish the calculation of node {person}, the probability of node {lab} needs to add a new probability $1 - 0.8$ as shown in Figure 11.

Case 5 (*r2* or *r3* contains distributional nodes). If the path from node v to any keyword matched node contains distributional nodes, it illustrates that this keyword matched



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

Figure 6: Distributional nodes.



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

Figure 7: Keyword nodes probabilistic tree.

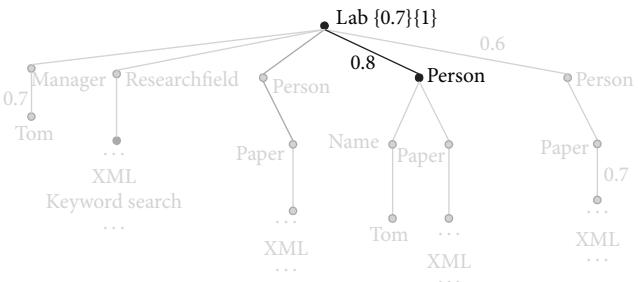


Figure 8: A keyword nodes probabilistic tree.

node will not appear in any possible worlds. In this situation, each probability of keyword matched node all need be recorded, just as in Figure 7(a) and Figure 7(b). Figure 7(a) shows the keyword {Tom} probabilistic tree. The node {lab} has two branches; one is the path which constituts nodes {lab} {manager} and {Tom}, and another one is the path that contains nodes {lab} {person} {name} and {Tom}. The probabilities in these two pathes are 0.7 and 0.8 as shown in Figure 7(a). The keyword {XML} is the same as shown in Figure 7(b). Figure 8 is a keyword nodes probabilistic tree. For the first keyword {Tom} in the second path ({lab} {person} and {name}), the node {person} contains another keyword XML. So, the node {lab} only need record the probability 0.7. For the second keyword XML, because the probability of the node {lab} containing it has the situation with value 1; other probabilities do not have to be recoded as shown in Figure 7(b).

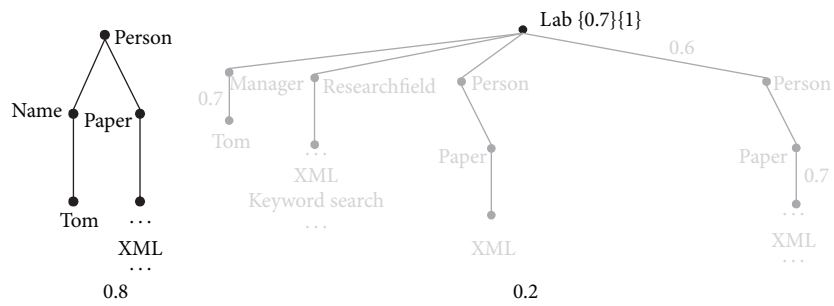
Case 6 (*r1 r2* and *r3* all contains distributional nodes). This situation can be seen as the synthesis of two kinds of aforementioned circumstances. The processing method has been introduced in the previous illustration, and in this section we only need to judge which one we need to compute firstly. Because SLCA nodes are the child nodes in all the common ancestor nodes, SLCA results need to be computed according to the path from bottom to up. In Figure 10, the nodes in r_2 and r_3 are child nodes of the nodes in r_1. The situation of r_2 r_3 has priority right. Next, the situation r_1 needs to be computed as the second step.

4. ELM Keyword Search on Probabilistic XML Data

Keyword search on probabilistic XML data based on classification mainly include four steps, they are shown as follows: (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) useing the set merging operation to structure the common ancestor nodes probabilistic tree, and (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 represents one keyword. For the given 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 using ELM to classify nodes on probabilistic XML data one by one.



(a) An SLCA node (b) Other nodes tree

Figure 9: SLCA of a common nodes probabilistic tree.

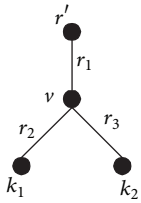


Figure 10: Node v tree.

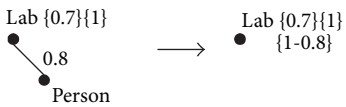


Figure 11: Calculation SLCA probability.

Table 1: The label of the nodes.

Node	Dummy node	Keyword
lab	{lab}	{Tom, XML}
IND2	{IND2}	{Tom, XML}
person	{person}	{Tom, XML}

First, adding dummy nodes according to the number of keywords. We can see that the probabilistic XML tree in Figure 1 contains two keywords {Tom, XML}. The algorithm uses 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. For example, in Figure 1, the node {lab}, {IND1}, and node {person} are added into the XML tree as a part of the dummy nodes tree. Each node on the probabilistic XML tree has a table. As shown in Figure 9. From the number of the nodes which contains keyword, the dummy nodes for adding are shown in Table 1.

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 consist of the classified nodes. ELM can classify the nodes to two sets such as shown in Figure 12(a). The first set represents keyword Tom, and the second set represents 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 to 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 12(b), the node {*Tom*} accepts the probability 0.7 from its father node {IND1}.

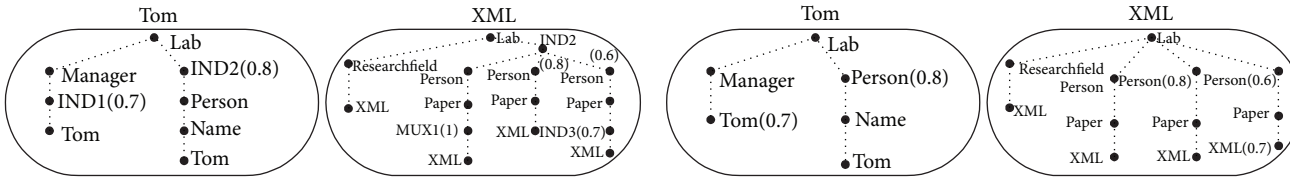
Third, use the set merging operation to structure the common ancestor nodes probabilistic tree. The intersection of the two sets is the set which includes node {lab} and {person}. Figure 13 shows the union set of two keyword sets.

Finally, repeat the operation of calculating SLCA and deleting the subtree; all the SLCA results will generate. Let us calculate SLCA result on the tree which is shown in Figure 13. 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 leaves to the other nodes tree. Next, the node {lab} is another result. The probability of this result is $0.2\times0.7=0.14$. 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.

From (4) we can see that the product of the extensive probability and the SLCA probability is the result probability. The extensive probability is 1 to the node {person}, so the result probability is $0.7\times1=0.7$. Moreover, the extensive probability of the node {lab} is 1, and the result probability of the node {lab} is $1\times0.14=0.14$.

5. ELM-Threshold Keyword Search on Probabilistic XML Data

Prunning can speed up the retrieval speed, we can delete any nodes that are not SLCA nodes in the processing of calculation SLCA results. According to the definition of SLCA, if a node is an SLCA node, its all ancestor nodes are not SLCA nodes. If there are no distributional nodes in r_1 r_2 and r_3, node v is an SLCA node and its father node r' is not SLCA node. For example, node {person} is an SLCA node, so its father node {lab} is not an SLCA result.



(a) A keyword p-tree

(b) P-tree without distributional nodes

Figure 12: Probabilistic tree.

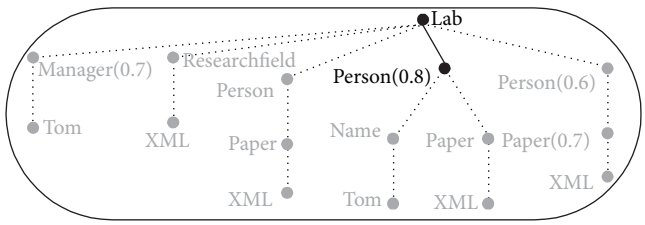


Figure 13: A common nodes p-tree.

Table 2: Properties of probabilistic XML data.

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

Some results' probability is very small, these results have very low extensive probability. For most users' query intension, if the probability is more than other nodes, that means that the node has high query value. So, this section introduces the concept of probability threshold.

Case 7 (*r1* contains distributional nodes). Because node v is an SLCA node, when we finish the calculation of node v, its ancestor nodes need to add the probability $1 - p(v)$. As shown in Figure 9, the value $(1 - 0.8) = 0.2$ has been retained in the other nodes tree in Figure 9(b). So, if the probability value $1 - p(v)$ is less than the probability threshold, the probability of node {lab} must be less than the probability threshold. All the ancestors must be deleted, because they are not threshold SLCA nodes.

Case 8 (*r2* or *r3* contains distributional nodes). There are distributional nodes in $r2$ or $r3$. When we finish the calculation of node v and 2 keywords, there are $3 = 2^2 - 1$ situations will be remained. We need to consider all these 3 situations. Each situation has the probability $1 - p_{\text{path}}(v \to k_1)$. If the probability value $1 - p_{\text{path}}(v \to k_1)$ is less than the probability threshold, the probability of node {lab} must be less than the probability threshold. All the ancestors must be deleted, because they are not threshold SLCA nodes. If there are no k_1 and k_2 in the subtree rooted at v, the probability is $(1 - p_{\text{path}}(v \to k_1)) \times (1 - p_{\text{path}}(v \to k_2))$. If the probability value $(1 - p_{\text{path}}(v \to k_1)) \times (1 - p_{\text{path}}(v \to k_2))$ is less than the probability threshold, the probability of node {lab} must be less than the probability threshold. All the ancestors must be deleted, because they are not threshold SLCA nodes.

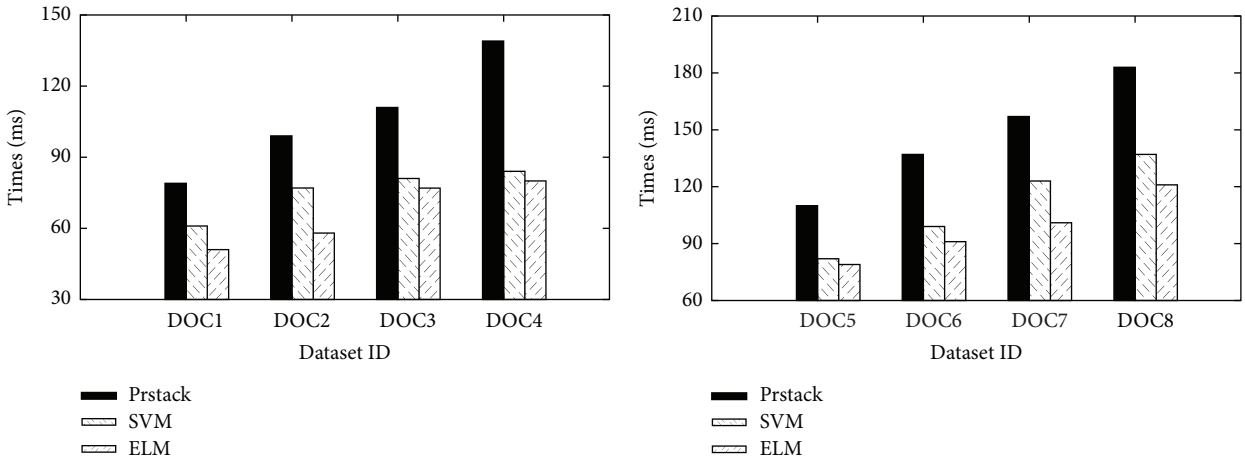
For a probability threshold θ, when we calculate the probability on the keyword nodes probabilistic tree, if the probability is less than the threshold θ, the result is not an SLCA node. For example, if the probability threshold is 0.3, the SLCA probability of the second result is $0.2 \times 0.7 = 0.14$ which is less than the probability threshold; the node {lab} is not a threshold SLCA node. When we compute the probability of the first result of the node {person}, because the probability $1 - 0.8 = 0.2$ will be leaves to the other nodes tree, and the probability value is less than the probability threshold, the probability of other nodes tree must be less than the probability threshold. So, we need not calculate SLCA on the other nodes tree. When the first result is computed, the threshold SLCA is finished.

6. Performance Verification

In this section, the performance of keyword search used ELM to classify on probabilistic XML data is shown as follows. All the experiments based on ELM for classification algorithms are carried out in MATLAB 2007 environment running in a Pentium 4, 2.53 GHZ CPU.

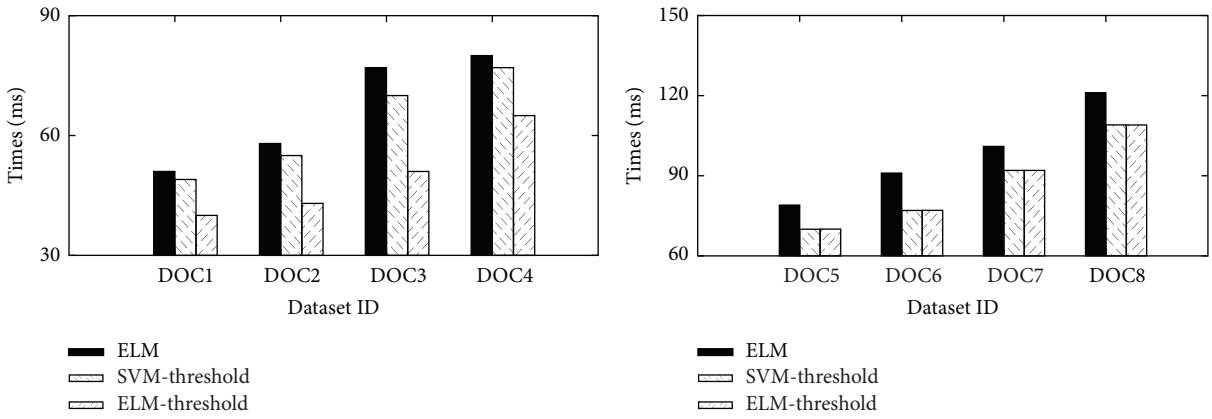
The dataset we used is shown in Table 2. In this paper, the algorithm selects two datasets XMARK and DBLP. For each XML dataset used, we generate the corresponding probabilistic XML tree, using the same method as used in [5]. Table 3 shows the dataset and keywords. We visit the nodes in the original XML tree in preorder 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 a restriction 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.

Compared with those traditional computational intelligence techniques, ELM provides better generalization performance at a much faster learning speed and with least human intervention. We compare the query times of two situations



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

Figure 14: Vary query over DOC1 to DOC8.



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

Figure 15: Vary query over DOC1 to DOC8.

Table 3: Keywords in probabilistic XML data.

ID	Name	Keyword query
DOC 1	XMARK	United States, Gredit
DOC 2	XMARK	United States, ship
DOC 3	XMARK	Ship, Credit, Alexas
DOC 4	XMARK	United States, ship, Gredit
DOC 5	DBLP	XML, Keyword
DOC 6	DBLP	Keyword, query
DOC 7	DBLP	XML, Keyword, probabilistic
DOC 8	DBLP	Keyword, query, XML

about keyword search in probabilistic XML data. The first situation is using the method in [17] to retrieve the SLCA nodes, and the second situation is using SVM and ELM to classify nodes for the keyword search on p-document. The second situation classifies nodes by using SVM and ELM. The speed of classification is shown in Figure 14.

From Figure 14 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 situations of the node which contains keywords. ELM can classify nodes according to the code and keywords by retrieving all the nodes once. So, the algorithm has high speed by using ELM to classify compared with SVM.

Next, we compare the query times of two situations about keyword search in probabilistic XML data. The first situation is using ELM to classify nodes for the keyword search on p-document, and the second situation is adding the probability threshold to classify nodes for the keyword search on p-document based on SVM. The third situation is adding the probability threshold to classify nodes for the keyword search on p-document based on ELM. The speed of classification is shown in Figure 15.

From Figure 15, we can see that the probability threshold has advantages. If the threshold is set to 0.4, from the results, we can see that compared with ELM algorithm,

the algorithm SVM-threshold and ELM-threshold which adds the probability threshold can also improve the time efficiency. ELM-threshold has advantages of speed compared with SVM-threshold. The pruning algorithm can be more than 1.3 times faster than the first algorithm. After adding probability threshold, the exist probability is less than probability threshold will be deleted; the probability will be also deleted when the probability is less than probability threshold on the process of computing probabilities.

7. Related Work

Recently, keyword search has been studied extensively in traditional XML data. For a keyword query and an XML tree, most of related work took SLCA and ELCA as the results to be returned. XRANK [1] developed stack-based algorithm to compute SLCA. Reference [2] introduced two algorithms, they are the Indexed Lookup Eager algorithm when the keywords appear with significantly different frequencies and the Scan Eager algorithm when the keywords have similar frequencies. Reference [3] designed an MS approach to compute SLCA for keyword queries in multiple ways. Reference [4] took set intersection to compute SLCA. Reference [18] proposed the Indexed Stack algorithm to find ELCA. The probabilistic XML model has been studied recently. In [6], they first introduced a probabilistic XML model with the probabilistic types IND and MUX. IND means independant and MUX means mutually exclusive. Reference [5] summarized and extended the probabilistic XML models previously proposed; the expressiveness and tractability of queries on different models are discussed with the consideration of IND and MUX. Reference [17] addressed the problem of keyword search in probabilistic XML data and computed SLCA by scanning the keyword inverted lists once. Different from all the above work, we adopt set intersection to compute SLCA in probabilistic XML data.

8. Conclusions

In this paper, we have addressed the problem of keyword search in a general probabilistic XML data. And we adopt probabilistic XML model $\text{PrXML}^{\{\text{ind,mux}\}}$. Given a probabilistic XML tree T, a set of keywords, and a probability threshold, we have discussed the challenges to find SLCA results with the probability which is more than probability threshold. Our algorithm has been proposed to compute the SLCA probabilities without generating possible worlds. We adopt set intersection to compute SLCA based on ELM. This paper uses ELM to classify nodes according to keyword search on probabilistic XML data. Keyword search on probabilistic XML data has 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. The experiments have demonstrated efficiency of our algorithms.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

Yue Zhao, Ye Yuan, and Guoren Wang were supported by the NSFC (Grant nos. 61025007, 61328202, and 61100024), National Basic Research Program of China p(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).

References

[1] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram, "XRANK: ranked keyword search over XML documents," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '03)*, pp. 16–27, 2003.

[2] 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.

[3] C. Sun, C.-Y. Chan, and A. K. Goenka, "Multiway SLCA-based keyword search in XML data," in *Proceedings of the 16th International World Wide Web Conference (WWW '07)*, pp. 1043–1052, Alberta, Canada, May 2007.

[4] 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.

[5] B. Kimelfeld, Y. Kosharovsky, and Y. Sagiv, "Query efficiency in probabilistic XML models," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '08)*, pp. 701–714, June 2008.

[6] A. Nierman and H. V. Jagadish, "ProTDB: probabilistic data in XML," in *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB '02)*, pp. 646–657, 2002.

[7] 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.

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

[9] G.-B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 274–281, 2003.

[10] G.-B. Huang and C.-K. Siew, "Extreme learning machine with randomly assigned RBF kernels," *International Journal of Information Technology*, vol. 11, pp. 16–24, 2005.

[11] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, no. 16–18, pp. 3460–3468, 2008.

[12] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16-18, pp. 3056–3062, 2007.

[13] J.-G. Taylor, "Cognitive computation," *Cognitive Computation*, vol. 1, no. 1, pp. 4–16, 2009.

[14] M. Wöllmer, F. Eyben, A. Graves, B. Schuller, and G. Rigoll, "Bidirectional lstm networks for context-sensitive keyword detection in a cognitive virtual agent framework," *Cognitive Computation*, vol. 2, no. 3, pp. 180–190, 2010.

[15] P. K. Mital, T. J. Smith, R. L. Hill, and J. M. Henderson, "Clustering of gaze during dynamic scene viewing is predicted by motion," *Cognitive Computation*, vol. 3, no. 1, pp. 5–24, 2011.

[16] E. Cambria and A. Hussain, *Sentic Computing: Techniques, Tools, and Applications*, 2012.

[17] J. Li, C. Liu, R. Zhou, and W. Wang, "Top-k keyword search over probabilistic XML data," in *Proceedings of the IEEE 27th International Conference on Data Engineering (ICDE '11)*, pp. 673–684, IEEE, Hannover, Germany, April 2011.

[18] Y. Xu and Y. Papakonstantinou, "Efficient lca based keyword search in xml data," in *Proceedings of the 11th International Conference on Extending Database Technology (EDBT '08)*, 2008.