A Semantic Tree Model-based Approach for XML Keyword Query *

Xin LI 1,∗, Zhiguang CHU 1, Xiangfu MENG 2

1 Computer Center, Liaoning University of Technology, Jinzhou 121001, China
2 College of Electronic and Information Engineering, Liaoning Technical University, Huludao 125105, China

Abstract

The dependences between structure and content of XML document cannot be effectively captured by the traditional tree-based modeling method, and thus leads to the XML keyword query difficult to achieve a high recall and precision. Using the semantic analysis techniques, this paper proposes a novel XML document modeling and keyword query approach, which takes full account of the nodes context to optimize the XML document tree modeling and construct the XML Semantic Tree (XST). Based on the XST, the set of Semantically Related (SR) node is firstly defined, and then the rules and algorithm for determining the set of SR nodes are developed. The XML keyword query method, which is based on XST modeling and set of SR nodes, is finally presented. The experimental results demonstrates that the approach proposed in this paper can improve the query precision and recall significantly comparing with the existing typical approaches, and it also has the high performance.

Keywords: XML; Keyword Query; XST; SR

1 Introduction

With XML data widely used on the internet, XML keyword query has been widespread concern. Keyword query is a user-friendly information retrieval technology, which allows users to search that users can not grasp query language (such as XPath, XQuery) and do not understand the structure of XML document. Because of XML datas semi-structured features, that making keyword query XML data has new feature. The new feature is that the result is no longer the entire XML document, but which compact full semantic document fragments. The information of user interested returned, not only to find the keyword content information, but also to structural information of the query to be a reasonable limit. Therefore, the key issues of improving the keyword query XML documents recall and accuracy is how to find the user a number of compact and fully reflects the semantic results.

*Project supported by the Natural Science Foundation of Liaoning Province, China (201302028).
*Corresponding author.
Email address: lgjsjlx@163.com (Xin LI).

1553–9105 / Copyright © 2014 Binary Information Press
DOI: 10.12733/jcis10359
June 15, 2014
There are several classical XML keyword search systems, such as XRank [1], XKeyword [2], XK-Search [3], Schema-Free XQuery [4] and XSeek [5]. XRank and XKeyword use a graph-based XML document model. XKSearch, Schema-Free XQuery and XSeek use the tree-based models. These methods result set was created through finding the sub tree containing the keyword in the document tree. Which is defined as LCA (XRank) or its variants such as SLCA (XKsearch), ML-CA (Schema-Free XQuery) and so on. In recent years, XML keyword proximity queries matching research work focuses on how to find the most compact of all keywords containing fragments SLCA research. Literature [6] proposed a statistical based SLCA and fragments keyword approximate query methods, and the use of query results TFIDF ideological sorting method. Literature [7] is based on the keyword content information to classify the query results. In order to improve query efficiency, the offline phases of the XML document contents are summarized. When a keyword query arrives, quickly find a matching keyword content summary according to their degree of correlation with the query results for classification. Literature [8] proposed a retrieval method that based on SLCA and merger-trim Top-K. Firstly, the method find keyword matching SLCA, then according to the degree of matching with the keyword SLCA its merger trim and a quick return to Top-K SLCA as the most relevant search results. Under the premise of without changing XML structure information, literature [9] proposed the use of XML data as a function dependencies and interaction between user and system information and query results approximate matching keyword coherent order. Literature [10] proposed a directed acyclic graph based on the index to effectively organize a large number of keywords-based queries to reduce the cost of query matching. And according to the characteristics of the data stream, we propose a stack-based temporary result caching method to filter a large number of independent data with the query node. Literature [11] designed a semantic-based XML keyword search engine-Ropeway. It uses XML data contains semantic information, analyze the semantics of XML data and user information needs, infer the user’s query intent to improve the retrieval quality. Literature [12] proposed an XML-based XLCA keyword search method. This document presents the results based on XLCA set definition, and gives a compact index structure and corresponding BuP algorithm to efficiently find the result set.

It should be noted that the above XML-based document tree modeling and SLCA keyword query method in determining the relationships between nodes, the nodes based entirely on the label and the relationship between them communicating, while ignoring the context node in the document relations. The query methods are difficult to capture semantic information inherent in the structure, so that the query accuracy and recall is limited to the lower level. For lack of existing methods, this paper proposes a new XML document model is implemented in this model as well as semantic keyword query approximation methods, and experimentally demonstrate the effectiveness of the method.

2 Solution

XML keyword search methods mentioned herein solution is as follows:

(i) Optimizing the model of the XML document-on the basis of XML document tree model to analysis the context node semantic features to build an XML semantic tree (XML Semantic Tree, XST).

(ii) Anglicizing the semantic correlation on the correlation of the XST-according to the nature of semantically related nodes, define semantically related set of nodes, and gives its established
rules and calculation methods.

(iii) XML-based semantic tree and semantically related set of nodes—implement an XML document using the stack keyword queries.

3 XST Modeling and Query Matching

3.1 XML semantic tree model

Semi-structured XML document determines the content and structure of the phase dependence. Valid XML document modeling to map the structure of information is the basis for processing XML data. A typical tree modeling approach is to build an XML document tree: \( T = (N, E, r, L) \), where \( N \) is a finite set of nodes, \( E \) is a finite set of \( T \); \( r \) is the root of \( T \), \( L \) is a finite set of the tag carried by nodes in \( T \). The advantage of the XML document tree is straightforward XML elements to reflect the level of nesting. XML document to element constituent units can be divided into various segments, which each segment contents and some documents related to the content. To some extent, XML document structure defines the semantic content. Although simple tree nesting able to clearly reflect the hierarchy of the document, it does not adequately reflect the content and structure of dependencies between.

By analyzing the XML document tree structure can be found in the XML hierarchy, each child node inherits properties from its parent and sibling nodes by sharing its parent node inherits the properties of the underlying. Thus, each of the parent node and its child nodes by a node share a common set of attributes associated parent node. Especially for data nodes, which are distributed in the XML tree leaf layer and its parent node is one to one, therefore regarded as dependent on its parent nodes are grouped together. In this way, each set can be a logical entity. Through specific context nodes to define the semantics of the node, in order to better reflect the content and structure of dependencies between. Defining the semantics of the context node is conducive to independent node in the query filter; however, the semantics of the node will be strictly limited to limited to the node limited to a specific environment. This also causes the node was spun off from the document, which led to significant loss of query results.

In order to strike a balance between the two, we introduce the concept of ontology; ontology is important concepts in specific areas of shared formal description. Ontology refers to a class share the same field that contains the attribute, cognitive characteristics of the entity set of classes. In practice, the two entities may belong to different types of the same name, or there are two entities of the same type have a different name. This method can overcome the ambiguity. Therefore, this article parent class to tag ontology defined logical entity.

Based on the foregoing analysis, optimize the XML document tree model. On this basis, it combined parent-child nodes for each group as a single logical entity. And the body of the parent node class mark as a semantic unit/semantic node. Then according to the original XML document tree of parent-child relationships, transforming XML documents tree into an XML semantics tree.

Definition 1 Semantic Unit/Semantic Node: \( sn = (OL(n_p), N_{pc}), OL(n_p) \), \( sn \) is the label, is the parent node of the body tag of the \( N_{pc} \), that \( L_{sn} = OL(n_p)N_{pc} \) is an XML document tree \( T \) on a parent-child nodes and sub-nodes attached to a limited set of data nodes.

Definition 2 XML Semantics Tree: \( XST=(N_{sn}, E_{sn}, sr, L_{sn}) \), \( XST \) is a four-tuple, which \( sn \) is a
finite set of $N_{sn}, N_{sn} l = (N_i, 1 \leq i \leq |N_{sn}|), E_{sn}$ is a finite set of $N_{sn}, E_{sn} \subseteq N_{sn} \times N_{sn}, sr$ is the root of XST, $L_{sn}$ is a finite set of labels carried by $sn, L_{sn} = (L_i, 1 \leq i \leq |L_{sn}|)$.

XST constructing principles: if the XML document tree $T$ has two internal nodes $n$ and $n'$, $(n, n')$ is an edge of XML document tree $T$, or $n$ is $n'$ ancestor, and the path from $n$ to $n'$ does not exist any node, then $(sn_i, sn_j)$ is an edge of XST. $sn_i= (OL(n), N_i)$ and $sn_j= (OL(n), N_j), N_i$ and $N_j$ are the set of data nodes of $n$ and $n'$.

### 3.2 Semantically related set of nodes

In this article, the XST node that contains the keyword referred to as $ksn$ node, all nodes with a semantically related $ksn$ node for the current node semantically related $ksn$ set of nodes, denoted $SR_{ksn} (sn \in SR_{ksn})$. It represents a $sn$ belongs to a set of nodes $ksn$ of semantically related. Snoeck et al proposed a type-oriented modeling is a correlation between the presences of an object type partial order. If two objects $O_i$ and $O_j$ belong to the same type, $O_i$ and $O_j$ relationship does not exist, and vice versa. It follows that the nature of semantically related nodes.

**Property 1** if a node named $sn$ is the semantic relevant node of $ksn (sn \in SR_{ksn}),$ the nodes $sn$ and $ksn$ have the different labels ($L_{sn} \neq L_{ksn}$).

**Property 2** if a node named $sn$ is the semantic relevant node of $ksn (sn \in SR_{ksn}),$ and $sn$ is the node between the $sn$ and $ksn$ on the XST, the nodes $sn$ and $sn$ have the different labels ($L_{sn} \neq L_{sn}$).

**Property 3** if a node named $sn$ is not the semantic relevant node of $ksn (sn'! \in SR_{ksn}),$ and $sn$ is that XST associated to $ksn$ by $sn, sn$ is not the semantic relevant node of $ksn (sn'! \in SR_{ksn})$.

Taking these properties, if two nodes $sn$ and $sn$ on the XST to their LCA does not contain multiple paths with the same label $sn$. That is, the LCA of $sn$ and $sn$ is the only node that contains the same label, then the two nodes $sn$ and $sn$ are the semantically related nodes. It can be concluded definition of semantically related set of nodes as follows:

**Definition 3** Set of Semantically Related Node (SR): Current $ksn$ semantic relevant nodes set on the XST is a finite set of $sn, SR_{ksn} = \{sn | sn is a node on the XST and L_{sn} \neq L_{sn}\}$.

Node set of semantically related concepts can make search focused on specific parts of an XML document, and thus improve the search efficiency and accuracy of the results.

From the above definition and nature come to the following pruning rules of determining the SR:

**Rule 1** From the XST cut any $sn$ that satisfy the following condition: ($L_{sn} = L_{ksn}$).

**Rule 2** From the XST cut $n'$ that satisfy the following condition. The condition is the other $n''$ is positioned between the $n'$ and the $ksn$, and $L_{n'} = L_{n''}$.

**Rule 3** From the XST cut any $sn$ that any through $n$ and $n! \in SR_{ksn}$ connected with $ksn$. 
The simple method of calculating the current SR of $k_sn$ is to use the three rules pruning to the $sn$ on the XST. Calculate the $s_n$ SR algorithm’s time complexity is $O(|N_{sn}|^2)$. However, can be drawn from the rule 3, if $n \in SR_{k_sn}$, the others that adjacent sn are not in the SR. Which can be optimized calculated as follow: do not check every $sn$ on the XST and only check the $sn$ with adjacent $n$. if $n \in SR_{k_sn}$, should check $sn$ which adjacent to the $n$, otherwise it would not check for any $n'$ that connect to $k_sn$ through $n$. The time complexity of the algorithm is $O(\sum_{i=1}^{|N|}(|SR_{N_i}|))$. An algorithm of computer SR set on the XST shown in Algorithm 1.

### Algorithm 1 Compute SR set on the XST algorithm

01: $n = k_sn$
02: $S = null$
03: $SR_{k_sn} = null$
04: Examine sn($n$)
05: While $n \in adj[n]$ do
06: If $L_{n'} \neq L_{k_sn} \& L_{n'} \not\in S$ then
07: $SR_{k_sn} = SR_{k_sn} \cup n'$
08: Examine sn($n'$)
09: end if
10: enddo
11: return $SR_{k_sn}$

### 3.3 Keyword search algorithm

#### Algorithm 2 Keyword query algorithm on the XST

01: stack=empty
02: for $(1 \leq k \leq Q.length)$
03: keyword=$Q[k]$
04: $p$=lca(stack, keyword)
05: textbf{while} (stack.size $>$ $p$) do
06: stackEntry=stack.pop()
07: If (is_Result(stackEntry) then
08: output stackEntry as a result
09: else
10: for $(1 \leq j \leq m)$
11: if (stackEntry.Keyword[$j$]=true) then
12: stack.top.Keywoed[$j$]=true
13: end if
14: end if
15: end if
16: end do
17: if keyword in $SR_{Stack}$ then
18: for $(p<j$) keyword.length)
19: stack.push(keyword[$j$], [])
20: stack.top.Keywords[$j$]=true
21: end for
22: end for
23: end do
24: end for
25: end if

For the answer to a keyword query of $n Q = \{k_1, k_2, \ldots, k_n\}$, in this paper, on the XST nodes DeweyID coding based on the use stack-based algorithm. Stack-based algorithm works as follows: the stack algorithm, each stack entry is a $(I_d, Keywords)$ tuple. $I_d$ from the bottom to the top of the stack was $I_d1, I_d2, \ldots, I_dn$, then the stack represents a DeweyID code of $I_d1, I_d2, \ldots, I_dn$. Keywords is a kBoolean array. When $Keywords[i] = T$, which means that the current stack node represented sub tree rooted directly or indirectly include this keyword $k_i$. For an array of input query $Q = k_1, k_2, \ldots, k_n$, which means that the keywords used DeweyID, each time a node containing the keyword stack. Then it searches the node that containing other keywords in the set of nodes. Each push it to determine whether the current node has a longer push common node, no further calculations exclude invalid node, pop longest common prefix that do not contain the top entries until the stack to save a result.

An algorithm of keyword query on the XST shown in Algorithm 2, the time complexity is $O(h|sn|)$, where $h$ is the depth XST, $|sn|$ is the number of $sn$ on the XST.
4 Experimental Evaluation

Experimental hardware environment: AMD Sempron(tm) X2 180 dual-core processor clocked at 2.41GHz, 2G DDR3 1333Hz RAM, Windows XP Professional operating system. Experimental test data set using the actual data set DBLP. We make a comparison with the typical method such as XSEarch, Schema-Free XQuery (SFXQuery) and XKSearch systems. And with the proposed method is based on the design and implementation of a keyword-based query XML semantic retrieval system (XML Semantic Search, XSS). XSS system consists of an XML document parsing module P, query processing module Q, and input and output modules I/O composition. XSS system architecture is shown in Fig. 1.

XSS system as a test prototype system that can accept user input and then generates a list of keywords to query processing module, the final module of the query results returned to user. Each module functions as follows:

XML document parsing module: XML documents and XML schema (schema or DTD) generated through the initial parsing XML document tree, and then by his son node XST construction method combined with ontology database and ontology editing tools to do the merged node body tag generation XST. Practice, most of the nodes (75%) of the body tag main body can use the existing data set is not a small part from the main body can be obtained by concentration of the rib body editor (the system uses a tool Protégé 4.0) generates. For practical system you can add it to generate a body tag ontology data for future use.

Query Processing Module: receive the keywords list of input/output Module and the XST coding of XML document parsing module; establish the inverted index table, filter nodes that contain the keyword on the XST, generate semantically related nodes that contain the keyword node set by the Algorithm_SR, then generate results for query results by the Algorithm_SR.

Input/Output Module: sort the results; output the results list to the user, and the system follows the XRank system sort function.

Experimental evaluation is mainly a test query results and execution efficiency. Query results returned by the system based on the results of four recall and precision to measure. Execution
efficiency is by comparing the four systems query response time to measure. The results are in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>XSS</th>
<th>Schema-Free</th>
<th>XKsearch</th>
<th>XSEarch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Averaged Recall</td>
<td>0.698</td>
<td>0.381</td>
<td>0.534</td>
<td>0.450</td>
</tr>
<tr>
<td>Averaged Precision</td>
<td>0.636</td>
<td>0.322</td>
<td>0.451</td>
<td>0.454</td>
</tr>
<tr>
<td>Response Time(s)</td>
<td>1.712</td>
<td>2.324</td>
<td>2.879</td>
<td>1.566</td>
</tr>
</tbody>
</table>

As can be seen from the data in the table XSS system query response time was less than XSEarch systems and SFXQuery system, slightly XKSearch system. The results of the query terms, XSS system compared to other systems of its recall and precision has been greatly improved, indicating the effectiveness of this method.

5 Summary

This paper presents a combination of semantic analysis and modeling of XML documents based on this keyword query methods. This major work is summarized as follows: (1) build the XML semantic tree by optimizing the XML document tree (XST); (2) propose the definition and calculation methods of the set of semantically related node (SR) according to the nature of the XST semantically related nodes; (3) achieve a keyword query of XML document based on XST and SR. Experimental results and comparative analysis shows that the proposed method over existing methods in the keyword query typically recall and precision has increased dramatically, and has a higher efficiency. How can we more effectively parse the XML document structure properties? How accurately capture the user’s query intent on the basis of the existing? And how to get better query results is the next research goal.

References


