J. Software Engineering & Applications, 2010, 3: 65-72
doi:10.4236/jsea.2010.31008 Published Online January 2010 (http://www.SciRP.org/journal/jsea)

Element Retrieval Using Namespace Based on Keyword Search over XML Documents

Yang WANG[1], Zhikui CHEN[1], Xiaodi HUANG[2]

[1]School of Software, Dalian University of Technology, Dalian, China; [2]School of Computing and Mathematics, Charles Sturt University, Australia.
Email: Wayag2000@yahoo.com.cn, zkchen@dlut.edu.cn, xdhuang@csu.edu.au

Received July 31st, 2009; revised September 12th, 2009; accepted September 22nd, 2009.

ABSTRACT

Querying over XML elements using keyword search is steadily gaining popularity. The traditional similarity measure is widely employed in order to effectively retrieve various XML documents. A number of authors have already proposed different similarity-measure methods that take advantage of the structure and content of XML documents. However, they do not consider the similarity between latent semantic information of element texts and that of keywords in a query. Although many algorithms on XML element search are available, some of them have the high computational complexity due to searching for a huge number of elements. In this paper, we propose a new algorithm that makes use of the semantic similarity between elements instead of between entire XML documents, considering not only the structure and content of an XML document, but also semantic information of namespaces in elements. We compare our algorithm with the three other algorithms by testing on real datasets. The experiments have demonstrated that our proposed method is able to improve the query accuracy, as well as to reduce the running time.

Keywords: *Semantics, Namespace, SVD, Text Matching*

1. Introduction

Keyword search querying over XML elements has emerged as one of the most effective paradigms in information retrieval. To identify relevant results for an XML keyword query, different approaches lead to various search results in general. Some authors calculated the similarity between the content of XML documents and query, only analyzing the content and structure of XML (e.g., [1–3]). Many algorithms calculate the degree of text of elements matching with the keywords to produce the ranked result-list (e.g., DIL Query processing algorithm [4] and Top-k algorithm [5]). The classical methods focus on TF-IEF formula to calculate the cosine similarity between elements and query (e.g., Tae-Soon Kim *et al*. [6]; Maria Izabel M *et al*. [7]; Yun-tao Zhang *et al*. [8]).

In particular, overlaps of elements in XML documents must be considered. For several overlapping relevant elements, we have to choose which one should be avoided to ensure that users do not see the same information for several times. Su Cheng Haw *et al*. [9] presented the TwigINLAB algorithm to improve XML Query processing. In this paper, we modify it to deal with the elements overlap occurring in keyword search results.

On the basis of previous work, we make the following contributions in this paper. Firstly, we utilize the semantic information of namespaces in elements to filter the relevant components since the text of elements are commonly related with semantic information of namespace. Secondly, the precision and recall of our algorithm show that the non-text matching but semantic relevant elements with respect to the keyword can be effectively retrieved. Compared with traditional work, our algorithm also shows the better performance on time execution over a large collection of elements.

The rest of this paper is organized as follows: Section 2 introduces the element-rank schema by keyword search. Section 3 presents the Namespace Filter Algorithm (NFA). The experiments on the comparison of NFA and related methods are reported in Section 4. Related work is presented in Section 5, followed by the conclusion.

2. Element-Rank Schema

In this section, we utilize the namespace of elements to describe our element rank schema. Another goal of utilizing namespace is to filter relevant elements with the keyword in a query to reduce time execution compared with traditional algorithms.

Interestingly, namespaces can distinguish different ele-

ments containing the same markup that refers to different semantic meanings. As an illustration, we consider two elements with the same markup of <table>:

```
<table>
    <td>apple</td>
    <td>banana<td>
</table>
<table>
    <name>coffee table</name>
    <width>80</width>
</table>
```

This will lead to the confliction when they are in the same XML document. Thus, we utilize different namespaces of 'h' and 'f' to distinguish them as below.

```
<h:table xmlns:h = "http://.../fruit">
    <h:td>apple</h:td>
    <h:td>banana</h:td>
</h:table>
<f:table xmlns:f = "http://.../furniture">
    <f:name>coffee table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
</f:table>
```

As discussed above, the text of elements is commonly related with the semantic information of their namespaces. Given the semantic information of namespace that is irrelevant to the keyword, it is not desirable to access all the elements containing this namespace. In order to calculate the semantic similarity between namespaces and keywords, we map semantic information of namespaces and keywords into different vectors in a concept vector space created by Singular Value Decomposition (SVD) [10] over a collection of elements. In order to do this, Definitions 1 and 2 are provided as follows.

Definition 1: $prefix(v)$: a function that maps the namespace of element v into a vector and represents a special meaning in the concept vector space created by SVD.

Definition 2: $correlation(prefix(v), keyword)$: the degree of relevance calculated by the cosine similarity between the namespace vector of element v and the keyword vector in concept vector space created by SVD.

The value of the correlation is commonly normalized between the range [-1,1]. If the semantic meaning of namespaces is very close to that of the keywords, the value of '*correlation*' will be around 1. Nobert Govert *et al*. [2] proposed the concept of degree of relevance between elements. We extend it to include several intervals in [-1,1] to describe the degree of the semantic similarity of namespaces and keywords. Without loss of generality, some definitions are provided to describe the degree of relevance between namespaces and keywords.

Definition 3: High relevance: the high correlation between namespaces and keywords which satisfies

$$\lambda_1 \le correlation(prefix(v), keyword) \le 1 \quad (1)$$

Definition 4: Common relevance: the median correlation between namespaces and keywords which satisfies

$$\lambda_2 \le correlation(prefix(v), keyword) < \lambda_1 \quad (2)$$

Definition 5: Irrelevance: the lower correlation between namespaces and keywords which satisfies

$$-1 \le correlation(prefix(v), keyword) < \lambda_2 \quad (3)$$

In the above equations, we have $0 \le \lambda_2 \le \lambda_1 \le 1$, our ranking algorithm accesses the elements containing the namespaces that satisfy either Equation (1) or Equation (2) rather than Equation (3).

3. Espace Filter Algorithm

In this section, we introduce some preliminary knowledge, followed by presenting our algorithm called the Namespace Filter Algorithm (NFA).

3.1 Preliminaries

The $tf-idf$ weight is commonly used to calculate the term weight in documents in the field of traditional information retrieval. The purpose of our work is to retrieve the appropriate nested elements that contain the relevant text to keywords instead of entire XML documents. So we extend $tf-idf$ to $tf_{t,e}-ief$ for elements in XML documents.

Notations:

$tf_{t,e}$ the number of times that keyword t occurs in the text of element e.

$tf_{t,q}$ the number of times that keyword t occurs in the query q.

$ief = \log_{10}\frac{N}{ef}$ where N is the total number of elements over a collection of XML documents, and ef is the number of elements that contain the keyword. We then give Definition 6 as below.

Definition 6: keyword weights in elements and query

$$W_{t,e} = \begin{cases} (1+\log_{10} tf_{t,e}) \times ief & if \quad tf_{t,e} > 0 \\ 0 & otherwise \end{cases} \quad (4)$$

$$W_{t,q} = (1+\log_{10} tf_{t,q}) \times ief \quad (5)$$

where $W_{t,e}$ is keyword weight in the text of element e, and $W_{t,q}$ keyword weight in a query q. We calculate the cosine similarity between query vector $\vec{q}$ and element vector $\vec{e}$ in Equation (6) on text matching factor.

$$score(q,e) = \frac{\vec{q} \bullet \vec{e}}{|q||e|} = \frac{\sum_{i=1}^{n} q_i e_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} e_i^2}} \quad (6)$$

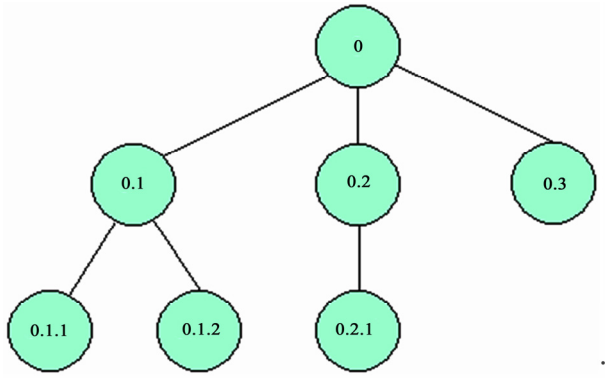

Figure 1. Example of elements with the label ID in the XML document tree

where e_i is the ith keyword weight in $\vec{e}$, and q_i is the ith keyword weight in $\vec{q}$. Their weight values are calculated using Equation (4) and Equation (5), respectively.

In XML documents, elements are of varying size and nested. Since relevant elements can be at any level of granularity, either an element or its children can be relevant to a given query. These facts commonly lead to a problem that the same resulting elements of a query based on keyword search will be presented to users for several times. As an illustration, Consider the structure of an XML document that is shown as the labeled tree in Figure 1.

Besides, let us suppose the relevant element list after keyword search are listed in Table 1.

Elements with ID 0.2.1 and 0.2 are overlapping, so are with 0.1, 0.1.1, and 0.12. If one element's parent is the component of another element, the two relevant components can be merged into one. An element will be merged into its parent only if the number of the keyword occurring in this particular element is less than that of its parent element. In this way, there will be no overlap in the resulting list shown in Table 2.

Furthermore, we denote value[v] calculated by NFA in Section 3.2 as element v. Combining with Definition 2. The final comprehensive evaluation formula about relevant elements ranking is given as Equation (7).

$$rank(v) = a_1 \times correlation(prefix(v), keyword) + a_2 \times value[v] \quad (7)$$

where $a_1 + a_2 = 1$. In order to highlight the factor of namespace's semantic, we have $0 \le a_2 \le a_1 \le 1$.

Table 1. Example of ranked list

Rank	Self	Parent
1	0.2.1	0.2
2	0.2	Root
3	0.1	Root
4	0.1.1	0.1
5	0.1.2	0.1

Table 2. Result list without overlap

Rank	Self	Parent
1	0.2.1	0.2
2	0.1	Root

3.2 NFA Description

In the following discussion, we will focus on presenting the Namespace Filter Algorithm (NFA) and how it performs based on the keyword search over a collection of elements.

Let A be a set consisting of different elements to be accessed by NFA, and the namespaces of elements in set A satisfy either Equation (1) or Equation (2). Other elements not included in A will be neglected by NFA. The length[e] in Equation (6) is defined as $\sqrt{\sum_{i=1}^{n} e_i^2}$.

Value[e] in Figure 2 gives the degree of text matching between the text of element and keywords.

3.3 An Example

To evaluate the effectiveness of NFA, using an example, we perform it with different pair values of λ_1 and λ_2 in Equations (1) and (2). We empirically provide an XML document named as record.xml in Figure 3 which consists of many elements with namespace 'c' describing semantic "computer" and 'n' describing "joy". Let the query be "data and space in algorithm". Meanwhile, we set λ_1 in Equation (1) and λ_2 in Equation (2) to 0.8 and 0.6, respectively. SVD is commonly applied to documents in traditional information retrieval. We extend it to

NFA : retrieve the ranked element based on the keyword

Input:query, a collection of relevant elements denoted as A

Output: top k elements of ranked result list

Description:

```
01    float value[N] = 0//N is the number of elements ∈ A
02    float Length[N]
03    for each keyword t in the query
04      do for each pair(element ∈ A,tf(t,e))
05        do value[e] += W_{t,e} × W_{t,q}  //Equations.(4) and (5)
06      end-for
07    end-for
08    for each element e
09      do value[e] = value[e] / length[e]
10    end-for
11    merge the overlap
12    calculate the rank[] with Equation (7)
13    return top K elements of rank[] over all documents
```

Figure 2. Namespace filter algorithm


```
<root1>
  <c:cs xmlns:c = "http://...../computer">
    <c:DBMS>
    <c:DB>attribute</c:DB>
   <c:DB>Management</c:DB>
  </c:DBMS>
 <c:programming>
    <c:complexity>data and space</c:complexity>
    <c:time>data in computer's Algorithm</c:time>
</c:programming>
<c:java>data of Algorithm in computer science</c:java>
</c:cs>
<n:joy xmlns:n = "http://...../happiness">
  <n:entertainment>
    <n:in>no space with audience's joy</n:in>
    <n:out>jackson    dance in large space</n:out>
  </n:entertainment>
</n:joy>
</root1>
```

Figure 3. Example of record.xml

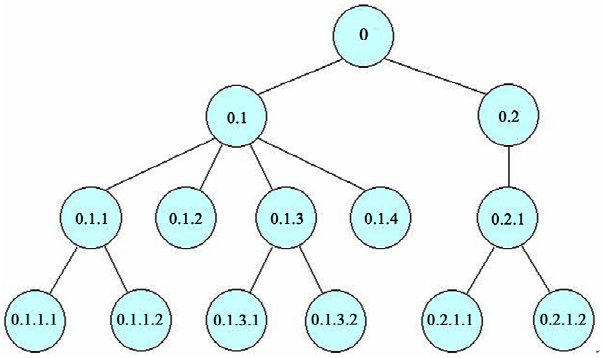

Figure 4. Tree structure of record.xml

Table 3. Term-element matrix M

	0.1.1.1	0.1.1.2	0.1.2	0.1.3.1	0.1.3.2	0.1.4	0.2.1.1	0.2.1.2
Computer	0	0	1	0	1	1	0	0
Data	0	0	1	1	1	1	0	0
Space	0	0	1	1	1	0	1	1
Algorithm	0	0	0	0	1	1	0	0
Joy	0	0	0	0	0	0	1	0

elements in this example.

Each element in record.xml corresponds to a node in the tree with labeled IDs in Figure 4.

Given the correlation value between the semantic meaning of namespaces: 'c', and 'n', and that of the keywords :"data","space", and "Algorithm", we construct a term-element matrix denoted as M, the elements of which are term frequencies occurring over all of elements in record.xml in Table 3.

Then we normalize matrix M denoted by M1 as follows

$$\begin{bmatrix} 0 & 0 & 0.5774 & 0 & 0.5774 & 0.5774 & 0 & 0 \\ 0 & 0 & 0.5774 & 0.7071 & 0.5774 & 0.5774 & 0 & 0 \\ 0 & 0 & 0.5774 & 0.7071 & 0 & 0 & 0.7071 & 1 \\ 0 & 0 & 0 & 0 & 0.5774 & 0.5774 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.7071 & 0 \end{bmatrix}$$

M1 is decomposed into the following three matrixes by SVD

$$U = \begin{bmatrix} -0.4050 & 0.4285 & -0.1839 & 0.4256 & 0.6614 \\ -0.5874 & 0.3361 & 0.3163 & -0.6617 & -0.0637 \\ -0.6474 & -0.6863 & 0.1155 & 0.2915 & -0.1073 \\ -0.2435 & 0.4147 & -0.3754 & 0.3172 & -0.7261 \\ -0.1122 & -0.2458 & -0.8437 & -0.4420 & 0.1403 \end{bmatrix}$$

$$S = \begin{bmatrix} 1.8397 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.3770 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.6569 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.4126 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.3433 & 0 & 0 & 0 \end{bmatrix}$$

$$V = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0.6428 & 0.2843 & -0.7113 \\ 0 & 0 & 0 & 0 & 0 & -0.2946 & -0.7654 & -0.5722 \\ 0.5146 & -0.0328 & -0.2179 & -0.0774 & -0.8250 & 0 & 0 & 0 \\ 0.4746 & 0.1798 & -0.4647 & 0.6345 & 0.3520 & 0 & 0 & 0 \\ 0.3878 & -0.4945 & 0.2135 & -0.1133 & 0.2159 & 0.5 & -0.4082 & 0.2887 \\ 0.3878 & -0.4945 & 0.2135 & -0.1133 & 0.2159 & -0.5 & 0.4082 & -0.2877 \\ 0.2920 & 0.4786 & 0.7839 & 0.2579 & -0.0681 & 0 & 0 & 0 \\ 0.3519 & 0.4984 & -0.1757 & -0.7066 & 0.3124 & 0 & 0 & 0 \end{bmatrix}$$

In the following, we consider the reduced semantic space with two most informative dimensions. Let U1 be first two columns of U, S1 be the diagonal square matrix that contains the first two biggest eigenvalues 1.8397, and 13770 of S as diagonal elements, and other elements in S1 are 0.V1 be the transpose of first two columns of V. We then build up a new term-element matrix M2 by using U1* S1*V1 as below.

$$\begin{bmatrix} 0 & 0 & 0.4024 & 0.2472 & 0.5804 & 0.5804 & -0.0650 & -0.0321 \\ 0 & 0 & 0.5707 & 0.4292 & 0.6475 & 0.6475 & 0.0937 & 0.1492 \\ 0 & 0 & 0.5813 & 0.7346 & -0.0059 & -0.0059 & 0.7997 & 0.8897 \\ 0 & 0 & 0.2490 & 0.1097 & 0.4559 & 0.4559 & -0.1426 & -0.1271 \\ 0 & 0 & 0.0950 & 0.1587 & -0.0874 & -0.0874 & 0.2222 & 0.2413 \end{bmatrix}$$

The correlation values between terms are shown in Table 4a by using $M2 \times M2^T$. We then normalize these values in table 4a to the range of [-1,1] as given in Table 4b.

According to Table 4b, the correlation values between the semantic meanings of namespaces 'c','n' and those of the keywords in the query are given in Table 5.

Consider the keyword search for "data" or "Algorithm" in a query. As shown in Table 5, both the values of correlation of Namespace 'c' vector with "data" and "Algorithm" vector satisfies Equation (1) and Equation (2). In contrast, the correlation value of Namespace 'n' vector and keyword vectors does not satisfy Equations (1) and (2). So we have $\{0.2,0.2.1,0.2.1.1,0.2.1.2\} \not\subset A$. The parameter values (in Section 3.1) of elements in set A are listed in Table 6.

From line 05 to 11 of NFA in Figure 2, combining with Table 6, the value[e] s of elements in set A are shown in Table 7.

As shown in Table 7, there exists the overlap between element 0.1 and other elements. After merging the overlap, the result is 0.1 including its descendent elements 0.1.2,0.1.3,0.1.4 as a whole components. The other resulting element is 0.1.1 including all its descendent elements. Let a_1, and a_2 in Equation (7) be 0.9 and 0.1, respectively, and the correlation value between namespace and keywords be 0.8085, which is the average correlation value between "data" and "Algorithm". Then we can get the final ranked result by using Equation (7) in Table 8.

In order to exploit the relation between λ_1 in Equation (1), and λ_2 in Equation (2) and search result, we assign different pair values to λ_1 and λ_2 such as 0.6 and 0.3. We still give the same query to perform NFA over record.xml. This time we focus on "space" in the query rather than "Algorithm" and "data". Table 5 shows that the correlation value between namespace 'c' vector and "space" vector is 0.3038, which satisfies Equation (2) and namespace 'n' vector and "space" vector is 0.5233, which satisfies Equation (2). Consequently, the search result performed by NFA is given in Figure 6.

As shown in Figures 5 and 6, the different degrees of semantic information relevance between the namespaces and keywords will lead to various search results by using NFA.

In summary, the degree of semantic relevance between the namespace and keywords depends not only on their semantic information similarity, but also on user-specified weights on other factors.

4. Experiments

In our experiments, we compare NFA with other related

Table 4. Correlation value between different pair of terms in record.xml

a

$$\begin{bmatrix} 0.9020 & 1.0765 & 0.3281 & 0.6699 & -0.0462 \\ 1.0765 & 1.3795 & 0.8471 & 0.7473 & 0.0660 \\ 0.3281 & 0.8471 & 2.3087 & -0.0072 & 0.5652 \\ 0.6699 & 0.7473 & -0.0072 & 0.5262 & -0.1010 \\ -0.0462 & 0.0660 & 0.5652 & -0.1010 & 0.1571 \end{bmatrix}$$

b

$$\begin{bmatrix} 0.8352 & 0.9967 & 0.3038 & 0.6203 & -0.0428 \\ 0.9967 & 1.2773 & 0.7843 & 0.6919 & 0.0611 \\ 0.3038 & 0.7843 & 2.1377 & -0.0066 & 0.5233 \\ 0.6203 & 0.6919 & -0.0066 & 0.4872 & -0.0935 \\ -0.0428 & 0.0611 & 0.5233 & -0.0935 & 0.1455 \end{bmatrix}$$

Table 5. Correlation value between semantic of namespace 'c', 'n' vectors and other three keyword vectors over elements in record.xml

Correlation	data	space	Algorithm
computer	0.9967	0.3038	0.6203
joy	0.0611	0.5233	-0.0935

Table 6. Times of "data","space","algorithm" occurring in query and revelant elements of record.xml

Dewey ID	$tf_{t,e}(data)$	$tf_{t,e}(space)$	$tf_{t,e}(a\lg orithm)$
0.1	3	3	2
0.1.2	1	1	0
0.1.3	1	1	1
0.1.3.1	1	1	0
0.1.3.2	0	0	1
0.1.4	1	1	1

Table 7. The ranked result-list with element overlap

Rank	Self	Parent	Value[e]
1	0.1	Root	1.6160
2	0.1.4	0.1	1.5779
3	0.1.3	0.1	1.5779
4	0.1.2	0.1	1.4145
5	0.1.3.1	0.1.3	1.4145
6	0.1.3.2	0.1.3	1

Table 8. Comprehensive ranking using Equation (7)

Rank	Dewey ID	Score
1	0.1	0.8893
2	0.1.1	0.7277

algorithms and methods on two metrics: precision and recall. The result of comparing NFA with the methods that have the similar Precision and Recall on aspect of time execution of algorithm is also presented. We set 0.9 to λ_1, 0.6 to λ_2, 0.9 to a_1, and 0.1 to a_2 in Equation (7) to perform NFA.

4.1 Experimental Setup and Results

Equipment: Our experiments are performed on a PC with a 2.33GHz Intel(R) Core(TM) 2 Duo CPU, 3.25 GB memory, and Microsoft Windows XP. The TermJoin algorithm [11], semantic tree creation algorithm [12], and NFA are all implemented in C++.

Data set: We have tested NFA on two data sets called Dataset1 [13] and Dataset2 [14], respectively. In order to show its performance, we add some namespaces to elements [13]. Each namespace represents the general idea of text embedded in elements [13].

Query set: the query set consists of two parts with 13 queries that represent all kinds of queries over Dataset1 and Dataset2 in Table 9.

4.1.1 Precision and Recall

Precision is defined as the number of relevant elements retrieved by keyword search divided by the total number of elements, while recall refers to as the number of relevant elements retrieved by keyword search divided by the total number of existing relevant elements. We compare the precision and recall of NFA with the Termjoin algorithm [11], semantic tree creation algorithm [12] on Dataset1 and CAS Query [7] on Dataset2. We then calculate the precision and recall of top 20 components retrieved by each algorithm as reported in Figure 7.

As shown in Figure 7, the Term-join algorithm retrieves the relevant elements. However, it also retrieves some non-relevant elements. The basic idea of the Termjoin algorithm is to calculate the degree of text matching of elements with keywords rather than the latent semantic information of text of elements. Furthermore, both NFA and semantic tree creation algorithm efficiently solve the semantic information similarity between text of elements and keyword. However, they do not have the equal running time as given in Section 4.1.2. In [6, 7], authors provide the methods that utilize the semantic

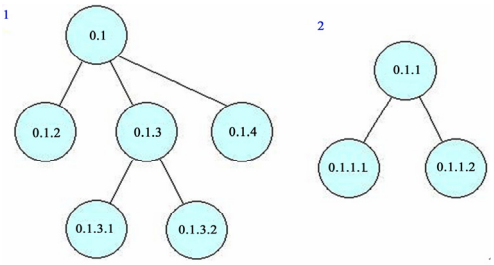

Figure 5. Experimental result elements in record.xml retrieved by NFA with λ_1 in Equation (1) be 0.8 and λ_2 in Equation (2) be 0.6

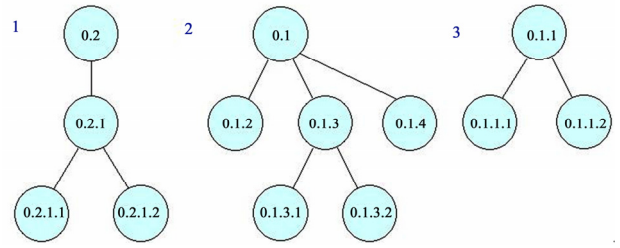

Figure 6. Experimental result elements in record.xml retrieved by NFA with λ_1 in Equation (1) be 0.6 and λ_2 in Equation (2) be 0.3

Table 9. Query set on Dataset 1 and Dataset 2

Dataset1

Q11: pitch step B and octave 2
Q12: natural type
Q13: voice 1 and type eighth
Q14: music with voice 1 staff 1
Q15: music with beam begin and down
Q16: 16th type in music
Q17: 16th type and type of beam
Q18: 16th type and duration 2

Dataset2

Q21: best table in furniture
Q22: best fruit table in furniture
Q23: eat apple at the table
Q24: have coffee at the table
Q25: the list of table

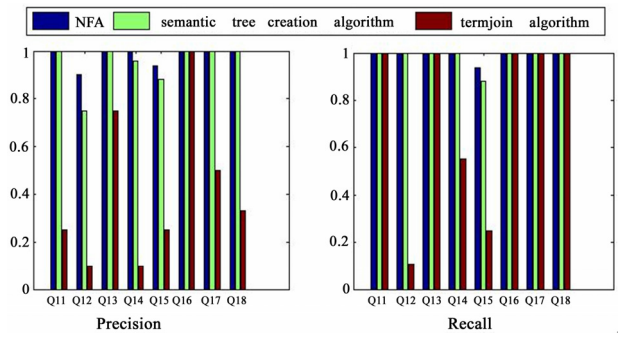

Figure 7. Precision and recall on Dataset1

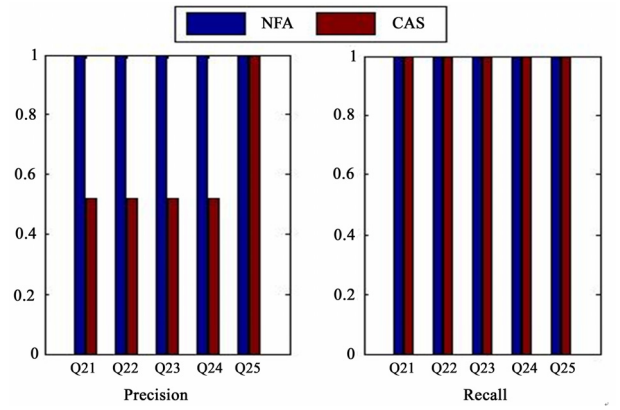

Figure 8. Precision and recall on Dataset2

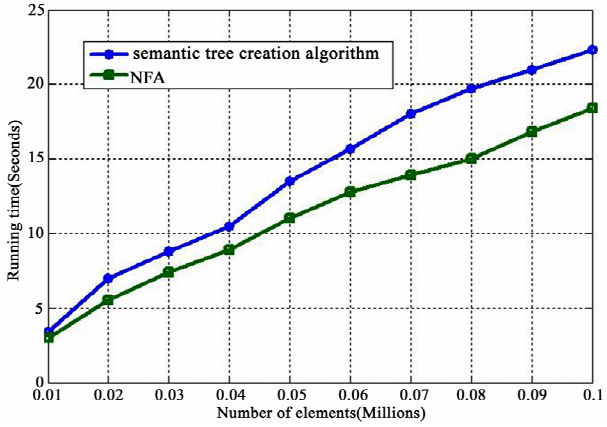

Figure 9. The average running time of NFA and semantic tree creation algorithm over 100 thousand elements from Dataset1 based on the queries from Q11 to Q18 in Table 9

information of markups in elements to calculate the semantic information similarity between the elements and query. However, sometimes it can only get the relevant components with various markups. In order to present the difference of search results of CAS Query [7] and NFA, we test both of them on Dataset2 consisting of elements with namespace 'h' and 'f' nested in the same markup <table>. Both of them are tested by queries from Q21 to Q25 in Table 9 and the precision and recall is shown in Figure 8.

As discussed in Section 2.1, namespace can distinguish different elements even with the same markup which leads to different precision of NFA and CAS in Figure 8.

4.1.2. Running Time of NFA and Semantic Tree Creation Algorithm

In terms of running time in practice, we compare NFA to the semantic tree creation algorithm. We test both of them on Dataset1, and plot the average running time based on the queries from Q11 to Q18 in Table 9 over 100 thousand elements from Dataset1 in Figure 9.

The idea of NFA is to filter the relevant elements with respect to keywords in order to reduce the running time of the semantic tree creation algorithm [12], which accesses all elements in a collection to get the semantic information similarity between the text of elements and keywords. Figure 9 shows that the semantic information of namespace in elements significantly reduces running time compared with the semantic tree creation algorithm over a large collection of elements.

5. Related Work

To be the best of our knowledge, no existing work has formally studied on the namespaces [14] for elements retrieval. There has been a large body of work on content-oriented of XML documents and corresponding ranking schema.

Substantial researches have been done on the area of taking relevant matches between the content and the query as the criteria. e.g., DIL Query processing algorithm [4], Termjoin Algorithm [11] and Top-k algorithm [5]. Jovan Pehcevski *et al*. [15] content that the purpose of XML retrieval task is to find elements that contain as much relevant information. However, some elements that are not keyword matches may be also relevant to the query but not return in those algorithms. The classical method is to calculate the value of consine similarity between the content and keyword utilizing the formula of TF-IEF, the related work have been reported in [6–8,11,16,18,19]. Unfortunately, most of them still cannot accurately calculate the similarity on semantic problem only by this formula. Li Deng *et al*. [12] present the semantic tree creation algorithm. Other proposals are given on semantic problem from the inner structure of XML document (e.g., Hongzhi Wang *et al.* [1]; Norbert Govert *et al.* [20]; Felix Weigel *et al.* [3]; Sihem Amer-Yahia *et al.* [5]; M.S.Ali *et al.* [21]). However, they have to do a large time execution. Benny Kimelfeld *et al.* [5] have observed this shortcoming. They presented the method which filters the relevant documents before processing the algorithm. Due to the notion of methods [22], we interestingly find that the namespace in elements not only solve the latent semantic problems between elements and keyword, but also filter the relevant elements based on the keyword to reduce time execution in the traditional algorithm. The most related work to this paper is [6,11], both of which have proposed the content of markup or frequency of markup as a factor contributed to semantic similarity between the content and query. However, It cannot effectively distinguish the elements with same markup representing different semantic information.

Another related area in elements retrieval is ranking schema based on keyword search. The classical scoring function is tf-ief (e.g., [5,23]) in information retrieval. However, many approaches simply calculate $tf_{t,e}$ with respect to all elements of the collection [9] or partly consider it by estimating $tf_{t,e}$ across elements of the same type [25]. $tf_{t,e}$ is also calculated based on the concentration of the text of the element and that of its descendants [25,26]. A different approach is to compute $tf_{t,e}$ for leaf-elements only, which are then used to score the leaf-elements themselves. All non-leaf elements are scored based on combination of the score of their descendants elements. The propagation of score starts from the leaf elements and can consider the distance between the element being considered and its descendent leaf-elements [27]. Similar notion is adopted by the DIL algorithm [4]. V. Mihajlovi *et al*. [28] rank elements using a utility function that is based not only on the relevance score of an element, but also on its size.

6. Conclusions

This paper addresses the keyword search over elements in XML documents. Using the namespaces of elements, we have presented the Namespace Filter Algorithm (NFA) that retrieves the relevant components of XML documents with respect to keyword queries. In addition, we provide a new approach that can remove effectively the element overlaps occurring in query results. Using an evaluation formula, our approach is able to produce a ranking result-list without element overlaps. Compared with previous algorithms, NFA has demonstrated a better performance not only on time execution but also on the precision and recall of query results. Our future work will study the relation of the previous factors on the background of graph structures in XML documents.

7. Acknowledgments

We are grateful to the anonymous reviewers for their helpful comments.

REFERENCES

[1] H. Z. Wang, J. Z. Li, W. Wang, and X. M. Lin, "Coding-based join algorithm for structure queries on graph-structured XML document," World Wide Web, Vol. 11, pp. 485–510, 2008.

[2] N. Govert, G. Kazai, N. Fuhr, and M. Lalmas, "Evaluating the effectiveness of content-oriented XML retrieval," Information Retrieval, Vol. 9, No. 6, pp. 699–722, 2006.

[3] F. Weigel, H. Meuss, and K. U. Schulz, "Francois bry content and structure in indexing and ranking XML," WebDB, Vol. 17–18, pp. 68–72, June 2004.

[4] G. Lin, S. Feng, C. Botev, and J. Shanmugasundaram, "XRank: Ranked keyword search over XML documents," ACM International Conference Proceeding, SIGMOD, pp. 7–11, June 9–12, 2003.

[5] S. A. Yahia, N. Koudas, A. Marian, D. Srivastava, and D. Toman, "Structure and content scoring for XML," Proceedings of the 31st VLDB Conference, pp. 362–372, 2005.

[6] T. S. Kim, J. H. Lee, J. W. Song, and D. H. Kim, "Similarity measurement of XML documents based on structure and contents," International Conference on Computational Science (ICCS), Part 3, LNCS 4489, pp. 902–905, 2007.

[7] M. Izabel, M. Azevedo, L. P. Amorim, and N. Ziviani, "A universal model for XML information retrieval," LNCS pp. 312–318, 2005.

[8] Y. T. Zhang, L. Gong, and Y. C. Wang, "An improved TF-IDF approach for text classification," Journal of Zhejiang University Science, Vol. 6A, No. 1, pp. 49–55, 2005.

[9] S. C. Haw and C. S. Lee, "TwigINLAB: A decomposition-matching-merging approach to improving XML query processing," American Journal of Applied Sciences, Vol. 5, No. 9, pp. 1199–1205, 2008.

[10] C. D. Manning, P. Raghavan, and H. Schutze," Introduction to information retrieval," Cambridge Press, April, 2008.

[11] S. Al-Khalifa, C. Yu, and H. V. Jagadish, "Querying structured text in an XML database," ACM International Conference Proceeding, SIGMOD, June 9–12, 2003.

[12] D. Li, X. J. Wang, and L. H. Wang, "Indexing temporal XML using semantic tree index," IEEE Xplore, pp. 448–451, 2008.

[13] http://www.musicxml.org/xml/elite.xml.

[14] Namespaces in XML Available: http://www.w3schools.com/XML/xml_namespaces.asp/.

[15] J. Pehcevski and J. A. Thom, "HixEval: Highlighting XML retrieval evaluation," LNCS 3977, pp. 43–57, 2006.

[16] T. S. Kim, J. H. Lee, J. W. Song, and S. L. Lee, "Semantic structural similarity for clustering XML documents," Inha University Technical Report, 2006. http://webbase.inha.ac.kr/TechnicalReport/tech_04.pdf.

[17] C. Yang and N. Liu, "Measuring similarity of semi-structured documents with context weights," ACM International Conference Proceeding, pp. 719–720, August 6–11, 2006.

[18] S. Feng, G. Lin, C. Botev, and J. Shanmugasundaram, "Efficient keyword search over virtual XML views," VLDB, pp. 1057–1065, September 23–28, 2007.

[19] K. Sauvagnat, L. Hlaoua, and M. Boughanem, "XML retrieval: What about using contextualrelevance?" SAC, pp. 1114–1115, April 23–27, 2006.

[20] B. Jeong, D. Lee, H. Cho, and J. Lee, "A novel method for measuring semantic similarity for XML schema matching," Expert Systems with Applications, Vol. 24, pp. 1651–1658, 2008.

[21] M. S. Ali, M. P. Consens, and M. Lalmas, "Structural relevance in XML retrieval evaluation," Proceedings of the SIGIR Workshop on XML and Information Retrieval, pp. 2–8, July 27, 2007.

[22] B. Kimelfeld, E. Kovacs, Y. Sagiv, and D. Yahav, "Using language models and the HITS algorithm for XML retrieval," LNCS 4518, Springer-Verlag Berlin Heidelberg, pp. 253–260, 2007.

[23] C. Botev and J. Shanmugasundaram, "Context-sensitive keyword search and ranking for XML," WebDB, 2005.

[24] B. Sigurbjornsson, J. Kamps, and M. de Rijke, "The effect of structured Queries and selective Indexing on XML retrieval," INEX'05, LNCS 3977, pp. 104–118, 2006.

[25] M. Theobald, R. Schenkel, and G. Weikum, TopX & XXL at INEX 2005.

[26] P. Ogilvie and J. Callan, "Parameter estimation for a simple hierarchical generative model for XML component retrieval," INEX, 2004.

[27] S. Geva, "GPX-gardens point XML," IR at INEX 2005.

[28] V. Mihajlovic, G. Ramirez, T. Westerveld, D. Hiemstra, H. E. Blok, and A. P. de Vries, "Vague element selection, image search, overlap, and relevance feedback," INEX 2005.

