論文 / 著書情報
Article / Book Information

論題(和文)	A Hybrid Approximate XML Subtree Matching MethodUsing Syntactic Features and Word Semantics
Title(English)	A Hybrid Approximate XML Subtree Matching MethodUsing Syntactic Features and Word Semantics
著者(和文)	梁文新, 横田治夫
Authors(English)	Wenxin Liang, Haruo Yokota
掲載誌(和文)	DEIM Forum 2009論文集
Citation(English)	Proc. of DEIM Forum 2009
Vol, no, pages	, ,
発行日 / Pub. date	2009, 3

A Hybrid Approximate XML Subtree Matching Method Using Syntactic Features and Word Semantics

Wenxin LIANG[†,††] and Haruo YOKOTA[††]

† CREST, Japan Science and Technology Agency
5 Sanbancho, Chiyoda-ku, Tokyo 102–0075, Japan
†† Global Scientific Information and Computing Center, Tokyo Institute of Technology
2–12–1 Ookayama, Meguro-ku, Tokyo 152–8552, Japan
E-mail: wxliang@de.cs.titech.ac.jp, yokota@cs.titech.ac.jp

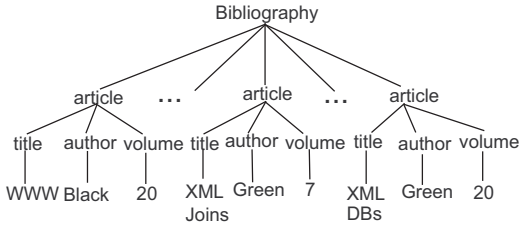
Abstract With the exponential increase in the amount and size of XML data on the Internet, XML subtree matching has become important for many application areas such as change detection, keyword retrieval and knowledge discoveries over XML documents. In our previous work, we have proposed leaf-clustering based approximate XML subtree matching methods using syntax information of both the clustered leaf nodes and the corresponding paths. In this paper, we propose a hybrid subtree matching method, in which subtree matching is determined by using the word semantics based on WordNet thesaurus in leaf nodes and the syntactic features in the relevant paths. We also propose a one-pass hash join technique to reduce the additional join cost caused by the extra words expanded by the WordNet. We perform experiments to evaluate performance and matching precision and recall comparing the hybrid method with the original syntax-based methods. The experimental results indicate that the proposed hybrid method with one-pass hash join, comparing with the existing path-based *SLAX* algorithm, can effectively improve the precision and recall with about only 5% increase of the execution time for the leaf-clustering based subtree matching.

Key words XML Subtree Matching, Syntactic Feature, Word Semantics, WordNet

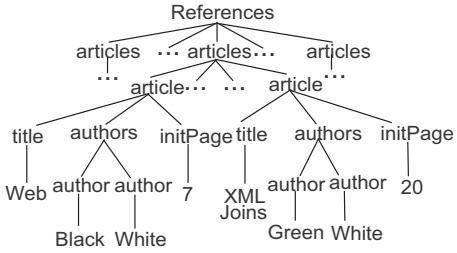
1. Introduction

XML has increasingly become a wildly popular standard and employed as a key technique for representing, exchanging and integrating data on the Internet, because it is portable for representing different types of data from multiple sources. Recently, a large number of data, for example free online encyclopedias such as Wikipedia [10], life science data such as Swiss-Prot [8], and bibliography data such as DBLP [12], are disseminated and exchanged on the Internet in the form of XML documents. With the exponential increase in the amount and size of XML data, XML subtree matching has become important for many application areas such as change detection, keyword retrieval and knowledge discoveries over XML documents. However, XML subtree matching is not an easy task because XML documents from different data sources representing nearly or exactly the same information may be constructed by different structures. Besides, even two XML documents contain the same information, each of them may have some extra information that the other does not do.

[Example 1] Figure 1 shows an example of two XML document trees with different DTDs. Although the two XML document trees have different structures, they represent very similar information. Besides, each document has some infor-



(a)



(b)

Figure 1 Example XML document trees

mation that the other does not do. For example, `volume` in Figure 1 (a); and `initPage` in Figure 1 (b).

In previous work, we have proposed leaf-clustering based approximate XML subtree matching methods using syntax features of the clustered leaf nodes [6] and the corresponding paths [7]. However, focusing on the syntactic features without considering the semantic similarity between words achieves high performance but may degrade matching precision and recall due to the indistinguishable matching problem. To solve the indistinguishable matching problem and achieve reasonable trade-off between performance and effectiveness, in this paper we propose a hybrid subtree matching method, in which subtree matching is determined by the word semantic similarity using WordNet thesaurus [11] of clustered leaf nodes and the syntactic features in the relevant paths. We perform experiments to evaluate the performance and matching precision and recall comparing the proposed hybrid method with the original ones, and the experimental results indicate that the proposed method can improve the matching precision and recall but takes more execution time.

The rest of the paper is organized as follows. Section 2 briefly introduces the related researches. Section 3 describes our previously proposed syntax-based methods and the indistinguishable matching problems. In Section 4, we propose the hybrid subtree matching method and discuss the one-pass join technique. Section 5 describes the experimental evaluation of the proposed method comparing with the original methods. Finally, Section 6 concludes this paper.

2. Related Work

XML subtree matching in RDBs is often applied in the area of change detection on XML documents [3, 9]. Wang et al. [9] proposed X-Diff, an effective change detection algorithm based on an unordered tree model. However, X-Diff cannot be applied to large-scale XML documents because of main memory limitation. Leonardi et al. [3] proposed schema-conscious approaches which can be applied to larger XML documents with better performance than X-Diff. However, the schema-conscious methods need the DTDs of the XML documents. In [6], we proposed *SLAX*, a syntax-based subtree matching algorithm, which is DTD-independent and cost-effective with focusing only on the syntax of clustered leaf nodes. However, *SLAX* sometimes cause imprecise matching problems that may degrade the effectiveness of subtree matching. To solve the imprecise matching problems, in [7] we proposed path-based *SLAX*, which utilizes the syntactic features of both the path information and the clustered leaf nodes. However, focusing on the syntactic feature may degrade matching precision and recall due to the indistinguishable matching problem which will be described in the next section.

3. Syntax-based Method

In this section, we briefly introduce our previously proposed syntax-based approximate XML subtree matching algorithms using syntax features of the clustered leaf nodes [6] and the corresponding paths [7], and the indistinguishable matching problems occurred in the syntax-based methods.

3.1 SLAX and Path-based SLAX

In [6], we have proposed the leaf-clustering based XML integration methods, in which the two XML documents to be joined are firstly segmented into independent meaningful subtrees [4]. Then, the subtree matching is determined by SSD (*Subtree Similarity Degree*) as shown in the following equation [5, 6], where n and n_{bi} denote the number of matched leaf nodes and the number of leaf nodes in the base subtree t_{bi}, respectively.

$$SSD(t_{bi}, t_{tj}) = \frac{n}{n_{bi}} \times 100\ (\%) \tag{1}$$

However, focusing only on the leaf nodes sometimes causes imprecise matching problems that may degrade the precision and recall of subtree matching. To solve the imprecise matching problems, in [7] we have proposed path-based *SLAX*, in which the approximate similarity between two segmented subtrees is determined based on the Path-based Subtree Similarity Degree (PSSD) which is defined as follows(1).

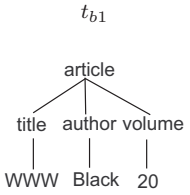
[Definition 1] ***(Path-based Subtree Similarity Degree (PSSD))*** For a pair of matched subtrees $T_M(i)$, assume the number of matched leaves is K, the full-path based subtree similarity degree $PSSD(t_{bi}, t_{tj})$ is determined by Equation 2, where $PS(i)$ is the path similarity determined by the number of matched tags out of the total number of tags in the path.

$$PSSD(t_{bi}, t_{tj}) = SSD(t_{bi}, t_{tj}) \times \frac{\sum_{i=1}^{K} PS(i)}{K} \tag{2}$$

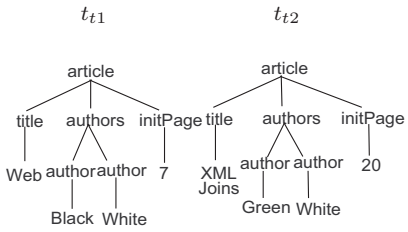
3.2 Indistinguishable Matching Problem

Focusing on the syntactic features without considering the semantic similarity between words achieves high performance but may degrade matching precision and recall due to the indistinguishable matching problem. Figure 2 shows an example of indistinguishable matching occurred in the syntax-based methods. According to the definition of SSD, subtree t_{b1} matches both target subtrees t_{t1} and t_{t2} because of the same $SSD = 33.33\%$; that is, the SSD cannot distinguish which pair of subtrees are more semantically relevant only by using the syntax information. However, since the word "WWW" is synonymous with the word "Web", the target

(1) : Note that the path-based *SLAX* in this paper only corresponds to the full-path based method using path information after matching leaf nodes proposed in [7].



(a) Base subtree



(b) Target subtrees

Figure 2 Example of indistinguishable matching

subtree t_{t1} is considered to be more semantically relevant to the base subtree t_{b1}.

4. Hybrid Subtree Matching Method

4.1 Hybrid Similarity Degree

WordNet [11] is one of most common used lexical database of English. Words are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets, interlinked by means of conceptual-semantic and lexical relations, are the general components for describing word semantics. We use the synsets returned by Java WordNet API [2] to determine the semantic similarity between the words in the leaf nodes. Given two words w_1 and w_2, assume w_1 and w_2 contain m and n synonyms $s_1(i)$ $(1 \leqq i \leqq m)$ and $s_2(j)$ $(1 \leqq j \leqq n)$ from the synsets of WordNet(2), the semantic similarity between the two words $\mathcal{S}(w_1, w_2)$ can be determined by the following equation, where σ $(0 < \sigma \leqq 1)$ is a user-defined constant(3).

$$\mathcal{S}(w_1, w_2) = \begin{cases} \sigma, & \text{if any } s_1(i) = s_2(j) \\ 0, & otherwise \end{cases}$$

In order to solve the indistinguishable matching problem, we propose a hybrid method using both the syntactic features and the word semantics. We first use WordNet thesaurus to determine the semantic similarity between leaf nodes, and then consider the syntactic features in the relevant paths. In the proposed method, the similarity between two subtrees are determined by the Hybrid Similarity Degree (HSD) which is defined as follows.

(2) : Note that if there are no synonyms returned from WordNet, $s_1 = w_1$ and $s_2 = w_2$.

(3) : For the sake of simplicity, we set $\sigma = 1$ in this paper.

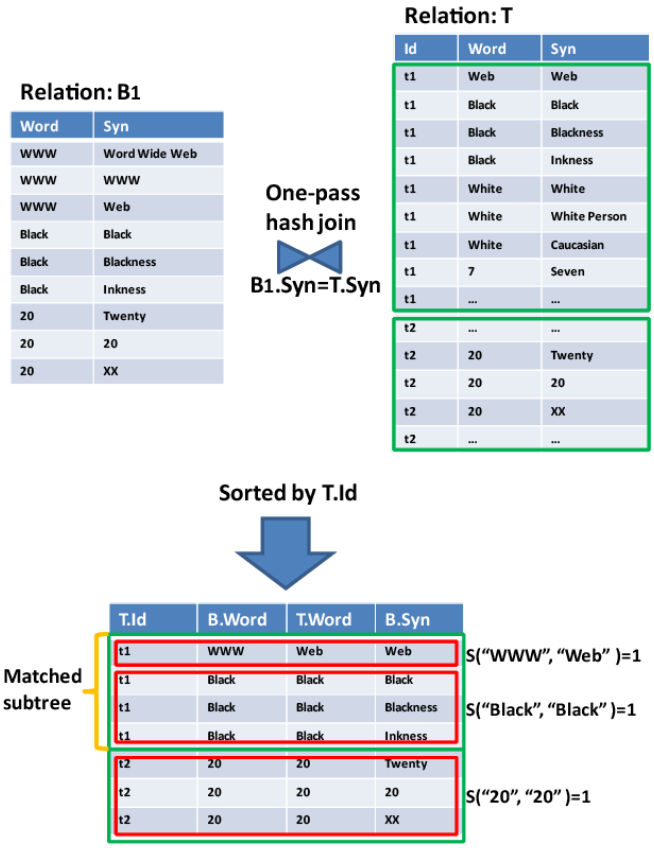


Figure 3 Example of join process

[Definition 2] ***(Hybrid Similarity Degree (HSD))*** For a pair of subtrees t_b and t_t, assume there are m and n words $w_b(i)$ and $w_t(j)$ in the leaf nodes of t_b and t_t is m and n, the hybrid similarity degree between t_b and t_t, $HSD(t_b, t_t)$ is determined by Equation 3, where $PS(k)$ and K are the same as those in Definition 1.

$$HSD(t_b, t_t) = \frac{\sum_{i=1}^{m} Max\{\mathcal{S}(w_b(i), w_t(j))\}}{m} \times \frac{\sum_{k=1}^{K} PS(k)}{K} \quad (3)$$

4.2 One-pass Join

In order to reduce the the additional join cost caused by the extra words expanded by the WordNet, we propose a one-pass hash join technique instead of joining each pair of base and target subtrees one by one. In the one-pass hash join, the words and their corresponding synonyms in the leaf nodes of each base subtree are stored into a relation table B_i and those in the leaf nodes of all the target subtrees together with their IDs are stored into a relation table T. During the join process, once a join base table B_i is built into memory by a hash function, the join table T containing all the target subtree information can be one-pass probed to achieve high-speed subtree matching.

Table 1 Experimental environment

CPU	Intel Core 2 Processor E6850 3.0 GHz
Memory	2.0 GB
Hard Disk	Seagate ST336607LC 37GB
OS	Linux 2.6.9
DBMS	PostgreSQL 8.1.3
Java	Sun JDK 1.6.0
Lexical Database	WordNet 2.1
Java API	MIT JWI 2.1.5

Figure 3 illustrates the process of joining base subtree t_{b1} with target ones t_{t1} and t_{t2} in Figure 2. Firstly, the synonyms of the words in the leaf nodes of t_{b1}, t_{t1} and t_{t2} are obtained from the synsets of WordNet, and they are stored together with the original words into two relational tables B_1 and T (Table T also stores the subtree ID information), respectively. Next, the two tables are hash joined by the condition that $B_1.Syn = T.Syn$. Finally, the matched subtree for the base subtree t_{b1} can be determined based on Equation (3) using the join results sorted by $T.Id$. In this example, according to Equation (1) and (3), $SSD(t_{b1}, t_{t1}) = SSD(t_{b1}, t_{t2}) = 33.3\%$, while $HSD(t_{b1}, t_{t1}) = 66.7\% > HSD(t_{b1}, t_{t2}) = 22.2\%$. This means the hybrid method can effectively solve the indistinguishable matching problem. Besides, since $HSD(t_{b1}, t_{t1}) = 66.7\% > PSSD(t_{b1}, t_{t2}) = SSD(t_{b1}, t_{t1}) = 33.3\%$, the hybrid method is more effective to evaluate the semantic similarity between subtrees than the syntax-based methods.

5. Experiments

5.1 Experimental Setup

The experiments were performed under the environment shown in Table 1. We used the XML version of SIGMOD Record [1], named sigmod.xml (482KB, about 20,000 nodes), and we segmented the XML version of DBLP [12] into 955 fragment documents, named dblp1∼955.xml (300KB each, about 15,000 nodes). We parsed the sigmod.xml and dblp290.xml into relational tables and then segmented them into independent meaningful subtrees(4).

5.2 Experimental Results

After the XML documents are stored and segmented into independent meaningful subtrees, we used the original algorithm *SLAX*, the path-based *SLAX*, and the proposed hybrid method to execute the subtree matching for the sigmod.xml and dblp290.xml with four different matching threshold from 0.1 to 0.4. The lexical database and Java API used in the hybrid method are WordNet 2.1 [11] and MIT Java Wordnet Interface JWI 2.1.5 [2]. The number of segmented subtrees

(4) : Refer to Reference [4] for the detailed data schema and subtree segmentation algorithms.

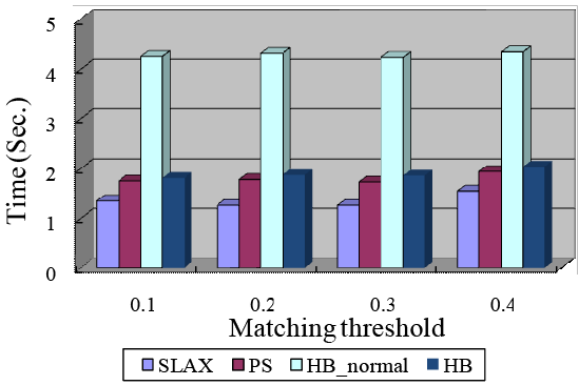


Figure 4 Time for subtree matching

of the two documents are 1504 and 698, and the number of correct answer is 142. In order to evaluate the effectiveness of subtree matching, we define *precision* and *recall* as follows.

[Definition 3] ***(Precision)*** The precision of subtree matching ($\mathcal{P}$) is the percentage of the number of correctly selected hit subtrees ($\mathcal{N}_s$) out of the total number of hit subtrees ($\mathcal{N}_h$).

$$\mathcal{P} = \frac{\mathcal{N}_s}{\mathcal{N}_h} \times 100\ (\%) \tag{4}$$

[Definition 4] ***(Recall)*** The recall of subtree matching ($\mathcal{R}$) is the percentage of the number of correctly selected hit subtrees ($\mathcal{N}_s$) out of the total number of correct answer($\mathcal{N}_c$).

$$\mathcal{R} = \frac{\mathcal{N}_s}{\mathcal{N}_c} \times 100\ (\%) \tag{5}$$

Figure 4 shows the execution time using *SLAX*, path-based *SLAX* (PS for short), the hybrid method using traditional join (HB_normal for short), and the hybrid method using one-pass hash join (HB for short). Figure 5 and Figure 6 show the precision and recall of subtree matching using *SLAX*, PS and HB methods. According to the experimental results, we can make the following discussions:

- Comparing with the PS method, the HB_normal method increases 138.30% of execution time on the average, because the extra words expanded by the WordNet increases the number of comparison in the join process. While, the hybrid method with one-pass hash join only increases 5.29% of execution time on the average, which indicates that using the one-pass hash join can effectively reduce the join cost even the number of the words to be compared becomes much larger due to the utilization of the WordNet thesaurus.
- Comparing with *SLAX*, the precision increases 19.68% and 20.37% on the average when using PS and HB, respectively. Figure 5 shows that HB is superior to PS when the matching threshold is greater than 0.2. However, PS achieves slightly better precision when the matching threshold is less than 0.2, because using the word semantics in HB generates

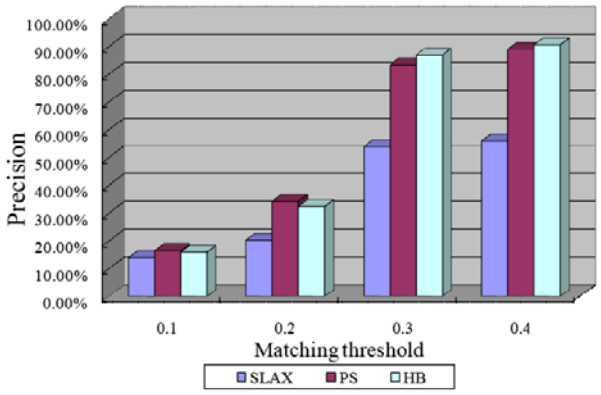


Figure 5 Precision of subtree matching

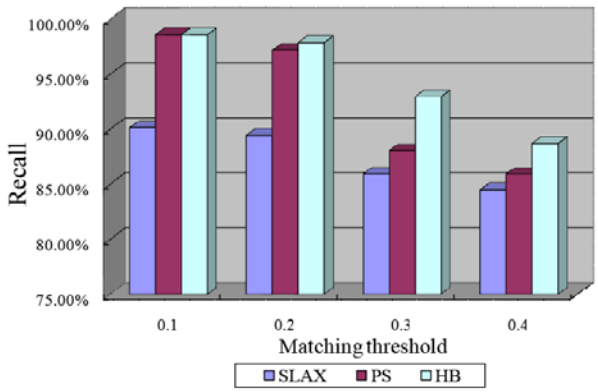


Figure 6 Recall of subtree matching

larger number of hit subtrees than PS does(5).

- Comparing with *SLAX*, the recall increases 2.09% and 4.93% when using PS and HB, respectively. The hybrid method achieves better recall than the *SLAX* and PS methods, particularly when the matching threshold becomes larger than 0.2.

From the above results, we can learn that the proposed hybrid method with one-pass hash join, comparing with the path-based *SLAX* algorithm, can effectively improve the precision and recall with about only 5% increase of the execution time for the leaf-clustering based subtree matching, particularly when the matching threshold becomes larger.

6. Conclusions

In this paper, we proposed a hybrid XML subtree matching method, in which subtree matching is determined by using word semantics in leaf nodes and the syntactic features in the relevant paths, to solve the indistinguishable matching problem in syntax-based approximate XML subtree matching algorithms. We also proposed a one-pass hash join technique to reduce the additional join cost caused by the extra words expanded by the WordNet.

We performed experiments to evaluate the trade-off between performance and matching precision and recall comparing the proposed hybrid method with the original ones, and the experimental results indicate that the proposed hybrid method with one-pass hash join, comparing with the path-based *SLAX* algorithm, can effectively improve the precision and recall with about only 5% increase of the execution time for the leaf-clustering based subtree matching, particularly when the matching threshold becomes larger.

(5) : Another reason is that the XML documents used in our experiments contain many computer terms, "XML" for example, that are not included in the WordNet database. Acturally, in the future we plan to conduct further evaluation on the effectiveness of using WordNet by more experiments using multiple kinds of XML data.

Acknowledgment

This work was partially supported by CREST of JST (Japan Science and Technology Agency), and by the Grant-in-Aid for Scientific Research of MEXT Japan #19024028.

References

[1] ACM SIGMOD Record in XML. Available at http://www.acm.org/sigmod/record/xml/.
[2] MIT Java WordNet Interface. Available at http://projects.csail.mit.edu/jwi/.
[3] Erwin Leonardi and Sourav S. Bhowmick. Detecting Changes on Unordered XML Documents Using Relational Databases: A Schema-conscious Approach. In *Proc. of CIKM*, pages 509–516, 2005.
[4] W. Liang, X. Ouyang, and H. Yokota. An XML Subtree Segmentation Method Based on Syntactic Segmentation Rate. In *Proc. of ADSS*, pages 551–558, 2007.
[5] W. Liang and H. Yokota. *LAX*: An Efficient Approximate XML Join Based on Clustered Leaf Nodes for XML Data Integration. In *Proc. of BNCOD*, pages 82–97, 2005.
[6] W. Liang and H. Yokota. *SLAX*: An Improved Leaf-Clustering Based Approximate XML Join Algorithm for Integrating XML Data at Subtree Classes. *IPSJ Trans. on Databases*, 47(SIG8(TOD30)):47–57, 2006.
[7] Wenxin Liang and Haruo Yokota. Exploiting Path Information for Syntax-based XML Subtree Matching in RDBs. In *Proc. of WAIM*, pages 105–112, 2008.
[8] Swiss-Prot. http://www.ebi.ac.uk/swissprot/.
[9] Y. Wang, D. J. DeWitt, and J. Cai. X-Diff: An Effective Change Detection Algorithm for XML Documents. In *Proc. of ICDE*, pages 519–530, Match 2003.
[10] Free Encyclopedia: Wikipedia. http://wikipedia.org/.
[11] WordNet. http://wordnetweb.princeton.edu/.
[12] XML Version of DBLP. Available at http://dblp.uni-trier.de/xml/.