Subtree-based XML Data Integration Using Leaf-clustering Based Approximate XML Join Algorithms

Wenxin LIANG♥ Haruo YOKOTA♠

In this paper, we discuss the subtree-based XML data integration using the leaf-clustering based approximate XML join algorithms (*LAX* and *SLAX*) for the fragments divided from large XML documents. We propose a subtree-based integration method for integrating the matched subtree pairs determined by both *LAX* and *SLAX*. We conduct experiments to compare the effectiveness of the subtree-based integration for real large bibliography XML documents using *LAX* and *SLAX*, respectively. We show the integration result for a sample subtree from the XML document of SIGMOD record. The experimental results show that the previous proposed algorithms are effective for the subtree-based XML data integration.

1. Introduction

XML has become the *de facto* standard for representing and exchanging data over the World Wide Web. Recently, a large number of data, for instance bibliography data such as DBLP [11] and ACM SIGMOD Record [1], and bioinformatics data such as TrEMBL [9] and Swiss-Prot [8], are published and exchanged by XML on the Internet. However, XML documents from different data sources may have nearly or exactly the same information but may be different on structures. Besides, even the two XML documents contain the same data, both of them may have some extra information what the other does not do. Fig.1 shows two example XML documents. These two documents are structurally different because of different DTDs, but they represent very similar information. In addition, each of them has some information what the other does not do. For example, *endPage* in Fig. 1 (a) and *year* in Fig. 1 (b).

In previous work [5], we have proposed *LAX* (Leaf-clustering based Approximate XML join algorithm) for measuring the approximate similarity between XML documents. In order to compare the work based on the tree edit distance [3, 4], the output of *LAX* is oriented to the pair of documents that has larger tree similarity degree than the user-defined threshold [5]. However, large XML documents must be divided into small fragments when they are too large to be loaded into the main memory. In this case, the target subtrees are distributed in each fragment document. Therefore, when we apply *LAX* to such fragment documents, the matched subtrees selected from the output pair of fragment documents that has large tree similarity degree might not be the proper one that should be integrated. To solve this problem, we have proposed an improved algorithm, *SLAX* (Subtree-class Leaf-clustering based Approximate XML join algorithm) in [6].

In this paper, we propose a subtree-based XML data integration method for integrating the matched subtree pairs determined by both *LAX* and *SLAX*. We conduct experiments to compare the effectiveness of the subtree-based integration for real large bibliography XML documents using *LAX* and *SLAX*, respectively. And we show the integration result for a sample subtree from the XML document of SIGMOD record. The experimental results indicate that the previous proposed algorithms are effective for the subtree-based XML data integration.

The remainder of this paper is organized as follows. Section 2 briefly reviews the work related to this paper. Section 3 shortly introduces the basic knowledge of the leaf-clustering based approximate XML join algorithms. In Section 4, we propose the subtree-based integration method. In Section 5, we do experiments using real large bibliography documents and show the integration result. Finally, Section 6 concludes this paper.

2. Related Work

A well formed XML document can be parsed into an ordered labeled tree [10]. The edit distance between two ordered labeled tree is defined as the minimum cost edit operation (insertions, deletions and substitutions) required to transform one tree to another [12]. The tree edit distance is regarded as an effective metric for measuring the structural similarity in XML documents [2, 3, 7]. However, the computational cost is extremely expensive; in the worst case, it is an $O(n^4)$ operation for the document of size n.

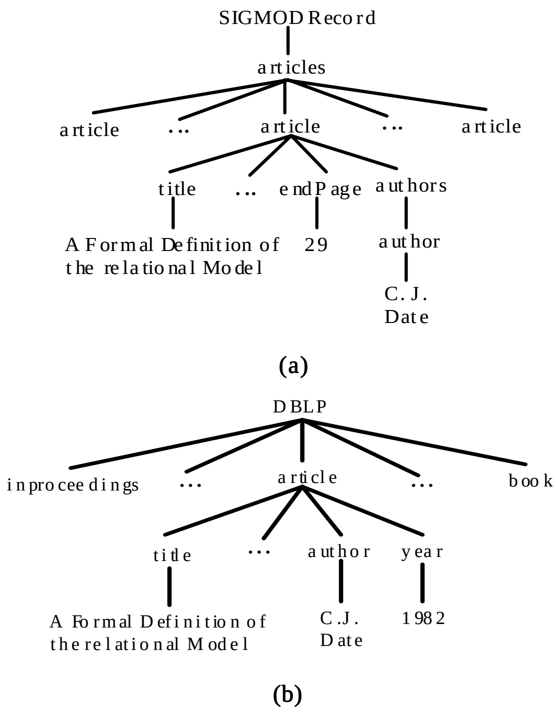


Fig.1 Example XML Documents

♥ Student Member Graduate School of Information Science and Engineering, Tokyo Institute of Technology wxliang@de.cs.titech.ac.jp
♠ Regular Member Global Scientific Information and Computing Center, Tokyo Institute of Technology yokota@cs.titech.ac.jp

S. Guha, et al. proposed the lower and upper bound as inexpensive substitutions for the tree edit distance operation [2]. However, when the upper bound is greater than the threshold τ and the lower bound is less than τ, the tree edit distance will still must be calculated.

3. Preliminaries

3.1 LAX

Let T_b and T_t (b denotes base and t denotes target) be two XML document trees. Assume T_b and T_t are segmented into k_b and k_t subtrees $t_{bi}(1 \le i \le k_b)$ and $t_{tj}(1 \le j \le k_t)$, respectively. The *subtree similarity degree* between t_{bi} and t_{tj}, $SSD(t_{bi}, t_{tj})$ is defined by Equation (1), where n and n_i denote the number of matched leaf nodes (the pair of leaf nodes that has the same PCDATA value) and the total number of leaf nodes in the base subtree .

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

The *matched subtree*, $T_M[i]$ is defined as the pair of subtrees that has the maximum subtree similarity degree in a join loop. The similarity degree of $T_M[i]$, $S_M[i]$ can be calculated by Equation (2).

$$S_M[i] = Max_{j=1}^{k_t}(SSD(t_{bi}, t_{tj})) \quad (2)$$

The *tree similarity degree* between T_b and T_t, $TSD(T_b, T_t)$ is determined by the following equation based on the mean value of the similarity degrees of matched subtrees.

$$TSD(T_b, T_t) = \frac{\sum_{i=1}^{k_b} S_M[i]}{k_b} \times 100\% \quad (3)$$

For two XML data sources S_b and S_t, assume each document $d_b \in S_b$ and $d_t \in S_t$ are parsed into XML document trees T_b and T_t. Let T_b and T_t be segmented into k_b and k_t subtrees t_b and t_t. Given a user-defined threshold τ, *LAX* will output the pair of documents that has larger tree similarity degree than the threshold τ in each join loop. It is an $O(n^2)$ operation in the worst case for the document of size n. The details of *LAX* are available in [5].

3.2 SLAX

When the XML documents are too large to be loaded into the main memory, they must be segmented into small fragments. Thus, when *LAX* is applied to such fragment documents, a mismatching problem sometimes may occur. Namely, the matched subtree selected from the output pair of documents that has larger tree similarity degree might not be the most similar one as the base one.

In [6], we have proposed *SLAX* to solve the mismatching problem in *LAX*. For two XML data sources S_b and S_t, assume each document $d_{bm} \in S_b$ and $d_{tn} \in S_t$ are parsed into XML document trees T_{bm} $(1 \le m \le K)$ and T_{tn} $(1 \le n \le L)$. Let T_{bm} and T_{tn} be segmented into k_b and k_t subtrees $t_{bmi}(1 \le i \le k_b)$ and $t_{tnj}(1 \le j \le k_t)$.

The *match value*, $M[n]$ for the base subtree t_{bmi} and each target tree T_{tn} $(1 \le n \le L)$ is defined by the following equation.

$$M[n] = Max_{j=0}^{k_t}(SSD(t_{bmi}, t_{tnj})) \quad (4)$$

The *maximum match value*, $MMV[i]$ for the base subtree t_{bmi} is defined as followings.

$$MMV[i] = Max_{n=0}^{L}(M[n]) \quad (5)$$

Given a user-defined threshold T, *SLAX* will output the matched subtree pairs whose maximum match values are larger or equal to the threshold T. The details of *SLAX* are available in [6].

4. Subtree-based XML Data Integration

In this section, we will propose a subtree-based XML data integration method for integrating the matched subtree pairs selected by *LAX* and *SLAX*.

For a matched subtree pair t_{bi} and t_{tj}, assume t_{bi} and t_{tj} have n_b and n_t leaf nodes, respectively.

Definition 1 (Matched Leaf). *For a pair of leaf nodes L_{bi} in t_{bi} and L_{tj} in t_{tj}, they are matched leaves, iff L_{bi} and L_{tj} have the same PCDATA value.*

Definition 2 (Nonmatched Leaf). *For a leaf node L_{bi} in t_{bi} or L_{tj} in t_{tj}, it is a nonmatched leaf, iff there is not any leaf node in t_{bj} or t_{ti} that has the same PCDATA value as L_{bi} or L_{tj}.*

Definition 3 (Insert Operation). *For a nonmatched leaf node L_{tj} in t_{tj}, the insert operation, ins (L_{tj}, t_{bi}) inserts the whole path from the child node of R_t to L_{tj} into t_{bi} below its root node R_b.*

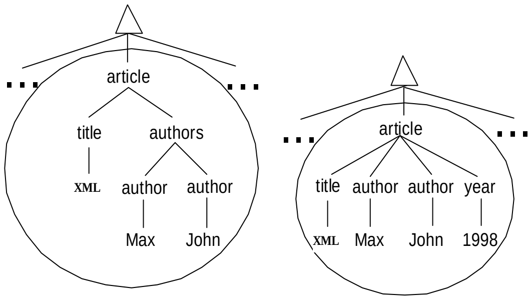
The procedure of the subtree-based XML data integration is illustrated in Fig. 2. Assume the number of matched leaves in the target subtree t_{tj} is n_m, the total number of leaf nodes in the integrated subtree, N can be figured out by the following equation.

$$N = n_b + n_t - n_m \quad (6)$$

Example 1. *For the matched subtree pair t_b and t_t shown in Fig. 3, the nonmatched leaf in the target subtree t_t is "1998". Therefore, the path {"year", "1998"} will be inserted into the base subtree t_b under its root node "article" as shown in the integrated subtree in Fig 4. According to Equation (6), the total number of the result subtree N = 3+4-3 = 4.*

```
Input: Matched subtree pairs (t_bi, t_tj)
Output: Integrated subtree t(i)
Assume t_bi and t_tj have n_b and n_t leaf nodes, respectively.
begin{
  t(i)=t_bi ; //Reserve the whole base subtree as the
              primal result
  for (k =1 to n_t) {
    if (L_tk is a nonmatched leaf){
      ins (L_tk, t(i)) ; //Insert the nonmatched leaves in
                          the target subtree
    }
  }
  return t(i);
} end
```

Fig. 2 Procedure of subtree-based integration



(a) Base subtree t_b (b) Target subtree t_t

Fig. 3 Example matched subtree pair

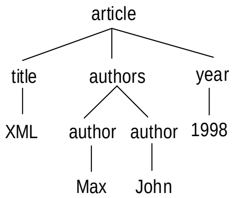


Fig. 4 Example integrated subtree

5. Experimental Evaluation

We conduct experiments to compare the effectiveness of the subtree-based integration for real large bibliography XML documents using *LAX* and *SLAX*, respectively. The experiments have been done under the environment as shown in Table 1.

In our experiments, we use a fragment file of SigmodRecord.xml [1] named sigmod.xml (240KB, about 10,000 nodes) as the base document and 20 fragments of DBLP.xml [11] named dblp1-20.xml (300KB, about 15,000 nodes) as the target ones. We sample a base subtree from sigmod.xml as shown in Fig. 5. The match values for the sample subtree and dblp1-20.xml are shown in Fig. 6 (a), and the tree similarity degrees for sigmod.xml and dblp1-20.xml are shown in Fig. 6 (b). The matched subtree for the sample one will be selected from dblp8.xml by *SLAX* because of the maximum match value. While the matched subtree for the sample one will be selected from dblp13.xml by *LAX*.

Fig. 7 shows the real XML code of the matched subtree for the sample one. From Fig. 7 (a), we can observe that the matched subtree selected from dblp8.xml by *SLAX* is exactly the same article as the sample subtree in Fig. 4. However, Fig. 7 (b) indicates that the matched subtree selected from dblp13.xml by *LAX* is not exactly the same article as the sample one but written by the same authors.

Fig. 8 indicates the integration results for the matched subtrees selected by *SLAX* and *LAX*. From Fig. 8 (a), we can see that the integration result using *SLAX* contains the complete information for the same article from both the base subtree and the matched target one. Although the result using *LAX* shown in Fig. 8 (b) indicates that the integrated subtree sometimes may not represent the same information of an identical article, it is still available for collecting valuable information, articles written by the same authors for example. Therefore, we consider that our previous proposed algorithms are effective for the subtree-based XML data integration.

Table 1 Experimental Environment

CPU	Intel Pentium 4 2.80 GHz
Memory	1.0 GB
OS	MS Windows XP Professional
Programming Environment	Sun JDK 1.4.2

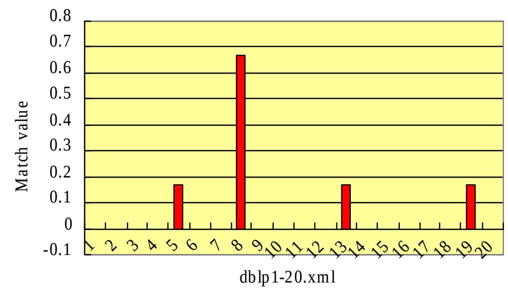
```
<article>
<title>On Global Multidatabase Query Optimization.</title>
<initPage>6</initPage>
<endPage>11</endPage>
<authors>
<author position="00">Hongjun Lu</author>
<author position="01">Beng Chin Ooi</author>
<author position="02">Cheng Hian Goh</author>
</authors>
</article>
```

Fig. 5 Sample subtree from sigmod.xml

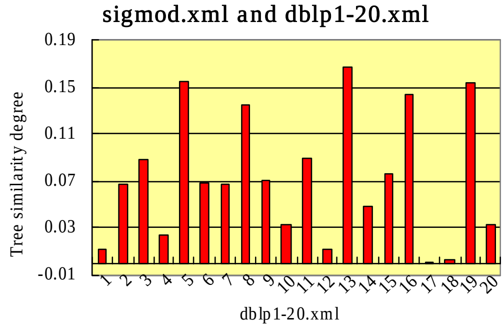
6. Concluding Remarks

As more and more data are represented and exchanged by XML on the World Wide Web, an algorithm that can efficiently measure the approximate similarity between XML documents for the subtree-based integration of multiple XML documents becomes important.

In this paper, we have proposed a subtree-based XML data integration method for integrating the matched subtree pairs determined by the previous proposed leaf-clustering based approximate XML join algorithms, *LAX* and *SLAX*. We conduct experiments to compare the effectiveness of the subtree-based integration for real large bibliography XML documents using *LAX* and *SLAX*, respectively. And we show the integration result for a sample subtree from the XML document of SIGMOD record. The experimental results indicate that our previous proposed algorithms are effective for the subtree-based XML data integration.



(a) Match values for the sample subtree from sigmod.xml and dblp1-20.xml



(b) Tree similarity degrees for sigmod.xml and dblp1-20.xml

Fig. 6 Results for bibliography data

```
<article mdate="2002-01-03" key="journals/sigmod/LuOG92">
<ee>db/journals/sigmod/LuOG92.html</ee>
<author>Hongjun Lu</author>
<author>Beng Chin Ooi</author>
<author>Cheng Hian Goh</author>
<title>On Global Multidatabase Query Optimization.</title>
<cdrom>sigmodR/21-4/P006.pdf</cdrom>
<year>1992</year>
<volume>21</volume>
<number>4</number>
……
</article>
```

(a) Selected from dblp8.xml by *SLAX*

```
<inproceedings mdate="2002-01-03" key="conf/sigmod/LowOL92">
<crossref>conf/sigmod/92</crossref>
<author>Chee Chin Low</author>
<author>Beng Chin Ooi</author>
<author>Hongjun Lu</author>
<title>H-trees: A Dynamic Associative Search Index for OODB.</title>
<year>1992</year>
……
</inproceedings>
```

(b) Selected from dblp13.xml by *LAX*

Fig. 7 Real XML code of the matched subtree for the bibliography documents

```
<article>
<title>On Global Multidatabase Query Optimization.</title>
<initPage>6</initPage>
<endPage>11</endPage>
<authors>
<author position="00">Hongjun Lu</author>
<author position="01">Beng Chin Ooi</author>
<author position="02">Cheng Hian Goh</author>
</authors>
<ee>db/journals/sigmod/LuOG92.html</ee>
<cdrom>sigmodR/21-4/P006.pdf</cdrom>
<year>1992</year>
<volume>21</volume>
<number>4</number>
……
</article>
```

(a) Integration result using *SLAX*

```
<article>
<title>On Global Multidatabase Query Optimization.</title>
<initPage>6</initPage>
<endPage>11</endPage>
<authors>
<author position="00">Hongjun Lu</author>
<author position="01">Beng Chin Ooi</author>
<author position="02">Cheng Hian Goh</author>
</authors>
<crossref>conf/sigmod/92</crossref>
<author>Chee Chin Low</author>
<title>H-trees: A Dynamic Associative Search Index for OODB.</title>
<year>1992</year>
……
</article>
```

(b) Integration result using *LAX*

Fig. 8 Integration result for the sample subtree

[Acknowledgements]

This work is partially supported by the Grant-in-Aid for Scientific Research of MEXT Japan (#16016232), by CREST of JST, and by the TokyoTech 21COE Program “Framework for Systematization and Application of Large-Scale Knowledge Resources”.

[References]

[1] ACM SIGMOD Record in XML. Available at http://www.acm.org/sigmod/record/xml/

[2] M. Garofalakis and A. Kumar. Correlating XML data streams using tree-edit distance embeddings. In Proc of PODS'03, page 143-154, 2003.

[3] S. Guha, H. V. Jagadish, N. Koudas, D. Srivastava and T. Yu. Approximate XML Joins. In Proc. of ACM SIGMOD 2002, pages 287-298, 2002.

[4] S. Guha, N. Koudas, D. Srivastava and T. Yu. Index-Based Approximate XML Joins. In Proc. of ICDE 2003, pages 708-710, 2003.

[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 2005, Springer LNCS 3567, pages 82-97, July 2005.

[6] W. Liang and H. Yokota. *SLAX*: An Improved Leaf-clustering Based Approximate XML Join Based on Clustered Leaf Nodes for XML Data Integration at Subtree Classes. In Proc. of DBWeb 2005, IPSJ Symposium Series, 2005(16):41-48, November 2005.

[7] A. Nierman and H. V. Jagadish. Evaluating Structural Similarity in XML Documents. In Proc. of WebDB 2002, pages 61-66, 2002.

[8] Swiss-Prot. http://www.ebi.ac.uk/swissprot/.

[9] TrEMBL. http://www.ebi.ac.uk/trembl/.

[10] World Wide Web Consortium (W3C). The Document Object Model (DOM). http://www.//w3.org/DOM/.

[11] XML Version of DBLP. Available at http://dblp.uni-trier.de/xml/.

[12] K. Zhang and D. Shasha. Tree Pattern Matching. Pattern Matching Algorithm, chapter 11. Oxford University Press, 1997.

Wenxin LIANG

He received the B.E. and M.E. degrees in biomedical electronic engineering from Xi’an Jiaotong University, China in 1998 and 2001, respectively. He is currently a Ph.D. student at the Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology, Japan. His current research interests include XML data integration and management, XML storage, indexing, labeling and querying techniques. He is a student member of DBSJ and SIGMOD-J.

Haruo YOKOTA

He received the B.E., M.E., and Dr. Eng. degrees from Tokyo Institute of Technology in 1980, 1982, and 1991, respectively. He joined Fujitsu Ltd. in 1982, and was a researcher at the ICOT for the Japanese 5th Generation Computer Project from 1982 to 1986, and at Fujitsu Laboratories Ltd. from 1986 to 1992. From 1992 to 1998, he was an associate professor in Japan Advanced Institute of Science and Technology (JAIST). He is currently a professor at Global Scientific Information and Computing Center, and Department of Computer Science in Graduate School of Information Science and Engineering in Tokyo Institute of Technology. His current research interests include general research area of data engineering, information storage systems and dependable computing. He is a member of DBSJ, IPSJ, IEICE, JSAI, IEEE, IEEE-CS, ACM, ACM-SIGMOD, and ACM-SIGARCH.