JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 19, No. 7 : 1599~1607 Jul. 2015

부모-자식 행렬을 사용한 XML 문서 유사도 측정과 군집 기법

이윤구[1] · 김우생[2*]

Similarity Measure and Clustering Technique for XML Documents by a Parent-Child Matrix

Yun-Gu Lee[1] · Woosaeng Kim[2*]

[1]Department of Computer Software, Kwangwoon University, Seoul 139-701, Korea
[2*]Department of Computer Software, Kwangwoon University, Seoul 139-701, Korea

요 약

최근 들어, 인터넷에서 자주 사용되는 XML 문서들에 대한 접근, 질의와 관리를 위한 효율적인 기법들이 연구 되어 왔다. 이 논문에서, 우리는 XML 문서를 효율적으로 군집화하기 위해 부모-자식 행렬 기법을 제안한다. 부모-자식 행렬은 XML 문서의 내용과 구조의 특징들을 분석한다. 부모-자식 행렬의 각 셀은 XML 트리 노드의 값이거나, 트리에서 부모-자식 관계가 존재할 때의 자식 노드의 값이 된다. 따라서 두 XML 문서의 유사도는 대응하는 부모-자식 행렬들의 유사도로 측정된다. 실험을 통해 우리가 제안하는 기법이 좋은 결과를 냄을 보였다.

ABSTRACT

Recently, researches have been developing efficient techniques for accessing, querying, and managing XML documents which are frequently used in the Internet. In this paper, we propose a parent-child matrix to cluster XML documents efficiently. A parent-child matrix analyzes both the content and structural features of an XML document. Each cell of a parent-child matrix has either the value of a node in an XML tree or the value of a child node, where a parent-child relationship exists in the XML tree. Then, the similarity between two XML documents can be measured by the similarity between two corresponding parent-child matrices. The experiment shows that our proposed method has good performance.

키워드 : XML, 군집화, XML 유사도

Key word : XML, Clustering, XML Similarity

Received 27 May 2015, **Revised** 19 June 2015, **Accepted** 30 June 2015

*** Corresponding Author** Woosaeng Kim(E-mail:kwsrain@kw.ac.kr, Tel:+82-2-940-5217)
Department of Computer Software, Kwangwoon University, Seoul, 139-701, Korea

Open Access **http://dx.doi.org/10.6109/jkiice.2015.19.7.1599** print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. INTRODUCTION

As the Internet continues to grow and evolve, XML has become the world's most widely accepted data representation and exchange format. An XML document consists of a content and a structure, and in this way differs from traditional data. This requires new effective methods to process, organize, and retrieve the semi-structural nature of XML data.

The general purpose of data clustering is to derive some relevant information from the various data for further data processing. The clustered data may show some tendency or regularity in the data and may even show some relevant knowledge worth noting. The clustering of XML documents is to group similar documents to facilitate searching because similar documents can be searched and processed within a specific category. The appropriate clustering of XML documents is also effective for systematic document management, data integration, XML retrieval, efficient storage of XML documents, and even for system protection purposes because unusual document can be discovered easily. XML clustering methods can be categorized into three main groups: methods based on content, methods based on structure, and methods based on both content and structure.

A nested structure of an XML document can be modeled as an ordered labeled tree. An element and an inclusion relationship of an XML document corresponds to a node and a level of the corresponding tree, respectively1). Therefore, we can find similar XML trees corresponding to XML documents for clustering. However, the difficult task is to find partially matched sub-trees between XML trees. The studies to find partially matched sub-trees between XML trees that have been made so far are complex, and have an overhead[1-3]. Thus, in this paper, we propose a parent-child matrix to find both semantically and structurally matched sub-trees between XML trees efficiently. In fact, a parent-child matrix can represent not only parent-child relationships but also ancestor-descendant relationships between nodes in an XML tree. Each cell of a parent-child matrix has either the value of a node in an XML tree or the value of a child node, where a parent-child relationship exists in the XML tree. Then, the similarity between two XML documents can be measured by the similarity between two corresponding parent-child matrices. The experiment shows that the clusters are formed well and efficiently when a parent-child matrix is used to cluster XML documents.

The rest of the paper is organized as follows. The related works are discussed in section 2. In section 3, we propose a parent-child matrix method to cluster XML documents. In section 4, we test and verify the effectiveness of our algorithm with synthetic and real data. Finally, we make a conclusion in section 5.

Ⅱ. RELATED WORKS

XML data can be represented using a common data model such as rooted labeled trees, directed acyclic graphs, or vector-based techniques for clustering. The data models capture either content, structure, or content and structural features of XML data, and it is the basis for the identification of the features to be exploited in the similarity computation. XML documents can be represented as labeled trees. Several authors have provided algorithms for computing the optimal edit distance between two XML trees, which is the generalization of the problem of computing the distance between two strings[4]. In general, edit distance measures the minimum number of node/sub-tree insertions, deletions, and updates required to convert one XML tree T1 into another XML tree T2. The distance between T1 and T2 is then defined to be the cost of such a

1) In a tree representation, attributes are not distinguished from elements, both are mapped to the tag name set; thus, attributes are handled as elements.

sequence[1-3, 5-9]. However, a clustering based on the notion of edit distance between the tree representations of an XML document is too costly to be practical. Thus, an effective summarization, which can distinguish documents among different clusters, would be highly desirable. Based on this direction, Ref. [10] developed the notion of the s-graph to represent XML data and suggested a distance metric to perform clustering on XML data. They have shown that the s-graph of an XML document can be encoded by a cheap bit string, and clustering can then be efficiently applied on the set of bit strings for the whole document collection.

XML data can be represented as vectors in an abstract n-dimensional feature space. A feature can be either element, text, level, path, or any component of an XML document/tree. Then, the feature vectors are compared to find similar XML documents. Ref. [11] applies a K-means clustering technique to XML documents represented in a vector-space model. In this representation, each document is represented by an N-dimensional vector with N being the number of document features such as text features, tag features, and a combination of both in the collection. However, they only consider the contents of XML. In Refs. [12, 13], a new bitmap indexing based technique to cluster XML documents is described. A BitCube is presented as a 3-dimensional bitmap index of triplets (document, XML-element path, word). BitCube indexes can be manipulated to partition documents into clusters by exploiting bit-wise distance and popularity measures. Ref. [14] devises features for XML data, focusing on content information extracted from textual elements and structural information derived from tag paths. They introduce the notion of tree tuple in the definition of an XML representation model that allows for mapping XML document trees into transactional data, i.e., variable length sequences of objects with categorical attributes. A partitional clustering approach has been developed and applied to the XML transactional domain. Ref. [15] transforms XML trees into vectors in a high-dimensional Euclidean space based on the occurrences of the features in the documents. Next, they apply principal component analysis (PCA) to the matrix to reduce its dimensionality. Finally, they use a K-means algorithm to cluster the vectors residing in the reduced dimensional space and place them in appropriate categories. Refs. [16, 17] propose a method to extract representative paths from the XML trees by various pattern mining techniques. Then, the XML documents are clustered by the path similarities.

Ⅲ. PARENT-CHILD MATRIX SIMILARITY MEASURE FOR XML DOCUMENTS

The similarity between two XML documents can be measured by their content and structure features. Let us assume that Figs. 2(a), 2(b) and 2(c) are the corresponding XML trees of the XML documents A1, A2, and A3 generated by the Actor DTD of Fig. 1. Let us also assume that Figs. 2(d) and 2(e) are the corresponding XML tress of the XML documents M1 and M2 generated by the Movie DTD of Fig. 1. These XML trees share the nodes of a, b, c, and f, and they also share nodes b and c as the child nodes of node a. In other words, they share node f and a sub_tree rooted at the node a. Therefore, the XML documents A1, A2, A3, M1, and M2 are similar to each other.

Actor	Movie
<!ELEMENT a (b, c)>	<!ELEMENT f (a, h?, i*)>
<!ELEMENT b (d*, e?, f?)>	<!ELEMENT a (b, c)>
<!ELEMENT c (e?, f?, g?)>	<!ELEMENT i (j)>
<!ELEMENT d (#PCDATA)>	<!ELEMENT b (#PCDATA)>
<!ELEMENT e (#PCDATA)>	<!ELEMENT c (#PCDATA)>
<!ELEMENT f (#PCDATA)>	<!ELEMENT h (#PCDATA)>
<!ELEMENT g (#PCDATA)>	<!ELEMENT j (#PCDATA)>

그림 1. Actor와 Movie DTD
Fig. 1 Actor and Movie DTD

We propose a parent-child matrix to represent this kind of content and structural features of the XML trees corresponding to the XML documents.

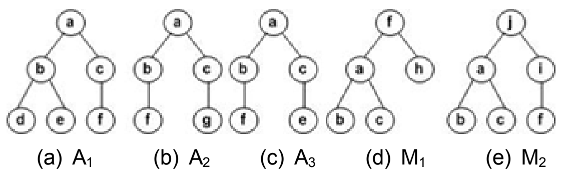

그림 **2.** XML 문서들 A_1, A_2, A_3, M_1, M_2의 트리들
Fig. 2 Trees of the XML documents A_1, A_2, A_3, M_1, and M_2

Both the vertical and horizontal axes of a parent-child matrix represent all the distinct nodes in the given XML trees. Each cell of a parent-child matrix has either the value of a node in an XML tree or the value of a child node, where a parent-child relationship exists in the XML tree. However, XML documents can encompass many optional and repeated elements. Thus, the value of a node becomes the summation of each node's value when the nodes with the same name occur multiple times in an XML tree in order to reflect repetitions of elements in similarity measure.

Definition 1: A parent-child $n \times n$ matrix M for an XML document with its corresponding XML tree T is defined by i) $M_{aa}=\sum$ (a's value) if node a is in T, ii) $M_{ab}=\sum$ (b's value) if a node b is a child of node a in T where n is the total number of all the distinct nodes in the given XML trees.

For example, when the value of each node is 1, Tables 1 and 2 are the parent-child matrices of the XML documents A_1 and A_2, respectively. Any parent-child relationship between two nodes in an XML tree is represented by a parent-child matrix. A parent-child relationship between node *a* and node *b* of Fig. 2(b) can be represented by the values of column *a* and column *b* of the row *a* in Table 2. In fact, a parent-child matrix can represent not only parent-child relationships but also ancestor-descendant relationships. For example, the ancestor-descendant relationships between node *a* and node *f* of Fig. 2(b) is represented by both a parent-child relationship between node *a* and node *b*, which in turn is represented by the values of column *a* and column *b* of the row *a* in Table 2 and a parent-child relationship between node *b* and node *f*, which in turn is represented by the values of column *b* and column *f* of the row *b* in Table 2.

표 **1.** XML 문서 A_1의 부모-자식 행렬
Table. 1 Parent-child Matrix of XML Document A_1

	a	b	c	d	e	f	g	h	i	j
a	1	1	1							
b		1		1	1					
c			1			1				
d				1						
e					1					
f						1				
g										
h										
i										
j										

표 **2.** XML 문서 A_2의 부모-자식 행렬
Table. 2 Parent-child Matrix of XML Document A_2.

	a	b	c	d	e	f	g	h	i	j
a	1	1	1							
b		1				1				
c			1				1			
d										
e										
f						1				
g							1			
h										
i										
j										

A similarity between two XML documents can be measured by the similarity between two corresponding parent-child matrices. In this paper, when both the corresponding cells between two parent-child matrices have some value, we call those cells as a paired cell. Then, the number of paired cells and the summation of all the paired cells' values between A_1 and A_2 is 6 and 12, respectively, because Figs. 2(a) and 2(b) share the nodes of *a*, *b*, *c*, and *f*, and they also share nodes *b* and *c* as the child nodes of node *a*. In the same manner, the number of paired cells and the summation of all the paired cells' values between A_1 and M_1 is 6 and 12, respectively. Therefore, the similarity between A_1 and A_2 is the same as the similarity between A_1 and M_1. However, in an XML tree, the closer to root node the more important to XML tree in general. Therefore, we can regard that the similarity between A_1 and A_2 is greater than the similarity between A_1 and M_1. In order to reflect this concept, it is necessary to give more values to those nodes near the root of XML tree. In the similar way, even though the summation of all the paired cells' values between A_1 and M_1, and between A_1 and M_2 has the same value of 12, the similarity between A_1 and M_1 is greater than the similarity between A_1 and M_2, because the size of M_1 is smaller than that of M_2. Therefore, it is necessary to give more values to those nodes in a smaller size XML tree.

Definition 2. The similarity S between two XML documents X_1 and X_2, whose corresponding parent-child matrices are M_{X1} and M_{X2}, is defined by

$$S(X_1,X_2) = \sum SPC \times \frac{|M_{X_1} \cap M_{X_2}|}{|M_{X_1}| \cup |M_{X_2}|} \qquad (1)$$

where $\sum SPC$ is the summation of all the paired cells' values between M_{X1} and M_{X2}, $|M_{X1}|$ is the number of cells which has a value in M_{X1}, $|M_{X2}|$ is the number of cells which has a value in M_{X2}, and $|M_{X1} \cap M_{X2}|$ is the number of paired cells between M_{X1} and M_{X2}.

For example, if we give a weighted value of 3, 2, and 1 to a node in the first, second, and third level of an XML tree, respectively, table 3 shows the combination of the five parent-child matrices for XML documents A_1, A_2, A_3, M_1, and M_2. In this case, the similarity between A_1 and A_2 is 24×(6/20), because their total cells' number, the paired cells' number and the summation of all the paired cells' values is 20, 6 and 24, respectively. On the other hand, the similarity between A_1 and M_1 is 21×(6/20) because their total cells' number, the paired cells' number and the summation of all the paired cells' values is 20, 6 and 21, respectively. Therefore, the similarity between A_1 and A_2 is greater than the similarity between A_1 and M_1. In the similar way, the similarity between A_1 and M_1 is 21×(6/20) and the similarity between A_1 and M_2 is 21×(6/22). Therefore, the similarity between A_1 and M_1 is greater than the similarity between A_1 and M_2. Since A_2 and A_3 have an additional ancestor-descendant relationships between node *a* and node *f*, their similarity has the largest value 25×(7/18).

표 3. XML 문서 A_1, A_2, A_3, M_1, M_2. 부모-자식 행렬들의 결합
Table. 3 Combination of the Parent-child Matrices for XML documents A_1, A_2, A_3, M_1, and M_2

	a	b	⋯	i	j
a	(3,3,3,2,2)	(2,2,2,1,1)	⋯		
b		(2,2,2,1,1)	⋯		
c			⋯		
d			⋯		
e			⋯		
f	(,,2,2)		⋯	(,,,,2)	
g			⋯		
h			⋯		
i			⋯	(,,,,2)	(,,,,1)
j			⋯		(,,,,1)

Ⅳ. CLUSTERING XML DOCUMENTS BY PARENT-CHILD MATRIX

We use a hierarchical clustering algorithm to cluster XML documents. The hierarchical clustering algorithms produce a hierarchy of nested clustering. A clustering $\mathcal{R}_1$ containing k clusters is said to be nested in the clustering $\mathcal{R}_2$, which contains r ($< k$) clusters if each cluster in $\mathcal{R}_1$ is a proper subset of $\mathcal{R}_2$. The hierarchical clustering algorithms are classified into two groups agglomerative and divisive in accordance with the building up direction of the clusters. The pseudo code of the general agglomerative clustering algorithm is described in Fig. 3 when the total number of patterns is n.

```
① Begin with $n$ clusters, each consists of one pattern.
② Repeat step ③ a total of $n$-1 times.
③ Find the most similar clusters $C_i$ and $C_j$ and merge $C_i$ and
   $C_j$ into one cluster.
```

그림 3. 응집형 군집화 알고리즘
Fig. 3 Agglomerative clustering algorithm

Different agglomerative clustering algorithms are obtained by using different methods to determine the similarity of clusters. The single-linkage algorithm is obtained by defining the distance between two clusters to be the smallest distance between two patterns such that one pattern is in each cluster. Therefore, if C_i and C_j are clusters, the distance between them is defined as in

$$d_{SL}(c_i,c_j) = \min_{X\in Ci,\, Y\in Cj} d(X,Y) \tag{2}$$

where, $d(X, Y)$ denotes the distance between the samples X and Y. On the other hand, the complete-linkage algorithm is obtained by defining the distance between two clusters to be the largest distance between a pattern in one cluster and a pattern in the other cluster. Therefore, if C_i and C_j are clusters, the distance between them is defined as in

$$d_{CL}(c_i,c_j) = \min_{X\in Ci,\, Y\in Cj} d(X,Y) \tag{3}$$

where, $d(X, Y)$ denotes the distance between the samples X and Y. One of the important issues for the clustering process is how a similarity measure between patterns is quantified. The most obvious measure of the similarity (or dissimilarity) between two patterns is the distance between them especially Euclidean distance. Since we represent an XML document as a parent-child matrix, Definition 2 is used as a similarity measure between two XML documents. It is always possible to represent an $n\times n$ parent-child matrix as a $1\times n^2$ parent-child vector. Therefore, we will use a parent-child vector in case of need. When Table 4 shows the corresponding parent-child vectors for XML documents A_1, A_2, A_3, M_1, and M_2, the process of clustering by a single-linkage algorithm is as follows. The most similar two vectors in Table 4 are A_2 and A_3, because their similarity has the largest value of 25×(7/18). Therefore, we form $\{A_2, A_3\}$ as the first cluster. In the similar way, the most similar two vectors are A_1 and A_2, because their similarity has the largest value of 24×(6/20). Therefore, we form $\{A_1,\{A_2,A_3\}\}$ as the second cluster. In the similar way, the most similar two vectors are $\{M_1,M_2\}$ because their similarity has the largest value of 22×(7/20). Therefore, we form $\{M_1,M_2\}$ as the third cluster. Finally, $\{\{A_1,\{A_2,A_3\}\},\{M_1,M_2\}\}$ is formed. Therefore, when we have two clusters from five documents with a hierarchical clustering technique, the clusters are formed correctly as $\{A_1,A_2,A_3\}$ and $\{M_1,M_2\}$.

표 4. XML 문서 A_1, A_2, A_3, M_1, M_2의 부모-자식 행렬들
Table. 4 Parent-child vectors for XML documents A_1, A_2, A_3, M_1, and M_2

	aa	ab	ac	ad	..	ba	bb	bc	bd	..	ji	jj
A_1	3	2	2		...		2		1	...		
A_2	3	2	2		...		2			...		
A_3	3	2	2		...		2			...		
M_1	2	1	1		...		1			...		1
M_2	2	1	1		...		1			...		

```
Input: X - the parent-child vectors for
           all the given XML documents
Output: Similarity - similarity between every two XML documents

 num=1;
 for i ← 1 to (docNum-1) do
  for j ← i+1 to docNum do
   cell1 ←0; cell2 ←0; pairedCell ←0; pairedCellSum ←0;
   for k ← 1 to (AllElement * AllElement) do
    if X(i,k)>0 then cell1 ← cell1+1;
    if X(j,k)>0 then cell2 ← cell2+1;
    if (X(i,k)>0) & (X(j,k)>0) then
      pairedCell ← pairedCell+1;
      pairedCellSum ← pairedCellSum+(X(i,k)+X(j,k));
    end
   end
   Similarity(num) ← pairedCellSum*(pairedCell/(cell1+cell2));
   num ← num+1;
  end
 end
```

그림 4. 유사도 계산 알고리즘
Fig. 4 Similarity calculation algorithm

A pseudo code of the algorithm for computing the similarity among all the given XML documents is given Fig. 4. The algorithm requires O(n2) for each comparison, where n is the total number of distinct elements of all the given XML documents.

Ⅴ. CLUSTERING EXPERIMENTS

We performed experiments on both synthetic and real data sets. We generated the synthetic data using DTDs of the XML data bank of the University of Wisconsin[18]. This data bank provides 8 DTDs, such as actor, bibliography, club, department, movies, personal information, company profiles, and stock quotes. A parent-child vector for each XML document is used to cluster XML documents. We give a weighted value of 5, 4, 3, 2, and 1 to a node in the first, second, third, fourth, and the remaining levels of an XML tree, respectively. Simple cardinality constraints can be imposed on the elements using regular expression operators ?, +, or * in DTD. As a result, algorithms for comparing XML documents should be aware of such repetitions to effectively assess structural similarity[1, 2]. We therefore test our algorithm by controlling the sub-tree repetitions in the XML tree. First, we restrict a special symbol such as ?, +, or * in DTD, which occurs at most one time in the generated XML document. Fig. 5 shows the dendrogram when a single-linkage algorithm is applied. In Fig. 5, the horizontal axis represents the XML documents and the vertical axis represents the dissimilarity measure. We test 40 XML documents generated from 8 DTDs in order to show the result in one diagram. All the XML documents are numbered in order to observe the result easily: Actor (1-5), Bibliography(6-10), Club(11-15), Department (16-20), Movies(21-25), Personal(26-30), Profile (31-35), and Stock(36-40). The diagram illustrates that all the documents are clustered correctly. Department documents are clustered first because they have the greatest number of elements; on the other hand, Actor documents are clustered last because they have the smallest number of elements. The similarities among Profile documents are the same because Profile DTD does not have a special symbol, such as ?, +, or *. Among the eight clusters, Department and Club documents are clustered first because they share many nodes with the same name such as name, phone, email, address, lastname, and firstname. Fig. 6 shows the dendrogram when a complete-linkage algorithm is applied to the same XML documents. Fig. 6 illustrates that all the documents are clustered correctly even though the sequence of the clustering process is different from the case of a single-linkage algorithm.

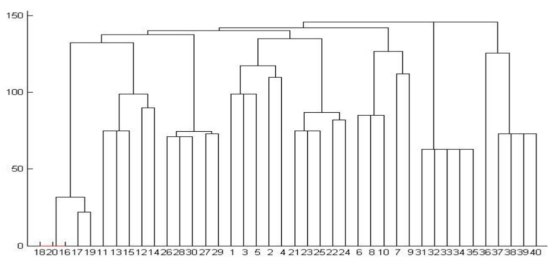

그림 5. 단일 연결 알고리즘에 의한 성장 그래프
Fig. 5 Dendrogram by single-linkage algorithm

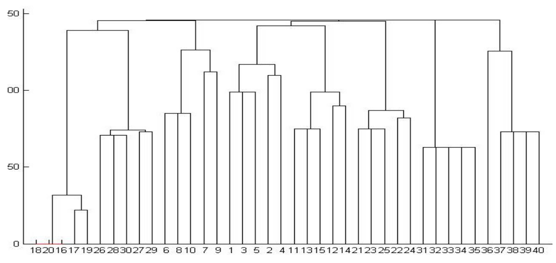

그림 6. 완전 연결 알고리즘에 의한 성장 그래프
Fig. 6 Dendrogram by complete-linkage algorithm

Since Actor documents are clustered last because they have the smallest number of elements, we intentionally allow a special symbol such as ?, +, or * in DTD to occur several times in the generated XML document. As shown in Figs. 7 and 8, Actor documents are clustered first because they have many repetitions of elements, and even sub-tress.

For a real data set, we used XML data obtained from the online XML version of the ACM SIGMOD Record[19]. Specifically, we sampled 15 documents randomly from each of the following DTDs: Index TermsPage.dtd, OrdinaryIssuePage.dtd, and Proceedings Page.dtd. Fig. 9 shows the dendrogram when an average-linkage algorithm is applied.

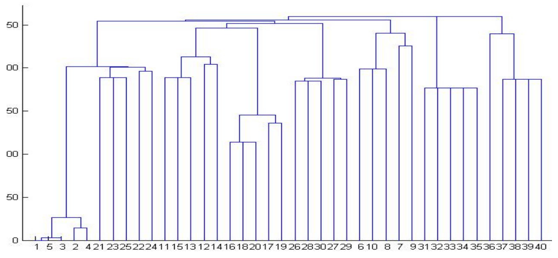

그림 7. 단일 연결 알고리즘에 의한 성장 그래프
Fig. 7 Dendrogram by single-linkage algorithm

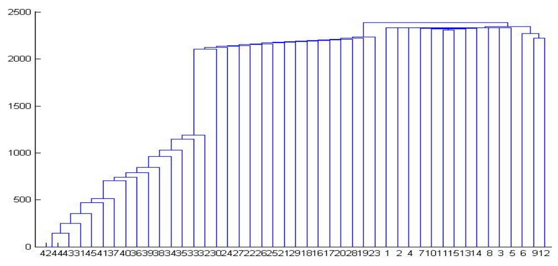

그림 8. 완전 연결 알고리즘에 의한 성장 그래프
Fig. 8 Dendrogram by complete-linkage algorithm.

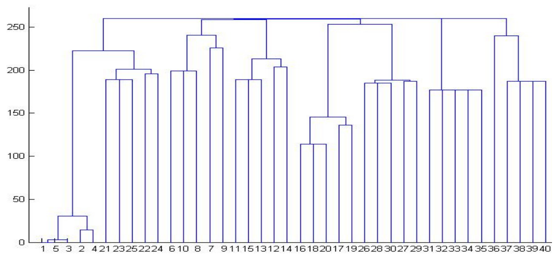

그림 9. 평균 연결 알고리즘에 의한 성장 그래프
Fig. 9 Dendrogram by average-linkage algorithm

All the XML documents are numbered in order to observe the result easily: IndexTermsPage(1-15), OrdinaryIssuePage (16-30), and ProceedingsPage(31-45). The diagram illustrates that all the documents are clustered correctly. ProceedingsPage documents are clustered first because they have the greatest number of elements; on the other hand, IndexTermsPage documents are clustered last because they have the smallest number of elements. The similarities among the IndexTermsPage documents or the OrdinaryIssuePage documents are almost the same because their contents and size are almost similar.

Ⅵ. CONCLUSION

An efficient clustering technique is required to process XML documents which are popular on the Internet for their data exchange format. Similar XML documents can easily be found if we can find partially matched sub-trees between the corresponding XML trees. The previous methods to find partially matched sub-trees between XML trees are complex and have an overhead. We therefore propose a parent-child matrix that represents not only a parent-child relationship but also ancestor-descendant relationships between nodes in XML trees. Since a parent-child matrix can find both semantically and structurally matched sub-trees between XML trees easily, it can cluster similar XML documents efficiently. A hierarchical clustering algorithm with a parent-child matrix is used to cluster XML documents. The experiment shows that by our method, clusters are formed correctly and efficiently.

ACKNOWLEDGMENTS

The present research has been conducted by research grant of Kwangwoon University in 2015.

REFERENCES

[1] A. Nierman and H. V. Jagadish, "Evaluating structural similarity in XML documents," *Fifth International Workshop on the Web and Databases,* 2002.

[2] J. Tekli and R. Chbeir, "A novel XML document structure comparison frmaework based-on sub-tree commonalities and label semantics," *Web Semantics: Science, Services*

and Agents on the WWW, 2012.

[3] D. Brzezinski, A. Lesniewska, T. Morzy, and M. Piernik, "XCleaner: A new method for clustering XML documents by structure," *Control and Cybernetics*, 2011.

[4] R. Wanger and M. Fischer, "The String-to-String Correction Problem," *Journal of the ACM*, 1974.

[5] S. Chawathe, A. Rajaraman, H. Garcia-Molina, and J. Widom, "Change Detection in Hierarchically Structure Information," in *Proceedings of the ACM International Conference on Management of Data*, 1996.

[6] S. Chawathe, "Comparing Hierarchical Data in External Memory," in *Proceedings of International Conference on Very Large Databases*, 1999.

[7] J. Wang, K. Zhang, and D. Shasha, "A System for Approximate Tree Matching," *IEEE TKDE*, 1994.

[8] A. Nierman and H. V. Jagadish, "Evaluating Structural Similarity in XML Documents," in *Proceedings of International Workshop on the Web and Databases*, 2002.

[9] T. Dalamagas, T. Cheng, K. J. Winkel, and T. Sellis, "A Methodology for Clustering XML Documents by Structure," *Information Systems*. In press, 2004.

[10] W. Lian, D. Cheung, N. Mamoulis, and S.-M. Yiu, “An Efficient and Scalable Algorithm for Clustering XML Documents by Structures,” *TKDEE*, 2004.

[11] A. Doucet and H. Ahonen-Myka, “Navie Clustering of a Large XML Document Collection,” *Proc. 1st Annual Workshop of the Initiative for the Evaluation of XML retrival(INEX)*, Germany, pp.81-88, Dec. 2002.

[12] J. Yoon, V. Raghavan, and V. Chakilam, “BitCube: Clustering and Statistical Analysis for XML Documents,” *Proc. of the 13th Int. Conf. on Scientific and Statistical Database Management*, Fairfax, Virginia, July 2001.

[13] J. Yoon, V. Raghavan, V. Chakilam, and L. Kerschberg, “BitCube: A 3-D Bitmap Indexing for XML Documents,” *Journal of Intelligent Information Systems*, Vol. 17, pp.241-254, November 2001.

[14] A. Tagarelli, and S. Greco, “Toward Semantic XML Clustering,” *6th SIAM International Conference on Data Mining (SDM ’06)*, Bethesda, Maryland, USA, pp. 188-199, April 2006.

[15] J. Liu, Jason T., L. Wang, W. Hsu, and K. G. Herbert, “XML Clustering by Principal Component Analysis,” *Proc. of the 26th IEEE International Conference on Tools with Artificial Intelligence(ICTAI)*, 2004.

[16] J. Hwang, and K. Ryu, “XML Document Clustering Based on Sequential Pattern,” *Journal of Korea Information Processing Society*, Dec. 2003.

[17] I. Choi, B. Moon, and H.-J. Kim, "A clustering method based on path similarities of XML data," *Data & Knowledge Engineering*, 2007.

[18] http://research.cs.wisc.edu/niagara/data.html

[19] http://www.sigmod.org/publications/sigmod-record/xml-edition.

이윤구(Yun-Gu Lee)

2000년 2월: 한국과학기술원 전기 및 전자공학과 졸업 (공학사)
2002년 2월: 한국과학기술원 전기 및 전자공학과 졸업 (공학석사)
2006년 8월: 한국과학기술원 전기 및 전자공학과 졸업 (공학박사)
2006년 9월 ~ 2013년 3월: 삼성전자 디지털 미디어 연구소 수석연구원
2013년 3월 ~ 현재: 광운대학교 컴퓨터 소프트웨어학과 부교수
※관심분야 : 영상 신호처리, 영상 시스템

김우생(Woosaeng Kim)

1985년 서울대 수료 및 텍사스 주립대학 전산학과 졸업 (학사)
1987년 미네소타 주립대학 전산학과 졸업 (석사)
1991년 미네소타 주립대학 전산학과 졸업 (박사)
1992년 - 현재: 광운대학교 컴퓨터 소프트웨어 학과 교수
관심분야: 데이터베이스, 멀티미디어