대표 경로에 기반한 XML 문서의 계층 군집화 기법☆

A Hierarchical Clustering Technique of XML Documents based on Representative Path

김 우 생*
Woosaeng Kim

요 약

XML은 데이터 교환과 정보 관리에 점차 중요해지고 있다. 근래에 XML 문서들에 대한 접근, 질의, 저장을 위한 효율적인 기법들을 개발하기 위해 많은 노력들이 이루어지고 있다. 이 논문에서 우리는 XML 문서들을 효율적으로 군집화하는 새로운 방법을 제안한다. XML 문서의 특징을 위해 XML 문서의 구조와 내용을 대표할 수 있는 새로운 대표 경로, 즉 가상 경로가 제안된다. XML 문서들을 군집화하기 위해 잘 알려진 계층 군집화 기법들을 대표 경로들에 적용하기 위한 방법도 제안된다. 실험을 통해 XML 문서의 특징으로 가상 경로를 사용했을 때 실제적인 군집들이 촘촘한 형상으로 잘 형성됨을 알 수 있다.

Abstract

XML is increasingly important in data exchange and information management. A large amount of efforts have been spent in developing efficient techniques for accessing, querying, and storing XML documents. In this paper, we propose a new method to cluster XML documents efficiently. A new prepresentative path called a virtul path which can represent both the structure and the contents of a XML document is proposed for the feature of a XML document. A method to apply the well known hierarchical clustering techniques to the representative paths to cluster XML documents is also proposed. The experiment shows that the true clusters are formed in a compact shape when a virtual path is used for the feature of a XML document.

☞ keyword : XML, XML clustering(XML 군집화)

1. Introduction

The growth of the Internet has greatly simplified access to existing information sources and spurred the creation of new sources. As the Internet continues to grow and evolve, more and more information is being placed in structurally rich documents such as XML. With the large number of documents on the Web, there is an increasing need to be able to automatically process those structurally rich documents for information retrieval and search applications.

XML consists of elements and each element is a pair of matching start- and end-tags and all the text that appears between them. Since users can define tags freely, tags represent not only data itself but also data meaning. The elements of XML are organized in a nested structure such that XML can be modeled as an ordered labeled tree[1]. Each node in this tree corresponds to a tag in the document and is labeled with the tag's name. Each edge in this tree represents inclusion of the tag corresponding to the child node under the tag corresponding to the parent node in the XML.

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

* 정 회 원 : 광운대학교 컴퓨터공학부 교수
kwsrain@kw.ac.kr
[2008/10/09 투고 - 2008/10/13 심사 - 2008/12/24 심사완료]
☆ 본 논문은 2009년도 광운대학교 교내 학술연구비 지원에 의해 연구되었음.

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 and the efficient storage of XML documents, and even for system protection purpose because unusual document can be discovered easily.

This paper proposes a new method to cluster XML documents efficiently. We propose a new representative path called a virtual path for the feature of a XML document. The virtual path which composes of the important node for each level of a XML tree can represent both the structure and the contents of a XML document. We then propose a method to apply the well known hierarchical clustering techniques to the representative paths to cluster XML documents. We will also explain how the hierarchical clustering process is terminated when the clustering that best fits the data has been achieved according to some criterion.

The rest of the paper is organized as follows. In section 2, we propose a method to extract representative paths such as a virtual path and a longest frequent path from a XML document. In section 3, we propose a method to apply the well known hierarchical clustering algorithm when representative paths are used as the feature of a XML document. In section 4, we test and verify the effectiveness of our algorithm with several examples. Finally, we make a conclusion in section 5.

2. Related Work

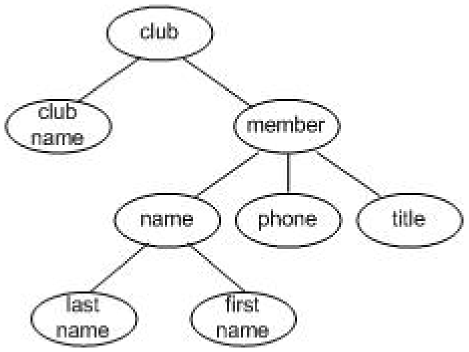
The need for organizing and clustering XML data has become challenging, due to the increase of heterogeneity of XML sources. Recently, several clustering techniques which consider the structure and/or the contents of XML documents are studied. Ref. [2] applies a K-means clustering technique to XML documents 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. They only consider the contents of XML. In [3,4], a new bitmap indexing based technique to cluster XML documents is described. A BitCube is presented in 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. However, this method needs manual operations to create a bitmap index. Ref. [5] devises features for XML data, focusing on content information extracted from textual elements and structure 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. On the other hand, ref. [6] transforms the structure of the XML document into a discrete function. The discrete function, then, is transformed into frequency domain by FFT. The result of FFT is a pair of complex numbers consisting of x and y values and considered to be a pair of n-dimensional vectors. The pairs of n-dimensional vectors are compared using a weighted Euclidean distance metric in an incremental and unsupervised fashion. This approach considers solely the structure of elements. Ref. [7] 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. However, their method only works for documents with the same DTD. Ref. [8] proposes a method to extract representative paths by considering both the sequence and occurrence frequency of elements of XML tree by sequential pattern mining technique. Then the paths composed of items are clustered by the notion of large items, i.e., items contained in some minimum fraction of transactions in a cluster to measure the similarity of a cluster of transactions[9].

3. Feature Extraction in XML

A feature extraction algorithm can be considered as a function, g, which maps a pattern P into a set of features $F=\{f_1,f_2,\dots,f_n\}$, viz. $g:P\rightarrow\{f_1,f_2,\dots,f_n\}$. We present methods to extract a feature which can represent a XML document. XML is composed of sequential and nested structure of elements contrast to the nonsequential structure of a traditional document. An element is composed of a pair of matching start and end tags, and all the text that appears between them. Since XML's tags express the meaning of a document, the sequential and nested structure of the important XML's tags can be used as a XML's feature. Fig. 1(a) is the part of the club XML document to explain the process of fining the structure which represents a given XML document.

```
<club>
    <clubname>....</clubname>
    <member>
        <name>
            <last name>...</last name>
            <first name>...</first name>
        </name>
        <phone>.....</phone>
        <title> ... </title>
    </member>
</club>
```

(a)



(b)

(Fig. 1) Club XML document and its corresponding tree

One way to get the structure that represents the document of Fig. 1(a) is to extract an appropriate representative path from the corresponding tree of Fig. 1(b). Table 1 shows all the paths composed of consecutive nodes from the root to leaf and those nodes' names are renamed to distinguish them easily.

(Table 1) Paths of the XML tree

Sequence_ID	Original path	Renamed path
1	Club/club name	a/b_1
2	Club/member/name/last name	a/b_2/c_1/c_4
3	Club/member/name/first name	a/b_2/c1/c_5
4	Club/member/phone	a/b_2/c_2
5	Club/member/title	a/b_2/c_3

The longest frequent paths are used as the features of XML documents to cluster them based on the large items in the following way[8]. When the consecutive nodes from the root are only considered, the path 'a' is occurred 5 times, the path 'a/b2' is occurred 4 times, the path 'a/b2/c1' is occurred 2 times, and the remaining paths are occurred only once. As a path includes more lower nodes its occurrence frequency becomes large and its importance becomes high. Also, the long path may represent the contents of a document better than the short path does. Therefore, the longest path which satisfies the minimum support of a sequential pattern mining technique can be used as a representative path. When the minimum support is 2, the longest frequent path of Fig. 1(b) is 'a/b2/c1', i.e., 'club/member/name' that can represent the document of Fig. 1(a). However, the computation overhead is very large to extract a longest frequent path by a sequential pattern mining technique such as a PrefixSpan[10].

We, therefore, propose a new method to find the representative path of a XML document more efficiently. In each level of a XML tree, the node which represents the level is extracted to form a virtual path. Since the node which has many children can be considered as an important node, the node which has the most children among the nodes in each level is extracted as the representative node for that level except the last level. If more than one node satisfy the condition in the same level, then an arbitrary node is selected. For example, the <club> node is selected because there is only one node in the first level of Fig. 1(b). The <member> node is selected because it has the most children in the second level. The <name> node is selected in the same way in the third level. The last level is ignored because it only consists of leaf nodes which are not important in a general sense. As a result 'club/member/name' is selected as the virtual path for representing the document of Fig. 1(a). On the assumption that each node knows about its children information, the pseudo code to extract a virtual path is illustrated in Fig. 2. In this method, the representative node in each level is selected as a node is deleted one by one from the queue after all the nodes of the tree are inserted into queue once. Therefore, the time complexity of the algorithm is O(2n) when the total number of nodes of a tree is n.

```
Procedure find_virtual_path (XML_tree: tree)
begin
    insert(rootnode, Que);
    nextLevel_nodeNum := 1;
    while Que <> empty
    begin
        curtLevel_ nodeNum := nextLevel_
        nodeNum;
        nextLevel_ nodeNum := 0;
        Max := 0;
        while curtLevel_ nodeNum > 0
        begin
            delete(node, Que);
            curtLevel_ nodeNum --;
            // if a node has children then insert
            children nodes to queue
            if node has children
            for i : = 1 to childNum {
                insert(child, Que)
                nextLevel_ nodeNum ++;}
        // a representative node for each level
        is extracted
        if childNum > Max {
            Max := childNum;
            rep_node := node; }
      end
      print(rep_node);
    end
end
```

(Fig. 2) Pseudo code to extract a virtual path

4. XML Clustering

We propose a method to apply the well known hierarchical clustering algorithm when a representative path is used as the feature of a XML document. The hierarchical clustering algorithms produce a hierarchy of nested clustering. A clustering $\Re_1$ containing k clusters is said to be nested in the clustering $\Re_2$, which contains r(<k) clusters, if each cluster in $\Re_1$ is proper subset of $\Re_2$. 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 general agglomerative clustering algorithm is described in Fig. 3 when the total number of patterns is n.

```
① Begin with n clusters, each consisting of one
   pattern.
② Repeat step ③ a total of n-1 times.
③ Find the most similar clusters Ci and Cj and
   merge Ci and Cj into one cluster.
   If there is a tie, merge the first pair found.
```

(Fig. 3) Agglomerative clustering algorithm

One of the important issues for clustering process is how the similarity measure between patterns is quantified. We consider not only the node's name but also the node's position in the path to measure the similarity between XML documents. The similarity between the names of the two compared nodes can be obtained by the set of synonyms or thesauruses. For example, suppose that the names of the compared nodes are different such that an 'actor' in one node and a 'star' in the other node. Then, the system assigns a value from 0 to 1 to the name weight according to the conformance level of the two names after the synonyms of 'star' are extracted from a synonym database such as the WordNet[11]. However, even if the names of the compared nodes are the same, the weight can be different according to the positions of the nodes in the paths. This is because as the path to the node becomes short, more weight is assigned when the similar XML documents are searched[12]. Let the level weight of a node x_i, ($1 \le i \le n$) in a representative path is Lev_{x_i}, then the level weights satisfy the following conditions.

$$Lev_{x_1} \ge Lev_{x_2} \ge \ldots \ge Lev_{x_n} \text{ and } \sum_{i=1}^{n} Lev_{x_i} = 1 \qquad (1)$$

Let the representative paths X and Y of the two XML documents are $x_1 / x_2 / \ldots / x_n$ and $y_1 / y_2 / \ldots / y_m$, respectively, and the name weight of the two nodes x_i and y_j is $Name(x_i, y_j)$. Then, the similarity between two XML documents $Sim(X,Y)$ ($0 \le Sim(X,Y) \le 1$) is defined in the following way.

$$Sim(X,Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} Name(x_i, y_i) \times \min(Lev_{x_i}, Lev_{y_j}) \quad (2)$$

The reason of choosing the minimum value of the two level weights is that we want to reduce the influence of a weight as the difference between the levels of the compared nodes is large. 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 Ci and Cj are clusters, the distance between them is defined as:

$$d_{SL}(C_i, C_j) = \max_{X \in C_i, Y \in C_j} Sim(X,Y) \quad (3)$$

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 Ci and Cj are clusters, the distance between them is defined as:

$$d_{CL}(C_i, C_j) = \min_{X \in C_i, Y \in C_j} Sim(X,Y) \quad (4)$$

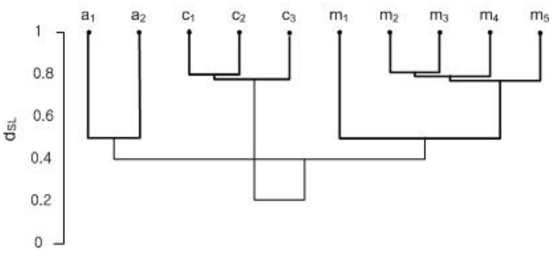
5. Clustering Experiments

We use the data provided by the XML data bank of the University of Wisconsin to measure the effectiveness of our algorithm[13]. This data bank provides several DTDs such as bibliography, club, company profiles, stock quotes, department, personal information, movies, and actors. Even though many XML documents can be generated from each DTD, the number of the representative paths available from those documents is limited. For example, the virtual paths available from the documents based on the club's DTD in Fig. 4 are 'club/member/name' and 'club/member/address'. On the other hand, the longest frequent paths available from the documents based on the club's DTD are 'club/member', 'club/member/name' and 'club/member/address' when the minimum support is 2.

```
Club
<?xml encoding="ISO-8859-1"?>
  <!ELEMENT club (clubname, member+)>
  <!ELEMENT clubname (#PCDATA)>
  <!ELEMENT member (name, phone, email,
  address?, title?)>
  <!ELEMENT name (lastname?, firstname)>
  <!ELEMENT address (city, state, zip)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT lastname (#PCDATA)>
  <!ELEMENT firstname (#PCDATA)>
  <!ELEMENT phone (#PCDATA)>
  <!ELEMENT email (#PCDATA)>
  <!ELEMENT city (#PCDATA)>
  <!ELEMENT state (#PCDATA)>
  <!ELEMENT zip (#PCDATA)>
```

(Fig. 4) Club's DTD

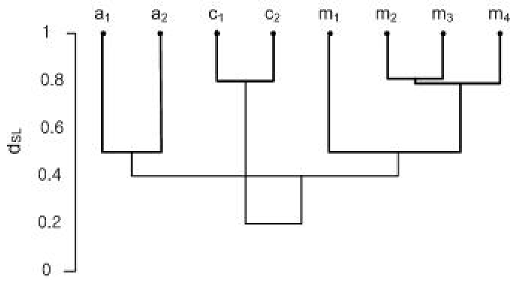
We experiment with the representative paths available from XML documents based on three DTDs of actor, club, and movies to see a hierarchical clustering process. When a path's length is 2, 3, or 4, the assigned level weight is (0.6, 0.4), (0.5, 0.3, 0.2), or (0.4, 0.3, 0.2, 0.1), respectively. The name weight between two compared nodes is set to 1 for convenience as long as the names of the two nodes are similar. The dendrogram of a virtual path by a single-linkage algorithm is shown in Fig. 5(a). The virtual paths available from the documents based on actor, club, and movies DTDs are renamed as (a1, a2), (c1, c2), and (m1, m2, m3, m4), respectively. The result of the clustering is {a1, a2}, {c1, c2}, and {m1, m2, m3, m4} when the three clusters are formed. We see that the clustering result is correct although there are some paths which are similar to each other but belong to different DTDs such as a2 = actor/filmography/movie and m1 = movie/cast/actor. The reason of the correct clustering regardless of those paths is that we consider how much the compared nodes' levels are close to each other as well as how much those names are similar to each other. The difference between the value of d_{SL} when three clusters are formed and that of d_{SL} when two clusters are formed is small, i.e., 0.1. The reason is that some paths which are similar to each other but belong to different DTDs such as a2 = actor/filmography/movie and m1 = movie/cast/actor are to be merged in the next step. This shows that a single-linkage algorithm has a tendency to favor elongated cluster. On the other hand, the dendrogram of a longest frequent path by a single-linkage algorithm is shown in Fig. 5(b). The longest frequent paths available from the documents based on actor, club, and movies DTDs are renamed as (a1, a2), (c1, c2, c3), and (m1, m2, m3, m4, m5), respectively. As we can see in Fig. 5, the clustering result of a longest path is almost the same as that of a virtual path except that the number of representative paths generated by a longest path is a little bit larger than that of a virtual path.



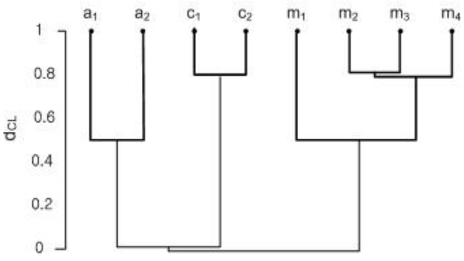
(b)

(Fig. 5) Dendrogram by a single-linkage algorithm

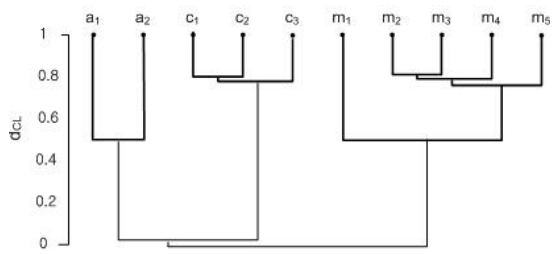
Fig. 6(a) is the dendrogram of a virtual path when a complete-linkage algorithm is used with the same data. When three clusters are formed, the clustering result is correct as {a1, a2}, {c1, c2}, and {m1, m2, m3, m4} although the order of the clusters' formation is different from that of the single-linkage algorithm. However, the difference between the value of d_{CL} when the three clusters are formed and that of d_{CL} when the two clusters are formed is large, i.e., 0.5. The reason is that there exists at least one path in each cluster which is not similar to others in other clusters. In this case, it is not desirable to merge those three clusters into two clusters. On the other hand, the dendrogram of a longest frequent path by a single-linkage algorithm is shown in Fig. 6(b). As we can see in Fig. 6, the clustering result of a longest path is almost the same as that of a virtual path.



(a)



(a)



(b)

(Fig. 6) Dendrogram by a complete -linkage algorithm

The hierarchical clustering algorithm does not necessarily produce the whole hierarchy of n clustering, but it can terminate when the clustering that best fits the data has been achieved according to some criterion[14]. For this, we define a function h(C), $(0 \le h(C) \le 1)$ that measures the dissimilarity between the documents of the same cluster C.

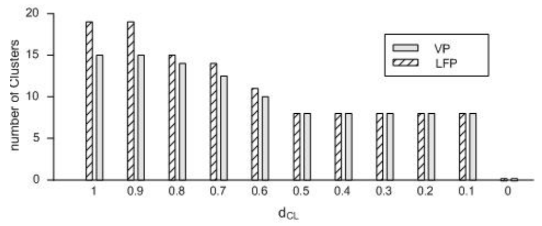
$$h(C) = \min_{X,Y \in C} Sim(X,Y) \qquad (5)$$

Let θ $(0 \le \theta \le 1)$ be an appropriate terminal threshold for the adopted h(C). Then, the algorithm terminates at the $\Re_t$ clustering if

$$\exists C_j \in \Re_{t+1} : \ h(C_j) < \theta \qquad (6)$$

In words, $\Re_t$ is the final clustering if there exists a cluster C in $\Re_{t+1}$ with dissimilarity between its patterns (h(C)) is less than θ. The dendrogram of Fig. 6 suggests that an appropriate value for θ is 0.5 when a complete-linkage algorithm is used.

Fig. 7 shows the number of the clusters formed according to the value of d_{CL} when 80 XML documents based on eight DTDs of the XML data bank are clustered. The number of the clusters formed by a virtual path(VP) is less than that of the clusters formed by a longest frequent path(LFP). This is because the number of the virtual paths available from a DTD is usually less than that of the longest frequent paths. Eight clusters are formed when the value of d_{CL} is 0.5 and all the clusters are merged into one when the value of d_{CL} becomes 0. Therefore, we know that the proper clusters are formed when a terminal threshold θ is set to 0.5 in case of our experimental data. It shows that true clusters are formed in a compact shape when a virtual path is used for the feature of a XML document and an appropriate terminal threshold can be obtained when a complete-linkage algorithm is used for XML clustering.



(Fig. 7) Number of the clusters

6. Conclusion

As the Internet continues to grow and evolve, more and more information is being placed in structurally rich documents such as XML. With the large number of documents on the Web, there is an increasing need to be able to automatically process those structurally rich documents for information retrieval and search applications. 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 and the efficient storage of XML documents, and even for system protection purpose because unusual document can be discovered easily.

This paper proposes a new method to cluster XML documents efficiently. We propose a new representative path called a virtual path which represents both the structure and the contents of a XML document for the feature of a XML document. We then propose a method to apply the well known hierarchical clustering algorithm to those representative paths to cluster XML documents. The experiment shows that an appropriate terminal threshold can be obtained to terminate the clustering process automatically and the true clusters are formed in a compact shape when a complete-linkage algorithm is used. We also show that it is much more efficient when a virtual path is used as a representative path becuase the time complexity to extract a virtual path is only O(2n) when the total number of nodes of a XML tree is n. In the future, we plan to investigate a method to consider the node which has the most decendants as the representative node for each level of the virtual path.

References

[1] R. Behrens, "A Grammar Based Model for XML Schema Integration," Proc. of the 17th British National Conf. on Databases, pp.172-190, 2000

[2] 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.

[3] 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.

[4] 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.

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

[6] H. Lee, "An Unsupervised Clustering Technique of XML Documents based on Function Transform and FFT," Journal of Korea Information Processing Society, 2007

[7] 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.

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

[9] K. Wang, C. Xu, and B. Liu, "Clustering Transactions Using Large Items," Proc. of ACM CIKM-99, 1999

[10] Jian Pei, and etc., "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth," Proc. 17th International Conference on Data Engineering, pp.215－224, April 2001.

[11] http://wordnet.princeton.edu/

[12] U. Park, and Y. Seo, "An Implementation of XML Document Searching System based on Structure and Semantics Similarity," Journal of Korean Society for Internet Information, Vol.6, No.2, April 2005.

[13] Niagara Query Engine, http://www.cs.wisc.edu/niagara/data.html

[14] Boberg J., and Salakoski T. "General formulation and evaluation of agglomerative clustering methods with metric and non-metric distances," Pattern Recognition, Vol.26(9), 1993.

◐ 저 자 소 개 ◑

김 우 생
1985년 U. of Texas at Austin 전산학과 졸업(학사)
1987년 U. of Minnesota 대학원 전산학과 졸업(석사)
1991년 U. of Minnesota 대학원 전산학과 졸업(박사)
1992～현재 광운대학 컴퓨터소프트웨어 학과 교수
관심분야 : 데이터베이스, 멀티미디어
E-mail : kwsrain@kw.ac.kr