An Adaptive School Web Site Construction Algorithm Using Association Rules

Jeongmin Lee[†], *and Woochun Jun*[††]

[†]*Yongin Kogi Elementary School, Yongin, Kyunggido, Korea*
[††] *Dept. of Computer Education, Seoul National University of Education, Seoul, Korea*

Summary

Advances in Web technologies have been changing school environment. Especially, school Web site is now a representative communication tool for teachers, students, and parents, etc. As more people are visiting school Web sites, it is necessary to provide right and up-to-date information by analyzing requirements and characteristics of visitors. In this paper, an adaptive school Web site construction algorithm is designed and implemented. For the proposed algorithm, meaningful behavior patterns are extracted by applying association rule, one of the data mining techniques, to log data. Based on the analyzed behavior patterns, the proposed algorithm is designed to reflect pattern documents on the school Web Site. It is proved that performance of the proposed system is better than that of the existing best algorithm. The proposed algorithm can provide popular information to visitors. Also, the proposed system can provide information associated each other to those visitors. In addition, teachers responsible for Web site management can be relieved from a tedious task such as periodic site update.

Key words

Web mining, Adaptive Web

1 Introduction

Due to the significant investment into the informatization of elementary education by the government during the 1990's, most schools have constructed their Web sites, which provide a wide variety of educational contents to the students and parents[6]. So, the school Web site needs to be automated from the analysis of visitors access patterns to the reflection of that pattern to the school Web site. Because firstly, for a web master the automation will lighten the burden of having to update the contents constantly. Secondly, the time consumed in analyzing and updating the Web sites will be greatly reduced. Thirdly, the ability to react to the ever-changing requirements and characteristics of the visitors will be much more effective. Fourthly, the Web site will be able to link related contents to the visitors, thus greatly increase the effectiveness of the way visitors can search for content. Lastly, a Web master may manage the sites scientifically in adding, deleting, updating of the school Web site by using extracted patterns.

The aims of this research are to design an algorithm that will extract to the related Web documents by analyzing the visitors access patterns, and to propose how to construct a school adaptive Web site that will reflect the pattern document on school Web site automatically. The proposed algorithm in this paper converts the data from a Web log file into a binary tree to the optimum pattern search and extracts the associated Web documents from the tree. The extracted pattern is then serviced to visitors as lists of recommendations. The above series of works will be implemented as an application program.

The remainder of the paper is structured as follows. In Section 2, we will present an overview of the existed adaptive Web site construction algorithms. In Section 3, we will explain the mining process by using the association rules. In Section 4, the proposed algorithm will be analyzed theoretically.. And the conclusion and future works will be presented in Section 5.

2 Related Works

Adaptive Web sites are defined by Perkowitz and Etzioni as Web sites that automatically improve their organization and presentation by learning from visitor access patterns[10]. Related works regarding adaptive Web sites are as such.

2.1 The PageGather Algorithm

Perkowitz and Etzioni have proposed the 'PageGather Algorithm' that uses the clustering mining[10]. Table 1 describes the four stages of the PageGather Algorithm. This algorithm uses a Web log access file and the basic information of clustering as inputs. As a result of this algorithm, candidate index-page contents are obtained.

The PageGather Algorithm contributed much to the development of adaptive Web site studies. The algorithm

Manuscript received January 5, 2008
Manuscript revised January 20, 2008

explains how to analyze visitor's access patterns and proposes the idea of synthesizing Web pages automatically by using the index page. However, this algorithm does not take into account the hyperlink's infrastructure, so meaningless sites, having high hits, were included in mining process.

Table 1. The PageGather Algorithm[10]

Step 1. Process the access log into visits.
Step 2. Compute the co-occurrence frequencies between pages and create a similarity matrix.
Step 3. Create the graph corresponding to the matrix, and find cliques (or connected components) in the graph.
Step 4. For each cluster found, create a Web page consisting of links to the documents in the cluster.

2.2 Web Documents Recommendation Algorithm

In [5], Web documents recommendation algorithm used clustering algorithm and Apriori algorithm without any modifications. This algorithm as in Table 2 recommends the frequent Web document to users.

However, the above algorithm did not take into account the hyperlink infrastructure in its filtering process therefore, the sites without intent of visit have a high value number of support. Furthermore, the Apriori is famous for its founding of the association rules but we can point out two short comings that it needs to scan the database multiple times and it has to generate a huge number of candidates[7]. Recently, diverse algorithms such as the FP-growth have been developed to reinforce the Apriori algorithm[8]. Because it furnishes a key for the construction of adaptive Web site to extract association rules effectively from a massive amount of information, it is necessary to modify the Apriori algorithm with it.

Table 2. Web Documents Recommendation Algorithm[5]

```
String[] RecomPages(String requested) {
  //Input : requested, Output : recoms[]
    int num=0;
    float th = 0.5;
    String[] recoms;
    Vector[] page_conf;
        page_conf = recomPage(requested);
        num = page_conf.length;
            if(num==1) return recoms;
            else if(num>1) {
                comp_threshold(page_conf, th);
                recoms = sort();
                return recoms;
            }
            else return null;
}
```

2.3 Web Documents Recommendation Algorithm Using Markov Chain

In case study [4], we can observe that the Markov chain has had developed for a considerable amount of change to have become stabilized. It is called the stable state. The algorithm in case study [4] proposed a Web document recommendation system that shows a meaningful document set to visitors automatically by using Markov properties. Table 3 is the algorithm proposed by case study [4]. This algorithm using the Markov chain has been known to mine information previously inaccessible by using the anti-frequent-item-set method.

However, the above algorithm requires extensive calculations of the conditional probability of each page. Recent trends of school Web sites carrying various contents and services, thus surmising a massive Web database, development of an algorithm that will reduce the amount of calculations is paramount.

Table 3. Web Documents Recommendation Algorithm Using Markov Chain[4]

(1) Conversion of log access file.
(2) Construction of matrix by calculating the support
(3) Calculation of the stable state probability by using Markov chain
(4) Decision of a candidate solution by using the founded frequent itemset.

2.4 TPA (Traversal Pattern Analysis) Algorithm

The suggested Traversal Pattern Analysis algorithm in [1] is a modified algorithm of the Apriori to extract the sequential pattern. Table 4 shows the TPA algorithm's forward phase and its backward phase.

When examining the TPA's forward phase, TPA forms a length-1-frequent itemset. Let L_1 be the complete set of length-1 frequent itemsets. Then the algorithm creates the set of length-2 candidates, denoted as C_2, and is generated from L_1. After scanning the database once more to count the support value of each itemset in C_2, The itemsets in C_2 passing the support threshold form the length-2 frequent itemsets, L_2. A similar process goes on until no candidate can be derived or no candidate is frequent. In the forward phase, similar candidate itemsets combination is avoided, in the backward phase, the TPA uses a function to reduce the cost of scanning the database. However, even the TPA algorithm is considered to have problems regarding time and space management due to constant database scans and the creation of candidate itemsets.

Therefore, this research gleans the meaningful Web document set from Web log access file, and proposes way

that will reduce the number of scans, so that the mining process will become more effective and streamline.

Table 4. TPA Algorithm[1]

```
/* Forward Phase */
L1 ={large 1-sequences};
C1 = L1;
last = 1;
For (k=2;Ck-1 ≠ {} and Llast ≠ {} ; k++) do
    Begin
    If (Lk-1 ≠{}) then Ck ={New candidates generated from Lk-1};
    Else Ck ={New candidates generated from Ck-1};
    If (k==Next(last)) then
        Begin
          Forall data sequence c in the data-sequences do
                Increment the count for all candidates in Ck that are contained in c
          Lk = candidates in Ck with minimum support
          last = k;
        End
    End

/* Backward Phase */

For (k=last ; k >=1; k--) do
  If (Lk not found in forward phase) then
     Begin
     Delete all sequences in Ck contained in some Li, i>k;
     Forall data sequence c in the data-sequences do
              Increment the count for all candidates in Ck that are contained in c
     Lk = candidates in Ck with minimum support.
     End
  Else /* Lk already known */ Delete sequences in Lk contained in some  Li, i>k;
     Answer = UkLk
Function Next(k: integer)
Begin
     If (hitk > Y) return k+2
     Else return k+1
End
```

3 Frequent Web Document Pattern Mining Method using Association Rule

Adaptive School Web site divides the processes into, the preprocessing of the access log file, the construction of the pattern tree from scanning of the database, the mining through the pattern tree, the storing and renewing process of the mined pattern, and lastly the presentation of the hyperlink on the Web site.

3.1 The Preprocessing of the Log File

The log file in its raw format consists of large mounts of unnecessary data for pattern mining. Therefore to insure an effective mining process, log file has to go under a preprocessing phase. The transaction database is formed from the preprocessing phase. However, it is necessary to have two hypotheses as thus, before the formation of transaction database

Identification of the Visitor : Each IP address will be treated on one situation. Therefore each different IP address will be treated as a new visitor.

Identification of the Session : the series of hits in the user specified maximum time gap will be considered to be a single session. When a visitor has exceeded his or her time gap, a new hit will induce a new session, resulting in a new transaction.

The preprocessing is divided in itself into two phases. In the first phase, the log file is cleaned up. And unwanted information and repeated information are discarded. In the second phase, the information resulting from the backtracking function of the Web-browser is eliminated. Table 5 is an example of a transaction database after the prepossessing.

Table 5 Transaction database

Transaction ID	Raw Transaction Database	Reconstructed Transaction Database
100	ABACGJGCF	BJF
200	ABEBADH	EDH
300	ACGJGI	ACJI
400	HDBABABACCCF	HDBCF
500	EBACGHGI	EBACHI
600	ACFCGI	AFGI
700	DABEBACF	DECF
800	ABEBACF	ECF
900	ADHDACGJ	HCGJ

If you follow the transaction ID number 100 in Table 5, we can see that the visitor had traversed the Web pages in the following path: ‘A-B-A-C-G-J-G-C-F’. (Web document are represented by letters) The path is transformed into the tree structure shown in Fig. 1. The algorithm will pick out 'B', 'J', 'F' as the meaningful documents in pattern mining. The algorithm will deduct in the sequence 'A-B-A', the user wanted data 'B', and in the sequence of 'C-G-J-G-C', the user went through the process to ultimately access data 'J'.

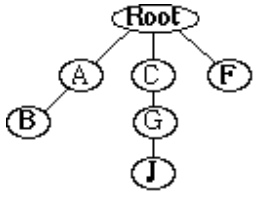

Fig. 1. Web document traversal tree in TID 100

3.2. The Formation of a Pattern Tree

Frequent Web document pattern mining consists of two phases. Firstly, an FWDP-tree is created with a preprocessed transaction database. Secondly through the traversal of FWDP-tree that has been already created, the frequent Web document pattern mining occurs.

The FWDP-tree consists of the header table, the root node, and the Web document-prefix-sub-tree as a child of root. The header table is created to help the traversal of the FWDP-tree, and is consisted of three fields. The first one is the document name field, which is used to save the name of frequent Web document. The second one is the head of the node-link field, which is used to store the link information. In the FWDP-tree formation process, if a new node is created, pointer information is stored to indicate the new node in this field. The other is the last of the node-link field, which is used to store the information indicating to the last node created. Regarding the colossal size of the data involved in pattern mining, it is time consuming and inefficient to write out the node-link fields of the node considering the whole pattern tree when a new node is inserted into the tree. To solve this problem, the FWDP-tree uses the last node field. That field has the pointer information of the latest node added in tree. So when a node with the same name is added, the node-link field of node may be modified easily.

The node of FWDP-tree is a structure, which are commonly organized into five different fields as in Fig. 2. The document name field is used to store the Web document name. The count field is used to store the number of transactions on the database. The parent link and the children link each pertain the parent and children node, and the node link field point to the nodes that have the same name as each other, and if there are not any that have the same name, the node link field gets the value 'null'. The FWDP-tree algorithm scans the database two times to load the information needed for the frequent pattern mining on memory.

Because the loaded information is huge, and the Web documents have the complicated reference relationship, we cannot accurately predict the number of referring to Web pages for each Web documents. The reason why is the number of linked nodes to the FWDP-tree node is not constant to each other. To remedy this, this research is proposing a binary tree that has a maximum number of nodes at two, unlike the FP-tree algorithm[8]. Therefore, the children-link field of FWDP-tree node is divided intotwo links, the child-link and the sibling-link.

If the same data are in transaction database, the FWDP-tree algorithm does not create a new node but change the count field. This is the Web document-prefix-subtree. The FWDP-tree will be smaller than the database at least by sharing nodes with repeated data. Table 6 describes the FWDP-tree algorithm.

Table 6. FWDP-tree Algorithm

```
Input : A transaction database DB and a minimum support threshold SUP_min
Output : FWDP-tree
Step1. Scan the transaction database DB once.
        Collect F. /* F is the set of frequent document */
        Sort F in support-descending order as FList.
        Create FArray. /* FArray is the array of FList */
Step2. Create the Header Table, H
       /* H is consist of three fields. H_name is the field of
          Document-Name. H_first is the first node link area.
          And H_last is the last node link area. */
       Set H_first with FList.
       Create the root of an FWDP-tree, T
Step3. For each transaction Trans in DB do {
          Select the frequent document in Trans and
                  sort them according to the order of FList.
          Let the sorted frequent-document list in Trans be [p|P]. /* p is the element of P */
                  For each p do {
                      Insert_tree(p, T)
                  }
          }
Function insert_tree(p, T) {
          If T has a child N such that N.Document-name=p.Document-name
                  then increment N's Count by 1.
          else {  Create a new node newNode.
                   Insert a newNode as the last right child of root.
                  After that, a newNode becomes a left child of parent node.

                  Count =1 ;
                  Children-link=null ;
                  Parent-link=parent node ;
        }
```

In step one of the algorithm, through a database scan, a frequent Web document set ***F*** is formed and the Web documents of ***F*** are sorted according to their support value in descending order. In step two, we can see that the root node is created, as the first operation of the FWDP-tree, ***T***. The root node has a null value. In step three, FWDP-tree algorithm is performed iteratively as the following. Firstly, the algorithm selects the frequent Web document in a transaction and sorts the documents according to their support in descending order. And then the ***intert_tree()*** function is constantly recalled to form the FWDP-tree.

In Table 5, if the threshold support is 3, ***F*** would be {(C.6), (F.5), (E.4), (A.3), (B.3), (D.3), (I.3)}. Afterward the algorithm will scan each transaction and match the frequent Web document list (C, F, E, A, B, D, I) and organize it accordingly.

Table 7 is reconstructed from <Table 5> to illustrate that the FWDP-tree is better. Fig. 2 is to illustrate the FWDP-tree based on Table 7. For practical reasons the header table is given in Fig. 2(i).

Table 7. Transaction sequence and sorted frequent Web documents

Transaction ID	Transaction Sequence	Sorted frequent Web document
100	B J F	F B
200	E D H	E H D
300	A C J I	C A I
400	H D B C F	C F H B D
500	E B A C H I	C E H A B I
600	A F G I	F A I
700	D E C F	C F E D
800	E C F	C F E
900	H C G	C H

3.3. Mining the Pattern Tree

By Using the FWDP-tree formed in the section above and the minimum threshold support, pattern is mined. In the adaptive Web site, to induce relevant frequent Web document pattern mining, a FP-growth algorithm[9] is modified and then applied.

This research in creating an adaptive school Web site, takes great interest in the Web document that the visitors have frequently visited. In this research, we design that an adaptive school Web site will allow a much more interactive, convenient and helpful Web site to its visitors through finding out the Web documents associated with a frequent Web document. Therefore to be effective, one has to find the Web site 'X' that is frequently visited, and the Web site 'Y' which is frequently visited by visitor who have visited Web site 'X.' Table 8 is FWDP-mine algorithm applied in pattern mining. According to the frequent Web document list, the complete set of frequent Web document can be divided into subsets (7 for our example) without overlap.

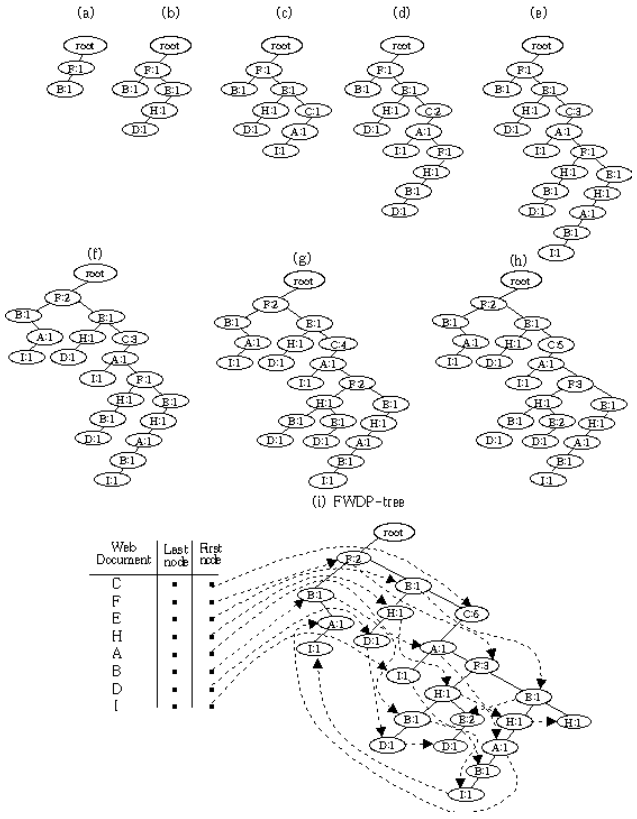

Fig. 2. FWDP-tree

Table 8. FWDP-mine Algorithm

```
Procedure FWDP-mine(){
    for each a_i do {  /* FList = a_1, a_2, ... a_n (1≤i≤n) */
        Pattern P={ } /* P = { (x_1,s_1), (x_2,s_2), ..., (x_n,s_n),}
x is node name, s is support */
        forall the path of a_i do {
            Let sup(a_i) be the support of a_i on the path.
            ptr = parent node of a_i. /* ptr is a pointer */
                forall each node from ptr to root do {
                    if    (ptr=ptr.parent-link.child-link)
then {
                        Create pattern node(x_ptr,s_ptr)
                         if  (node(x_ptr,s_ptr)=node(X,S)  in  P)
then S += s_ptr
                         else insert node(x_ptr,s_ptr)in P
                    }
                    ptr = ptr.parent-link
                }
        }
        Organize the frequent pattern set freq_pattern for a
a_i
    }
    return freq_pattern
}
```

First frequent Web document sets having 'I' are mined. And then the sets having document 'D' but not 'I' are mined. In this manner FWDP-mine algorithm is performed iteratively to mine patterns for each frequent Web documents. So in case Fig. 2, it is repeated eight times.

After processing the FWDP-mine algorithm, the F and freq_pattern are presented in Table 9. We can see that four frequent Web documents were mined. Examining the algorithm process, people who have visited Web site 'I,' also tended to visit Web site 'A' and people who have visited Web site 'C' tended to visit Web site 'H,' 'E,' and 'F'.

Table 9. The Set P and the maximum length frequent Web document pattern

Web Doc	P	freq_pattern
I	{(C:2),(A:3),(E:1),(H:1),(B:1),(F:1)}	{(A:3),I}
D	{(E:2),(H:2),(C:2),(F:2),(B:1)}	Ø
B	{(F:2),(C:2),(H:2),(E:1),(A:1)}	Ø
A	{(C:2),(E:1),(H:1),(F:1)}	Ø
H	{(E:2),(C:3),(F:1)}	{(C:3),H}
E	{(C:3),(F:2)}	{(C:3),E}
F	{(C:3)}	{(C:3),F}
C	Ø	Ø

3.4. Storing of the Pattern and Updating the Site

This process consists of the processes of storing the mined patterns and the automatic integration of the information into the school Web sites. The mined patterns are stored in pattern tables having formats such as Table 10.

Table 10. The example of pattern table

Document	Pattern
I	A
A	I
H	C
E	C
F	C

The pattern saving module, if there are more than one patterns extracted, will sort the patterns into descending order and save it into the database.

The mined pattern is serviced to the visitors in various forms. Some ways could be by the use of index pages that has no need of destroying the original Web infrastructure, or by encoding and inserting hyperlinks, or by changing the font size, type or color [2,3]. By using the index page method, one can avoid destroying the Web site's infrastructure or change the content. However, in recent PC oriented environment, for personal ease or for security reasons pop-ups are sometimes disabled and frequent index pages may cause the visitors to tire from having to click the mouse button frequently and having a hard time navigating.

To remedy this, in this research, we chose the method of inserting hyperlinks into the Web document. This method by having hyper-links inserted into the Web document can keep the Web site design congruent from before and can overcome the shortcomings of creating index pages.

In another words, when one visits Web site 'X,' the host, by consulting the Web server database pattern table, the server examines for related Web site 'Y'. If Web site 'Y' exists then the title of 'Y' would be added into 'X' with hyper-link information. Fig. 3 demonstrates the new hyperlink infrastructure.

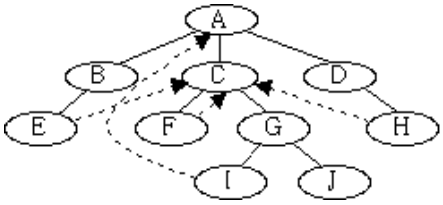

Fig. 3. Web site infrastructure and pattern link

4 Analysis of Pattern-mining Algorithm

FWDP-mine algorithm is a modification of the FP-growth algorithm for the use of school Web sites. But in the FWDP-mine algorithm does not have the 'Growth'[7] idea of the association rule patterns that the FP-growth used. Instead the FWDP-mine algorithm only calculates the maximum length patterns. This is so to maximize the capabilities to use the hyper-link method of updating the adaptive school Web site.

Theorem: the maximum length frequent Web document pattern extraction of FWDP-mine takes considerably less time complexity than when the FP-growth is applied.

Proof: let us assume that a certain FP-tree's ***tree*** existed and that a Sup_{min} existed. '***F***' is a set of frequent Web documents and $\boldsymbol{F}$= { a_1, a_2, ⋯, a_n }. For some frequent Web document, a_i (only if, $a_i \in \boldsymbol{F}$, $1 \leq i \leq n$), path(a_i) will be all possibilities including path a_i. Also as the current time pathNode(a_i) will mean all possibilities between the root node and the a_i 's parent node. The sum of all the root of the path of a_i will be pathNode(a_i) ($k \geq 1$), k (only if, k>=1), and set 'm', the number of node in the k's nodes as the least possible the minimum support value.

Even in a worse case scenario in the FWDP-mine algorithm, the time complexity is O(n×k). On the other hand, in the FWDP-tree, even when dealing with a single path situation, the time calculation comes out to be O(n×(2m-1)) even in a best case scenario. Therefore FWDP-mine algorithm is better than FP-growth to mine the conditional pattern.

5 Conclusions and Future Research

In this research, we proposed an algorithm that can extract the user's access patterns and a method in which to apply it into the adaptive school Web sites. From the Web site that the visitors frequently visit, we were able to extract the associated sites that are important and reflected the information gathered back into the Web sites by the means of hyper-links, thus providing the users, a more efficient and user-friendly venue.

The suggested FWDP-tree algorithm transforms a raw data into a suitable data structure. And FWDP-mine algorithm mines the frequent Web document patterns through traversing the FWDP-tree. This algorithm does not create multiple database scans nor does it create candidates, yet able to mine the maximum length patterns. Mined frequent pattern is a set of the associated Web documents and a list of recommendation. The research, for the construction of an adaptive Web site create a pattern miner application. Developed pattern miner automates a series of process from the preprocessing of a log file to the updating a school Web site.

Through the adaptive Web site used in school Web sites, users will be able to experience short-cut access interfaces and the provider will be able to experience less workload concerning data analysis, maintenance and presentation.

In the future, the research has to earn support and confidence from its users, and use it to further mine behavior pattern mine and integrate it into the adaptive school Web site. In addition, to develop an algorithm that will personalize the Web data pattern and to present a personal Web site will be our task for the future.

References

[1] Ko, K., A Study on Adaptive Web Site Construction by Analyzing User Access Patterns. Master's Thesis, Kyonggi University, Suwon, Kyunggido, Korea (2001)

[2] Kim, J., Data Analysis Using Web Mining. Master's Thesis, University of Seoul, Seoul, Korea, (2001)

[3] Song, M., A Study on Efficient Data Structure for Mining Association Rules. Master's Thesis, Hongik University, Seoul, Korea (2001).

[4] Lee, S., A study for adaptive Web-site construction through Web-mining. Master's Thesis, Hanyang University,, Seoul, Korea, (2000).

[5] Lee, Y., A personalized Web recommender system using clustering and association rules. Master's Thesis, Inha University, Inchon, Korea, (2002)

[6] Lee, J., Design and implementation of a School Web-board for Adaptive Web Site Construction, Master's Thesis, Graduate School of Seoul National University of Education, Seoul, Korea, (2004).

[7] Kantardzic, M. Data Mining, WILEY-INTERSCIENCE, (2003)

[8] Han, J. and Pei, J. Mining Frequent Patterns by Pattern-Growth : Methodology and Implications. ACM SIGKDD, Vol. 2,. Issue. 2, (2000) 14-20.

[9] Pei, J. Pattern-growth Methods for Frequent Pattern Mining, The Degree of Doctor of Philosophy, Simon Fraser University, Vancouver, BC Canada, (2002)

[10] Perkowitz, M. and Etzioni, O. Adaptive Web Site: Automatically Synthesizing Web Page. IJCAI:Proceedings of the conference, Vol.15, No. 1, (1998) 727-732

Jeongmin Lee received the Bachelor and Master degrees, from Korea National University of Education and Seoul National University of Education, in 1997 and 2005, respectively. She has been a teacher at Yongin Kogi Elementary school since 2006. Her research interests include web mining and web-based instruction.

Woochun Jun has been an a professor in Dept. of Computer Education at Seoul National University of Education , Seoul, Korea, since 1998.. His areas of interest include web-based instruction, mobile learning, web mining, semantic web. He holds a Ph.D. degree in Computer Science from University of Oklahoma, USA in 1997. He also received a Master's degree and BS degree in Computer Science from Sogang University, Seoul, Korea, in 1987 and 1985, respectively.