Huang, Y.-M., Chen, J.-N., & Cheng, S.-C. (2007). A Method of Cross-level Frequent Pattern Mining for Web-based Instruction. *Educational Technology & Society*, 10 (3), 305-319.

A Method of Cross-level Frequent Pattern Mining for Web-based Instruction

Yueh-Min Huang[1], Juei-Nan Chen[1] and Shu-Chen Cheng[2]
[1]Department of Engineering Science, National Cheng Kung University, Taiwan // Tel: +886-6-2757575 ext. 63336 // Fax: +886-6-2766549 // huang@mail.ncku.edu.tw // rnchen@gmail.com
[2]Department of Computer Science and Information Engineering, Southern Taiwan University of Technology, Taiwan // Tel: +886-6-2533131 ext. 3228 // kittyc@mail.stut.edu.tw

ABSTRACT

Due to the rise of e-Learning, more and more useful learning materials are open to public access. Therefore, an appropriate learning suggestion mechanism is an important tool to enable learners to work more efficiently. A smoother learning process increases the learning effect, avoiding unnecessarily difficult concepts and disorientation during learning. However, many suggestion demands come from different abstraction levels, and traditional single level frequent pattern mining is not sufficient. This paper proposes a methodology for mining frequent patterns of learners' behavior which connote a hierarchical scheme to provide cross-level learning suggestions for the next learning course. With this system, a learner can get multiple levels of abstract suggestions instead of merely single level frequent pattern mining results. Our study shows that the algorithms can mine considerable quantities of frequent patterns from real life learning data. The experimental data are collected from a Web learning system originating from National Cheng Kung University in Taiwan. The proposed methodology gives learners many suggestions to help them learn more effectively and efficiently. Finally, we collect some representative cases to realize different requirements which are extracted from a learners' access database. These cases are classified into three types; notably, type three generalized four meaningful external factors which are inferred by our observations from these cross-level frequent patterns.

Keywords

e-Learning, Cross-level frequent pattern, FP-tree, FP-growth, Data mining

1. Introduction

At the dawn of the 21st Century, the education landscape is changing, in large part due to the explosive growth of Web technology. Many of the traditional institutions of higher education - universities, colleges, and professional development centers - are now beginning to develop and deliver Web-based courses via the Internet (Chen et al., 2005b; Choi, 1999; Huang et al., 2004; Khalifa and Lam, 2002; Jeng et al., 2005; Pahl, 2003; Song et al., 2004). Therefore, groups such as SCORM (2004), IEEE LTSC (2002), IMS (2003), and AICC (2003) have undertaken significant work on learning objects schemas. A learning object is a modular data unit that encapsulates information to describe a concept, skill, operation or procedure. Accordingly, these learning objects can be easily reused and shared over the Internet.

Although numerous Web-based learning systems have been developed, the vast number of learning objects have created disorientation and cognitive overload (Berghel, 1997; Borchers et al., 1998), so that learners may find it hard to know what course to take. To increase the learning effect, many powerful assistant mechanisms have been proposed. Huang et al. (2006) proposed a sequential mining algorithm to analyze learning behaviors to discover frequent sequential patterns. By these patterns, the system can provide suggestions for learners to select their learning contents. Papanikolaou et al. (2002) presented an approach to adaptive hypermedia learning environments design and illustrated how learners' knowledge level and individual traits can be exploited to guide the adaptive dimension of a hypermedia system. Tang and Mccalla (2003) proposed an evolving Web-based learning system which can adapt itself not only to its users, but also to the open Web in response to the usage of its learning materials. Cheng et al. (2005) presented an automatic leveling system for an e-Learning examination pool using the algorithm of the decision tree. The automatic leveling system is built to automatically level each question in the examination pool according its difficulty. Thus, an e-Learning system can choose questions that are suitable for each learner according to individual background. Chen et al. (2005a) proposed a personalized e-Learning system based on IRT (Item Response Theory) providing learning paths that can be adapted to various levels of difficulty of course materials and various abilities of learners. Karampiperis and Sampson (2005) presented an alternative sequencing method, instead of generating the learning path by populating a concept sequence with available learning resources

ISSN 1436-4522 (online) and 1176-3647 (print). © International Forum of Educational Technology & Society (IFETS). The authors and the forum jointly retain the copyright of the articles. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear the full citation on the first page. Copyrights for components of this work owned by others than IFETS must be honoured. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from the editors at kinshuk@ieee.org.

based on pre-defined adaptation rules, it first generates all possible learning paths that match the learning goal, and then selects the desired one to estimate the suitability of learning resources for a targeted learner.

Consider an e-Learning system in which various learning objects are provided for the learners via the Internet. A learner may not easily acquire suitable learning objects or may lose his/her way in this vast space. Therefore, providing suitable learning suggestions for the learners becomes an important mechanism in a comprehensive web learning system. In this study, we attempt to use the idea of Han and Fu's (1999) multiple-level hierarchy information encoded method. Following the hierarchical taxonomy, an encoded string needs fewer bits than the corresponding object-identifier. Moreover, the connotation of an encoded string and their correlations may lead to the discovery of more specific and important knowledge from the data. Relevant item taxonomies are usually predefined in real-world applications, and can be represented using hierarchy trees. Terminal nodes on the trees represent actual items appearing in transactions; internal nodes represent classes or concepts formed by lower-level nodes. Unlike Han and Fu (1999), this study attempts to find the frequent patterns with crossing-level relations rather than look for the association rules in a specific level.

Since the early work in Agrawal et al. (1993), various studies have adopted or modified the Apriori candidate set generation-and-test approach to mine association rules. Kim and Kim (2003) proposed a recommendation algorithm using multi-level association rules to predict user's preference, and then suggest items by analyzing the past preference information of users. Bonchi et al. (2003) introduced a fast algorithm for frequent pattern mining that exploits anti-monotone and monotone constraints to optimize a level-wise, Apriori-like computation. Kaya and Alhajj (2004) presented a fuzzy weighted multi-cross-level association rules mining approach which integrates several concepts, including fuzziness, cross-level mining, weighted mining and linguistic terms. However, Apriori-like candidate set generation is still costly, especially when there are many patterns and/or long patterns.

In this study, we apply a frequent pattern tree (FP-tree) structure, which is an extended prefix-tree structure for storing compressed, crucial information about frequent patterns, and employ FP-growth to mine the complete set of frequent patterns by pattern fragment growth (Han et al., 2000). The FP-growth method can get more efficient and scalable results for mining both long and short frequent patterns than the Apriori algorithm, and it is also faster than some recently reported new frequent pattern mining methods. This study provides a useful application applying data mining methodology in the e-Learning field. Our study shows that cross-level frequent pattern mining can give learners various learning suggestions. Furthermore, it succeeds in joining the methods of multiple-level hierarchy information encoded technique and frequent pattern mining, offering a new utilization in data mining field.

We ran experiments using data from an e-Learning system originating at National Cheng Kung University in Taiwan (NCKU, 2005). The data were daily collected from January 1, 2005 to December 31, 2005. Although the system is still serving, we retrieved the log data from the database and evaluated its features. At the end of 2005, the system contained 15,858 students, 528 lecturers, and 1,916 courses.

This paper has three primary research contributions:
1. Integrating multi-level hierarchy information encoded technique and frequent pattern mining method to provide learning suggestions;
2. Conducting some experiments to evaluate the proposed methodology using real-world data and realize what learners want;
3. Collecting some representative cases to understand different learning requirements.

The rest of this article is organized as follows. Section 2 shows the taxonomy of the curriculum scheme. Section 3 introduces the FP-tree construction process, and a method of cross-level frequent pattern mining is proposed in Section 4. The experimental results are presented in Section 5 and discussed in Section 6. Finally, Section 7 draws conclusions.

2. Taxonomy of the Curriculum Scheme

In the first place, the taxonomy information of the curriculum scheme should be constructed, and each position in this academic hierarchy should be given a unique encoded string which requires fewer bits than the corresponding accessing-identifier. We assume that the database contains: 1) a learning item data set which contains the description

of each learning item in I in the form of (ES_i, $description_i$), where encoded string $ES_i \in I$, 2) an access database, ADB_{origin}, which consists of a set of user access data (UID_i, $\{ES_x, \ldots, ES_y\}$), where UID_i is a user's identifier and $ES_i \in I$ (for $i=x, \ldots, y$), and 3) the notation of '*' represents a class or concept in an encoded string ES_i. To clarify our discussion, an abstract example is introduced in the following sections.

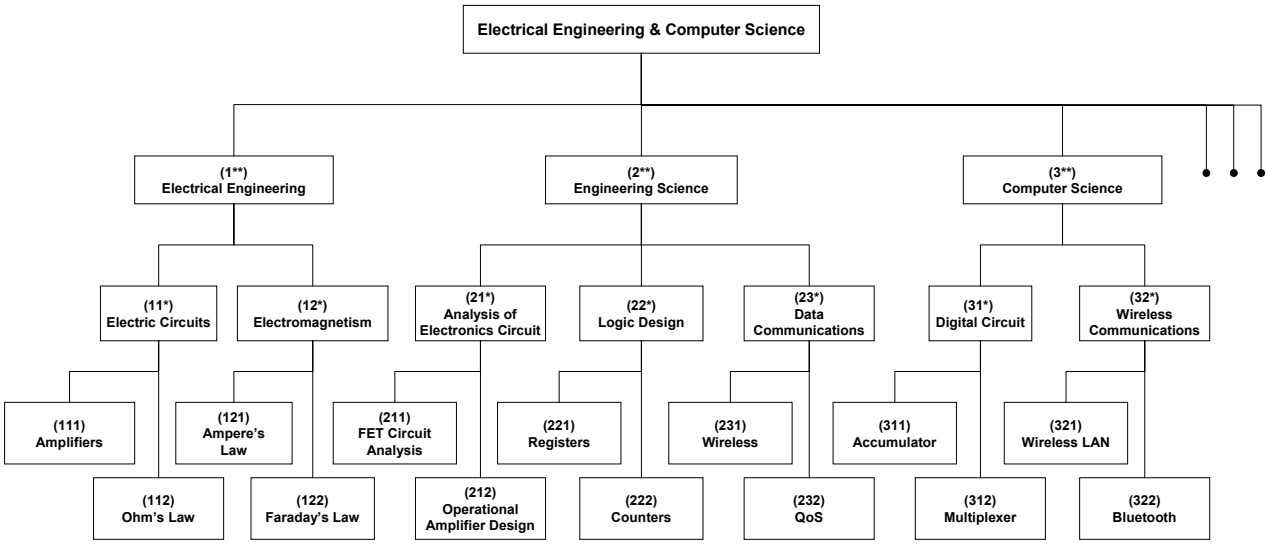

Figure 1. An example of a curriculum scheme

Example 1: For instance, the taxonomy information of the curriculum scheme is constructed as shown in Figure 1, and the encoded access database ADB_{origin} is shown in Table 1. We use a unique encoded string to represent a corresponding accessing-identifier, which is shown in the parenthesis in front of each accessing-identifier. In this example scheme, the Electrical Engineering & Computer Science curriculum is given in three layers. The first layer enumerates all the departments in this Electrical Engineering & Computer Science curriculum. The second layer lists all the courses in the corresponding departments. The third layer itemizes all the learning items including in the courses. For instance, the learning item 'Wireless' is encoded as '231' in which the first digit, '2', represent 'Engineering Science' at layer 1, the second, '3', for 'Data Communications' at layer 2, and the third, '1', for the learning item 'Wireless' at layer 3.

Table 1. An example of user access database ADB_{origin}.

UID	Items
U_1	{231, 321, 322, 312}
U_2	{111, 112, 212, 321, 322, 231}
U_3	{322, 321, 111, 212, 231}
U_4	{111, 222, 112, 212,211}
U_5	{112, 211,212, 311}
U_6	{111, 311, 212, 112, 231}
U_7	{231, 121, 321}

In this research, the frequent patterns are not restricted to the same layer (Han and Fu, 1999) but extend to the different layers. For this reason, we increase the higher-level items in the access database ADB_{origin} to be an enriched access database, $ADB_{enriched}$. Whenever the lower-level items achieve a specified portion of count, the higher-level items should be added into the enriched access database $ADB_{enriched}$ (shown in Table 2). For example, the access data U_6 in Table 2 contains the encoded items - 111, 112, 121, and 122, and these items fulfill the state we mentioned.

Therefore, the higher-level items of 11* and 12* should be added into $ADB_{enriched}$. Moreover, 1** also should be added into $ADB_{enriched}$ as a result of the 11* and 12* being satisfied with the same condition.

Table 2. An example of enriched access database $ADB_{enriched}$.

UID	Items
U_1	{231, 321, 322, 312, 32*}
U_2	{111, 112, 212, 321, 322, 231, 11*, 32*}
U_3	{322, 321, 111, 212, 231, 32*}
U_4	{111, 222, 112, 212, 211, 11*, 21*}
U_5	{112, 211,212, 311, 21*}
U_6	{111, 311, 212, 112, 231, 121, 122, 11*, 12*, 1**}
U_7	{231, 121, 321}

Definition 1: An encoded string set, *ES*, is one item ES_i or a set of conjunctive items $ES_i \cap \cdots \cap ES_j$, where $ES_i,...,ES_j \in I$. The *support* of a frequent pattern *X* in a set *S*, $\rho(X/S)$, is the number of transactions (in *S*) which contain *X* versus the total number of transactions in *S*. The *confidence* of $X \to Y$ in *S*, $\delta(X \to Y/S)$, is the ratio of $\rho(X \cap Y/S)$ versus $\rho(X/S)$, i.e., the probability that pattern *Y* occurs in *S* when pattern *X* occurs in *S*.

To find relatively frequently occurring patterns and reasonably strong rule implications, a user or an expert may specify two thresholds: minimum support, *minsup*[*l*], and minimum confidence, *minconf*[*l*], where *l* specifies different level of a curriculum scheme. Notice that for finding cross-level frequent patterns, different minimum support and/or minimum confidence can be specified at different levels.

The qualification of the cross-level frequent item sets proceeds as follows. Let the minimum support at level 1, 2 and 3 be 3, 3 and 4 access data respectively (i.e., minsup[1]=3, minsup[2]=3, and minsup[3]=4). For the sake of simplicity, notice that since the total number of access data is fixed, the support is expressed as an absolute value rather than a relative percentage. In accordance with each level's minimum support threshold, the items which are lower than the minimum support should be filtered out (presented in a single deletion line), and the results are shown in Table 3. For instance, the learning item '322' occurs in U_1, U_2 and U_3, and the minimum support at level 3 is 4 (minsup[3]=4). Hence, the learning item '322' should be removed, as its occurrence is lower than its threshold level.

Table 3. A filtered example of enriched access database $ADB_{enriched}$.

UID	Items
U_1	{231, 321, ~~322~~, ~~312~~, 32*}
U_2	{111, 112, 212, 321, ~~322~~, 231, 11*, 32*}
U_3	{~~322~~, 321, 111, 212, 231, 32*}
U_4	{111, ~~222~~, 112, 212, ~~211~~, 11*, ~~21*~~}
U_5	{112, ~~211~~,212, ~~311~~, ~~21*~~}
U_6	{111, ~~311~~, 212, 112, 231, ~~121~~, ~~122~~, 11*, ~~12*~~, ~~1**~~}
U_7	{231, ~~121~~, 321}

3. FP-tree Construction

In this section, we employ the frequent pattern tree (FP-tree) structure to compress a large database into a highly condensed, much smaller data structure, which avoids costly, repeated database scans (Han et al., 2000). If two access data share a common prefix, according to some sorted order of frequent items, the shared parts can be merged using one prefix structure as long as the count is registered properly. If the frequent items are sorted in descending order of their frequency, there is a better chance that more prefix strings can be shared. Figure 2 shows the algorithm

of FP-tree construction from Han et al. (2000). We use the same example from Section 2 to illustrate the FP-tree construction process.

The frequent items in each access data are listed in descending order, and shown in Table 4. Let the minimum support of a frequent pattern be 3 (i.e., minsup[FP] = 3). With these observations, one may construct a frequent pattern tree as follows. First, a scan of sorted $ADB_{enriched}$ derives a list of frequent items, (212:5), (231:5), (111:4), (112:4), (321:4), (11*:3), (32*:3), (the number after ":" indicates the support), in which items are in order of descending frequency. This ordering is important, since each path of a tree will follow this order.

Definition 2: A frequent pattern tree (or FP-tree in short) is a tree structure as defined below.

1. It consists of one root labeled as "*null*", a set of item prefix sub-trees as the children of the root, and a frequent-item header table.
2. Each node in the item prefix sub-tree consists of three fields: *item-name*, *count*, and *node-link*, where *item-name* registers which item this node represents, *count* registers the number of transactions represented by the portion of the path reaching this node, and *node-link* links to the next node in the FP-tree carrying the same item-name, or null if there is none.
3. Each entry in the frequent-item header table consists of two fields, (1) *item-name* and (2) *head of node-link*, which points to the first node in the FP-tree carrying the *item-name*.

Based on this definition, we have the following FP-tree construction algorithm.

Figure 2. The algorithm for constructing the FP-tree (Han et al., 2000)

Algorithm 1 (FP-tree construction)
Input: An enriched access database $ADB_{enriched}$ and a minimum support threshold minsup[FP].
Output: Its frequent pattern tree, FP-tree.
Method: The FP-tree is constructed in the following steps.

1. Scan the access database $ADB_{enriched}$ once. Collect the set of frequent items F and their supports. Sort F in descending order as L, the list of frequent items.
2. Create the root of an FP-tree, T, and label it as "*null*". For each access data *Acc* in $ADB_{enriched}$ do the following.
Select and sort the frequent items in *Acc* according to the order of L. Let the sorted frequent item list in *Acc* be $[p|P]$, where p is the first element and P is the remaining list. Call *insert_tree*($[p|P]$, T).
The function *insert_tree*($[p|P]$, T) is performed as follows. If T has a child N so that *N.item-name = p.item-name*, then increment N's count by 1; otherwise create a new node N and let its count be 1, its parent link be linked to T, and its node-link be linked to the nodes with the same *item-name* via the node-link structure. If P is not empty, call *insert_tree*(P, N) recursively.

Table 4. A filtered example of enriched access database $ADB_{enriched}$ with sorted items

UID	Items
U_1	{231, 321, 32*}
U_2	{212, 231, 111, 112, 321, 11*, 32*}
U_3	{212, 231, 111, 321, 32*}
U_4	{212, 111, 112, 11*}
U_5	{212, 112}
U_6	{212, 231, 111, 112, 11*}
U_7	{231, 321}

Second, one may create the root of a tree, labeled "*null*". Scan the $ADB_{enriched}$ a second time. The scan of the first access data leads to the construction of the first branch of the tree:<(231:1), (321:1), (32*:1)>. Notice that the frequent items in the access data are ordered according to the order in the list of frequent items. For the second access of the data, we construct the second branch of the tree: <(212: 1), (231:1), (111:1), (112:1), (321:1), (11*:1), (32*:1)>. For the third access of the data, since its (ordered) frequent item list <212, 231, 111, 321, 32*> shares a

common prefix <212, 231, 111> with the existing path <212, 231, 111, 112, 321, 11*, 32*>, the count of each node along the prefix is increased by 1, and one new node (321:1) is created and linked as the child of (111:2) and another new node (32*:1) is created and linked as the child of (321:1). Working with the same process, we can get the data structure which is shown in Figure 3 until we have scanned all the access data in Table 4.

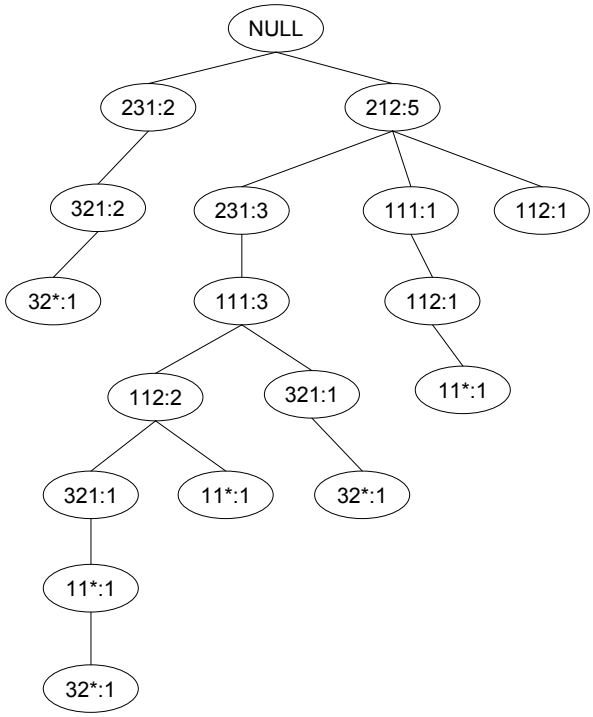

Figure 3. The FP-tree in Example 1

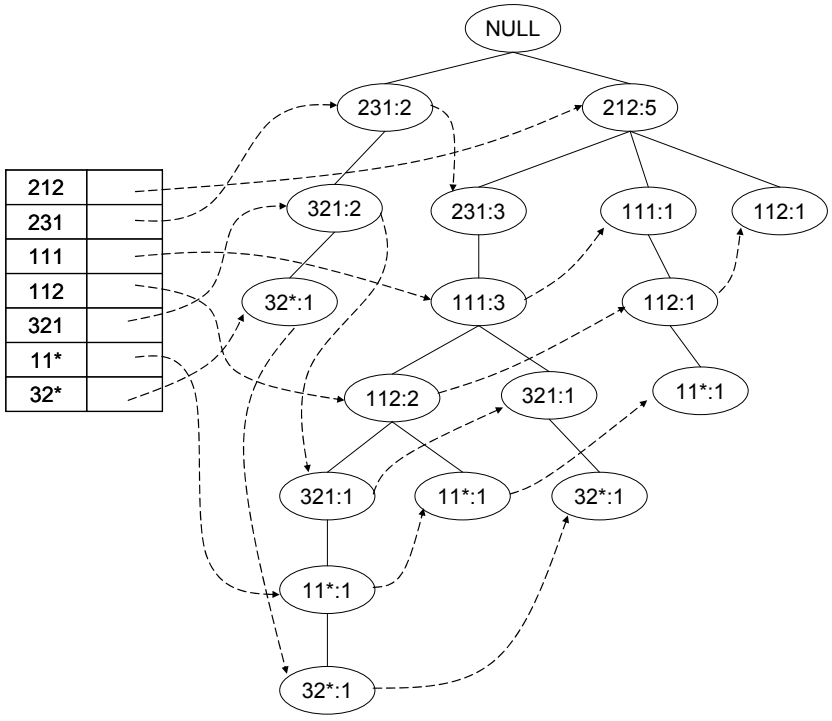

Figure 4. The FP-tree with header table in Example 1

Third, to facilitate tree traversal, an item header table is built in which each item points to its occurrence in the tree via a head of node-link. Nodes with the same item-name are linked in sequence via such node-links. After scanning all the access data, the tree with the associated node-links is shown in Figure 4.

4. Mining Association Rules using the FP-tree

In this section, we study how to explore the compact information stored in an FP-tree, mine the frequent patterns with an efficient method based on Han et al. (2000) as shown in Figure 5, and then generate the association rules from conditional frequent patterns.

I={ES_1, …, ES_n} is a set of learning items. α and β are frequent learning items in $ADB_{enriched}$. P is a single path of the FP-tree structure. The support (or occurrence frequency) of a pattern A, which is a set of items, is the number of transactions containing A in $ADB_{enriched}$. A is a frequent pattern if A's support is no less than a predefined minimum support threshold, minsup[FP].

Algorithm 2 (FP-growth: Mining frequent patterns with FP-tree by pattern fragment growth)
Input: FP-tree constructed based on Algorithm 1, using $ADB_{enriched}$ and a minimum support threshold minsup[FP].
Output: The complete set of conditional frequent patterns.
Method: Call FP-growth (FP-tree, *null*).

Procedure FP-growth (*Tree*, α)
{
(1) **if** *Tree* contains a single path *P*
(2) **then for each** combination (denoted as β) of the nodes in the path *P* **do**
(3) generate pattern $\beta \cup \alpha$ with *support = minimum support of nodes in* β ;
(4) **else for each** ES_i in the header of *Tree* **do** {
(5) generate pattern $\beta = ES_i \cup \alpha$ with *support* = ES_i.*support* ;
(6) construct β 's conditional pattern base and then β 's conditional FP-tree $Tree_\beta$;
(7) **if** $Tree_\beta \neq \phi$
(8) **then** call FP-growth ($Tree_\beta$, β) }
}

Figure 5. The FP-growth algorithm (Han et al., 2000)

Keeping the previous example, we examine the mining process by starting from the bottom of the header table. For node 32*, it derives a frequent pattern (32*: 3) and three paths in the FP-tree: <231: 2, 321: 2, 32*: 1>, <212: 5, 231: 3, 111: 3, 112: 2, 321: 1, 11*: 1, 32*: 1> and <212: 5, 231: 3, 111: 3, 321: 1, 32*: 1>. The third path indicates that string "(212, 231, 111, 321, 32*)" appears once in the access database. Notice that although string <231, 111> appears three times and <212> itself appears five times, they only appear once together with 32*. After proceeding the remaining node of 11*, 321, 112, 111, 231 and 212, the conditional pattern bases and conditional frequent patterns can be generated and are shown in Table. 5.

Figure 6 shows the association rule generation process from conditional frequent patterns. First, the frequent access rules are generated by combining all the items in each conditional frequent pattern, and pruning the duplicate rules. Next, in each frequent access rule, all the corresponding encoded digits except the last digit should be compared. If the compared results are all equal, this rule should be pruned off; otherwise, it should be added into the pruned frequent access rule table. Finally, the pruned frequent access rules can be formed as association rules by inserting an implication relation between each gap. Furthermore, the remaining gaps of each association rule should be laid down with the interconnection relation of '∩' which ties the related items together. In Table 6, the conditional frequent patterns can be broken into many frequent access rules. If the frequent access rules already exist, the rules produced

later should be omitted (presented in a single deletion line). Furthermore, since each learning item is represented by an encoded string, we can easily deduce their correlations. Since we want to provide useful recommendations to the learner, it should be omitted from the rules of great generality, especially since these items lie in the same category. Therefore, we prune the rules presented in a double deletion line.

Table 5. Mining of all-patterns by creating conditional (sub)-pattern bases

Item	Conditional pattern base	Conditional frequent pattern
32*	{(231: 1, 321:1, 32*: 1), (212: 1, 231: 1, 111: 1, 112: 1, 321: 1, 11*: 1, 32*: 1), (212: 1, 231: 1, 111: 1, 321: 1, 32*: 1)}	{(231: 3, 321: 3)}| 32*
11*	{(212: 1, 231: 1, 111: 1, 112: 1, 321: 1, 11*: 1), (212: 1, 231: 1, 111: 1, 112: 1, 11*: 1), (212: 1, 111: 1, 112: 1, 11*: 1)}	{(212: 3, 111: 3, 112: 3)}| 11*
321	{(231: 2, 321: 2), (212: 1, 231: 1, 111: 1, 112: 1, 321: 1), (212: 1, 231: 1, 111: 1, 321: 1)}	{(231: 4)}| 321
112	{(212: 2, 231: 2, 111: 2, 112: 2), (212: 1, 111: 1, 112: 1), (212: 1, 112: 1)}	{(212: 3, 111: 3)}| 112
111	{(212: 3, 231: 3, 111: 3), (212: 1, 111: 1)}	{(212: 3, 231: 3)}| 111
231	{(212: 3, 231: 3)}	{(212: 3)}| 231
212	ϕ	ϕ

Algorithm 3 (Association rule generation process from conditional frequent patterns)
Input: Conditional frequent patterns.
Output: Association rules.
Method: The association rule generation process from conditional frequent patterns is constructed in the following steps.

1. Scan the conditional frequent pattern table once. Combine all the encoded strings, *ES*, in each conditional frequent pattern to generate frequent access rules (denoted as τ). If the frequent access rule has been generated, then ignore reproduced rule.
2. **for each** τ_i **do**
   if all the corresponding encoded digits except the last one are the same **then**
      Prune τ_i;
   else add τ_i to pruned frequent access rule (denoted as ω);
3. **for each** ω_j **do**
   Insert the implication relation between each gap, and the remaining gaps lay down the interconnection relation '∩';

Figure 6. Association rule generation algorithm

5. Experimental Results

In this section, we study the learning results by varying different database parameters. All experiments are performed on a computer with a CPU clock rate of 2.8GHz and 1 GB of main memory. The program is written in Java programming language and running on Microsoft Windows XP. In this study, we change the following parameters to observe the data precision and the learning effects, which are (1) the average length of each sequential transaction,

(2) the upgrading proposition of each level, (3) confidence analysis, and (4) the accuracy between multiple support threshold and single support threshold.

Table 6. Association rules generation from conditional frequent pattern

Conditional frequent pattern	Frequent access rules	Pruned frequent access rules	Association rules
{(231: 3, 321: 3)}| 32*	231, 321 231, 32* 321, 32* 231, 321, 32*	231, 321 231, 32* ~~321, 32*~~ 231, 321, 32*	231 → 321 231 → 32* 231 → 321∩32* 231∩321 → 32*
{(212: 3, 111: 3, 112: 3)}| 11*	212, 111 212, 112 212, 11* 111, 112 111, 11* 112, 11* 212, 111, 112 212, 111, 11* 212, 112, 11* 111, 112, 11* 212, 111, 112, 11*	212, 111 212, 112 212, 11* ~~111, 112~~ ~~111, 11*~~ ~~112, 11*~~ 212, 111, 112 212, 111, 11* 212, 112, 11* ~~111, 112, 11*~~ 212, 111, 112, 11*	212 → 111 212 → 112 212 → 11* 212 → 111∩112 212∩111 → 112 212 → 111∩11* 212∩111 → 11* 212 → 112∩11* 212∩112 → 11* 212 → 111∩112∩11* 212∩111 → 112∩11* 212∩111∩112 → 11*
{(231: 4)}| 321	~~231, 321~~	ϕ	ϕ
{(212: 3, 111: 3)}| 112	~~212, 111, 112~~	ϕ	ϕ
{(212: 3, 231: 3)}| 111	212, 231 ~~212, 111~~ 231, 111 212, 231, 111	212, 231 231, 111 212, 231, 111	212 → 231 231 → 111 212 → 231∩111 212∩231 → 111
{(212: 3)}| 231	~~212, 231~~	ϕ	ϕ

The historical access data was drawn from an e-Learning system of National Cheng Kung University in Taiwan (NCKU, 2005). At the end of 2005, the system contained 15,858 students, 528 lecturers, and 1,916 courses.

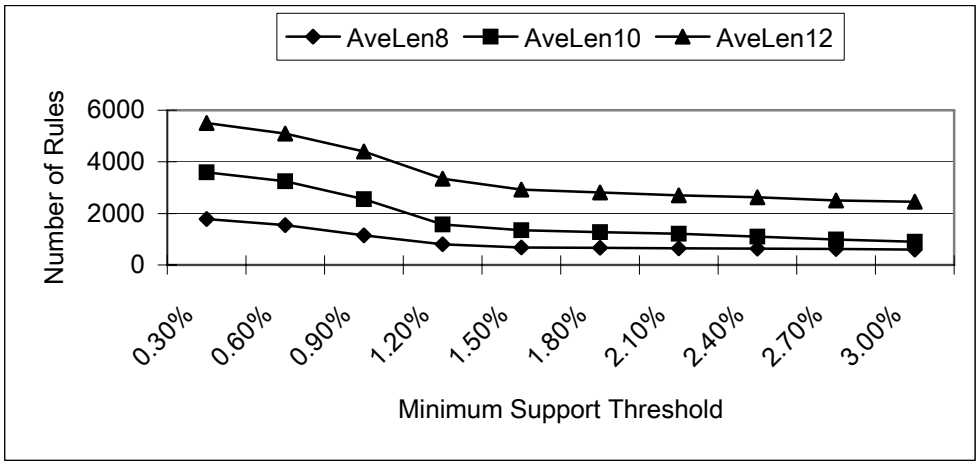

Figure 7. Average length with variant sequential transaction lengths

5.1 The Average Length of each Sequential Transaction

This experiment wants to observe the influence between the sequential transaction length and number of mining results. We fix the average data length up with 8, 10 and 12 items in a transaction. Figure 7 shows the correlation between minimum support threshold and number of rules. It is clear that the higher minimum support threshold value leads to the fewer number of rules. There is a tardy trend with the curve of AveLen8, and the visible decrease continued until the minimum support threshold reached 1.2%. Likewise, the curve of AveLen10 became gentle after the minimum support threshold also arrived at 1.2% as well. Finally, the curve of AveLen12 fell lightly after the minimum support threshold approached 1.5%.

5.2 The Upgrading Proposition of each Level

In this experiment, we want to realize each level's upgrading influence related to the number of rules. The upgrading proposition is highly correlated with the amount of higher-level items in the enriched access database $ADB_{enriched}$. Conversely, if the upgrading proposition holds great values, it will cause fewer higher-level items to exist in the enriched access database $ADB_{enriched}$. Hence, the higher learning concepts can not be extracted easily. This experiment arranges 160,000 sequential transactions which indicate the number of access data in the database. Figure 8 shows that the curve of Chapter -> Course decreases slightly since the value of the upgrading threshold is higher than 0.7. The curve of Course -> Department clearly diminishes before the upgrading threshold reaches 0.5. The curve of Department -> College decreases slowly after the upgrading threshold reaches 0.3.

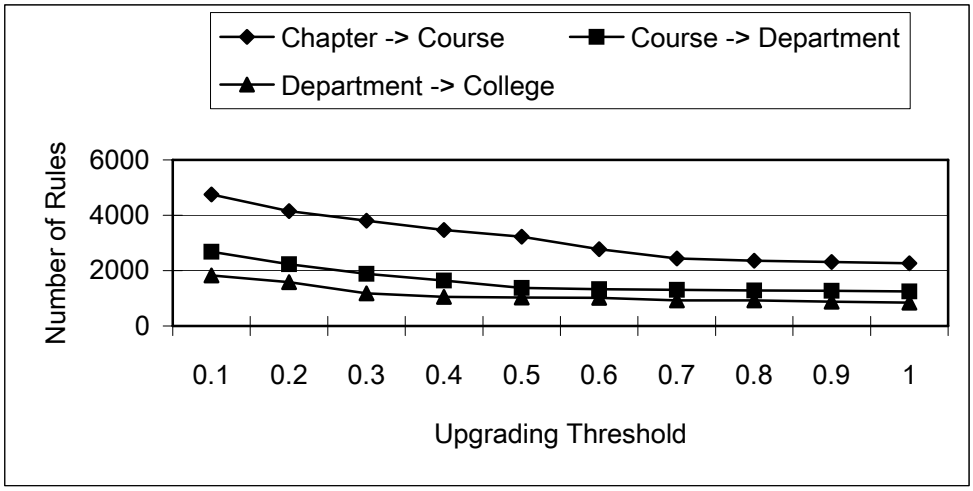

Figure 8. Upgrading threshold values with variant academic hierarchy levels

5.3 Confidence Analysis

The confidence value is the reliability of an association rule. In this experiment, we study the correlations between different confidence values and the number of rules. A higher confidence value leads to a fewer number of rules. In this experiment, when the minimum support of frequent pattern (minsup[FP]) is higher than 5%, then the number of rules decreases slightly, as shown in Figure 9. Therefore, if the system wants to provide various recommendation results, the minimum support value should not be higher than 5%. Because of the higher minimum support of a frequent pattern (minsup[FP]), more connoted rules could be filtered.

5.4 The Accuracy between Multiple Support Threshold and Single Support Threshold

We divide the data into training and testing samples to evaluate the precision and the recall of this recommendation mechanism. Precision is the number of relevant items retrieved over the total number of items retrieved. Recall is the

number of relevant items retrieved over the number of relevant items in the database. In this experiment, we use different recommendation threshold (confidence) values to evaluate the accuracy of the mining results, and fix the minimum support threshold value at 0.8%. A higher precision value means that the results are more accurate for the user. In Figure 10, the precision value clearly rises after the recommendation threshold (confidence) value reaches 0.3. We can easily recognize that frequent patterns with multiple support thresholds would get better results than when every hierarchy level holds the same support value.

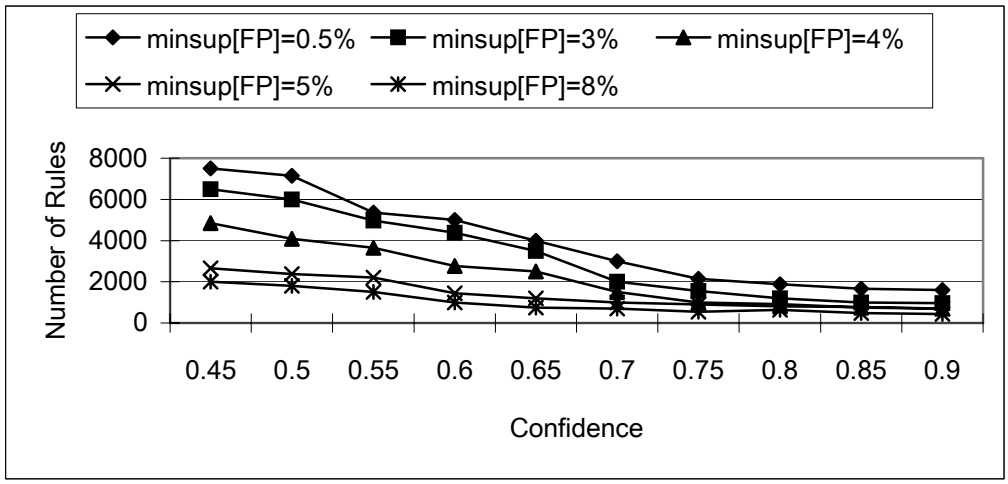

Figure 9. Confidence values with variant minimum support threshold values

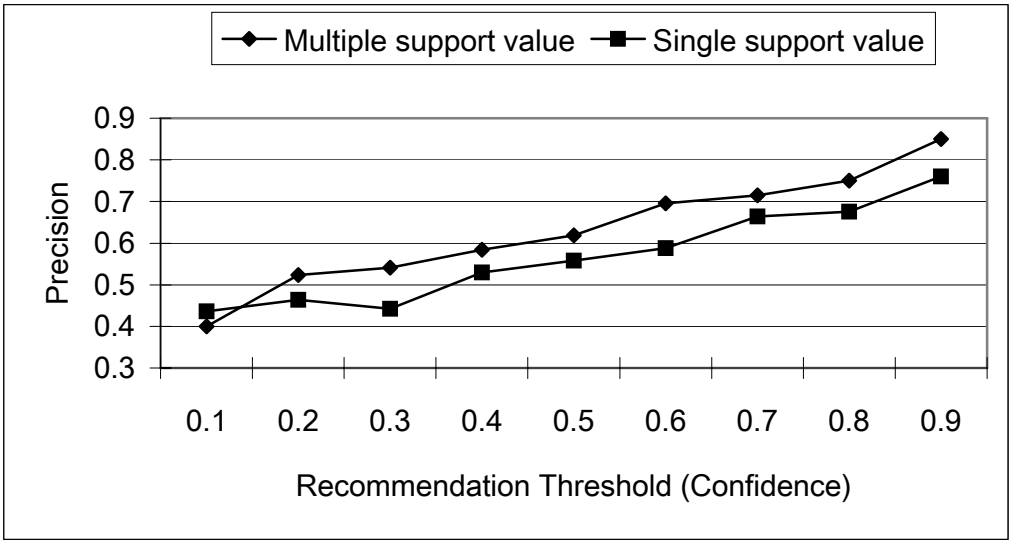

Figure 10. The precision with variant confidence values

In Figure 11, the recall is also mainly influenced on the recommendation threshold (confidence) value. The higher recommendation threshold (confidence) value means more restriction of each frequent pattern. Therefore, the higher recommendation threshold (confidence) value causes the lower recall percentage. Figure 11 shows that the recall percentage with multiple support thresholds is still 50%, since the threshold value is lower than 0.5.

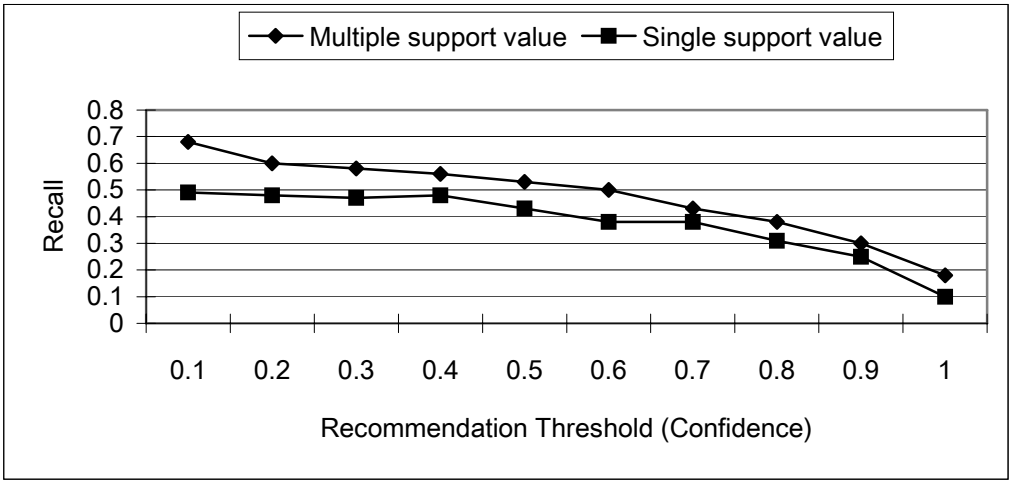

Figure 11. The recall with variant confidence values

6. Discussion

From the research results, we collect some representative cases to realize different requirements which are extracted from the learners' access database. These cross-level frequent patterns can be roughly divided into three kinds.

Type 1: High correlation between similar learning items

The meaningful type of frequent patterns is when the learning items are related to the similar concepts. This type of frequent patterns is 37% of the mining results, and is also the most important proportion of learners' behavior. The main reason is some courses contain the same learning concepts, and these learning items are explored by experienced learners. Whenever these kinds of patterns come up frequently, the mining process can easily find them from the access database. Learners can go a step further to study similar courses which involve similar learning materials as soon as they can not understand the meaning of a specific topic. For example, courses created by teachers using the authoring tool can be categorized with titles like 'Neural Networks' and 'Artificial Intelligence', etc. However, these courses involve many reduplicated concepts, but such learning items are associated with different hierarchy branches. This type of frequent pattern mainly collects the lower level relations of this academic hierarchy. This often takes place when the learner has an explicit learning target or they know what to learn, therefore, they can clearly find the related learning items to clarify and extend their learning.

Type 2: High correlation between complementary learning items

Another type of frequent pattern is learning items which have complementary relations. This kind of frequent patterns accounts for 9.3% for our experimental results. Excision is also a useful learning methodology when you want to understand a specific area of knowledge. By following an indirect approach, learners can get knowledge in a comparative way. This type of learning works best with the courses those are complementary with each other. For instance, the course of 'Solid State Physics' is theoretical, and 'Processing of Ceramics' is applied scientific knowledge. These two courses may contain the same subject matter, but presented in different ways. We think these two courses thus have a complementary relationship. Similar with type 1, the learner can also clearly identify the learning targets. Accordingly, the frequent patterns also primarily focus on the lower level relations of the academic hierarchy.

Type 3: Special frequent patterns which may imply particular factors

The greatest numbers of frequent patterns are this type. We generally categorize the remainder of rules as this type, which may not be classified into type 1 or type 2. The rules in this type may provide learners with unexpected

results, and these results are not restricted to a specific level but provide level crossing relations. Unlike type 1 and type 2, most of frequent patterns in this group seem to have occurred in a haphazard way; nevertheless, some of their relations can be figured out. These relations may be connected with certain external factors in the environment. In the following, we summarize several meaningful external factors (not including all of external factors) and give illustrations to show the mining results from this group.

1) **A time-related factor**: This refers to a point in time, that is, an important occurrence may cause some learners to have the same behavior at that particular period. Therefore, in a specified time, we can collect these meaningful rules. For example, enteroviruses are small viruses that are made of ribonucleic acid (RNA) and protein, and the infections are most likely to occur during the summer. Furthermore, these viruses easily multiply in a wet and hot environment. Therefore, in summer time, a learner may study a course which is talking about the Infectious Diseases. They may associate it with environmental sanitation, and then start to survey the courses in the Department of Environmental Engineering. In this case, a cross-level frequent pattern is "Infectious Diseases (course level) → Environmental Engineering (department level)".
2) **A policy or political affairs alternation**: This type of pattern is influenced by a public policy or political affairs. For instance, in 2005, TANFB (Taiwan Area National Freeway Bureau) announced that the national highway will put the ETC (Electronic Toll Collection) system into practice by January 1, 2006 with BOT (Build-Operate-Transfer) policy. The ETC is a system which allows drivers with an OBU (On-Board Unit) installed in their vehicles to pay their tolls electronically, without having to stop at the tollgates. FETC (Far Eastern Electronic Toll Collection) company was commissioned to build the system with infrared DSRC (Dedicated Short Range Communications) standard in 2003. Based on this policy implementation, after studying the course of Data Communications, which includes the descriptions of infrared DSRC, some learners then checked the courses of Infrastructural Engineering and Management. In this case, a cross-level frequent pattern is "Data Communications (course level) → Infrastructural Engineering and Management (course level)".
3) **Natural disasters**: There have been many disasters over the years resulting from its geographical location. Taiwan has a very special geographical position, and is often threatened by natural disasters such as earthquakes, typhoons, and floods. As a result of the data collected from universities in Taiwan, we conclude that natural disasters are an important factor. For instance, the 2004 Indian Ocean earthquake, known by the scientific community as the Sumatra-Andaman earthquake, was an undersea earthquake that occurred on December 26, 2004. The earthquake was reported as 9.3 on the Richter scale. The sudden vertical rise of the seabed by several meters during the earthquake displaced massive volumes of water, resulting in a tsunami that struck the coasts of the Indian Ocean. This event may cause a learner studying the Geology course to survey the Introduction of Natural Disasters course, and then inquire into the International Perspectives on Natural Disasters course. In this case, a cross-level frequent pattern is "Geology (course level) ∩ Introduction of Natural Disasters (course level) → International Perspectives on Natural Disasters (course level)".
4) **Mass media report or daily life**: Sometimes, our thinking is deeply influenced by the mass media or our daily life. For example, Artificial Intelligence (also known as machine intelligence and often abbreviated as AI) encompasses computer science, neuroscience, philosophy, psychology, robotics, and linguistics, and is devoted to the reproduction of the methods or results of human reasoning and brain activity. There was famous science fiction film that was released in 2001, which was called A.I.: Artificial Intelligence, and directed by Steven Spielberg. This film used lots of digital image processing and took an imaginative view of AI. This may cause a learner who is taking the AI course to then study about digital image processing in Department of Industrial and Commercial Design. In this case, a cross-level frequent pattern is "Artificial Intelligence (course level) → Industrial and Commercial Design (department level)".

7. Conclusions

A method of cross-level frequent pattern mining has been described, which extends the scope of mining frequent patterns from the same level of a hierarchy to the concept of mining learning relations among a curricular taxonomy. In this study, we integrate a multi-level hierarchy information encoded technique and frequent pattern mining method to provide users with experienced learning suggestions. The FP-tree construction algorithm, FP-growth algorithm and a complete case example clarify this research idea. We also conducted some experiments to evaluate the proposed methodology using real-world data from an e-Learning system of National Cheng Kung University in Taiwan. Based on the mining results, we generalized three types of frequent patterns, and reasoned out why the patterns happen and what the learners need. Such a learning mechanism is essential in this age of Web-based instruction, where there are more and more reusable learning materials shared over the Internet. Learners can not

easily take lessons in a good learning sequence, and may lose their way in this vast space. Our study initiates an innovative research direction of e-Learning field by applying data mining techniques, and gives learners many suggestions to help them learn more effectively and efficiently.

Acknowledgment

The authors would like to thank the National Science Council of the Republic of China for financially supporting this research under Contract No. NSC 94-2524-S-006-001.

References

Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining Association Rules between Sets of Items in Large Database. *Paper presented at the 1993 ACM SIGMOD conference on management of data*, May 26-28, 1993, Washington, D.C.

AICC (2003). Aviation Industry CBT Committee, Retrieved June 7, 2007, from http://www.aicc.org/.

Berghel, H. (1997). Cyberspace 2000: Dealing with information overload. *Communications of the ACM, 40*(2), 19–24.

Bonchi, F., Giannotti, F., Mazzanti, A., & Pedreschi, D. (2003). ExAMiner: Optimized Level-wise Frequent Pattern Mining with Monotone Constraints. *Paper presented at the Third IEEE International Conference on Data Mining*, December 19-22, 2003, Melbourne, Florida, USA.

Borchers, A., Herlocker, J., Konstanand, J., & Riedl, J. (1998). Ganging up on information overload. *Computer, 31*(4), 106–108.

Chen, C. M., Lee, H. M., & Chen, Y. H. (2005a). Personalized e-learning system using Item Response Theory. *Computers & Education, 44*(3), 237–255.

Chen, J. N., Huang, Y. M., & Chu, W. C. (2005b). Applying Dynamic Fuzzy Petri Net to Web Learning System. *Interactive Learning Environments, 13*(3), 159–178.

Cheng, S. C., Huang, Y. M., Chen, J. N., & Lin, Y. T. (2005). Automatic Leveling System for E-Learning Examination Pool Using Entropy-based Decision Tree. *Lecture Notes in Computer Science, 3583*, 273–278.

Choi, J. (1999). A study on instruction strategy to improve interactivity in web-based instruction. *Journal of Educational Technology, 15*(3), 129–154.

Han, J., & Fu, Y. (1999). Mining Multiple-Level Association Rules in Large Databases. *IEEE Transactions on Knowledge and Data Engineering, 11*(5), 798–805.

Han, J., Pei, J., & Yin, Y. (2000). Mining Frequent Patterns without Candidate Generation. *Paper presented at the 2000 ACM SIGMOD International Conference on Management of Data*, May 14-19, 2000, Dallas, TX, USA.

Huang, Y. M., Chen, J. N., Cheng, S. C., & Chu, W. C. (2004). Agent-Based Web Learning System Applying Dynamic Fuzzy Petri Net. *Lecture Notes in Computer Science, 3143*, 338–345.

Huang, Y. M., Kuo, Y. H., Chen, J. N., & Jeng, Y. L. (2006). NP-miner: A Real-time Recommendation Algorithm by Using Web Usage Mining. *Knowledge-Based Systems, 19*(4), 272–286.

IEEE LTSC (2002). *IEEE Learning Technology Standards Committee*, Retrieved June 7, 2007, from http://ltsc.ieee.org/.

IMS (2003). *Content Packaging Best Practice Guide, Version 1.1.3 Final Specification*, Retrieved June 7, 2007, from http://www.imsglobal.org/content/packaging/.

Jeng, Y. L., Huang, Y. M., Kuo, Y. H., Chen, J. N., & Chu, W. C. (2005). ANTS: Agent-based Navigational Training System. *Lecture Notes in Computer Science, 3583*, 320–325.

Karampiperis, P., & Sampson, D. (2005). Adaptive Learning Resources Sequencing in Educational Hypermedia Systems. *Educational Technology & Society, 8*(4), 128–147.

Kaya, M., & Alhajj, R. (2004). Mining Multi-Cross-Level Fuzzy Weighted Association Rules. *Paper presented at the Second IEEE International Conference on Intelligent Systems*, June 22-24, 2004, Varna, Bulgaria.

Khalifa, M., & Lam, R. (2002). Web-based learning: effects on learning process and outcome. *IEEE Transactions on Education, 45*(4), 350–356.

Kim, C., & Kim, J. (2003). A Recommendation Algorithm Using Multi-Level Association Rules. *Paper presented at the IEEE/WIC International Conference on Web Intelligence*, October 13-16, 2003, Halifax, Canada.

NCKU (2005). *Web Learning System*, Retrieved June 7, 2007, from http://iteach.ncku.edu.tw/.

Pahl, C. (2003). Managing evolution and change in web-based teaching and learning environments. *Computers and Education, 40*(2), 99–114.

Papanikolaou, K. A., Grigoriadou, M., Magoulas, G. D., & Kornilakis, H. (2002). Towards new forms of knowledge communication: the adaptive dimension of a web-based learning environment. *Computers & Education, 39*, 333–360.

SCORM (2004). *Shareable Content Object Reference Model, Version 1.3*, Retrieved June 7, 2007, from http://www.adlnet.gov/scorm/index.aspx.

Song, K. S., Hu, X., Olney, A., & Graesser, A. C. (2004). A framework of synthesizing tutoring conversation capability with web-based distance education courseware. *Computers and Education, 42*(4), 375–388.

Tang, T. Y., & Mccalla, G. (2003). Smart recommendation for evolving E-learning system. *Paper presented at the 11th international conference on artificial intelligence in education, workshop on technologies for electronic documents for supporting learning*, July 20-24, Sydney, Australia.