IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITJ2438

Automatic Re-Organization of Group-Wised Web Courseware

Changjie Tang, Sichuan University, China
Rynson W. H. Lau, City University of Hong Kong, China
Qing Li, City University of Hong Kong, China
Tianqing Zhang, Sichuan University, China
Danny Kilis, Pacific Century CyberWorks (PCCW), Hong Kong

ABSTRACT

With the increasing popularity of the Internet, there is a growing demand for web-based education, which allows students to study and learn at their own pace over the Internet. However in order to improve the teaching quality, such systems should be able to adapt the teaching in accordance with individual students' ability and progress. Focusing on this objective, this paper proposes a new method to construct group-wised courseware by mining both context and structure of the courseware to build personalized Web tutor trees. To this end, the concept of Web tutor units and the notion of similarity are presented. Five algoriths, including the Naive Algorithm for tutor concept tree and the Level-generate Algorithm to generate Web tutor units of K+1 levels, are proposed. Experimental results are presented to demonstrate the effectiveness of the new method.

Keywords: distance learning, student profiling, web tutor unit, group-wised tutor tree.

INTRODUCTION

With the rapid advancement in multimedia technologies and the availability of Web infrastructure, distance learning is now widely adopted in the higher education in China. The *e-Teacher system*, an experimental software for distance learning, has been jointly developed by the City University of Hong Kong and Sichuan University. It runs on the software platform with Windows NT plus IIS (Internet Information Server) and ASP (Active Server Page). The main idea of the e-Teacher is its capability in adapting the teaching in accordance with the progress of individual students. Its key mechanisms are as follows:

1) *Clustering students into different groups according to their abilities.* For example, group11 = (Theory, Excellent),

group12 = (Theory, Medium), group13 = (Theory, Not good), group21 = (Practice, Excellent), group22 = (Practice, Medium), group23 = (Practice, Not good), etc. Thus, the teaching style is called "group-wised teaching." In the extreme situation when each group has only one student, it becomes a "personalized teaching." In the e-Teacher system, this function is implemented in a data warehouse called ETDW.

2) *Constructing a group-wised courseware that can be accessed through common web browsers, such as Microsoft IE and Netscape Navigator.* It is implemented based on our experience in the course "Reading Selected Articles on Web" (RSAW). RSAW is one of the core courses of the distant learning M.Sc. and Ph.D. degree programs. One of the authors is currently teaching this course to students across several provinces in China. To organize the course RSAW in group-wised style, the distance-teacher needs to have the following:

- **A set of profiles:** To store the profile information of each student, such as name, age, class, interests, background, academic records, etc.
- **A tutor tree:** This is a learning schema designed for each cluster of students in accordance with their abilities. Each tree node is a 2-tuple <WTUnit, Weight>, where WTUnit is a Web tutor unit (an article or a sub tutor tree) organized in a multi-resolution form, and Weight is an array of integers (containing the cluster number, course importance, teaching hours).
- **A set of evaluation and upgrading facilities:** To automatically evaluate the answer sheets and exercise forms for each student, dynamically upgrade the student profiles (as a feedback of evaluation), and reorganize student grouping based on the evaluation results.

In this paper we focus on the design of a good tutor tree. A group-wised tutor tree allows students to find the articles satisfying personal demands in a short time. The basic idea and main steps to construct the tutor tree are as follows:

1) Use an existing (usually naive) URL tree as the initial tutor tree.
2) Configure the model to calculate the similarity by adjusting the weights of the Web tutor units, and to evaluate the similarity of Web tutor units.
3) Reorganize the tutor tree by similarity and group-wised keywords.
4) Establish the new tutor tree.

In current practice, distance-teachers use an existing collection of URLs in a way that the collection may be considered as a naive form of the tutor tree. As shown in Table 1, it works in an "eagerly collecting style" by collecting everything available with low efficiency. Our interest in this study is on how to build efficient and group-wised tutor trees for effective distance learning.

Table 1: Sample of naive tutor tree of RSAW

Topic of selected articles	The root of tutor tree
Knowledge Discovery	http://www.kdnuggets.com/
Machine Learning Database Repositories	http://www.ics.uci.edu/~mlrepository.html

Related Work

There have been some research efforts conducted that are related to our work. Jiang et al. (1999) attempted to discover structures from documents. They expounded the concept of *Structural Document* and developed a formula to calculate the similarity of two structural documents. The authors made a similarity matrix that can be updated by different clustering algorithms. Mannila and Toivonen (1999) proposed a method to discover generalized episodes using minimal occurrence. Agrawal et al. (1995) proposed a fast similarity search in the presence of noise in a time-series database. Tang et al. (1999, 2000) investigated methods to extract knowledge from semi-structural Web data and to discover the quasi-periodicity from Web data. Spertus (1997) considered information clustering (grouping Web documents) according to some predefined profiles. Song et al. (2000) proposed a model to analyze the semantic similarity between Web documents, and their system supports manipulation of Web documents such as exchange, search and evolution. Unfortunately, most of these existing systems are developed for specific purposes, and there is no satisfactory way for constructing efficient group-wised (or personalized) tutor trees for distance learning.

Paper Organization

The rest of the paper is organized as follows. The next section introduces the concepts of group-wised tutor tree and Web tutor unit. Then we present five algorithms for constructing the group-wised tutor tree and show some experimental results of the new method. Finally the last section draws a brief conclusion of the paper.

WEB-BASED TUTORING FACILITIES

Figure 1 gives a sample courseware of course RSAW for the distance-learning students. There are eight articles, some of these appeared in the *Proceedings of the 16th National Database Conference* in China. It includes URLs, names of the proceedings, pages, titles, keywords, author names, first authors' sexes, names of supervisors (for student authors), fields, and special topics.

Group-Wised Tutor Tree

Two group-wised tutor trees for the course, organized in different ways, are shown in Figures 2(a) and 2(b), where symbol '^' indicates "unknown." However, there may be professors and students who may prefer to access the Web tutor tree in a way similar to the one shown in Figure 2(c), which organizes selected articles according to the subject. The objective of group-wised courseware is to meet such *individual* needs. As mentioned before, we cluster students into different groups according to their ability. When there is only one member in each group, "group-wised" becomes "personalized." In this paper, "personalized" will be considered as a special case of "group-wised" without further explanations.

Basic Concepts and Definitions

In general, the selected articles for distance leaning students include HTML files, bookmarks, personal home pages, images, and voice files. To formalize the observations, we now define *Web Tutor Unit*.

Figure 1: RSAW courseware for distance learning.

1.www.pru.edu.cn, 16DB, P1, "Implementation of Storage of Large Object Data", {large object, spatial multi-pointer, bitmap page}, Zhang Xiao, Male, Prof. Wang, DB, Database theory.
2.www.fudan.edu.cn, 16DB, P6, "The Query Language and Data Model of Constraint Based Spatial-Temporal Data in Digital Library", {constraint database, query language, data model}, Wang Wei, Male, Prof. Shi, DB, Database theory.
3.www.fudan.edu.cn, 16DB, P77, "Non-Monotonic Inheritance of Objects", {Deductive OO database, non-monotonic, inheritance canonical model, the inheritance diagram}, Liu Hong Liang, Male, Prof. Shi, DB, Advanced Database.
4.www.nju.edu.cn, 16DB, P93, "The Implementation of EXPRESS Object-Oriented Data Model with Relational Database Systems", {OO data model, relational data model, EXPRESS modeling language, RDBS}, Yu Yong Hong, Male, Prof. Xu, DB, Advanced Database.
5.www.scu.edu.cn, 16DB, P215, "Aggregation on Data Cube", {KDD, cube OLAP, B-Tree, dependency tree}, Liu Xin, Male, Prof. Tang, DB, Data Warehouse.
6.www.pku.edu.cn, 16DB, P308, "A Client Analysis System Prototype Based on Spatial Data Mining", {attribute-oriented induction, spatial data mining, spatial attribute-oriented induction}, Xu Qi Chang, Male, Prof. Yang, DB, Data mining.
7.www.fudan.edu.cn, 16DB, P319, "Scaling DBSCAN Algorithm to Large-scale Database by Data Sampling", {spatial database, data clustering, sampling, DBSCAN}, Fan Ye, Male, Prof. Zhou, DB, Data mining.
8.www.scu.edu.cn, 16DB, P250, "Mining Associations of Objects with Relaxed Periodicity and its Applications in Seismic Research", {KDD, relaxed periodicity, Seism}, Yang Lu, Female, Prof. Tang, DB, Data mining.

Definition 1 (Web Tutor Unit).

1. A *Web Tutor Unit* (abbreviated as WTUnit) is a recursive structure:

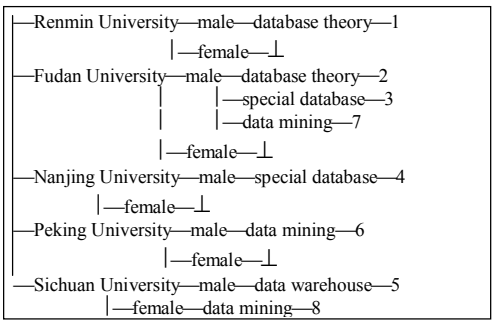
```
struct WTUnit {
    CString ObjTitle;          // title of web page or bookmark
    SetOfCString Keywords;     // std keywords given by author
    Cstring URL_Dir;           // such as www.scu.edu.cn\CS\DB\
    WTUnit *pChildrenUnit[]; // array of children unit
}
```

2. Let $N_k\backslash N_{k-1}\backslash ...\backslash N_2\backslash N_1\backslash$ be the URL_Dir of the Web tutor unit **w**. The ordered set $\{N_1, N_2, ..., N_k\}$, arranged from the last to the first, is called an *Ordered Ancestor Set* of a Web tutor unit, and is abbreviated as *Ancestor Set*. •

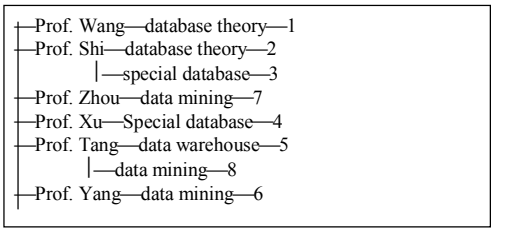
It is clear that there is a 1-1 corre-

Figure 2. Different organizations of web tutor tree. (a,b, and c)

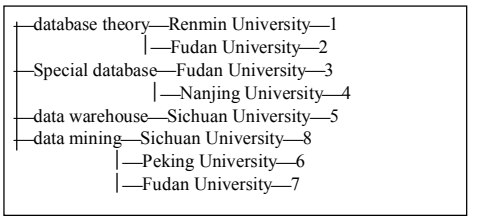
(a) By "University→Sex-Topic→PaperID"



(b) By "Director→Topic→PaperID"



(c) By "Field→University→PaperID"



spondence between a Web tutor unit and its URL. Thus, we can refer to a Web tutor unit by its URL when needed.

Example 1. Let **w** be the web tutor unit with URL = www.scu.edu.cn\CS\DB\KDD.html#papers99. ObjTitle is bookmarked as "papers99" in file "KDD.html". The string "papers99" is shown as the title of the current page. URL_Dir is www.scu.edu.cn\CS\DB\. Assuming that there are three hyperlinks named as "Relations," "Time Series," and "Classify" in **w**, pChildrenUnit is then the set of hyperlinks. The author of the article gives the set of keywords. Finally {"CS", "DB"} is the ordered ancestor set. •

We make the following observations:

1. Let **w** and **w'** be two web tutor units. If the first two parameters (i.e., ObjTitle and Keywords) are similar, then from the student's point of view, **w** and **w'** are similar in contents.
2. Let URL_Dir of **w** be $N_m\backslash N_{m-1}\backslash...N_k...\backslash N_2\backslash N_1\backslash$, URL_Dir of **w'** be $N'_m\backslash N'_{m-1}\backslash...N'_k...\backslash N'_2\backslash N'_1\backslash$, and $N_i = N'_i$, where 1 £ i £ k. This indicates that the positions of **w** and **w'** in the Web organization are similar, and the larger the k is, the more similar they are. Let $b(N_i)$ be the degree of contribution of N_i to this similarity. Obviously, $\beta(N_k) \le ... \le \beta(N_2) \le \beta(N_1)$.

Example 2. Suppose that **w** and **w'** correspond to URLs "www.scu.edu.cn\CS\KDD\A.html" and "www.pku.edu.cn\Math\KDD\B.html", respectively. From observation (2), we have $N_1=N'_1$=KDD, N_2 = CS (Department of Computer Science), N'_2 = Math (Department. of Mathematics), N_3 is Sichuan University, and N'_3 is Peking University. Files A.html and B.html are in different pages of different departments in different universities, but are similar in terms of having the same immediate ancestor (i.e., 'KDD'). •

Definition 2. Let UnitSet be a set of Web unit, the function *Same* is defined as:

Same: UnitSet × UnitSet →{1, 0}

The value of Same($UnitSet_1$, $UnitSet_2$) is 1 if $UnitSet_1$ $UnitSet_2$, or 0 otherwise. •

Definition 3 (Similarity of Web tutor units). Let w_1 and w_2 be two Web tutor units.

1 The set of personalized keywords, PersKeySet = {K_1, K_2,…, K_n}, is selected from domain standard keywords by users according to the personalized guideline.
2 Let K_SET_i = w_i.Keywords ∩ PersKeySet, where i = 1 or 2. $K(w_1,w_2)$ = $|K\_SET_1 \cap K\_SET_2| / |PersKeySet|$ is called the *personalized keyword similarity* of w_1 and w_2.
3 Let w_i have n_i child web units, and the children units are C_i = {w_i.pChildrenUnit[k] | 0 ≤ k ≤ n_i} for i = 1 to 2. Then, $C(w_1,w_2)= | C_1 \cap C_2 | / | C_1 \cup C_2 |$ is called the *children similarity* of w_1 and w_2.
4 Let ancestor sets of w_1 and w_2 be {N_{11}, N_{12}, ..., N_{1p}} and {N_{21}, N_{22}, ..., N_{2q}}, respectively. If there exists a number k, 0 ≤ k ≤ min (p, q) such that $N_{1i} = N_{2i}$ for 0 ≤ i ≤ k, and $N_{1(k+1)}$ ¹ $N_{2(k+1)}$, then $r = 1/2 + 1/4 + ... + 1/2^k$ is called the *inheritance similarity* of w_1 and w_2, denoted as $A(w_1, w_2)$.
5 Let k be the number described above and a, c, be non-negative numbers, and k+c+a=1. Then Group_Similarity (w_1,w_2) = k x$K(w_1,w_2)$ + c x $C(w_1,w_2)$ + a x (w_1,w_2) is called the *group-wised (or personalized when the group size is one) similarity* of w_1 and w_2.

Note that:

1) Inheritance similarity is a binary num-

ber $0.11\ldots1=2^{-1}+2^{-2}+2^{-3}+\ldots+2^{-k}$. Its length of fractional part is k. Contributions of ancestors to similarity are decreasing as series of 2^{-n}.

2) Inheritance similarity and children unit similarity reflect the resemblance of units in a web organization. The personalized keyword similarity describes resemblance of units under users' guideline.
3) Parameters a, c and k are given by the distance-teacher. (The default values used in our experiment are a=0.2, c=0.1, and k=0.7.)

ALGORITHMS FOR GROUP-WISED TUTOR TREE

We now proceed to describe how to construct the group-wised tutor tree. We do this by introducing five algorithms that we have developed for this purpose.

A Naive Way to Construct Web Tutor Tree

Parameters

In the Naive Algorithm for group-wised tutor tree, the parameters are set as follows: a=c=0 and k=1 (see Definition 3). That is, the role of the old organization of web courseware is ignored; only the keywords of the Web tutor unit are used as clustering criteria. Thus, the personalized similarity of w_1 and w_2 is: $P(w_1, w_2) = K(w_1, w_2) = |K\_SET_1 \cap K\_SET_2| / |PersKeySet|$.

Table 2: Tutor Unit-Key Matrix

Unit-Key	$w_1, w_2, ..., w_j, .., w_m$	Sum
k_1	.	.
...		..
k_i	... ... a_{ij} ..	s_i
...	.	
k_n	.	
Total keys	t_j	

Training Set and Personalized Order of Keywords

The primary training set, denoted as *PrimaryTutorSet*, is selected by the distance-teacher from the Web courseware under the following criteria:

The size of PrimaryTutorSet is big enough, say more than 100 pages.

The PrimaryTutorSet must be typical enough. It involves typical tutor contents, with typical keywords, ancestors and children units.

Let PrimaryTutorSet = $\{w_1, w_2, ...,w_n\}$ and the personalized keyword set PersKeySet=$\{K_1, K_2, ..., K_m\}$. We construct the *Tutor Unit-Key Matrix* as shown in Table 2. If there exists a keyword k_i contained in w_j, then $a_{ij} = 1$, otherwise, $a_{ij} = 0$. The number $s_i = a_{i1} + a_{i2} +...+ a_{in}$ is the total number of w_j containing k_i, called key activity of k_i. The number $t_j = a_{1j} + a_{2j} +...+ a_{nj}$ indicates the number of keywords contained in w_j. Formally, we have:

Definition 4. Let the context be as that of Table 2.

1) The set of personalized keywords with the descending order of s_i is called *Descending Key set*. t_j in Table 2 is called group-wised intensity of Web tutor unit w_j.
2) The function to arrange the web tutor unit set WTUnitSet with descending order of group-wised intensity is denoted as Ordered_WTUnitSet =PD_Sort (WTUnitSet).

3) Let Delta be the threshold of group-wised intensity. The set TrainWTUnitSet = GD_Sort($\{w_i \mid t_i(w_i) > Delta\}$) is called personalized training set with respect to Delta. •

It is straightforward to build the Tutor Unit-Key Matrix, to sort keywords, and to generate personalized training set. We thus assume that these processes are done in a preprocessing phase.

Candidate Tutor Concept Set

During the procedure to reorganize web tutor tree, a set of similar web tutor units can be viewed as a (new) tutor concept. To prepare and simplify the concept-generating procedure, we define the following:

Definition 5. Let TrainWTUnitSet = $\{w_1, w_2, ..., w_m\}$ and ancestor set of w_i be $A_i=\{S_{i1}, S_{i2}, ..., S_{ik}\}$. Then the Candidate Concept Set is defined as:

$$CandidateConceptSet=A_1 \cup A_2 \cup \ldots \cup A_m \text{ •}$$

Note that, S_{ik} is a candidate concept if and only if S_{ik} is an ancestor of some W_i. By "candidate" we mean that it can be selected as the basic material to compose a new concept. It is illustrated by the following example.

Example 3. (1) Consider Figure 2(a). CandidateConceptSet = {Renmin University, Fudan University, Nanjing University, Peking University, Sichuan University, male, female, Database theory, Special database, Data mining, Data warehouse}. (2) Consider Figure 2(b). CandidateConceptSet = {Prof. Wang, Prof. Shi, Prof. Xu, Prof. Tang, Prof. Yang, Prof. Zhou, database theory, special database, data mining, data warehouse}.•

Definition 6. Let S_1, S_2, and S_3 be three strings of characters, S_3 be the longest common sub-string of S_1 and S_2. If $|S_3| \geq 0$, then S_1 and S_2 are said to be partially similar with similarity $Sim(S_1, S_2) = |S_3| / \max(|S_1|, |S_2|)$. •

Based on a set of similar Web tutor units, new tutor concepts (or topics) can be generalized. The function *GenerateConcept_Similarity(S_1, S_2, NewConcept, Sim)* is defined as follows and explained in Example 4.

Function GenerateConcept _ Similarity
Input: Web tutor concept name (or string) S_1, S_2.
Output: NewConcept and Similarity Sim of S_1 and S_2 as return value.
Steps:
1. L = 0; // Initialization of the common feature
2. for (i = 1; $i \leq |S_1|$; i++)
3. for (j = 1; $j \leq |S_1|$-i+1; j++) {
4. Extract sub-string S_3 with length of j and start from S_1[i];
5. if (S_2 including S_3 and j >L) then L = j;
6. }
7. Sim = L / max($|S_1|$, $|S_2|$);
8. NewConcept = S_3 +"_Set";
9. Output NewConcept and Sim; □•

Example 4. Based on Example 3(1) and function GenerateConcept_Similarity, we can generate a new concept from similar concepts with similarity Sim ≥ 0.4, as shown below:

1)University_Set = {Renmin University, Fudan University, Nanjing University, Peking University, Sichuan University}, Sim = 0.5.
2)database_Set = {database theory, special database}, Sim = 0.5.
3)data_Set = {database theory, special database, data mining, data warehouse}, Sim = 0.5.

The Meta Concept Base

Some concept names are derived

from their elements semantically but not lexically, such as "Sex" ={"male," "female"}. To generate such tutor concepts (or topic) automatically, we need a *Meta concept base* as shown in Table 3.

Algorithm 1 (Generate new concept)

Input: CandidateConceptSet, Meta concept base, threshold of similarity $\delta>0$.

Output: The set of new web tutor concepts NewConcepts within which Similarity $\geq \delta$, and all the elements are sorted by descending group-wised intensity.

Steps:

1. NewConcept=NULL;
   // initiate it as NULL
2. for each C_i and C_j (i<j) in CandidateConceptSet {
3. GenerateConcept_Similarity(C_i, C_j, $NewConcept_{ij}$, Sim_{ij});
   // pair-wised check similarity
4. if ($Sim_{ij} \geq \delta$) then
5. $NewConcept=NewConcept \cup NewConcept_{ij}$;
   // NewConcept increases by threshold
6. }
7. Sort NewConcepts by name;
8. Merging courseware in NewConcepts with same name;
9. Output NewConcepts in the format of "NewConceptName_SET = {a, b, c, d, ...}";

In the above algorithm, the main computation cost is in the for-loop. Clearly the complexity is $O(n^2)$, where n is the size of CandidateConceptSet.

In Example 4, the Web tutor Concept database Set = {"database theory," "special database"}. Suppose that "database theory" and "special database" consist of papers $\{p_1, p_2, ..., p_n\}$ and $\{q_1, q_2, ..., q_n\}$, respectively. To evaluate the group-wised intensity of database_Set, we sum all intensity of all p_i's and q_j's. Formally, we have:

Definition 7. Let the context be the same as in Algorithm 1 and the NewConcept be $C_0 = \{t_1, t_2, ..., t_n\}$, where t_i is a concept or a title of a Web tutor unit. The group-wised intensity function, denoted as GI, is recursively defined as follows:

1 If t_i is the title of a Web tutor unit u_i, $GI(t_i)$ is the group-wised intensity as defined in Definition 4(2), i.e., the number of keywords contained in C_i.
2 $GI(C_0) = \Sigma GI(t_i)$.
3 SortedNewTopicSet is the set of NewConcept generated by Algorithm 1 and sorted according to the descending order of group-wised intensity. •

Intuitively, GI(NewConcept) is the total number of personalized keys appearing in the Web tutor tree with NewConcept as the root of tutor tree. To prepare Algorithm 3 (Naive Algorithm), we need another observation. Let w be the Web tutor unit with URL Renmin_University \ CS \ database \ Prof.Wang \ database theory \ male \ 1.htm." Its concept hierarchy model is "University→Deparment→ Prof→

Table 3: A part of meta topic base

Complex Tutor Concept Name	Contents of Concept	Inside Similarity
Sex	Male, Female	0.8
PaperID	1, 2, 3, 4, 5, ...	1
Fruit	Apple, Banana, Pear, Grape, Orange	0.8

Field→Sex→PaperID." Then, "\ database_ theory \ Renmin_University \ 1.htm" is a valid URL to locate 1.htm. Hence it is a valid reorganization under the concept hierarchy model "Field→University→ PaperID." Thus we have:

Algorithm 2 (Insert training units)
Input:

1) Concept hierarchy model $C_1 \rightarrow C_2 \rightarrow ... \rightarrow C_m$, (such as "Field →University→ PaperID").
2) A Web tutor unit w with URL "N_1\ N_2\ ...\ N_p\ leaf_file" in TrainWTUnitSet.

Output: Insert w into Web tutor tree under given concept hierarchy model, return true if successful or false otherwise.

Steps:

1. if not (each C_i is in $\{N_1, N_2, ..., N_p\}$) then return false; // C_i is invalid
2. if not exist group-wised tutor unit then GroupWisedTutorUnit = new WTUnit;
3. w_0 = GroupWisedTutorUnit;
4. w_0.ObjTitle = "Personalized web tutor page"; // default title
5. for (i = 1; i < p; i ++)
6. if not exist w_i {
7. w_i = new WTUnit;
8. w_i.ObjTitle = Name of the C_i;
9. w_{i-1}.pChildrenUnit = w_{i-1}.pChildrenUnit∪{&w_i}; // Add address of w_i as a new hyperlink
10. }
11. Insert leaf_file as the last child unit of w_p.

In Algorithm 2, the complexity of step 1 is m ´ p and the complexity of step 2 is O(p), where m ≤ p. Thus, the total complexity is $O(p^2)$, where p is the number of ancestors of w.

The Naive Algorithm

This algorithm is to first construct a Web tutor concept hierarchy model and concept tree based on TrainWTUnitSet, and then insert the remainder w_i of WTUnitSet to the concept tree (i.e., the Web tutor tree).

Algorithm 3 (Naive Algorithm)

Input: WTUnitSet, TrainWTU intSet, threshold δ >0, Max_L (the maximum level of Web tutor tree), PersKeySet ordered by key activity

Output: Concept hierarchy model (such as "Topic→University →PaperID" and Tutor tree (such as Figure 2(c)).

Procedure:

1. Build CandidateConceptSet from TrainWTUintSet; // See Example 3
2. Invoke Algorithm 1, sort its result, and generate SortedNewConceptSet with similarity ≥ δ;
3. Assume SortedNewConceptSet = $\{C_1, C_2, ..., C_n\}$, and m = Min(n, Max_L). The concept hierarchy model is then $C_1 \rightarrow C_2 \rightarrow ... \rightarrow C_m$.
4. For each Web tutor unit w in TrainWTUintSet, invoke Algorithm 2 and insert tutor unit according to the concept hierarchy model $C_1 \rightarrow C_2 \rightarrow ... \rightarrow C_m$. Denote the resulting tree as ConceptTree.
5. // Insert all remaining Web tutor units of WTUnitSet into ConceptTree
   TempUnit=NULL; TempSim=0;
   for (each $w_i \in$ WTUnitSet - TrainWTUintSet and each Unit w_j in ConceptTree) {Sim_{ij} = Group_ Similarity(w_i, w_j); // personalized similarity, see Definition 3 (5)
   if (Sim_{ij}>δ and Sim_{ij}>TempSim) {
   TempUnit=w_j; TempSim= Sim_{ij}; // Now TempUnit is the web tutor Unit with // Maxmum similarity to w_i.
   }
   }
6. Output ConceptTree as tutor tree. •

Proposition 1. Let *n* be the number of Web tutor units. Assume *t* is the max number of ancestors of tutor unit, *r* is the max length of keywords and concept names, and *m* is the levels of concept tree. Then the complexity of the Naive Algorithm is $O(p^4+n\times2^m)$, where $p = \max(n, r, t)$.

Proof. The complexity of step (1) is $n \times t$. Consider step (2) of the Naive Algorithm. The size of CandidateConceptSet is not greater than $n \times t$. As the complexity of function GenerateConcept_Similarity is $O(r^2)$, the complexity of Step (2) is $O(r^2\times(n\times t)^2)$. In Step (3), the concept tree has m levels, thus, the complexity of Step (4) and that of Step (5) are both $O(n\times2^m)$. Let $p = \max(n, r, t)$ in the worst case. The complexity of the Naive Algorithm is then $O(p^4+n\times2^m)$. •

In practice, usually $t \le 10$, $r \le 10$, $l \le 4$, and $m < n$. The complexity can be simply evaluated as $O(n^4+ n\times2^m)$ for the worst case.

Example 5. Consider the Web tutor units (i.e., the selected articles) in Figure 1. The WTUnitSet = $\{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8\}$, where

w_1= Renmin University \ CS \ database \ Prof. Wang \ database theory \ male \ 1.htm,
w_2= Fudan university \ CS \ database \ Prof. Shi \ database theory \ male \ 2.htm,
w_3= Fudan University \ CS \ database \ Prof. Shi \ special database \ male \ 3.htm,
w_4= Nanjing University \ CS \ database \ Prof. Xu \ special database \ male \ 4.htm,
w_5= Sichuan University \ CS \ database \ Prof. Tang \ data warehouse \ male \ 5.htm,
w_6= Peking University \ CS \ database \ Prof. Yang \ data mining \ male \ 6.htm,
w_7= Fudan University \ CS \ database \ Prof. Zhou \ data mining \ male \ 7.htm,
w_8= Sichuan University \ CS \ database \ Prof. Tang \ data mining \ female \ 8.htm.

The inputs of Algorithm 3 are given below:

- WTUnitSet=$\{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8\}$.
- TrainWTUintSet=$\{w_1, w_2, w_4, w_5, w_6, w_8\}$, selected by the teacher on the demands of the students and the personalized rules.
- PersKeySet={data mining, database theory, special database, data warehouse, Sichuan University, 4}.

The stepwise outputs of Algorithm 3 are as follows:

1. TrainWTUintSet = $\{w_1, w_2, w_4, w_5, w_6, w_8\}$.
2. CandidateConceptSet = {Renmin University, Fudan University, Nanjing University, Peking University, Sichuan University, CS, database, database theory, special database, data mining, data warehouse, male, female, Prof. Wang, Prof. Shi, Prof. Xu, Prof. Tang, Prof. Yang}.
3. Similar concept sets are:
   - University_Set = {Renmin University, Fudan University, Nanjing Uni versity, Peking University, Sichuan University}.
   - database_Set = {database theory, special database}.
   - data_Set = {database theory, special database, data mining, data warehouse}.
   - Prof._Set = {Prof. Wang, Prof. Shi, Prof. Xu, Prof. Tang, Prof. Yang}.

- Sex_Set = {male, female}.

4. Insert w_3 and w_7 after obtaining the concept tree. The final concept tree is as shown in Figure 2(c).•

Group-Wised Algorithm

Level Generating Algorithm

The Naive Algorithm (Algorithm 3) is simple and with acceptable speed, but the concept hierarchy cannot be modified once it is constructed. In particular, step (5) of the Naive Algorithm only inserts Web tutor units according to similarity; thus it cannot change the concept hierarchy. To develop a more flexible algorithm, we define the following:

Definition 8 (multi-level tutor units).

1. The tutor unit at *0-th level*, v^0, is a usual Web tutor unit v, and v^0. Keywords = v.Keywords.
2. Let $v_1, v_2, ..., v_n$ be the tutor unit at *k-th level*. Then *(k+1)-th level* tutor unit $v^{(k+1)}$ composed from $v_1, v_2, ..., v_n$ is a Web unit satisfying following conditions:
   a. $v^{(k+1)}$ has exactly n children units $v_1, v_2, ..., v_n$, i.e., *pChildrenUnit[k]=v_k, for k=1,2,…n.
   b. $v^{(k+1)}$.Keywords =)"v_i.*Keywords,* ( )" for all i, 0<i<n+1) $\bigcap_{i=1}^{n} v_i.Keywords$.
   c. $v^{(k+1)}$.ObjTitle is the concept with maximum frequency in NewConcepts, where NewConcepts is the set of concepts generated by applying GenerateConcept_Similarity on each pair in set {v_i.ObjTitle| 0<i<n+1}. •

Algorithm 4 generates *(k+1)-th level* Web tutor unit from *k-th level* of web tutor units. The idea is: (1) to evaluate the similarity of any two tutor units of *k-th level* in CurrWTUnitSet; (2) if similarity is higher than the threshold, then combine them as a *(k+1)-th level*, denoted as $v_i^{(k+1)}$; (3) prune the combined units from CurrWTUnitSet. The initial CurrWTUnitSet is the set of *k-th level* Web tutor units ordered by the descending personalized intensity. In the algorithm, the brackets "*{*" and "*}*" in italic font indicate that the contents inside them form a set.

Algorithm 4 (Level_Generate Algorithm for Web tutor unit of (k+1)-th Level)

Input: Web tutor units at *k-th level*: $v_1^k, v_2^k, ..., v_n^k$, Similarity threshold $\delta > 0$, and the personalized keyword set PersKeySet.

Output: Tutor unit at *(k+1)-th level*: $v_1^{(k+1)}, v_2^{(k+1)}, ..., v_n^{(k+1)}$, such that the similarity within each $v_i^{(k+1)}$ is not less than δ.

Procedure: Level_Generate(k+1)

1. CurrWTUnitSet = GD_Sort (*{*$v_1^k, v_2^k, ..., v_n^k$*}*, t_i); // t_i and GD_Sort same as in Definition 4
   // Initialize CurrWTUnitSet as the input and ordered by descending personalized intensity
2. OutputWTUnitSet = NULL;
3. for (each v_i^k in CurrWTUnitSet) {
4. CurrWTUnitSet = CurrWTUnitSet – {v_i^k}; // to avoid dead loop
5. v = new WTUnit; // generate a new Web tutor unit v as buffer
6. v.Keywords= PersKeySet∩v_i. Keywords; // Initialize the keywords of new WTUnit.
7. v.pChildrenUnit = *{*v_i^k*}*; // insert v_i^k as the first Child-tutor unit of v.
8. v.ObjTitle = "Title_I"; // set default title; it is modifiable
9. for (each v_j^k in CurrWTUnitSet)

10. if (GroupSimilarity (v_i^k, v_j^k) > δ) {
/ personalized similarity, see Def. 3 (5)
11. CurrWTUnitSet = CurrWTUnitSet – {v_j^k};
12. v.pChildrenUnit = v.pChildrenUnit ∪ {v_j^k}; // insert v_j^k as the child tutor Unit of v.
13. v.Keywords=v.Keywords Çv_j^k.Keywords;
14. } // end of if
15. } // end of for
16. For each pair (v_j^p ,v_j^q) in the set { v_j^k<n+1} {
17. GenerateConcept_Similarity (v_j^p.ObjTitle, v_j^q.ObjTitle, $TempNewTitle_{pq}$, $TempSimilarity_{pq}$);
18. Find the concept in { $TempNewTitle_{pq}$ } with maximum frequency and denote it as NewTitle;
19. v.ObjTitle = NewTitle; // it is similar to most of { v_j^k | 0<j<n+1}
20. if (2 > the number of Children Units in v) then v = v ∪ {ε}; // ε is a zero unit
21. $v^{(k+1)}$ = v; // it is web tutor unit of level (k+1)
22. OutputWTUnitSet = OutputWTUnitSet ∪{$v^{(k+1)}$};
23. }
24. Output OutputWTUnitSet; •

Proposition 2. Let n be the number of the k-th level Web units inputted to Algorithm 4. The complexity of Algorithm 4 is $O(n^2)$.
Proof. In Algorithm 4, the complexity of line 1 is O(n×log(n)). The complexity of lines 3 to 15 is O(n). Line 17 (function GenerateConcept_ Similarity) will be called for at most n(n-1)/2 times and its cost is $O(n^2)$. The cost comparisons in line 18 are no greater than $O(n^2)$. Thus the total complexity can be evaluated as $O(n^2)$. •

In order to simplify the algorithm, a zero tutor unit e is introduced here, which can be viewed as a bookmark pointing to an empty URL. ε is inserted to OutputWTUnitSet if it has only one child unit. In the following, Algorithm 5 will call the procedure Level_Generate (k+1) from k = 0 until its output set equals the input set. To simplify Algorithm 5, we need the concept of *embedded level tree*. Suppose that there are three sets: A = {1, 2, 3, 4}, B = {{1, 2}, {3, 4}}, and C = {1, {2, {3, 4}}}. The results of expanding them are shown in Figures 3(a), 3(b), and 3(c), which are called embedded level tree. The procedure to expand embedded level tree is denoted as Tree_Expand, the details of which are omitted due to its simplicity.

Algorithm 5 (Group-wis3ed Algorithm for generating multi-level Web tutor tree)

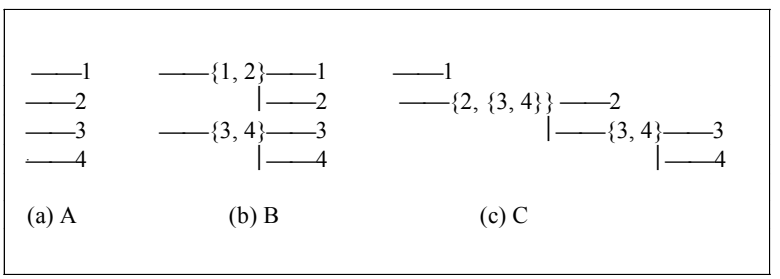
Input: 0-level Web tutor unit set {v_1^0, v_2^0, ..., v_n^0}, threshold of similarity δ > 0, and personalized keyword set PersKeySet.

Output: Web tutor tree with multi-level and its root node.

Procedure:

1. InputWTUnitSet = {v_1^0, v_2^0, ..., v_n^0};
2. OutputWTUnitSet = NULL;
3. Root = NULL;
/

Figure 3: Embedded Level Tree



```
/ Init multi-level Web tutor unit
4. k = 0;
5. while (k ≥ 0) {
6. CurrWTUnitSet = InputWTUnitSet;
   // Initialize it
7. Level_Generate(k+1);
   // Algorithm 4, generate Web tutor unit
   of (k+1)-th level
8. if (OutputWTUnitSet = =
   InputWTUnitSet) {
9. Insert each element of OutputWTUnit-
   Set to Root;
10. Exit;
11.}else InputWTUnitSet = OutputWTUnit-
   Set;
12.} // end of while
13. Tree_Expand(Root);
   //Use Level_Expand to expand Root to
   a tree; see Figure 3
14. Delete all zero units e in root and out
   put it; •
```

Proposition 3. Let n be the number of Web tutor units considered. The complexity of Algorithm 5 is $O(n^3)$.

Proof. In Algorithm 5, the complexity of lines 1-4 and lines 13-14 are O(n). In the "while" loop, line 7 incurs the maximum complexity. The complexity of other lines are O(n). In the worst case, n Web tutor units can be embedded in n levels. Thus the while statement will loop at most n times. By Proposition 2, for each call of line 7, the complexity is $O(n^2)$. Thus the total complexity can be evaluated as $O(n^3)$. •

Comparison of Algorithms 3 and 5

Tables 4-7 list the comparison of Algorithm 3 (the Naive Algorithm) and Algorithm 5 (the Group-Wised Algorithm). There is an interesting observation that Algorithm 5 looks simpler than Algorithm 3, but with higher efficiency. The reason is that Algorithm 4 (which generates the *(k+1)-th level* WTUnit) has already absorbed the difficulties.

EXPERIMENTAL RESULTS AND ANALYSES

We have done an initial experimental study based on an implementation of Algorithm 5. To avoid the side effect of network bottleneck, we first downloaded the selected Web tutor units from the Web. The format is as illustrated in Figure 1. During the testing, all the inputs are available from a local computer. The experimental result of algorithm 5 is shown in Figure 4. The descriptions of the experiment are given in Figure 4.

Input: 0-level web tutor units

Figure 4. Calculated Result

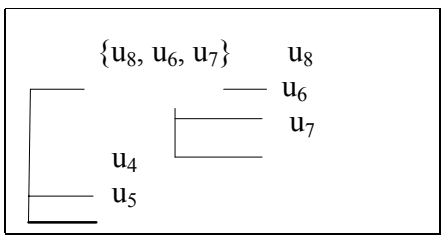


Table 4: The Input of Algorithms 3 and 5

Algorithm 3	Algorithm 5
WTUnitSet, TrainWTUintSet, threshold of similarity $\delta > 0$, PersKeySet	WTUnitSet, as 0-level web tutor unit set (no need for TrainWTUintSet), threshold of similarity $\delta > 0$, PersKeySet.

WTUnitSet = {u_4, u_5, u_6, u_7, u_8}; k = 0.7, c = 0.1, a = 0.2, δ = 0.3; PersKeySet = {Data Mining, Association Rule}. The item "keywords" and "field (of research)"- and abstracts of all articles in Figure 1 are taken as the Keywords of the tutor units.

Output: Shown in Figure 4.

Stepwise Analysis: Because there is no child tutor unit in those WTUnits, the children similarity of all WTUnits C = 0. The program based on the algorithm can be traced in a stepwise manner, which allows us to view the intermediate results as follows:

1. *k = 0, call Level_Generate (1):*
   After GD_Sort(WTUnitSet, t_i), CurrWTUnitSet = {u_8, u_6, u_7, u_4, u_5}.
   Using the functions K(x,y),C(x,y) and A(x,y) defined in the Definition 3, we have:
   $K(u_8, u_6) = 2/2$, $A(u_8, u_6) = 0$, we get Group_Similarity $(u_8, u_6) = 0.7'2/2+0.2'0 = 0.7$, Group_Similarity $(u_8, u_7) = 0.7'1/2 = 0.35$ and Group_Similarity $(u_8, u_4) = 0$.
   Since $K(u_8, u_5) = 0$ and $A(u_8, u_5)=1/2+1/4+1/8+1/16=15/16$, we get Group_Similarity$(u_8, u_5)=0.7'0+0.2'15/16=0.19$.
   Since Group_Similarity $(u_8, u_6) = 0.7>\delta$ and Group_Similarity $(u_8, u_7) = 0.35>\delta$, v_1^1={u_8, u_6, u_7} and v_1^1.Keywords = {Data Mining}, where CurrWTUnitSet = {u_4, u_5}.
   The same as above, we have v_2^1={u_4, ε}, v21.Keywords = ∅, v_3^1={u_5, ε}, and v31.Keywords = ∅. So OutputWTUnitSet = {{u_8, u_6, u_7}, {u_4, ε}, {u_5, ε}}. Since OutputWTUnitSet and InputWTUnitSet are of different values, we have the further step below:
2. *k = 1, InputWTUnitSet = OutputWTUnitSet, call Level_Generate (2).*
   After calling Level_Generate (2), the set OutputWTUnitSet is equal to the set InputWTUnitSet. Hence, Root = OutputWTUnitSet = {{u_8, u_6, u_7}, {u_4, ε}, {u_5, ε}}.
3. Deleting all zero Web tutor units, we have Root = {{u_8, u_6, u_7}, {u_4}, {u_5}}.

Table 5: The output of Algorithms 3 and 5

Algorithm 3	Algorithm 5
Topic hierarchy model, such as field→University→PaperID, Web Tutor tree (cf. Figure 1 (c))	Web tutor tree

Table 6: The calling features of Algorithms 3 and 5

Algorithm 3	Algorithm 5
Calls procedure Group_Similarity (...), and meta concept base. It works in the style of supervised classification.	Calls Algorithm 4 to generate tutor unit of (k+1)-th level from units of k-th level. It works in the style of unsupervised classification.

Table 7. The Data-flow features of Algorithms 3 and 5

Algorithm 3	Algorithm 5
Primary training set → Candidate concept Set→ tutor concept hierarchy model by Alg 1→ tutor concept Tree by Alg 2.	0-th Level obj→1 Level obj→,...→K Level obj by Alg 4.

Then, call Tree_Expand (Root), and the final result is as shown in Figure 4.

CONCLUSION

In this paper, we have presented concepts for web-based group-wised distance learning, such as *Web Tutor Unit with multi-levels*, its *Similarity*, and *Personalized Intensity*. We presented five algorithms, including the *Naive Algorithm* for simple Web tutor tree, *Level-Generate Algorithm* to generate Web tutor concept of k+1 level, and *Group-wised Algorithm* for personalized Web tutor tree. Experimental results are provided to illustrate our approach. We are currently studying several further research issues such as performance improvement, and the design and incorporation of the weighting concept into the personalized Web tutor tree.

ACKNOWLEDGMENTS

This project is in part supported by a CERG grant from the Research Grants Council of Hong Kong (RGC Reference Number: CPHK 77/95E) and the National Science Foundation of China grant #60073046.

REFERENCES

Agrawal, A., & Mannila, H. (1996). Fast Discovery of Association Rules. *Advances in Knowledge Discovery and Data Mining*, AAAI Press and The MIT Press.

Agrawal, A., Ling, K., Harpreet, S., & Shim, S. (1995). Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases. *Proc. of VLDB*.

Chim, J., Lau, R.W.H., Si, A., Leong, H.V., Green, M., & Lam, M. (1998).Multi-Resolution Model Transmission in Distributed Virtual Environments. *Proceedings of ACM VRST'99*, 25-34.

Fayyad, U. & Piatetsky-Shapiro, G. (1996). *Advances in Knowledge Discover and Data Mining (Eds)*, AAAI Press and The MIT Press, 1-5.

Guo, Y., Zhang, T., Yin, H., & Tang, C. (2000). The Implementation of Data Warehouse ETDW in Distance Education System E-Teacher.–*Proc. of National Database Conference of China (NDBC'00)*, B-Series, 249-252.

Jiang, M., Tseng, S., & Tsai, C. (1999). Discovering Structure from Document Databases. *Proc. of PAKDD'99*, 169-173.

Mannila, H. & Toivonen, H. (1999). Discovering Generalized Episodes Using Minimal Occurrence. *Proc. of Int' Conf. on Knowledge Discovery and Data Mining*.

Song, W., Cheung, D., & Tan, C. (2000). A Semantic Similarity Approach to Electronic Document Modeling and Integration. *Proceedings of International Conference on Web Information System Engineering (WISE'00)*, 108-116.

Spertus, E. (1997). Parasite: Mining Structural Information on the Web. *Proc.of International World Wide Web Conference*.

Tang, C., Yu, Z., You, Z., Zhang, T., & Lu, Y. (2000). Mine the Quasi-Periodicity From Web Data. *Journal of Computer*, 23(1), 52-59.

Tang, C., Lau, R.W.H., Yin, H., Li, Q., Lu, Y., Yu, Z., Xiang, L., & Zhang, T. (1999). Discovering Tendency Association Between Objects with Relaxed Periodicity and its Application in Seismology. *Proc. of ICSC'99*, LNCS 1749, 51-62.

Changjie Tang received his MSc. from the Department of Mathematics, Sichuan University in 1981. He is a Professor and the Head of Computer Science Department of Sichuan University. His current interests are in Web data mining. He has published eight books and more than 100 research papers in journals and international conferences, such as FODO, IFIP, SIGMOD, DASFAA, ICSC, TCS (Theoretical Computer Science), LNCS, JOS, JCST, SC (Science of China). Prof. Tang has served as a PC member of VLDV'97, DASFAA'99, ICSC'99, Pakdd2001, DASFAA2001, WAIM2000, WAIM02 and WAIM03. He is also a vice director of the Database Society of Chinese Computer Federation.(Email: chjtang2002@sohu.com or chjtang@scu.edu.cn)

Qing Li received his BEng. degree from Hunan University (Changsha, China), MSc and PhD degrees from the University of Souther California (Los Angeles, USA), all in computer science. He is currently an Associate Professor at the City University of Hong Kong, as well as an adjunct Professor of the Hunan University. His research interests include database modeling, multimedia retrieval and management, and e-learning systems. Dr Li has published over 100 papers in technical journals and international conferences in these areas, and is actively involved in the research community by serving as a journal reviewer, programme committee chair/co-chair, and as an organizer/co-organizer of several international conferences. Currently he serves as a councillor of the Database Society of Chinese Computer Federation, and as a Steering Committee member of the international WISE Society.(Email: itqli@cityu.edu.hk)

Rynson W.H. Lau received a (top) first class honors degree in Computer Systems Engineering from the University of Kent at Canterbury, England, and a Ph.D. degree in Computer Graphics from the University of Cambridge, England. He is currently an Associate Professor at the City University of Hong Kong. His research interests include computer graphics, virtual reality and multimedia systems.(Email: rynson@cs.cityu.edu.hk)

Tianqing Zhang received the B.E. and M.E degree in computer science from Sichuan University, China, in 1993 and 1996 respectively. He is currently a PhD candidate in the Computer Science Department, Sichuan University. His main research interests are database systems. (Email: zhangtianqing@cs.scu.edu.cn)

Danny Kilis received advanced academic training in both Computer Science as well as Electrical Engineering fields from the University of Minnesota, Minneapolis. He has over 17 years of experience in leading and managing eCommerce and Information Technology projects. His work experience spans across a wide spectrum of functional areas, including eCommerce (B2B and B2C) product development, pre- and post-sales consulting, product marketing and academic research. He is currently the Senior Vice President for the eProduct Development and Services teams in Hong Kong and Shenzhen (China) for PCCW. His teams focus in developing the company's strategic eCommerce software products. He has pioneered the product strategy on the company's ConXerto Product Suite (http://conxerto.pccw.com,) which is a set of software products targeted for the collaborative commerce area. In addition to his product role, Dr. Kilis advocates the group's object-oriented and component development strategies and processes. He is well experienced in such technologies as Internet/extranet architecture, object-orientation, component software engineering, software reuse, and artificial intelligence. Prior to joining PCCW, he also co-founded an offshore eCommerce Solutions Development Center in Manila for GE Information Services, which, in turn, developed the company's global eCommerce software products. He also single-authored some U.S. patents on software technology for IBM.