User Model User-Adap Inter (2009) 19:99–132
DOI 10.1007/s11257-008-9047-z

ORIGINAL PAPER

An architecture for making recommendations to courseware authors using association rule mining and collaborative filtering

Enrique García · Cristóbal Romero · Sebastián Ventura · Carlos de Castro

Received: 19 February 2007 / Revised: 10 September 2007 / Accepted: 21 May 2008 /
Published online: 11 July 2008

Abstract Nowadays we find more and more applications for data mining techniques in e-learning and web-based adaptive educational systems. The useful information discovered can be used directly by the teacher or author of the course in order to improve instructional/learning performance. This can, however, imply a lot of work for the teacher who can greatly benefit from the help of educational recommender systems for doing this task. In this paper we propose a system oriented to find, share and suggest the most appropriate modifications to improve the effectiveness of the course. We describe an iterative methodology to develop and carry out the maintenance of web-based courses to which we have added a specific data mining step. We apply association rule mining to discover interesting information through students' usage data in the form of IF-THEN recommendation rules. We have also used a collaborative recommender system to share and score the recommendation rules obtained by teachers with similar profiles along with other experts in education. Finally, we have carried out experiments with several real groups of students using a web-based adaptive course. The results obtained demonstrate that the proposed architecture constitutes a good starting point to future investigations in order to generalize the results over many course contents.

E. García · C. Romero (✉) · S. Ventura · C. de Castro
University of Córdoba, Campus de Rabanales, Ctra Madrid-Cádiz Km 396, 14071 Córdoba, Spain
e-mail: cromero@uco.es

E. García
e-mail: egsalcines@uco.es

S. Ventura
e-mail: sventura@uco.es

C. de Castro
e-mail: cdecastro@uco.es

Keywords Association rule mining · Recommender systems · e-learning · Web-based adaptive education · Courseware design

1 Introduction

Recently, the huge increase in Internet accessibility has made the concept of online education or e-learning a reality (Itmazi 2005). This is a form of computer-aided instruction that virtually does not depend on the need for a specific location or any special hardware platform (Brusilovsky 2003). Public and private schools are increasingly providing their students with e-learning systems that are also called Learning Management System (LMS). LMSs are software tools designed to manage user learning interventions that offer an extensive range of complementary functionality. Some examples of commercial LMSs are WebCT, Virtual-U, and TopClass, although open source systems such as Moodle, ATutor and ILIAS are gradually becoming more widespread (Itmazi 2005). Although LMSs provide useful tools for computer-supported collaborative learning (such as forums, chat rooms, discussion groups and e-mail), most of them show their contents and educational material to all students in the same way. At the same time students are also completely free to choose their own learning pathway through the course, which is not necessarily the most effective one taking into account their previous knowledge or needs.

One possible solution for this problem is the use of Adaptive and Intelligent Web-Based Educational Systems (AIWBES) (Brusilovsky 2003), which combine the techniques of adaptive systems (Brusilovsky et al. 2007; De Bra and Calvi 1998). These systems build a model for the objectives, preferences and knowledge of an individual user in order to adapt the system to his or her learning needs by means of Artificial Intelligence (AI) techniques from intelligent systems (Brusilovsky et al. 1996; Heift and Nicholson 2001) such as machine learning and data mining (DM). Data mining is part of the process of Knowledge Discovery in Databases (KDD) and is understood to be the non-trivial extraction of previously unknown and potentially useful, valid and comprehensible information from a large volume of data (Klösgen and Zytkow 2002). Hence, certain activities traditionally carried out by the teacher, such as training and monitoring students and diagnosing their limitations, can now be performed by the system.

Many of the systems mentioned above use data mining techniques in order to personalise the output data obtained, avoiding information overload and recommending items required by the current user based on previous interactions of other users with similar profiles (Costaguta 2006). There are different recommendation strategies for user requirements (Zanker and Jessenitschnig 2009) such as knowledge- and utility-based methods, collaborative filtering, association rule mining as well as hybrid variants. Recommendation systems can assist the natural process of relying on friends, classmates, lecturers, and other sources to make choices for learning (Lu 2004). In the educational setting, these recommendation systems can be classified into two types according to their target users (Romero and Ventura 2006). The first is student-oriented (Gaudioso et al. 2003; Zaiane 2002) in order to suggest good learning experiences for the students according to their preferences, needs and level of knowledge, and the second is teacher-oriented (Chen and Wasson 2002; Romero et al. 2003) to help teachers and/or authors of e-learning systems to improve the performance and functions of these systems based on student data.

The application of data mining in e-learning, particularly the teacher-centred approach aimed at improving courses, involves a series of hurdles that need to be overcome (Romero and Ventura 2006). Data mining tools are normally designed more for power and flexibility than for simplicity. Most of the current data mining tools are too complex for educators to use and their features do not cover the scope of what an educator might require (Romero et al. 2008). On one hand, there is a wide variety of e-learning and web-based adaptive courses that can apply data mining which is influenced by three key aspects: first of all, the field of knowledge covered by the course; secondly, the course level (university, secondary or primary school level, special education or any other kind of course); and finally, the level of difficulty of the course, that is, if it is a basic or beginner, intermediate, advanced or an expert's course. On the other hand, the wide range of results that can be obtained, depending on these factors, means that it can be fairly tricky to find therein general repeatable patterns which can be applied to any type of course. Furthermore, educational data sets are normally small (Hamalainen and Vinni 2006) if we compare them to databases used in other data mining fields, such as e-commerce applications, that involve thousands of clients. So, applying data mining with specific filtering parameters can cause problems for association rule discovery in small databases (Zhang and Zhang 2002) where the initial information is insufficient to construct a model that will infer future behaviour.

In this paper we propose a recommender system that uses data mining techniques to provide feedback to courseware authors. The paper is organized as follows: Sect. 2 describes some previous research related to our proposal while Sects. 3 and 4 describe the architecture of the system and its implementation. Experimental tests to prove the validity of the system are described in Sect. 5. Finally, Sect. 6 outlines conclusions and further research.

2 Related works

There are many models or techniques have been used in data mining. In the following subsections, we describe some of the techniques and work most directly related to our proposal.

2.1 Association rule mining

One of the most commonly used data mining techniques in the above-mentioned systems is association rule discovery (Agrawal et al. 1996). Association rules are one of the most popular ways of representing discovered knowledge and describe a close correlation between frequent items in a database. An $X \Rightarrow Y$ type association rule expresses a close correlation between items (attribute-value) in a database. There are many association rule discovery algorithms (Zheng et al. 2001) but Apriori is the first and foremost among them (Agrawal et al. 1996).

Most association rule mining algorithms require the user to set at least two thresholds, one of minimum support and the other of minimum confidence. The support S of a rule is defined as the probability that an entry has of satisfying both X and Y. Confidence is defined as the probability an entry has of satisfying Y when it satisfies X. Therefore the aim is to find all the association rules that satisfy certain minimum support and confidence restrictions, with parameters specified by the user. Therefore, the user must possess a certain amount of expertise in order to find the right support and confidence settings to obtain the best rules.

One possible solution to this problem can be to use an algorithm with less and/or more intuitive parameters. For example, the Weka (Weka 2008) package implements an Apriori-type algorithm that partially solves this problem. This algorithm iteratively reduces minimum support, by a delta factor support (Δs) introduced by the user, until a minimum support is reached or a maximum number of rules (NR) has been discovered. However, it needs some other parameters, such as lower and upper bound support and minimum confidence.

Often, the user can assume that the resulting association rules provide information about the process that generated the database, and that they will be valid in the future, too. However, confidence in training data is only an estimate of the rules' accuracy in the future, and since the space of association rules is searched to maximize confidence, the estimate is optimistically biased. A really important improvement to the Apriori algorithm for use in educational environments is the Predictive Apriori (Scheffer 2005) because it does not require the user to specify any of these parameters (either the minimum support threshold or confidence values).The algorithm aims to find the N best association rules, where N is a fixed number. This setting is more appropriate in many situations because these thresholds may not be easy to specify and a teacher may not be satisfied with either an empty or an outrageously large set of rules. The only parameter entered by the teacher is the number of rules to be discovered, which is a more intuitive parameter. The Predictive Apriori (PA) algorithm strikes an appropriate balance between support and confidence to maximize the probability of accurately predicting the dataset. In order to achieve this, the PA algorithm, using the Bayesian method, proposes a solution that quantifies the expected predictive accuracy ($E(c|\hat{c}, s)$ of an association rule $[x \Rightarrow y]$ with given confidence $\hat{c}$ and the support of the rule's body (the left hand side of the rule) of s. This parameter thus quantifies just how strongly the confidence of a rule has to be corrected given the support of that rule, and it depends on the prior $\pi(c)$ which is the histogram of accuracies of all association rules over the given items for the given database. The PA algorithm is displayed in Table 1.

We can estimate π(c) by drawing many hypotheses at random under uniform distribution, measuring their confidence, and recording the resulting histogram. However, there are many more long rules than there are short ones (the number of distinct item sets grows exponentially in the length). If we drew rules at random, we would almost never get to see short rules; our estimate of $\pi(c)$ for short rules would be poor. In order to avoid this problem, the author of PA algorithm (Scheffer 2005) proposed to run a loop over the length of the rule and, given that length, draw a fixed number of rules. He determines the items and the split into body and head by drawing at random (Step 2).

Table 1 Algorithm Predictive Apriori: discovery of n most predictive association rules

```
Input: $n$ (desired number of association rules), database with items $a_1, \ldots, a_k$
1) Let $\tau = 1$; // initial support
2) For $i = 1$ to $k$ Do: Draw a fix number of association rules [x→y]. Measure their confidence (provided
   $s(x) > 0$). Let $\pi_i(c)$ be the distribution of confidences.
3) For all $c$, Let $\pi(c) = \frac{\sum_{i=1}^{k} \pi_i(c) \binom{k}{i} (2^i - 1)}{\sum_{i=1}^{k} \binom{k}{i} (2^i - 1)}$
4) Let $X_0 = \{\emptyset\}$; $X_1 = \{\{a_1\},\ldots,\{a_k\}\}$ be all item sets with one single element.
5) For $i = 1$ to $k - 1$ While ($i = 1$ or $X_{i-1} \neq \emptyset$)
      (a) If $i > 1$ Then determine the sets of candidate item sets of length $i$
          as$X_i = \{x \cup x' \mid x, x' \in X_{i-1}, |x \cup x'| = i\}$. Eliminate double occurrences of item sets in $X_i$
      (b) Run a database pass and determine the support of the generated items sets. Eliminate item sets
          with support less than $\tau$ from $X_i$.
      (c) For all $x \in X_i$ Call best = GenRule($x$)*;
      (d) If best has been changed, Then Increase $\tau$ to be the smallest number such that $E(c|1, \tau) >$
          $E(c(best[n])|\hat{c}(best[n]), s(best[n]))$. If $\tau >$ database size Then Exit.
      (e) If $\tau$ has been increased in the last step, Then eliminate all item sets from $X_i$ which have support
          below $\tau$.
6) Output $best[1], best[2]\ldots best[n]$, the list of the n best association rules.
* GenRule($x$): find the best rules with body x efficiently
```

We have now drawn equally many rules for each size while the uniform distribution requires us to prefer long rules. There are $\binom{K}{i}$ I item sets of size i over k database items, and given i items, there are $2^i - 1$ distinct association rules (each item can be located on the left or right hand side of the rule but the right hand side must be nonempty). Hence, the following equation gives the probability that exactly i items occur in a rule which is drawn at random under uniform distribution from the space of all association rules over k items.

$$P[i\ items] = \frac{\binom{k}{i} \left(2^i - 1\right)}{\sum_{j=1}^{k} \binom{k}{j} \left(2^j - 1\right)}$$

Therefore, the author of PA estimates the prior over all association rules (Step 3) in a way that accounts for the number of rules with a specific length that exist by weighting each prior for rule length i by the probability of a rule length of i. This can be seen as a Markov Chain Monte Carlo style correction to the prior. Then, the PA generates the frequent item sets, pruning the hypothesis space by dynamically adjusting the minimum support threshold, generating association rules, and removing redundant association rules interleave.

Association rule mining algorithms normally discover a huge quantity of rules and do not guarantee that all the rules found are relevant. Therefore, they must be evaluated in order to find the best rules for a specific problem. Traditionally, the use of objective measures has been suggested (Tan and Kumar 2000), such as support and confidence, mentioned previously, as well as other measures such as Laplace, chi-square statistics, correlation coefficients, entropy gain, interest, conviction, etc. These measures can be used to rank the rules obtained so that the user can select those with the highest values for the most appropriate measures. On the other hand, subjective measures are becoming increasingly important (Silberschatz and Tuzhilin 1996). These measures are based on subjective factors controlled by the user. Most subjective approaches involve user participation in order to express which rules are of the most interest for clarifying and updating previous knowledge.

An Interestingness Analysis System (IAS) was proposed by (Liu et al. 2000). IAS compares the newly discovered rules to the user's current knowledge about the area of interest. Using their own specification language, they indicate their level of knowledge about the matter in question through relationships between the fields or items in the database. Let U be the set of user's specifications representing his knowledge space, and A be the set of newly found association rules. This algorithm implements a pruning technique to remove redundant or insignificant rules by ranking and classifying them into four categories:

Conforming rules: A discovered rule $A_i \in A$ conforms to a piece of user's knowledge $U_j \in U$ if both the conditional and consequent parts of A_i match those of $U_j \in U$ well. They use $conform_{ij}$ to denote the degree of the conforming match.

Unexpected consequent rules: A discovered rule $A_i \in A$ has unexpected consequents with respect to a $U_j \in A$ if the conditional part of A_i matches that of U_j well although the consequent part does not. They use $unexpConseq_{ij}$ to denote the degree of unexpected consequent match.

Unexpected condition rules: A newly found rule $A_i \in A$ has unexpected conditions with respect to a $U_j \in U$ if the consequent part of A_i does matches that of U_j well while the conditional part does not. They use $unexpCond_{ij}$ to denote the degree of unexpected condition match.

Both-side unexpected rules: A discovered rule $A_i \in A$ is unexpected on both-side with respect to a $U_j \in U$ if neither the conditional nor the consequent parts of rule A_i match those of U_j well. They use $bsUnexp_{ij}$ to denote the degree of both-side unexpected match.

The values for $conform_{ij}$, $unexpConseq_{ij}$, $unexpCond_{ij}$, and $bsUnexp_{ij}$ are between 0 and 1. The value "1" represents a complete match, either a completely conforming or a completely unexpected match, and the value "0" represents no match. The user can indicate his knowledge about the matter in question through relationships among the fields or items in the database. After the newly found rules have been analyzed, IAS displays different types of rules that are potentially interesting to the user. IAS shows the essential aspects of the rules in such a way that it can take advantage of human visual capabilities to enable the user to identify the truly helpful rules easily and quickly. These essential aspects are:

1. Types of potentially interesting rules: Different types of pertinent rules should be separated because they give the user different kinds of pertinent knowledge.
2. Degrees of interestingness ("match" values): Rules should be grouped according to their degrees of interestingness. This enables the user to focus his/her attention on the most unexpected (or conforming) rules first and to decide whether to view these rules as being less interesting.
3. Items of interest: showing preferably the items of interest in a rule can be better than seeing the whole rule.

2.2 Collaborative recommender systems

In general, frequent item sets are useful for revealing association rules in large databases. However, when working with separate, relatively small databases, it is essential to learn how to use experience, common sense and models created by other users who have already worked with these databases in the past (Klösgen and Zytkow 2002). There are pro-active methods that use tools to support collaborative work: this multidisciplinary development normally involves experts from different areas of knowledge such as: knowledge engineers in charge of modelling knowledge; knowledge database developers who construct, organise, annotate and maintain these databases; and teams of validating experts who validate elements of knowledge before they are entered into the contents repository. Collaborative Recommender Systems (Mobasher 2006) are based on opinions provided by experts, through explicit or implicit voting systems. The main goal is to suggest better solutions based on overall experience. They are based on social networking, so they are also vulnerable to social attacks (Mehta and Nejdl 2009).

Recommendation techniques for personalization can be classified in different ways (Mobasher 2006) based on data sources themselves as well as on the use made of this data. The Collaborative Filtering System (CFS), also referred to as social filtering, depends on a product database as well as on demographic data and potential consumer evaluations of certain products that have not yet been put to trial. This is perhaps the most familiar, widespread and fully developed of all recommendation techniques (Burke 2000a). The main idea of CFS revolves around computerising the "word of mouth" process that people use to recommend products or services to one another. If users need to choose between various options they have no experience about, they are likely to trust the opinions of those who do have experience. The Knowledge Based System (KBS), on the other hand, aims to suggest objects based on inferences about the user's preferences and needs. Unlike other techniques, it has prior functional knowledge about how a particular item can satisfy a user's needs and therefore can make reasoned judgements about the relationship between this need and a possible recommendation. The user profile can be any knowledge structure that supports this inference. In the case of Google, this would simply be the query entered by the user. In other cases, it might be a more detailed representation of the user's needs. The Entree system (Burke 2000b) uses Case-Based Reasoning (CBR) techniques to make recommendations based on knowledge.

Recommender Systems (RS) are currently applied to many web based sectors, for example, in e-commerce in order to offer personalised client services (Zan 2004), in webpage search engines in order to avoid information overload (Eliassi-Rad and Shavlik 2003), and in digital libraries in order to help users find desirable books or articles (Geyer-Schulz 2003). Another recent field of application for the currently booming RS is e-learning (Rosta and Brusilovsky 2006; Tang and McCalla 2005) which uses different recommendation techniques in order to suggest online learning activities or optimum browsing pathways to students, based on their preferences, knowledge and the browsing history of other students with similar characteristics.

There are several specific research projects on the application of recommender systems and association rule mining in e-learning systems. Wang (2002) developed a portfolio analysis tool based on associative material clusters and the sequences found therein. This knowledge allows educators to study dynamic browsing structures and to identify interesting or unexpected learning patterns. In order to achieve that, Wang discovers two types of relationships: association relations and sequence relations among documents. Minaei-Bidgoli et al. 2004 proposed mining contrast rules that are of interest for web-based educational systems. Contrast rules help to identify attributes that characterize patterns of performance disparity between different groups of students. Markellou et al. (2005) proposed an ontology-based framework and elaborate association rules, using the Apriori algorithm. The role of ontology is to determine which learning materials are the most suitable to recommend to the user. Zaïane and Luo (2001) proposed the discovery of useful patterns based on restrictions in order to help educators evaluate students' activities in web courses. Li and Zaïane (2004) also used recommender agents for e-learning systems which use association rule mining to reveal associations between user actions and URLs. The agent recommends online learning activities or shortcuts on a course web-site based on a learner's access history. Lu (2004) used association fuzzy rules in a personalized e-learning material recommender system. He uses fuzzy matching rules to discover associations between a student's requirements and a list of learning materials. Romero et al. (2003, 2004) proposed the use of grammar-based genetic programming with multi-objective optimization techniques to provide feedback to courseware authors. They discover interesting association rules in students' usage information. Merceron and Yacef (2004) used association rule and symbolic data analysis as well as traditional SQL queries in order to mine student data captured from a web-based tutoring tool. Their goal is to find mistakes that often occur together. Freyberger et al. (2004) use association rules to guide a search for the best fitting transfer model of student learning in intelligent tutoring systems. The association rules determine the operation that needs to be performed on the transfer model to predict a student's possibility of success. Finally, Srivastava et al. (2000) used clustering and association rule mining to extract usage knowledge for the purpose of web personalization. This personalization system can also be used to adapt courses to each student's needs.

3 Architecture of the system

In order to tackle the problems discussed in the introduction section, we are going to propose a collaborative recommender system applied to education. The objective is to help teachers to continually improve and maintain adaptive and non-adaptive e-learning courses. We have used a hybrid recommender system based on CFS and KBS in order to add a feedback stage in two ways. First of all, collaborative filtering will help to discover pertinent relationships among different teachers with similar profiles, each working with their own databases. These similarities or useful relationships will be available to other teachers to assess in terms of applicability and relevance. Secondly, the knowledge database will be strengthened with experiences that, due to their significance, satisfy the needs of many teachers and therefore can give rise to increasingly effective recommendations.

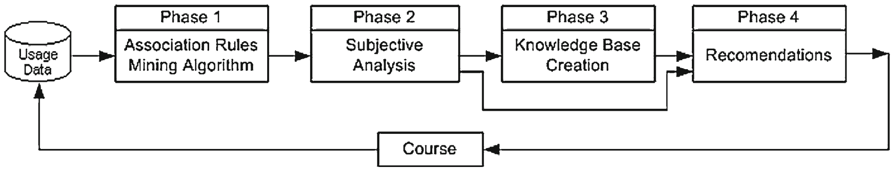


Fig. 1 Main phases of CIECoF architecture

The main phases used in the CIECoF (Continuous improvement of e-learning course framework) architecture are (Fig. 1):

- *Association rules mining*: This phase aims to find association rules on the data set generated as the students complete the course. Once the data has been pre-processed, it is used as input of the Predictive Apriori algorithm, the nucleus of this phase. Also, the teacher could select specific data and attributes in order to restrict the search domain. The output of this module (rules found) is then analyzed by the subjective analysis module.
- *Subjective analysis*: This phase uses a subjective rule evaluation measure (Sect. 3.2) to determine the interestingness of the rules found by association rule mining. It also applies the IAS algorithm to classify the rules in expected or unexpected comparing them with the rules stored in the knowledge base.
- *Knowledge base creation*: This phase combines collaborative filtering techniques with knowledge based techniques to create and to manage the rules repository. The information in the knowledge base is stored in form of tuples (rule-problem-recommendation-relevance) which are classified according to a specific course profile. In order to avoid the cold start issue of collaborative filtering systems, the experts propose the first tuples of the repository and also vote for those tuples proposed by other experts. On the other hand, the teachers could discover new tuples that must be validated by the experts before being inserted in the repository and also votes for the others tuples.
- *Recommendations*: The expected rules found by the phase 2 joined to the more intuitive tuples format mentioned in phase 3, are then used in this last phase to show the teacher, in most of the cases non expert in data mining, possible solutions to some problems detected in the course. The teacher analyzes the recommendation and he determines if it is relevant or not.

The system is based on client-server architecture with N clients, which applies an association rule mining algorithm locally on students' data using an online course. In the server application are included two modules. The first is a web application server so the experts can manage a knowledge base (KB) and can add, delete or edit tuples, as well as being able to vote on the contributions made by other experts in the team. The second module is a web service, which allows the server to share the updated KB with the client in PMML format (Data Mining Group 2006). PMML (Predictive Model Markup Language) is an XML-based language that enables the definition and sharing of predictive models between applications, establishing a vendor-independent means of defining these models, so that problems with proprietary applications and compatibility issues can be circumvented. So, once the updated version of the KB has

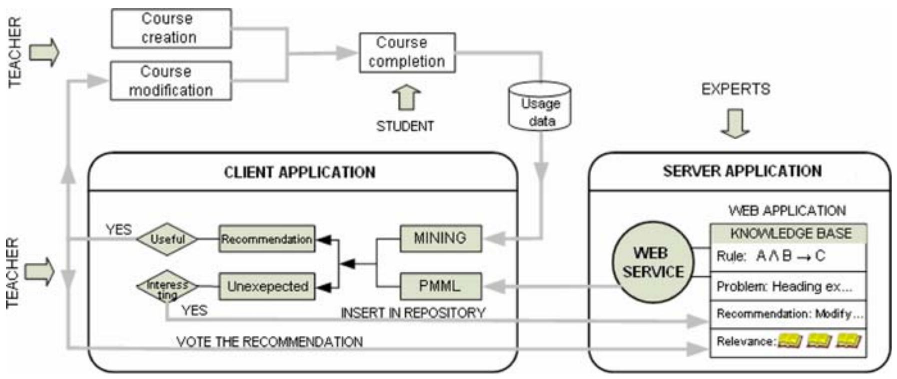


Fig. 2 CIECoF client-server architecture

been downloaded from the server, the client can apply the mining algorithm offline. Client application is part of the iterative methodology (García et al. 2006) that teachers use to develop courses. It is capable of detecting possible problems in the design and content of an e-learning course by adding a feedback or maintenance stage to the course.

As we can see in Fig. 2, there are several stages in this methodology: (1) the initial construction of a course; (2) the completion of the course by the students, during which usage information is transparently compiled and stored in a database; (3) the ongoing improvement stage, which coincides with client application. This last stage contains the core of the rule mining algorithm used (Sect. 3.1). The algorithm together with the KB classifies the rules found as being either expected (if they coincide with the KB), or unexpected (if they do not). If teachers apply a recommendation to the course, they are also implicitly voting on its usefulness in the server knowledge database. Unexpected tuples are ranked according to the IAS algorithm and teachers can tag any that are found interesting. The experts then analyse these unexpectedly 'interesting' tuples and can choose to include them in the KB.

3.1 Association rule mining algorithm

We have implemented an association rule mining algorithm oriented to education which is based on the following algorithms: (1) Predictive Apriori for association rule discovery without parameters; and (2) IAS for subjective analysis and classification of unexpected rules by comparing them to a previously defined knowledge database on the field. The algorithm also includes a new weight-based interestingness measurements presented in the Sect. 3.2, to recommend to the teacher any rules according to:

(a) Other teachers with a similar profile have found useful. The teacher profile is represented as a three-dimensional vector related with the following characteristic of his/her course: Topic (the area of knowledge, e.g. Computer Science or Biology); Level (level of the course, e.g. Universitary, High School, Elementary or Special Education); and Difficulty (the difficulty of the course, e.g., Low or

Table 2 Main algorithm

```
Input: Topic, Level, Difficulty: teacher profile;
       N: number of rules to discover;
1)  Iters = 0;
2)  KB = Get_Rules_fromServer( Topic, Level, Difficulty);
3)  While (teacher doesn't stop) do
4)    Re, Rne = Rules_Mining_Algorithm(N, KB, Iters);
          where $Re_{iters} \neq Re_{iters+1}$, $Rne_{iters} \neq Rne_{iters+1}$
5)    For each i-rule in Re do
6)      Teacher_Vote_Recommendation($Re_i$)
7)    End
8)    For each i-rule in Rne do
9)      If (Interesting($Rne_i$)) then
10)       Add_to_KnowledgeBase($Rne_i$);
11)     End if
12)   End
13)   Iters ++;
14) End while
15) End all
```

High). We use static classification to compare teachers, so similar profile refers to an exact coincidence between one profile and other.

(b) A team of validating experts has voted for in terms of interest or validity.

The algorithm implemented is especially useful in collaborative recommender systems, which can take advantage of the synergies offered by the network, in order to produce recommendations that are increasingly useful and precise.

The main algorithm is interactive and iterative (see Table 2). In each iteration, the teacher runs the mining algorithm in order to find the rules that will act as a basis for recommendations; this can be done as often as necessary.

In step (1) the variable *Iters*, which counts the number of iterations, is initialised at zero; in step (2) the teacher downloads the knowledge base (KB) from the server corresponding to his/her course profile; in step (3) the main loop starts and all its instructions will be executed until the teacher decides to stop it. Step (4) calls up the rule mining algorithm described in Table 3, which returns the sets of recommendations (R_e) and unexpected rules (R_{ne}) discovered where R_e and R_{ne} are different from one iteration to another. From steps (5–7), the teacher votes on whether the recommendation has been useful or not, and in steps (8–12), he/she evaluates unexpected rules to determine whether or not they are useful; unexpected rules might be added to the knowledge base (KB), subject to prior validation by the experts. Finally, in step (13), the *Iters* variable is incremented.

The rule mining algorithm implement is described as follows (see Table 3). Let $accR_i$ (i$=1, 2, \ldots n$) be the predictive accuracy of R_i; R the set of rules discovered by the current teacher, R_e the set of expected rules, and R_{ne} the set of unexpected rules, then $R = R_e \cup R_{ne}$; KB is the set of rules that makes up the knowledge database concerning this field.

In step (1), the *GenRules* function reveals the association rules; this function is provided with the desired number of rules and calls on the PA algorithm.

In step (2), the rule found is classified as being expected if it syntactically matches rule in the current knowledge database, that is, if it has both the same antecedent and

Table 3 Rule mining algorithm

```
Input:  N: number of rules to discover; Iters: number of iterations
        KB: knowledge base;
Ouput:  $R_e$: recommendations set; $R_{ne}$: unexpected rules;
1)  R, accR = GenRules (N, Iters); // Call to Predictive Apriori
2)  $R_e$, $R_{ne}$ = Classify(R);
3)  For each i-rule in $R_e$ do
4)      $WAcc_{Ri}$ = CalculateWeightedAccuracy ($R_i$);
5)  End
6)  For each i-rule in $R_{ne}$ do
7)    For each j-rule in KB do
8)      $conform_{ij}$, $unexpConseq_{ij}$, $unexCond_{ij}$, $bsUnexp_{ij}$ = IAS( );
9)    End
10) End
11) Order all the rules in $R_e$ from largest to smaller Wacc
12) Output the set $R_e$ as the set of recommendations
13) Ouput the unexpected rules $R_{ne}$ according to IAS
14) End all
```

consequent. The rule is classified as unexpected if it does not. From steps (3) to (5), for each rule $R_i \in R_e$, the new weight-based interestingness measurement W*Acc* is calculated (see Sect. 3.2).

From steps (6) to (10) the IAS algorithm is used to calculate the degree to which each unexpected rule Rne coincides with the rules stored in the knowledge base (KB). In our system, all the unexpected rules are ordered as follows: (a) the conformed rules that are the basis of recommendations to the professor; (b) unexpected both-sided rules whose antecedent and consequence have never been mentioned in our knowledge base; (c) the unexpected consequent rules that show us those rules found to be contrary to our existing knowledge; and (d) the unexpected condition rules show us that there are other conditions outside of our specified knowledge range that could be pertinent and conducive to learning.

In step (11), the set *Re* is ordered from highest to lowest based on the previously calculated W*Acc*. Step (12) displays all the recommendations corresponding to each of the previously ordered rules. Finally, in step (13), the teacher is given the chance to view the set of unexpected rules in order to assess which candidates are feasible and desirable for our knowledge database.

3.2 Weight-based rule evaluation measure

In order to help teachers make decisions about which rules to apply, the rules must be ordered in terms of interest. Therefore, a measurement of this interestingness must be established based on the weights reflected by the following parameters:

1) Rule's accuracy calculated by the Predictive Apriori algorithm.
2) How useful this rule has been to other teachers based on their votes.
3) How interesting the rule is according to a team of experts, also using a voting system.

Let $U_1, U_2, \ldots, U_m$, be m different teachers with different data-sources, S_i the set of expected association rules found by U_i $(i = 1, 2, \ldots m)$, $S = \{S_1, S_2, \ldots, S_m\}$; and let $E_1, E_2, \ldots, E_k$, be k different experts. According to Good's definition of weight (Good 1950), the voting for rule R in S can be used to assign the weight W_R to R. In practice, teachers are more interested in applying rules that have received greater support, or more votes, from other teachers.

Let $R = \{R_1, R_2, \ldots, R_n\}$ represent all the rules in S, then the weight of R_i can be defined as:

$$Wteachers_{Ri} = \frac{NumVotesTeachers(R_i)}{\sum_{j=1}^{m} NumVotesTeachers(R_j)} \quad (1)$$

where $i = 1, 2, \ldots, n$ and $NumVotesTeachers(R_i)$ is the number of teachers that have voted for rule R_i in S.

By applying the same reasoning to the experts' votes:

$$Wexperts_{Ri} = \frac{NumVotesExperts(R_i)}{\sum_{j=1}^{k} NumVotesExperts(R_j)} \quad (2)$$

where $i = 1, 2, \ldots, n$ and $NumVotesExperts(R_i)$ is the number of experts that have voted for rule R_i in R.

Therefore, the weight of rule R_i can be expressed as a weighted measurement of the votes registered by the teachers and experts, so that:

$$W_{Ri} = Wteachers_{Ri} * C_u + Wexperts_{Ri} * C_e : C_u + C_e = 1 \quad (3)$$

where C_u and C_e are the weighted coefficients representing the opinions of the teachers and experts respectively.

Once the weight of each rule has been calculated, an interestingness measurement can be devised, which we shall call weighted accuracy (*WAcc*) which includes the first factor mentioned at the start of this section: the predictive accuracy of the rule according to the PA algorithm.

We can define $WAcc_{R_i}$ for rule R_i as:

$$WAcc_{R_i} = W_{R_i} * \frac{\sum_{j=1}^{m} accR_{ij}}{m}$$

where W_{Ri} is the weight of the rule according to Eq. 3, and $accR_{ij}$ are the predictive accuracy results returned by the PA algorithm for each teacher that has voted for the rule R_i.

Next, we describe an example of how all the previously described equations are applied when three experts and three teachers evaluate the tuples or rules. Let U_1, U_2, U_3 designate three different teachers who vote (the rule was useful or not) on a set of rules in $S = \{S_1, S_2, S_3\}$:

S_1 is a set of useful association rules obtained by teacher U_1:

$A \wedge C \rightarrow D$; acc = 0.85
$A \rightarrow B$; acc = 0.70
$B \wedge C \rightarrow E$; acc = 0.75

S_2 is a set of useful association rules obtained by teacher U_2:

$B \rightarrow C$; acc = 0.88
$A \rightarrow B$; acc = 0.76
$B \wedge C \rightarrow E$; acc = 0.71

S_3 is a set of useful association rules obtained by teacher U_3:

$A \wedge C \rightarrow D$; acc = 0.82
$A \rightarrow B$; acc = 0.72

There are a total of four rules in S:

R_1: $A \wedge C \rightarrow D$
R_2: $A \rightarrow B$
R_3: $B \wedge C \rightarrow E$
R_4: $B \rightarrow C$

Starting from the above rules we can see that there are two teachers that vote/support for rule R_1, three teachers that votes for rule R_2, two teachers in favour of rule R_3 and one teacher that votes for the rule R_4. Thus the weight of R_i can be calculated as follows:

$$Wteachers_{\mathrm{R1}} = 2/(2+3+2+1) = 0.25$$
$$Wteachers_{\mathrm{R3}} = 2/(2+3+2+1) = 0.25$$

$$Wteachers_{\mathrm{R2}} = 3/(2+3+2+1) = 0.38$$
$$Wteachers_{\mathrm{R4}} = 1/(2+3+2+1) = 0.13$$

After normalizing between 0 and 1, the weights of the teachers are assigned as follows:

$$Wteachers_{\mathrm{R1}} = 0.66 \quad Wteachers_{\mathrm{R3}} = 0.66$$
$$Wteachers_{\mathrm{R2}} = 1.00 \quad Wteachers_{\mathrm{R4}} = 0.34$$

Experts vote in a similar way but in an explicit way. For example, in Table 4, we can see the votes of three experts for each of the four previous rules. They assign each rule a value from 1 to 5, where 1 represents the lowest value and 5 the highest.

$$Wexperts_{\mathrm{R1}} = 10/42 = 0.24 \quad Wexperts_{\mathrm{R3}} = 10/42 = 0.24$$
$$Wexperts_{\mathrm{R2}} = 15/42 = 0.36 \quad Wexperts_{\mathrm{R4}} = 7/42 = 0.17$$

After normalizing between 0 and 1, the experts' weights are assigned as follows:

$$Wexperts_{\mathrm{R1}} = 0.67 \quad Wexperts_{\mathrm{R3}} = 0.67$$
$$Wexperts_{\mathrm{R2}} = 1.00 \quad Wexperts_{\mathrm{R4}} = 0.47$$

Table 4 Example of experts' voting

Rule	Expert 1	Expert 2	Expert 3	Total
R1	3	3	4	10
R2	5	5	5	15
R3	4	3	3	10
R4	2	3	2	7
			Total Votes:	42

If we fix the values of $C_t = C_e = 0.5$, then we calculate W_{Ri} values as:

$$W_{\mathrm{R1}} = 0.67;\ W_{\mathrm{R2}} = 1.00;\ W_{\mathrm{R3}} = 0.67;\ W_{\mathrm{R4}} = 0.41$$

As we have seen, rule R_2 has the highest voting and the highest weight; and R_4 has the lowest voting and the lowest weight. Once the weight of each rule has been calculated, then we calculate the weighted accuracy (*WAcc*) of each rule as:

$$\begin{aligned}
W\mathrm{Acc}_{\mathrm{R1}} &= 0.67 * (0.85 + 0.00 + 0.82)/3 = 0.37\\
W\mathrm{Acc}_{\mathrm{R2}} &= 1.00 * (0.70 + 0.76 + 0.72)/3 = 0.73\\
W\mathrm{Acc}_{\mathrm{R3}} &= 0.67 * (0.75 + 0.71 + 0.00)/3 = 0.33\\
W\mathrm{Acc}_{\mathrm{R4}} &= 0.41 * (0.00 + 0.88 + 0.00)/3 = 0.12
\end{aligned}$$

After we apply the accuracy corrections to W_{Ri} we can see that the rule R_2 is still the most exact; and R_4 the least exact.

4 Implementation of the system

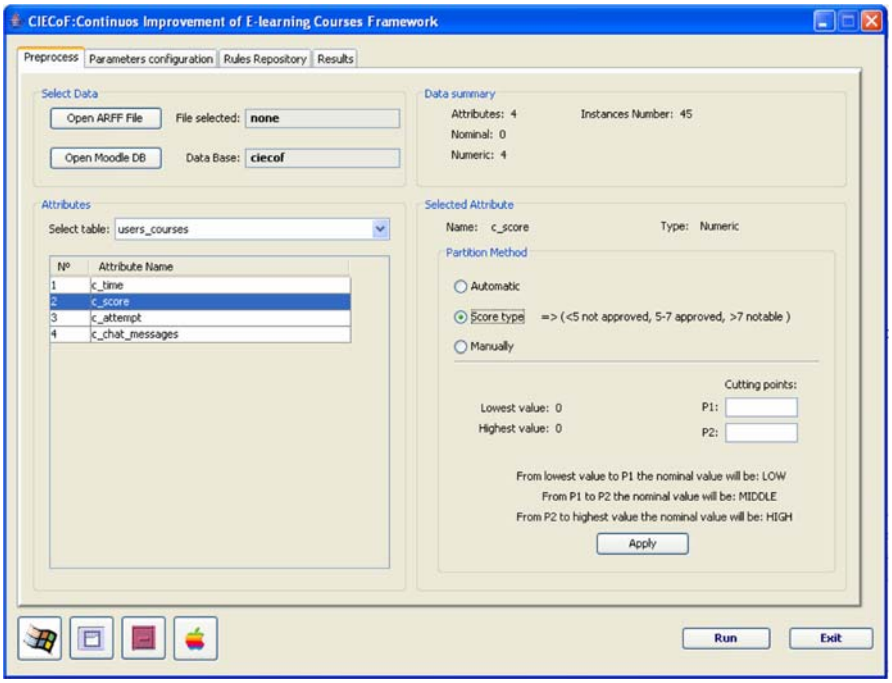
We have implemented a hybrid recommender system based on KBS and CFS in order to avoid the cold-start issue which is presented when the CFS is installed for the first time and we don't have data of any user's votes. Recommendations are made based on the knowledge database created and managed on the server according to different teacher profiles. Furthermore, collaborative filtering is used as a complementary approach, which filters and organises recommendation priority depending on the votes registered by experts and teachers with similar profiles. The experts explicitly vote for tuples by indicating degrees of preference on a form in the web application; however, the teachers vote implicitly to side-step one of the main problems for CFS (how to encourage teachers to vote or evaluate). In this case, if teachers apply one of the recommendations to their course, they are implicitly voting to apply this tuple.

Our system has both a client and a server application that we have implemented in Java language because of their multi-platform characteristics and which will now be described in more detail.

4.1 Client application

The main feature of the client application is its specialization in educational environments. To achieve this, we have used domain specific attributes, filters and restrictions for the rules, and the student's usage dataset from the e-learning course. The interface for client application has four basic panels:

- *Pre-processing*: Before applying a data mining algorithm, the data have to be pre-processed in order to adapt them to our data model. First, the teacher has to select the origin of the data to be mined (see Fig. 3). We have two different formats available for input data: (1) the Moodle relational database, for teachers that work with Moodle as well as the INDESAHC authoring tool (De Castro et al. 2004), so all our attributes are used directly; or (2) a Weka (Weka 2008) ARFF text file, for teachers that use other LMSs and, therefore, other attributes. When the data have been selected, the application shows the teacher only the numerical attributes (see Sect. 5.1) in order to transform them into discreet variables. The objective is to make the rules discovered easier to understand and also to significantly reduce the mining algorithm's running time. We have used three possible nominal values: LOW, MEDIUM and HIGH.
- *Configuration parameters*: The teacher has to set up the parameters and restriction that he/she wants the association rule mining algorithm to use (Fig. 4): the maximum number of rules to be discovered, maximum number of antecedent and consequent elements or items in the rule, the specific attributes that do or do not have to appear in the rule antecedent or consequent. In order to restrict the search field, we have also added a few parameters related with the analysis depth. Firstly, the teacher must select the level to carry out the analysis: course, unit, lesson and



Select Data
File selected: none
Data Base: ciecof

Data summary
Attributes: 4 Instances Number: 45
Nominal: 0
Numeric: 4

Attributes
Select table: users_courses

Nº	Attribute Name
1	c_time
2	c_score
3	c_attempt
4	c_chat_messages

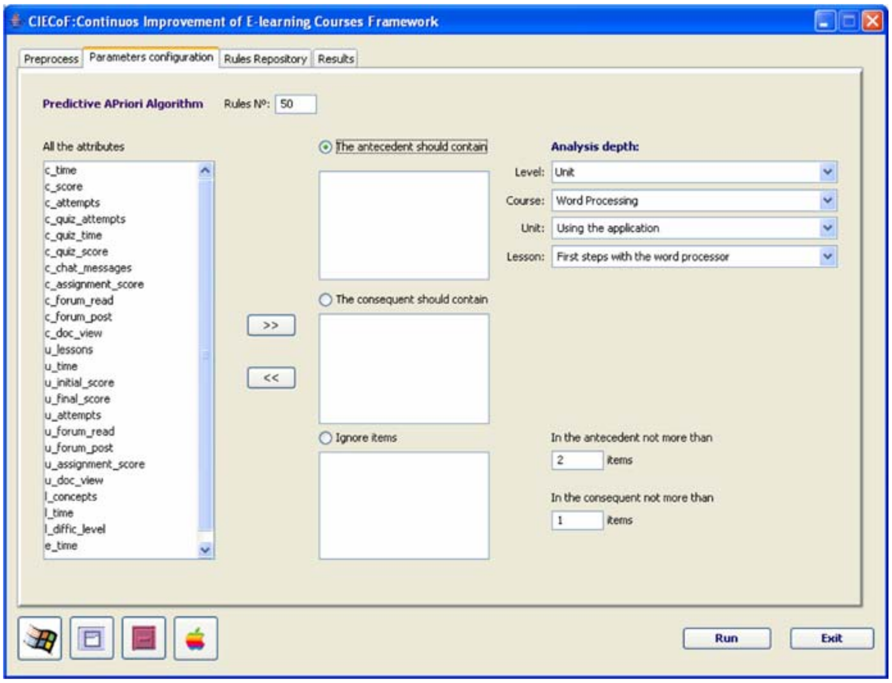
Selected Attribute
Name: c_score Type: Numeric

Partition Method
○ Automatic
[✓ Score type] => (<5 not approved, 5-7 approved, >7 notable)
○ Manually

Cutting points:
Lowest value: 0 P1:
Highest value: 0 P2:

From lowest value to P1 the nominal value will be: LOW
From P1 to P2 the nominal value will be: MIDDLE
From P2 to highest value the nominal value will be: HIGH

Fig. 3 Pre-process panel



CIECoF:Continuos Improvement of E-learning Courses Framework

Predictive APriori Algorithm   Rules Nº: 50

All the attributes

c_time
c_score
c_attempts
c_quiz_attempts
c_quiz_time
c_quiz_score
c_chat_messages
c_assignment_score
c_forum_read
c_forum_post
c_doc_view
u_lessons
u_time
u_initial_score
u_final_score
u_attempts
u_forum_read
u_forum_post
u_assignment_score
u_doc_view
l_concepts
l_time
l_diffic_level
e_time

[✓ The antecedent should contain]
The consequent should contain
Ignore items

Analysis depth:
Level: Unit
Course: Word Processing
Unit: Using the application
Lesson: First steps with the word processor

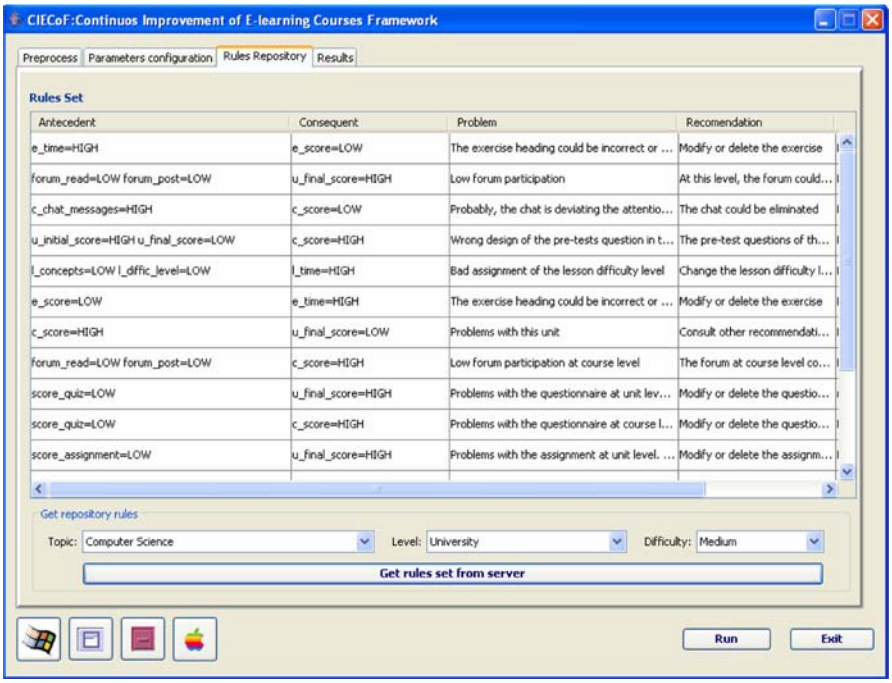
In the antecedent not more than 2 items

In the consequent not more than 1 items

Fig. 4 Parameters configuration panel

others tables such as course-unit, course-lesson, course-exercise, course-forum, unit-exercise, unit-lesson, lesson-exercise among others. Then, the teacher must select a particular course, unit or lesson in order to do rule mining only with the specified data at the specified level.

– *Rules Repository*: The rules repository (see Fig. 5) is the knowledge database upon which the subjective analysis of the discovered rules is based. Since a specific rule and/or specific recommendation that has been discovered in one course does not necessarily have to be valid or applicable to another different course, so we classify the rules in the repository according to the teacher profile: Topic, Level and Difficulty. Before running the algorithm, the teacher downloads the current knowledge database from the server (button *Get rules set from server*), according to his/her course profile. The personalisation of the tuples returned by the server is based on these three filtering parameters, along with the type of course to be analysed. So, the teacher only downloads tuples that match each profile. The information provided by the system for each tuple of the repository is: the rule itself (antecedent and consequent), the problem detected by the rule and an associated recommendation for its solution. In order to identify each tuple, additional information is also included, such as the name of the author, the date and evaluation of the rule. The rules repository is created on the server (Sect. 4.2), based on the educational considerations of experts and the experience garnered from other similar e-learning courses.



Rules Set

Antecedent	Consequent	Problem	Recomendation
e_time=HIGH	e_score=LOW	The exercise heading could be incorrect or ...	Modify or delete the exercise
forum_read=LOW forum_post=LOW	u_final_score=HIGH	Low forum participation	At this level, the forum could...
c_chat_messages=HIGH	c_score=LOW	Probably, the chat is deviating the attentio...	The chat could be eliminated
u_initial_score=HIGH u_final_score=LOW	c_score=HIGH	Wrong design of the pre-tests question in t...	The pre-test questions of th...
l_concepts=LOW l_diffic_level=LOW	l_time=HIGH	Bad assignment of the lesson difficulty level	Change the lesson difficulty l...
e_score=LOW	e_time=HIGH	The exercise heading could be incorrect or ...	Modify or delete the exercise
c_score=HIGH	u_final_score=LOW	Problems with this unit	Consult other recommendati...
forum_read=LOW forum_post=LOW	c_score=HIGH	Low forum participation at course level	The forum at course level co...
score_quiz=LOW	u_final_score=HIGH	Problems with the questionnaire at unit lev...	Modify or delete the questio...
score_quiz=LOW	c_score=HIGH	Problems with the questionnaire at course l...	Modify or delete the questio...
score_assignment=LOW	u_final_score=HIGH	Problems with the assignment at unit level. ...	Modify or delete the assignm...

Get repository rules

Topic: Computer Science Level: University Difficulty: Medium

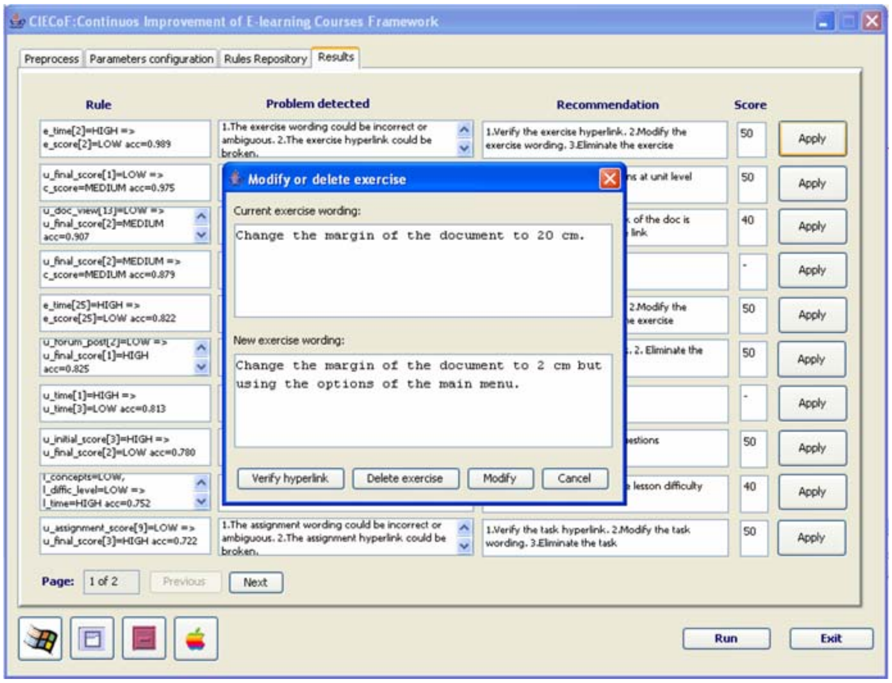
Get rules set from server

Fig. 5 Rules repository panel

– *Results*: Finally, after downloading the rule repository and configuring the application parameters or using default values, the teacher executes the association rule algorithm. Then, client application shows the results obtained in a table (see Fig. 6), with the following fields: rule, problem, recommendation, score and apply button. There are two types of recommendations:

  (1) Active, if it implies a direct modification of the course content or structure. Active recommendations can be linked to: modifications in the formulation of the questions or the practical exercises/tasks assigned to the students; changes in previously assigned parameters such as course duration or the level of lesson difficulty; or the elimination of a resource such as a forum or a chat room. For example, we can see in Fig. 6 that the exercise wording had a writing error (20 cm instead of 2 cm) and then the teacher has corrected it and also has added some more information.
  (2) Passive, if they detect a more general problem and point the teacher towards more specific recommendations.

For active recommendations, by clicking the *Apply* button, the teacher will be shown the area of the course that the recommendation refers to (see Fig. 6) so that he/she can carry out the modification, change, elimination, etc. Each time a teacher applies an active recommendation, he/she is implicitly voting for that tuple.


CIECoF:Continuos Improvement of E-learning Courses Framework

Rule	Problem detected	Recommendation	Score
e_time[2]=HIGH => e_score[2]=LOW acc=0.989	1.The exercise wording could be incorrect or ambiguous. 2.The exercise hyperlink could be broken.	1.Verify the exercise hyperlink. 2.Modify the exercise wording. 3.Eliminate the exercise	50
u_final_score[1]=LOW => c_score=MEDIUM acc=0.975		ns at unit level	50
u_doc_view[13]=LOW => u_final_score[2]=MEDIUM acc=0.907		of the doc is link	40
u_final_score[2]=MEDIUM => c_score=MEDIUM acc=0.879			-
e_time[25]=HIGH => e_score[25]=LOW acc=0.822		2.Modify the e exercise	50
u_forum_post[2]=LOW => u_final_score[1]=HIGH acc=0.825		. 2. Eliminate the	50
u_time[1]=HIGH => u_time[3]=LOW acc=0.813			-
u_initial_score[3]=HIGH => u_final_score[2]=LOW acc=0.780		estions	50
l_concepts=LOW, l_diffic_level=LOW => l_time=HIGH acc=0.752		lesson difficulty	40
u_assignment_score[9]=LOW => u_final_score[3]=HIGH acc=0.722	1.The assignment wording could be incorrect or ambiguous. 2.The assignment hyperlink could be broken.	1.Verify the task hyperlink. 2.Modify the task wording. 3.Eliminate the task	50

Page: 1 of 2

Modify or delete exercise

Current exercise wording:

```
Change the margin of the document to 20 cm.
```

New exercise wording:

```
Change the margin of the document to 2 cm but
using the options of the main menu.
```

Fig. 6 Results panel

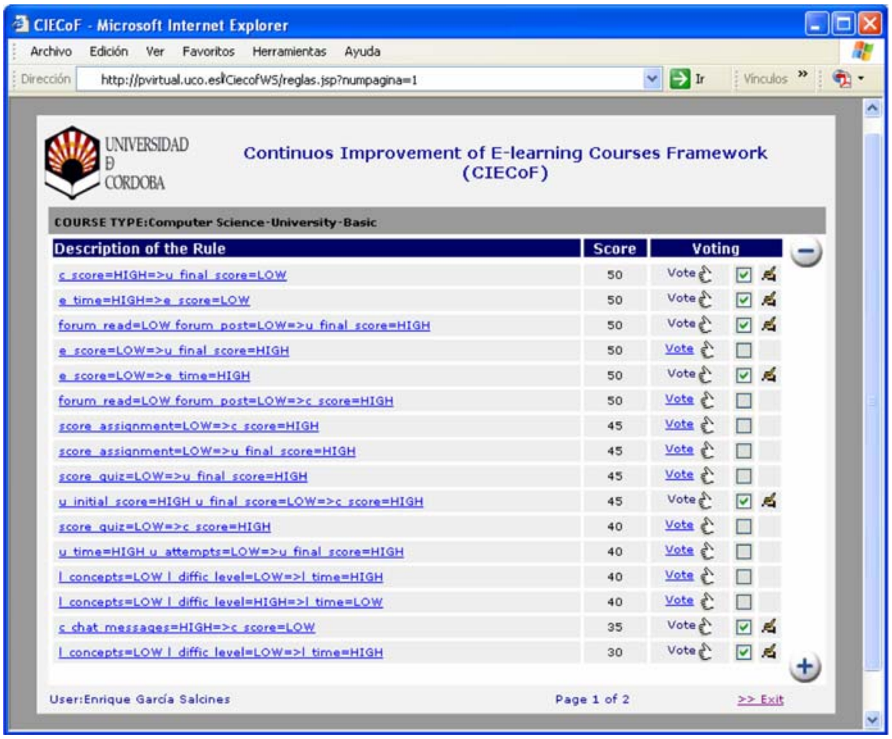
4.2 Server application

On the server side, we have implemented a web application (see Fig. 7) to manage the knowledge database or repository. In order to access absolutely all the editing options for the repository, a basic profile was created, which is the profile of the experts in the educational domain. These experts have permission to introduce new tuples into the rule repository and vote for existing ones. Based on the votes registered by experts, the $W_{experts}$ parameter is calculated. Implicit votes are also stored, which are registered by clients in their local analyses; based on these votes, $W_{teachers}$ is calculated.

In order to allow information exchange (tuples) between client and server, we have developed a web service. It keeps the current repository updated in a PMML file. Each time that a client application updates its repository, the parameters used in the algorithm described in Sect. 3.1 are recalculated and the tuples are reordered in the repository, taking into account the W*Acc* accuracy parameter.

Both experts and teachers participate in the creation of the knowledge base. Initially the knowledge base was empty and experts proposed tuples. Let's see how experts and teachers vote.

On one hand, each expert, using the server application, voted for each tuple in the repository, according to the approaches specified in Fig. 8. Expert evaluation has been divided into two groups of evaluation criteria or approaches: A_1 (expert evaluation) and A_2(expert decision), with three options or questions each. Be making W_1, W_2 the


CIECoF - Microsoft Internet Explorer

Archivo Edición Ver Favoritos Herramientas Ayuda

Dirección http://pvirtual.uco.es/CiecofWS/reglas.jsp?numpagina=1

UNIVERSIDAD D CORDOBA

Continuos Improvement of E-learning Courses Framework (CIECoF)

COURSE TYPE:Computer Science-University-Basic

Description of the Rule	Score	Voting		
c score=HIGH=>u final score=LOW	50	Vote	☑	
e time=HIGH=>e score=LOW	50	Vote	☑	
forum read=LOW forum post=LOW=>u final score=HIGH	50	Vote	☑	
e score=LOW=>u final score=HIGH	50	Vote	☐	
e score=LOW=>e time=HIGH	50	Vote	☑	
forum read=LOW forum post=LOW=>c score=HIGH	50	Vote	☐	
score assignment=LOW=>c score=HIGH	45	Vote	☐	
score assignment=LOW=>u final score=HIGH	45	Vote	☐	
score quiz=LOW=>u final score=HIGH	45	Vote	☐	
u initial score=HIGH u final score=LOW=>c score=HIGH	45	Vote	☑	
score quiz=LOW=>c score=HIGH	40	Vote	☐	
u time=HIGH u attempts=LOW=>u final score=HIGH	40	Vote	☐	
l concepts=LOW l diffic level=LOW=>l time=HIGH	40	Vote	☐	
l concepts=LOW l diffic level=HIGH=>l time=LOW	40	Vote	☐	
c chat messages=HIGH=>c score=LOW	35	Vote	☑	
l concepts=LOW l diffic level=LOW=>l time=HIGH	30	Vote	☑	

User:Enrique García Salcines Page 1 of 2 >> Exit

Fig. 7 Server application interface

weights assigned by the system administrator to the two groups of options A_1 and A_2, we can calculate the total score of a tuple according to:

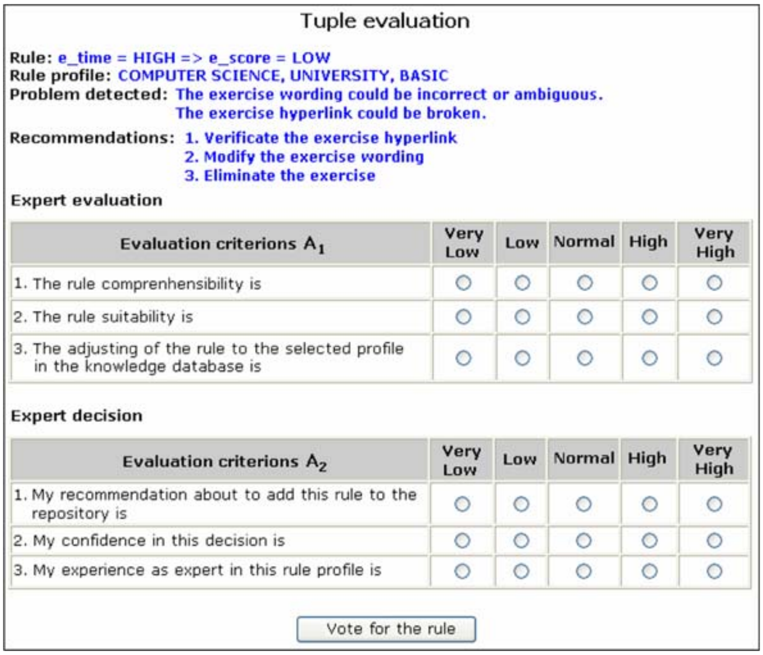
$$NumVotesExpert = W_1{}^*\overline{A}_1 + W_2{}^*\overline{A}_2$$

where $\bar{A}_1$ and $\bar{A}_2$ are the average score given by experts to each option in the group. In our experiment the we have fixed values of $W_1 = W_2 = 0.5$, due to we consider the two groups of evaluation criteria have the same importance. The *NumVotesExpert* values are between 0 and 100 and they are distributed, depending of the vote, in the following way: Very Low option (20 points), Low (40 points), Normal (60 points), High (80 points), and Very High (100 points).

On the other hand, as we have said previously, teachers vote implicitly; that is, if teachers apply one of the recommendations to their course, they are automatically voting for its applicability to this tuple:

$$NumVotesTeacher = 100 * TeacherVote$$

where *TeacherVote* is a binary variable with values of true (1) or false (0) according to whether the teacher votes for the rule or not. Once we have calculated the *NumVotes-Expert* which is used in Eq. 1 and *NumVotesTeachers* used in Eq. 2, we can calculate



Tuple evaluation

Rule: e_time = HIGH => e_score = LOW
Rule profile: COMPUTER SCIENCE, UNIVERSITY, BASIC
Problem detected: The exercise wording could be incorrect or ambiguous.
The exercise hyperlink could be broken.
Recommendations: 1. Verificate the exercise hyperlink
2. Modify the exercise wording
3. Eliminate the exercise

Expert evaluation

Evaluation criterions A_1	Very Low	Low	Normal	High	Very High
1. The rule comprenhensibility is	○	○	○	○	○
2. The rule suitability is	○	○	○	○	○
3. The adjusting of the rule to the selected profile in the knowledge database is	○	○	○	○	○

Expert decision

Evaluation criterions A_2	Very Low	Low	Normal	High	Very High
1. My recommendation about to add this rule to the repository is	○	○	○	○	○
2. My confidence in this decision is	○	○	○	○	○
3. My experience as expert in this rule profile is	○	○	○	○	○

Fig. 8 Form used by to the expert for evaluating tuples

the W_R (Eq. 3). In this case, we have fixed the values of $C_t = C_e = 0.5$, granting the same importance or weight to the vote of the teachers and of the experts. Finally, the score of each rule (see Fig. 7) is obtained by multiplying the *WAcc* values by 100 in order to show them in the range 0 and 100 instead of 0 and 1.

5 Experimental results

In order to test our system, we have carried out some experiments on an educational dataset. We have used real data gathered from students in a pilot experiment , called "*Cordobesas Enredadas*" and carried out in Cordoba (Spain) in 2004–2005, with respect to the technological literacy of women in rural settings,. In this project, 7 adaptive web-based courses were developed based on subjects included in the ECDL (*European Computer Driving Licence*) and Open Office, the free-distribution office package. The courses were developed using INDESAHC (De Castro et al. 2004), an authoring tool to create adaptive hypermedia courses compatible with Moodle. In our experiment, three experts in the Computer Sciences and Artificial Intelligence area in Cordoba University, Spain have also participated and were responsible for proposing the initial tuples in the repository. And there have also been two other teachers involved from the same area (the authors of the courses themselves), so the teacher profile is thus fixed at: Computer Science (Topic), Universitary (Level), and Basic (Difficulty).

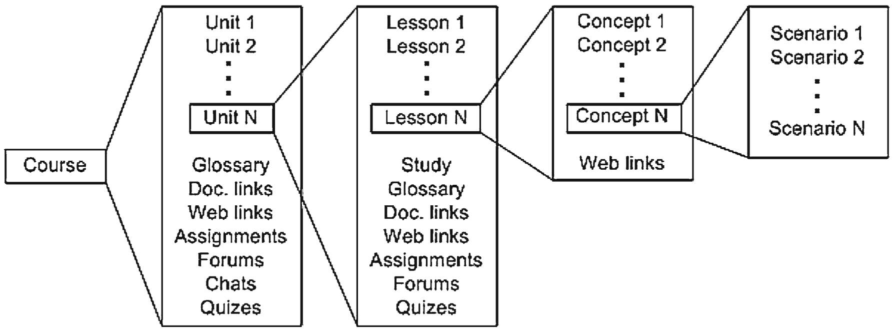


Fig. 9 INDESAHC domain model

5.1 INDESAHC data

The definition of the course syllabus in INDESAHC is based on a hierarchical domain model in which a course is composed of teaching units divided into lessons, each of which containing a series of concepts explained or assessed through scenarios or web pages (see Fig. 9). An adaptation model was also included in order to adapt all the contents to each student's level of knowledge. The specific adaptive techniques that we have used are adaptive link hiding and annotation (De Bra and Calvi 1998). We have classified all the contents of the course into different levels of difficulty (3 levels in this case). Thus, the system adapts the contents of the course (difficulty level) depending on each student's current level of knowledge.

Table 5 shows, on one hand, attributes related to adaptive hypermedia courses which have been added to the Moodle database as new tables. On the other hand, we can see attributes related to teaching resources such as forums, chat rooms, questionnaires and tasks, which have been also introduced from the INDESAHC interface.

5.2 Data pre-processing

Data pre-processing of LMS is a little simpler due to Moodle, and most LMS employ user authentication (password protection) in which logs have entries identified by users, since users have to log-in (Romero et al. 2008). In this way, sessions are already identified since users may also have to log-out and this eliminates the need for typical user and session identification tasks. So, the data gathered by an LMS may require less cleaning and pre-processing than data collected by other web-based systems. Although the amount of work required in data preparation is less, we have carried out two main pre-processing tasks:

- Data selection. It is necessary to decide which courses can be most benefited by mining. From the 7 courses available, we have selected the "Word Processing" one since it has the greatest number of activities and resources.
- Data cleaning. We have carried out cleaning for two main reasons. First, it was discovered that very high values were often recorded for attribute *time* because the student had left the computer without first exiting the exercise, concept or section.

Table 5 Attributes used in association rules mining process

Level	Attribute	Description
Course	c_time	Time taken by the student to complete the course
	c_score	Average final score for the course
	c_attempts	Number of attempts before passing the course
	c_quiz_attempt	Total number of attempts in the quiz
	c_quiz_time	Total time taken in the quiz
	c_quiz_score	Score obtained in the quiz
	c_chat_messages	Number of messages sent in the chat room
	c_assignment_score	Score in the assignment
	c_forum_read	Number of messages read in the forum
	c_forum_post	Number of messages posted in the forum
	c-doc_view	If the document or web link has been viewed
Unit	u_lessons	Number of lessons in a unit
	u_time	Time taken by the student to complete the learning unit
	u_initial_score	Student's score in the unit pre-test
	u_final_score	Student's final score on completing the unit
	u_attempts	Number of attempts before passing the unit
	u_forum_read	Number of messages read in the forum
	u_forum_post	Number of messages posted in the forum
	u_assignment_score	Score in the assignment
	u_doc_view	If the document or web link has been viewed
Lesson	l_concepts	Number of concepts in the lesson
	l_time	Time taken by the student to complete the lesson
	l_diffic_level	Level of difficulty of the lesson as defined by the teacher
Exercise	e_time	Time taken by the student to complete the exercise
	e_score	Score obtained in the exercise

In order to correct this, any times that exceeded a maximum established value were considered noisy data, and this maximum value was assigned to any apparently erroneous data. Secondly, it was discovered that some students had not completed all the course activities. Whenever possible, the students were contacted and asked to complete the course so that their information could be used. When this was not possible, the information regarding these students was discarded.

- Data discretization. The transformation into discreet variables can be seen as a categorisation of attributes that takes a small set of values. The basic idea involves partitioning the values of continuous attributes within a small list of intervals. Our process of discretization used three possible nominal values: LOW, MEDIUM and HIGH. And we have used three partition methods (Liu et al. 2002): equal width method, score type method and a manual method (where the teacher sets the limits of the categories manually).
- Data Integration. Normally, in a data mining problem, a single dataset must first be established if there are data that come from different sources. In this case, we have data from two sources: (1) the tables that stored student monitoring data in the specific attributes of INDESAHC; and (2) the tables used by Moodle, which stored the other information about the course such as forums, chat rooms and tasks. Using these data, a temporary database was created where rule mining was applied.

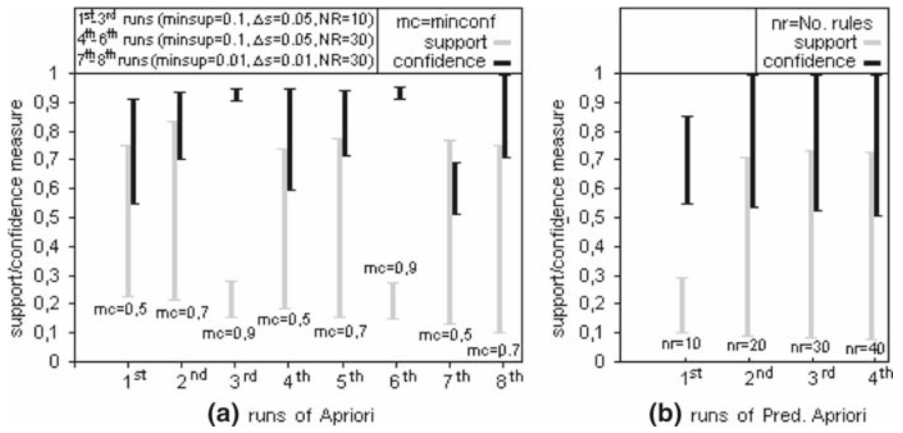


Fig. 10 Results of running the Apriori and Predictive Apriori algorithms on the query table *courses_exercises*

– Data Filtering. Before applying the rule mining algorithm, the teacher could also restrict the search domain by specifying the level of granularity of the analysis, for example, at a subject, lesson or exercise level. The resulting temporary table in this case would, therefore, only contain attributes and transactions from students with respect to the level selected. The system could also find interesting relationships between attributes from different tables, for example if the teacher selected a course-subject or subject-exercises, the temporary table created would contain attributes and transactions from more than one table.

5.3 Comparing the performance of association rule mining algorithms

In order to select the association rule mining algorithm for our CIECoF system, we have performed some tests with the course usage data found in students' tables such as: *students_courses, students_units, students_lessons, students_exercises, students_forums, students_quiz, students_task*, among others. In Fig. 10 we show the results obtained when comparing the support/confidence measure obtained by several runs of the Apriori and the Predictive Apriori algorithms using the data from the students' interaction with the first exercise in the query table *courses_exercises*, which contains 90 transactions with the following attributes: *c_time, c_score, e_time, e_score*. Figure 10a shows the initial execution for Apriori (Weka implementation), varying parameters. Figure 10b shows Predictive Apriori results, varying the number of rules (NR) to be discovered. In this case, starting from the second run (20 best solutions), the support ranges of the rules found are more uniform, varying from 0.08 to 0.7.

By comparing these results obtained in Fig. 10, some conclusions can be reached, which were found also in other tests and are described here. (1) The performance of Apriori depends heavily upon the choice of minimum support and confidence: we cannot be sure that a professor who is not an expert in data mining will obtain the best rules when assigning default values to input parameters. (2) The first execution of the PA algorithm obtains rules that, regardless of a low degree of support, present a high degree of confidence. As the first execution of the Apriori algorithm does not obtain these rules, the Apriori had to be run several times, varying its input parameters to obtain similar results to the PA. (3) The PA also discovers rules with low support and high confidence, which are not found by the Apriori. These specific rules are very interesting in education because they detect small groups of students who differ from the average (students with some type of problem). In fact, when teachers find these types of rules, they can identify those students in order to give them more personalised attention. Hence, for all the abovementioned reasons, we have used the PA as the basic rule algorithm in our CIECoF system.

5.4 Analysis of the recommendation effectiveness

In order to verify the effectiveness of the changes made by the teachers in the course, based on the recommendations suggested by the system, it is important to bear two points of view in mind: (1) the teacher's perspective, in terms of the percentage of apparently corrected problems, based on initial recommendations, that reappear in successive courses with different groups of students; and (2) the perspective of the students with respect to how the removal of those problems based on the recommendations influences their final score. Two hypotheses can initially be drawn from these aspects. Firstly, if the changes made by the teacher are 100% effective, then these problems should not be detected again in subsequent groups of students doing a course that has already been updated by applying the corrections. And secondly, if these problems do not happen again, then students' scores should improve.

We have implemented an iterative methodology to improve the course gradually with use (see Fig. 11). Using the recommendations obtained from the usage data of different groups of students, successive corrections to the course improve it step by step. In order to calculate the effectiveness of these recommendations (EfecRec$_{1,i}$), we use Eq. 4 where *TotalNew*$_1$ represents the total number of recommendations found when the usage data of the first group of students were analysed, which led to changes in the structure or content of the course. *TotalRep*$_{1,i}$ is the total number of recommendations that are repeated in consecutive runs of the same course, always applying the corrections with each different group of students. Thus, the effectiveness of the changes made can be calculated, based on the recommendations proposed in the initial stage (the first course run) with respect to stage $i\,(i = 2, 3, \ldots, N)$, which corresponds to subsequent runs, as follows:

$$EfectRec_{1,i} = \frac{TotalNew_1 - TotalRep_{1,i}}{TotalNew_1} \tag{4}$$

On the other hand, we can also evaluate the effectiveness of the corrections made following the recommendations in terms of the students. To do so, we compare the final marks obtained by students (average score and standard deviation) in a subsequent improved version of the same courses. We have used three different groups of 45 students who completed the course in the way indicated in Fig. 11. In order to eliminate

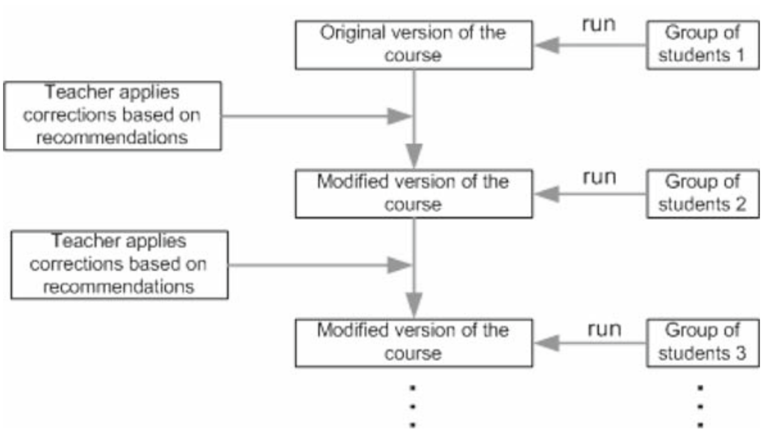

Original version of the course
run
Group of students 1
Teacher applies corrections based on recommendations
Modified version of the course
run
Group of students 2
Teacher applies corrections based on recommendations
Modified version of the course
run
Group of students 3

Fig. 11 Iterative methodology for improving the course

Table 6 Results from the teacher's point of view when applying our system consecutively to data from the three groups of students

Group	TotalRec	TotalNew	TotalRep	TotalRelevant (%)	Relevants (%)	EfectRec (%)
1	50	21	–	21	42.0	–
2	50	5	6	11	22.0	71.4
3	50	5	3	8	16.0	85.7

TotalRec refers to the total recommendations provided by the system. TotalNew refers to the total recommendations provided by the system which the teacher considered useful and applicable. TotalRep refers to initial recommendations that, even though applied by the teacher, reappeared in the same tuples in consecutive runs of the course

the influence of some external factors which might alter the results of the research, such as previous computing knowledge, average age of the group and level of education, the composition of the two groups was forced to fit the following requirements: (1) students with no prior knowledge of computers, which was relatively easy since the courses were aimed at computer literacy in rural settings; (2) the average age of the group had to be very similar; (3) the level of education was similar and above intermediate.

In Table 6 we show the effectiveness percentages of recommendations (column referenced as *EfectRec*) according to (4), as well as the percentages of relevant recommendations found in consecutive improvements to the course (column referenced as *Relevants*). All the changes made in different versions were attributed to the recommendations and therefore no modification was based on initiatives coming from the teachers themselves.

In Table 7 we compare the students' marks in order to determine the effectiveness of recommendations from the students' point of view.

We have reached several conclusions when analysing the results of Tables 6 and 7:

1) As we foresaw in the initial hypothesis, the effectiveness percentage veers towards 100% with subsequent improved versions of the course.

Table 7 Results from the point of view of the student

Group	Mark	*p*-value 1-2	*p*-value 1-3	*p*-value 2-3
1	6.55 ± 0.30			
		<0.0001		
2	6.95 ± 0.56		<0.0001	
				>0.05
3	7.10 ± 0.42			

The mark refers to the final average scores and standard deviation of each group. p-Value 1-2 and 1-3 show the p-values using t-Student's test comparing group 1 to group 2, group 1 to group 3 and group 2 to group 3

2) Not only did the effectiveness percentage increase, but there was also a corresponding decrease in the total number of recommendations associated with the problems detected. This is an indication that the course went on improving.
3) When the marks achieved by the three different groups of students were compared (*p*-values), the slight improvement observed is a further indication of the effectiveness of the system. Mainly if we compare the different modified versions of the course with the original course (group 1 vs. 2 and 3). Also, we can see that there aren't significant differences between the next consecutives modifications (group 2 vs. 3). Therefore, the first modification in the course (more relevant rules discovered) affects more in the effectiveness of the system than the following ones (less relevant rules discovered).
4) The percentages of relevant recommendation get lower throughout the different versions of the courses, so the proportion of change in course content also decreases.
5) New problems are detected with each new group of students. One possible reason might be the different prerequisite skills among students.
6) Some problems reoccurred throughout several improvements. These problems could be due to some of the changes made in course design, which were actually quite subjective, i.e. a change in the classification of lesson difficulty or of the estimated duration of a subject.

5.5 Interpretation of discovered rules

The teacher or course author has a crucial role in our methodology because he/she can also guide the search of rules by imposing some subjective restrictions (see Fig. 5). To do so, the teacher uses his own knowledge and experience in education. For example, he/she can decide to use data about one specific unit, lesson or even of the whole course, and whether or not to use data only about times, score or participation to construct rule antecedents and consequents.

It is important to point out that the comprehensibility and interestingness of rules are subjective concepts that are difficult to quantify effectively. Due to this, we have used constraint-based mining (Han et al. 1999), in which the teacher provides constraints that guide the search. We use three types of constraints:

1. Data constraints: the teacher can specify the relevant data set for the mining task.

2. Rule constraints: the teacher can select specific constraints for the rules to be mined.
3. Interestingness constraints: the teacher can specify the values or ranges of a measure interesting for himself.

As we have mentioned previously, our objective is to show a group of useful rules to the teacher, so that he/she can make decisions about which changes would improve the performance of the course. From a semantic point of view, our resulting rules match the following pattern:

$$\textbf{IF } \textit{Time}|\textit{Score}|\textit{Participation} \textbf{ AND} \ldots \textbf{ THEN } \textit{Time}|\textit{Score}|\textit{Participation}$$

Where *Time*, *Score* and *Participation* are thereby generic attributes referring to: the reading time for course, units, lessons and exercises (HIGH, MEDIUM and LOW values); information on students' scores in the test and activities questions (HIGH, MEDIUM and LOW values); and lastly, participation refers to how the students have used the collaborative resources such as forum and chat (HIGH, MEDIUM and LOW values). Based on the rules discovered, the teacher can decide which of the relationships expressed are desirable or undesirable, and whether or not to apply the recommendation in order to strengthen or weaken the relationship (namely changing or modifying the contents, structure and adaptation of the course, etc.).

The relationships that are shown in discovered rules can refer to the course, units, lessons, or scenarios of concepts (namely instructional and activity pages related to concepts). Next, we describe some examples of the general patterns found in rules of interest offering the teacher useful information about how to improve a course. We also describe some of their possible interpretations. It is important to highlight that a single rule can have several interpretations. Therefore the system will always show all the recommendations related to a detected problem, and it is the teacher him/herself who actually decides what recommendations to use. We should also mention that all the following examples always correspond to rules with a high degree of support, that is, they are confirmed by most of the students.

$$\textbf{IF } \textit{ExerciseTime} = \text{ HIGH } \textbf{THEN } \textit{ExerciseScore} = \text{ LOW}$$

This pattern indicates that the students have spent a long time doing the exercise although the final score has been low. Two possible interpretations of this pattern are:

1) The wording of this exercise could be incorrect or ambiguous, giving place to several interpretations. In this case the teacher can correct the exercise's wording or eliminate it altogether if necessary.
2) The exercise is quite difficult and for this reason the students spend relatively more time than on other exercises, resulting in a lower score. In this case, the teacher will determine if the exercise is in accordance or not with the difficulty level of the lesson.
3) The students were weak on prerequisite skills. In this case, the teacher should consult other recommendations of higher level such as the level obtained in the unit pre-test, in order to confirm that interpretation. From here on, we will present

only those interpretations that could be difficult to detect and represent a possible problem to be corrected.

An example of this type of rule is:

$$\textbf{IF}(\text{e\_time}[25] = \text{HIGH})\ \textbf{THEN}\ (\text{e\_score}[25] = \text{LOW}),\ \text{supp.} = 0.91,\ \text{accur.} = 0.82$$

This rule means that if students took a long time to complete exercise number 25, then they got a low score in this exercise. This rule can indicate that there is a problem with this specific exercise, which was part of the: "application use" subject; "first steps with the word processor" lesson; and "renaming and saving a document" concept. The exercise was an INDESAHC interactive video scenario in which the student had to simulate the necessary steps for completing an activity using the mouse. In this specific case, the question was confirmed to be ambiguous and interpretable in several ways, so the wording was changed. Other rules with a similar pattern were also found in multiple-choice or linking type questions.

$$\textbf{IF}\ \textit{UnitForumParticipation} = \text{LOW}\ \textbf{THEN}\ \textit{UniFinalScore} = \text{HIGH}$$

This pattern indicates that there was not much participation in the unit forum although the students obtained a high final score for the unit in question. Three possible interpretations of this pattern are:

1) The forum is not necessary for this unit, so the teacher can eliminate it.
2) There are problems concerning the tutors responsible for forum maintenance, so the teacher should analyze the causes of these problems in detail.
3) Strong students are more autonomous while weaker students are more inclined to use and consult the forum.

An example of this type of rule is:

$$\textbf{IF}\ (\text{u\_forum\_read}\ [2] = \text{LOW})\ \textbf{AND}\ (\text{u\_forum\_post}\ [2] = \text{LOW})$$
$$\textbf{THEN}\ (\text{u\_final\_score}\ [1] = \text{HIGH}),\ \text{supp.} = 0.85,\ \text{accur.} = 0.83$$

This rule shows that if students send or read few messages in forum 2 (unit 1), then they get a high score for this unit. This rule shows that the forum may not be necessary or that there were problems with it. This type of rule raises the issue about whether the forum is really necessary at certain levels of the domain hierarchy. In fact, the forum was removed in this case.

There are also patterns which did not provide any useful information for problem detection or that only provided the teacher with obvious information. For example:

$$\textbf{IF}\ \textit{AssignmentScore} = \text{HIGH}\ \textbf{THEN}\ \textit{UnitScore} = \text{HIGH},$$

This relationship indicates that if students obtain a good score in the assignment, then they also obtain a good score in the unit. An example of this type of discovered rules is the following:

$$\textbf{IF}\ (\text{u\_assignment\_score}\ [9] =\ \text{HIGH})$$
$$\textbf{THEN}\ (\text{u\_final\_score}[3] =\ \text{HIGH}),\ \text{supp.}\ = 0.75,\ \text{accur.} = 0.72$$

This rule shows that if the score of assignment 9 is high, then the final score obtained in unit 3 is high. This rule is totally logical for the teacher and it does not contribute any new information about how to improve the course.

There are patterns that can generate recommendations of a passive type. For example:

$$\textbf{IF}\ \textit{UnitFinalScore} =\ \text{LOW}\ \textbf{THEN}\ CourseScore = \text{MEDIUM or HIGH}$$

This pattern indicates that if students obtain a low score in a specific unit, then they obtain a medium or high final score in the course. This rule can generate a passive type of recommendation because it could indicate the possibly of problems in the unit and that other more specific recommendations should be consulted at unit level. An example of this type of discovered rule is:

$$\textbf{IF}\ (\text{u\_final\_score}\ [1] = \text{LOW})\ \textbf{THEN}\ (\text{c\_score}\ = \text{MEDIUM}),$$
$$\text{supp.} = 0.80,\ \text{accur.} = 0.88$$

This rule shows that if the score of unit 1 is low, then the final score of the course is average. This rule detects a possible problem with unit 1 but in order to detect more specific problems, the teacher must consult other tuples.

Lastly, we show an example of an unexpected rule:

$$\textbf{IF}\ (\text{l\_concepts}[13] = \text{LOW}\ \textbf{AND}\ \text{l\_diffic\_level}[13] =\ \text{LOW}$$
$$\textbf{THEN}\ (\text{l\_time}\ [13] = \text{HIGH}),\ \text{supp.} = 0.6,\ \text{accur.} = 0.85$$

This rule shows that if the number of concepts included in the lesson is LOW and the level of difficulty assigned to that lesson is LOW, then the time taken to complete the lesson is HIGH. The fact that students have spent a long time completing a lesson that supposedly is not very difficult and contains few new concepts could indicate that the level of difficulty has been incorrectly classified. In fact, in this case, the course designer decided that the level of difficulty for this lesson should be changed to MEDIUM.

6 Conclusions and future work

This paper describes a recommender system that uses interactive iterative association rule mining and collaborative filtering in order to help the teacher maintain and continuously improve e-learning courses. The system enables the locally obtained rules to be shared by other teachers and experts with a similar profile. It uses a weight-based evaluation measurement to rank the rules discovered, taking into account the opinion of both experts and teachers to produce more effective recommendations.

We have carried out several experiments using data from real students in order to test our system. First, we compared the classical Apriori algorithm to the Predictive Apriori algorithm. We show that the Predictive Apriori resulted in a better performance than the Apriori and required fewer parameters, making it more intuitive for a non-expert in data mining. Then, we carried out other experiments to evaluate the performance of the system from the points of view of both teacher and student. The results demonstrated our starting hypotheses: fewer problems are detected in subsequent improved versions of the courses and the students' final marks improve as the teacher corrects problems. Finally, the general opinion of both teachers and experts has been very positive. They have demonstrated a high degree of motivation and have especially liked the novelty of using students' data to improve e-learning courses, to be able to apply modifications to courses directly from the system and have the possibility of working and sharing information with other teachers and educational experts. However, experts have indicated that the creation of the repository or knowledge database is a hard task.

For future work, we aim to carry out a more detailed study involving more students, more groups and more experts and teachers from other areas (unrelated to computer science) in order to obtain a more heterogeneous teacher's profile. This will allow the study of other interesting questions such as: Is it possible that different teachers in different areas might coincide in their evaluation of patterns?; What is the behavior of experts and teachers as they progress through a course?; Can tuples that are found to be valid and useful in one course later be applied to another course with a different profile?; How many false positives are generated (i.e. rules the system generates that are rejected by the user)?. These aspects could lead to a validation that would focus solely on a detailed analysis of the changes made and whether the process is efficient and likely to be complementary to non guided course content revision. Finally, we also want to strongly emphasise the collaborative measures of the approach analysing the relevance ratio of expert and teacher votes (C_t and C_e) and how much work can be facilitated for experts.

Acknowledgements The authors gratefully acknowledge the subsidy provided by the Spanish Department of Research under TIN2005–08386-C05–03 Project and to the subsidy provide by the Provincial Government of Cordoba under the project reference *ECDL/DIPUCO/MEM/04–0001bis*.

References

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.: Fast discovery of association rules. Advances in Knowledge Discovery and Data Mining, pp. 307–328. AAAI Press, Menlo Park, CA (1996)

Brusilovsky, P., Kobsa A., Nejdl W., (eds.): The adaptive web: methods and strategies of web personalization. Lecture Notes on Computer Science, vol. 4321. Springer-Verlag, Heidelberg (2007)

Brusilovsky, P.: Adaptive and intelligent web-based educational systems. Int. J. Artif. Intell. Educ. **13**, 159–169 (2003)

Brusilovsky, P., Schwarz, E., Weber, G.: ELM-ART: an intelligent tutoring system on World Wide Web. Third International Conference on Intelligent Tutoring Systems, pp. 261–269. Montreal, Canada (1996)

Burke, R.: Semantic ratings and heuristic similarity for collaborative filtering. Seventeenth National Conference on Artificial Intelligence, pp. 1–7. Austin, Texas, July 30th–August 3rd (2000a)

Burke, R.: Knowledge-based Recomendador Systems. In A. Kent (ed.) Encyclopedia of Library and Information Systems, vol. 69, Suppl 32, pp. 1–23. Marcel Dekker, New York (2000b)

Chen, W., Wasson, B.: Coordinating collaborative knowledge building. International Conference Applied Informatics, vol. 25(1), pp. 1–10. Innsbruck, Austria (2002)

Costaguta, R.: Una Revisión de Desarrollos Inteligentes para Aprendizaje Colaborativo Soportado por Computadora. Revista Ingeniería Informática, No. 13, available at http://www.inf.udec.cl/revista (2006)

Data Mining Group: Predictive Model Markup Language (PMML), available at http://www.dmg.org/pmml-v3-0.html (2006)

De Bra, P., Calvi, L.: AHA! An open Adaptive Hipermedia Architecture. New Rev. Hipermedia Multimedia **4**, 115–139 (1998)

De Castro, C., García, E., Romero, C., Ventura, S.: Herramienta autor INDESAHC para la creación de cursos hipermedia adaptativos. Revista Latinoamericana de tecnología Educativa **3**, 349–367 (2004)

Eliassi-Rad, T., Shavlik, J.: A system for building intelligent agents that learn to retrieve and extract information. Int. J. User Model User-Adapted Interact. Special Issue User Model. Intell. Agents **13**(4), 35–88 (2003)

Freyberger, J., Heffernan, N., Ruiz, C.: Using association rules to guide a search for best fitting transfer models of student learning. Workshop on analyzing student–tutor interactions logs to improve educational outcomes at ITS conference, pp. 1–4. Maceio, Brazil (2004)

García, E., Romero, C., Ventura, S., de Castro, C.: Using rules discovery for the continuous improvement of e-learning courses. 7th International Conference on Intelligent Data Engineering and Automated Learning – IDEAL 2006, pp. 887–895. Burgos, Spain, LNCS, vol. 4224 (2006)

Gaudioso, E., Santos, O., Rodriguez, A., Boticario, J.: A proposal for modeling a collaborative task in a web-based collaborative learning environment. 9th International Conference on User Modeling, pp. 70–80. Johnston, PA, USA (2003)

Geyer-Schulz, A.: An architecture for behavior-based library recomendador systems. Inf. Technol. Libraries **22**, 165–174 (2003)

Good, I.: Probability and The Weighting of Evidence. Charles Griffin & Co. Ltd., London (1950)

Hamalainen, W., Vinni, M.: Comparison of machine learning methods for intelligent tutoring systems. The 8th International Conference in Intelligent Tutoring Systems, pp. 525–534. Jhongli, Taiwan (2006)

Heift, T., Nicholson, D.: Web delivery of adaptive and interactive language tutoring. Int. J. Artif. Intell. Educ. **12**, 310–324 (2001)

Itmazi, J.A.S.: Sistema Flexible de gestión del e-learning para soportar el aprendizaje en las universidades tradicionales y abiertas. PhD Thesis, University of Granada, Spain (2005)

Klösgen, W., Zytkow, J.M.: Handbook of Data Mining and Knowledge Discovery. Oxford University Press (2002)

Li, J., Zaïane, O.R.: Combining usage, content, and structure data to improve web site recommendation. The 5th International Conference on E-Commerce and Web Technologies, pp. 305–315. Zaragoza, Spain, Springer, LNCS, vol. 3182 (2004)

Liu, B., Wynne, H., Shu, C., Yiming, M.: Analyzing the subjective interestingness of association rules. IEEE Intel. Syst. **15**(5), 47–55 (2000)

Liu, H., Hussain, F., Tan, C.L., Dash, M.: Discretization: an enabling technique. J. Data Mining Knowledge Discov. **6**(4), 393–423 (2002)

Lu, J.: Personalized e-learning material recommender system. International Conference on Information Technology for Application, pp. 374–379. Harbin, China (2004)

Markellou, P., Mousourouli, I., Spiros, S., Tsakalidis, A.: Using semantic web mining technologies for personalized e-learning experiences, pp. 461–826. Web-Based Education, Grindelwald, *Switzerland* (2005)

Mehta, B., Nejdl, W.: Unsupervised strategies for shilling detection and robust collaborative filtering. Int. J. User Model. User-Adapted Interact. Special Issue Data Mining Personalization doi:10.1007/s11257-008-9050-4

Merceron, A., Yacef, K.: Mining student data captured from a web-based tutoring tool: initial exploration and results. J. Interactive Learning Res. **15**(4), 319–346 (2004)

Minaei-Bidgoli, B., Tan, P., Punch, W.: Mining interesting contrast rules for a web-based educational system. The Twenty-First International Conference on Machine Learning Applications, pp. 1–8. Alberta, Canada (2004)

Mobasher, B.: Data mining for web personalization. In: Brusilovsky P., Kobsa A., Nejdl W. (eds.), The Adaptive Web: Methods and Strategies of Web Personalization, pp. 90–135. Lecture Notes in Computer Science, vol. 4321. Springer-Verlag, Heidelberg (2006)

Romero, C., Ventura, S., de Bra, P., Castro, C.: Discovering prediction rules in AHA! Courses. 9th International User Modeling Conference, pp. 25–34. Johnston, PA, USA (2003)

Romero, C., Ventura, S., de Bra, P.: Knowledge discovery with genetic programming for providing feedback to courseware author. User Modeling and User-Adapted Interaction: J. Personalization Res. **14**(5), 425–464 (2004)

Romero, C., Ventura, S.: Educational data mining: a survey from 1995 to 2005. Expert Syst. Appl. **33**(1), 135–146 (2006)

Romero, C., Ventura, S., Garcia, E.: Data mining in course management systems: Moodle case study and tutorial. Comput. Educ. **51**, 368–384 (2008)

Rosta, F., Brusilovsky, P.: Social navigation support in a course recommendation system. Adaptive Hypermedia and Adaptive Web-Based Systems: 4th International Conference, AH 2006, pp. 91–100. Dublin, Ireland, (2006)

Scheffer, T.: Finding association rules that trade support optimally against confidence. Intell. Data Anal. **9**(4), 381–395 (2005)

Silberschatz, A., Tuzhilin, A.: What makes pattterns interesting in Knowledge discovery systems. IEEE Trans Knowledge Data Eng. **8**(6), 970–974 (1996)

Srivastava, J., Mobasher, B., Cooley, R.: Automatic personalization based on web usage mining. Commun. Assoc. Computing Machinery **43**(8), 142–151 (2000)

Tan, P., Kumar, V.: Interesting measures for association patterns: a perspectiva. Technical Report TR00-036, Department of Computer Science, University of Minnnesota, USA (2000)

Tang, T., McCalla, G.: Smart recommendation for an evolving E-Learning system: architecture and experiment. Int. J. E-Learning **4**(1), 105–129 (2005)

Wang, F.: On using data-mining technology for browsing log file analysis in asynchronous learning environment. Conference on Educational Multimedia, Hypermedia and Telecommunication, pp. 2005–2006. Denver, USA (2002)

Weka: Weka project available at http://www.cs.waikato.ac.nz/ml/weka/ (2008)

Zaiane, O.: Building a recommender agent for e-learning systems. International Conference on Computer in Education, pp. 55–59. Auckland, New Zealand (2002)

Zaïane, O., Luo, J.: Web usage mining for a better web-based learning environment. Conference on Advanced Technology for Education, pp. 60–64. Banff, Alberta, (2001)

Zan, H.: A graph model for E-commerce recomendador systems. J. Am. Soc. Inf. Sci. Technol. **55**(3), 259–274 (2004)

Zanker, M., Jessenitschnig, M.: Case-studies on explicit customer requirements in recommender systems. Int. J. User Modeling and User-Adapted Interact. Special Issue Data Mining Personalization doi:10.1007/s11257-008-9048-y

Zhang, C., Zhang, S.: Association Rule Mining. Springer, Berlin (2002)

Zheng, Z., Kohavi, R., Mason, L.: Real world performance of association rules. Sixth ACM SIGKDD International Conference on Knowledge Discovery & Data Mining **2**(2), 86–98 (2001)

Authors' vitae

Enrique García is an Assistant Professor in the Computer Science Department of the Cordoba University, Spain, and a Ph.D. candidate in Computer Science at Granada University. His primary interests lie in the areas of data mining and recommender systems. His paper summarizes the current state of his thesis work on the field of educational data mining for e-learning improvement.

Cristóbal Romero is an Assistant Professor in the Computer Science Department of the Cordoba University, Spain. He received his Ph.D. in Computer Science from the University of Granada in 2003. His research interests lie in artificial intelligence and data mining in education. He has published several papers about Educational Data Mining in international journals and conferences and he has co-edited the book Data Mining in e-learning.

Sebastián Ventura is an Associate Professor in the Computer Science Department of the Cordoba University, Spain. He received his Ph.D. in the Sciences from the Cordoba University in 1996. His research interests lie in soft-computing, machine learning, data mining and its applications. He has published several papers about Educational Data Mining in international journals and conferences and he has co-edited the book Data Mining in e-learning.

Carlos de Castro is an Associate Professor in the Computer Science Department of the Cordoba University, Spain. He received his Ph. D. in the Sciences from the University of Cordoba in 1983. His research interests lie in e-learning methodologies, resources and accessibility.