Predicting Student Actions in a Procedural Training Environment

Diego Riofrío-Luzcando, Jaime Ramírez and Marta Berrocal-Lobo

Abstract—Data mining is known to have a potential for predicting user performance. However, there are few studies that explore its potential for predicting student behavior in a procedural training environment. This paper presents a collective student model, which is built from past student logs. These logs are firstly grouped into clusters. Then an extended automaton is created for each cluster based on the sequences of events found in the cluster logs. The main objective of this model is to predict the actions of new students for improving the tutoring feedback provided by an intelligent tutoring system. The proposed model has been validated using student logs collected in a 3D virtual laboratory for teaching biotechnology. As a result of this validation, we concluded that the model can provide reasonably good predictions and can support tutoring feedback that is better adapted to each student type.

Index Terms—Educational Data Mining, e-learning, Procedural Training, Intelligent Tutoring Systems

1 Introduction

Educational data mining has already achieved promising results, for example, with regard to the analysis of student performance or the prediction of student grades, especially in the field of web e-learning [1], [2]. However, there is hardly any research in the literature that has integrated data mining techniques into intelligent tutoring systems (ITSs) [3], for example, to provide customized tutoring for each student.

This paper presents a collective student model that has been designed to anticipate the actions that students are likely to take while completing a practical assignment in an educational environment for procedural training. This model is created from activity records or logs collected from students with a similar background that previously completed the same practical assignment. As we will see later, an ITS equipped with this collective student model can use hints to stop students from making certain errors or from floundering with the practical assignment.

It is sometimes a good idea to let students make mistakes from which they learn. In other cases, however, it is better to give students the minimum amount of support that they need to progress independently towards problem solving and overcome their zones of proximal development [4]. In this way, each student learns not from his or her mistakes but with a little bit of help. If necessary, the tutor gradually increases the level of support or scaffolding every time the student makes a mistake or gradually reduces the amount of help provided when the student makes progress [5], [6]. Another reason for helping students not to make mistakes is to prevent student frustration when they fail too often.

The proposed collective student model consists of several clusters of students (Section 4), each of which contains an extended automaton (Section 4.1). This automaton is a directed graph adapted for our purposes. As explained later, these clusters will help to provide automatic tutoring adapted to each student type. In order to confirm this claim, we validated the model using student logs collected in a 3D virtual laboratory for teaching biotechnology. This validation had two main goals: i) verify that the prediction error is acceptable for tutoring purposes; and ii) check whether clustering methods can classify students into groups that require different tutoring feedback. As we will see later, although students had a lot of freedom of action in this virtual laboratory, the model was reasonably reliable at predicting student actions and provided a useful classification of students into clusters according to their performance.

The structure of the remainder of the paper is as follows. Section 2 shows relevant works in the field of educational data mining. Section 3 describes the proposed ITS architecture, which would be able to leverage the collective student model detailed later in Section 4. Section 5 reports model validation detailing the method followed in this study and discussing its results. Finally, Section 6 outlines the conclusions of this research and some future work.

- *D. Riofrío-Luzcando and J. Ramírez are with ETS de Ingenieros Informáticos, UPM, Spain.*
  E-mail: {driofrio, jramirez}@fi.upm.es
- *Marta Berrocal-Lobo is with E.T.S.I. Forestal y del Medio Natural, UPM, Spain, and Center for Plant Biotechnology and Genomics, UPM-INIA, Spain.*
  E-mail: m.berrocal@upm.es

2 Related Work

The related work is divided into two sections. Section 2.1 briefly presents the main goals of educational data mining and mentions some of the key results with respect to web based e-learning systems. Section 2.2 focuses on systems for procedural training equipped with ITSs, whose student logs have been analyzed by means of data mining.

2.1 Educational Data Mining (EDM)

EDM tries to use data sourced from the repositories of different types of learning environments to better understand learners and learning [2]. Some general applications of EDM are [7]: communicate student activities and usage of online courses to educational stakeholders; help with course maintenance and improvement by analyzing usage data; analyze how well the domain is structured by student performance prediction; generate recommendations for students; predict student grades and learning outcomes; and model students.

Given the scope of our research, the literature review will focus on the last three EDM applications.

Some researchers [8], [9], [10] use data mining to provide hints, feedback or recommendations about which content is best for each student. Some of these use an ITS [11]. The most frequently used data mining techniques for this purpose are association, sequencing, classification and clustering.

Other researchers [12], [13], [14] try to predict different kinds of student learning outcomes such as final grades or dropouts. The most frequent data mining techniques used in this group are association, classification and clustering.

Student modeling [15], [16], [17], [18], [19], [20] has several applications such as the detection of student behavior or learning problems. This group most frequently uses the same data mining techniques as above, plus statistical analyses, Bayes networks, psychometric models and reinforcement learning.

One noteworthy paper in the last group processes Moodle logs to discover a specific student behavior model [21]. They divide these logs into student groups with similar characteristics using a clustering method and then apply process mining to each cluster to create a model (represented by a directed acyclic graph) that shows the most frequent sequences of student actions. An interesting conclusion of this paper, which is relevant for our research, is that graphs, models or visual representations are easier to comprehend. Teachers and students find this summarized information more accessible. Therefore, this information could be very useful for monitoring the learning process and providing feedback.

As we find from the works referenced in this section, most research in EDM has focused on studying data or logs registered by web e-learning systems, like Moodle or MOOCs, or data collected from student curricula [1], [2], [22], [23].

To the best of our knowledge, there is no any other proposal of predictive model in the literature to support procedural training environments that relies on data mining. Hence, next section will focus on a closed related area where we have actually been able to find some interesting contributions, procedural training environments equipped with ITSs, whose student logs have been processed through data mining.

2.2 Intelligent Tutoring Systems with EDM in Procedural Training

Two well-known ITSs that employ EDM are Assistment [24] and Cognitive Tutor Algebra I [25], [26]. Both are web tools that guide students through the process of solving math exercises. There are several data mining studies using data collected from these two environments, but they do not report whether or not the results of these studies have been used to improve the tutoring services. For example, data from Assistment were used to create a model to predict when a student is about to ask for a hint [27]. EDM is applied in Cognitive Tutor Algebra I to create a model that detects student attitudes/feelings such as engagement, concentration, confusion, frustration, and boredom solely from student interactions within the tutor [28].

To the best of our knowledge, there is only one learning environment for procedural training equipped with an ITS that relies on EDM. It is called CanadarmTutor [11]. It simulates the Canadarm2 robotic arm used in the International Space Station. This ITS provides assistance to users on how to perform a correct sequence of arm operations to reach a goal. To do this, it integrates a cognitive model to assess skills and spatial reasoning, and an expert system that automatically generates solution paths. Because of the ill-defined characteristics of the problem-solving procedure, the ITS uses data mining techniques to extract a partial task model from past user solutions. Using this model the ITS can recognize the learner plan and provides assistance based on the prediction of the user's next action.

Despite the research that has already been conducted in this area, the community is missing more generally applicable results [29], for example, predictive models that can be used in more than one different context. There is also a remarkable shortage of intelligent educational systems that take advantage of models developed by EDM [3].

The research presented in this paper represents a step forward towards the development of an ITS that leverages a collective student model computed by means of EDM to offer better tutoring feedback. Moreover, this model is intended for procedural training in learning environments and is domain independent.

3 ITS Architecture Proposal

In order to leverage the presented collective student model, we propose an extension of a previous ITS architecture, the MAEVIF architecture [30], [31], [32], which is depicted in Figure 1. MAEVIF is a multi-agent architecture that is an adaptation of the classic ITS architecture for learning environments specialized in training. Within the extension of MAEVIF, this model encompasses a new agent, called Collective Student Agent.

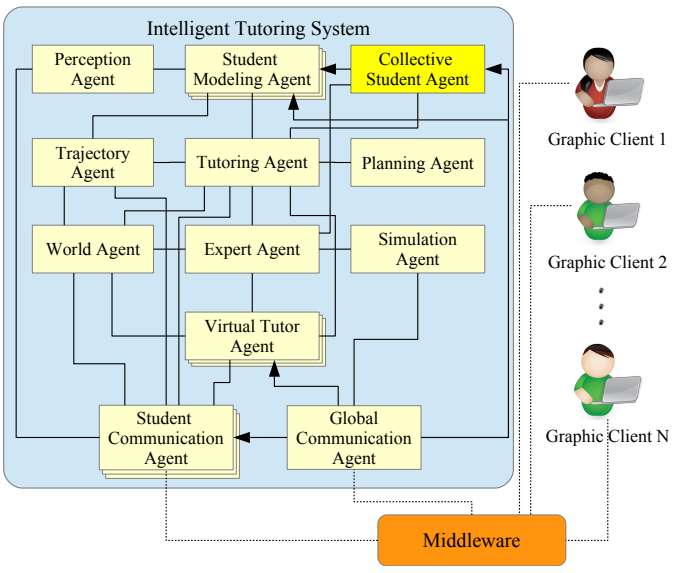

Fig. 1. MAEVIF architecture with the new agent

Of the MAEVIF agents, let us focus on the Student Modeling Agent [32], [33], [34]. The Student Modeling Agent

contains an adaptive, extensible and reusable student model that infers the student knowledge state using a pedagogic-cognitive diagnosis with non-monotonic reasoning abilities. The purpose of this agent is to discover each student's learning status, that is, what he or she does or does not know about the subject. This can serve as support for the personalized automatic tutoring of each student. In this way, if the student model contains enough information on a particular student, it will provide good predictions of his/her behavior. For example, if the student model knows that a student recently performed a task correctly, it is very likely that this student will perform the same (or a very similar) task correctly again.

One disadvantage of this student model is, however, that, if queried about the attainment level of a particular learning objective, it will need a lot of background information about the student with respect to that learning objective in order to give a reliable enough response, and this information will often not be available. For example, this may be the case if it is the student's first attempt at an exercise. This may constitute a problem when the tutoring agent needs to predict the student's next actions, because if the student modeling agent is not confident enough that the student knows which actions to take next, it will not be able to provide a good prediction.

As this paper shows, if the student model does not possess enough information on a particular student, the collective student model will be a reasonably good alternative. The collective student model comprises summarized data on past student action events that are used to predict the actions that a student under supervision is most likely to take next. The premise for creating this model is that the behavior of past students doing a practical assignment should be similar to current students with the same training completing the same practical assignment.

4 Description of the Collective Student Model

This model is created using historical data from past students and is continually refined with the actions from students under supervision. Our model can be considered as the result of the models/patterns discovery phase of the knowledge discovery in databases process adapted to EDM as formulated by Romero and Ventura [29].

The idea of building this model was inspired by our experiences evaluating the 3D virtual lab for biotechnology [35]. During model design, we observed the behavior of students in the virtual world following the ethnographic method. Subsequently, as recommended by Mostow et al. [36], the student logs were analyzed by hand to identify interesting phenomena. One of the conclusions drawn from this analysis was that students tend to fall into different groups depending on their performance in the practical assignment. In addition, it was clear that different groups required different levels of help to complete the practical assignment. This idea led us to apply clustering techniques to student logs.

Besides, sequence models can be viewed in data mining as graph-based models [37], and they could be described as directed graphs for discovering frequent events [38].

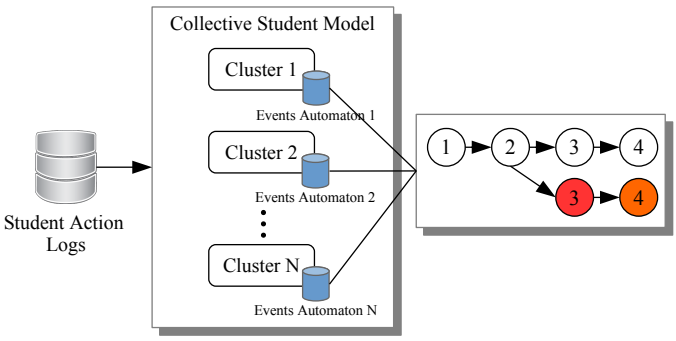

Fig. 2. Collective Student Model

Hence, the proposed model (see Figure 2) consists of several clusters of students, each of which contains an extended automaton (detailed in Section 4.1), which is a directed graph fit for our purposes. As discussed later, these clusters will help to provide automatic tutoring adapted to each student type.

Model creation shares the same main phases as the process proposed by Bogarín et al. [21]. It is executed when the tutor is launched. To create the model, it is necessary to access student log events stored in the student model ontology [32], [33]. Firstly, student logs are clustered based on any of the clustering methods detailed in Section 5.1.2. Then, for each cluster, an automaton is built from the student logs of this cluster. Next, the model is updated with each new student action at training time. In this way, the model can adapt to the students under supervision better and therefore deal with differences of behavior between current students and past students.

4.1 Extended Automaton Definition

States are represented by circles and transitions by arrows as shown in Figure 3. Furthermore, states are grouped into three zones: Correct Flow Zone, Irrelevant Errors Zone and Relevant Errors Zone.

Transitions denote events generated by students throughout an exercise, such as actions or attempted actions that past students have performed so far and new students may repeat in the future. Within the automaton, an event represents one of the following situations:

- a valid action for an exercise (do X event in Figure 3);
- an attempted action blocked by the intelligent tutor (try X event in Figure 4), because this action is wrong and the tutoring strategy has been configured to prevent students from performing this action; or
- an error detected by the intelligent tutor at the time of validating an incorrect action that has not been blocked (fail X event in Figure 3). A wrong action may involve more than one error, each of which will be considered as an event.

Student logs can also contain irrelevant actions. This type of student actions does not have any influence on the development of the practical assignment. Depending on the pedagogical value of these actions, they will be considered for creating the collective student model (and treated as right actions) or will be discarded.

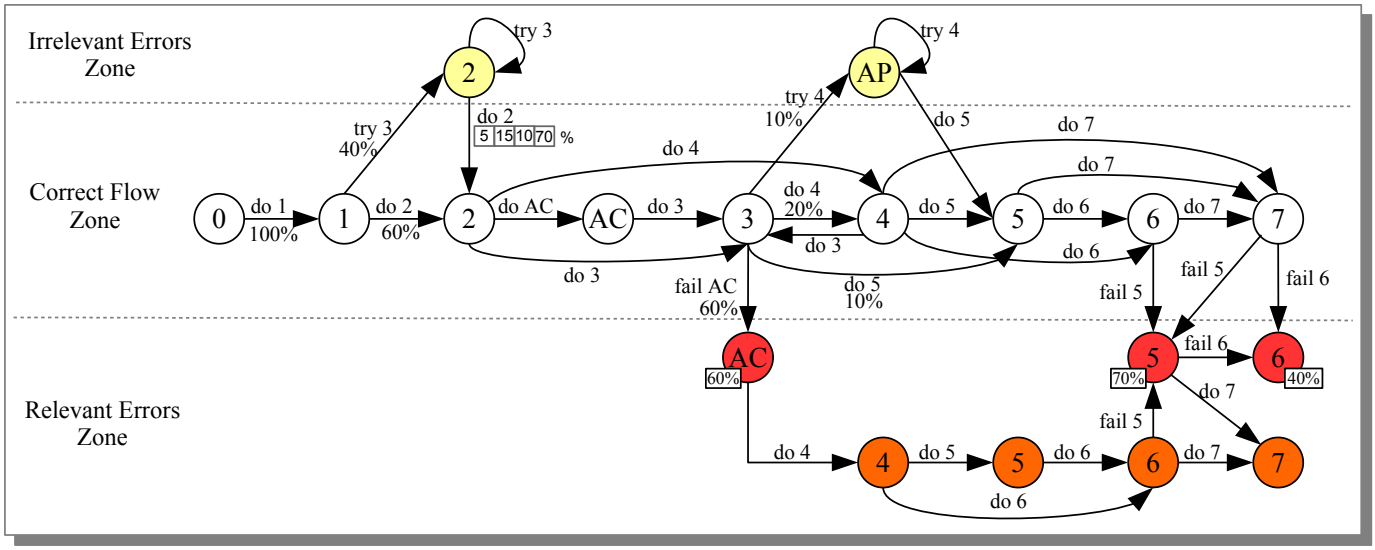

Fig. 3. Example automaton

Accordingly, states represent the different situations derived from the events generated by students.

Each state s contains the number of students whose logged sequences of events have passed through that state, which is denoted by $\gamma(s)$. The support of a state s is defined as the rate of $\gamma(s)$ with respect to the total number of students in the same cluster. Likewise, each transition t also contains its student frequency, denoted by $\phi(t)$, which is the number of logged sequences of events have passed through that transition. From this frequency, the confidence of the transition t is defined as the rate of $\phi(t)$ with respect to $\gamma(s1)$ where $s1$ is the source state in transition t. Figure 3 denotes support and confidence as percentages in the states and the transitions, respectively.

4.1.1 *Correct Flow Zone*

This area includes the states that constitute the valid sequences of actions for an exercise, which logically end up with a satisfactory final state. These states are depicted by white circles in Figure 3.

4.1.2 *Irrelevant Errors Zone*

This area groups states derived from error events that do not influence the final result. These error events are associated with attempted actions blocked by the tutor. States in this zone are depicted by yellow circles in Figure 3.

As a student may attempt the same wrong action more than once consecutively, vector transitions are employed for outgoing transitions from states of this zone to represent event frequencies. Therefore, the first position of the vector contains the number of students that exit the state without ever repeating the wrong action; the second position contains the number of students that exit the state after repeating the wrong action once and so forth.

For example, Figure 3 illustrates the vector transition from yellow state 2 to white state 2 for students that reached state 2 by attempted action 3, where 5% exited that state without repeating the attempted action "try 3" and 15% exited that state after repeating the attempted action "try 3" once.

4.1.3 *Relevant Errors Zone*

This area comprises states derived from error events that actually influence the final result, i.e., if an event of this type occurs the final result will be wrong, unless a repair action is taken. If the error is not repaired, it will be propagated to the subsequent states, which will also be considered erroneous, no matter what the student does afterwards (unless it is a repair action). The states derived directly from relevant errors are depicted by red circles, and the subsequent states that are the result of applying a right action to a state in this zone are denoted by orange circles.

4.2 Example Automaton

As an illustrative example of this automaton, we take an excerpt from the biotechnology virtual laboratory protocol [35], which is shown in Figure 4.

Figure 4 shows several actions, all of which, except for two (actions 3 and 4), must be executed in a specific order. These two actions can be performed in any order, but both must always be executed before action 5 and after action 2.

Figure 3 shows an example of the extended automaton described above. This automaton has been created from a subset of the logs of students who have performed the process shown in Figure 4 within the virtual laboratory for teaching biotechnology. Hence, it reflects all the actions and errors that students have performed in this part of the practical assignment (we have not included the frequencies of all events for the sake of clarity).

Figure 3 shows how some events associated with right actions recorded in student logs produce correct states. For example, event "do 1" (drop the glass on the shaker) leads to state 1, and event "do 3" leads to state 3.

In contrast, the event that leads to yellow state 2 does not represent a right action, but the error event of trying to do action 3 instead of action 2. Likewise, yellow state AP (already performed) is the result of another irrelevant error, i.e., trying to do action 4 when it has already been performed previously, because some students performed $do\ 4 \rightarrow do\ 3$ from white state 2.

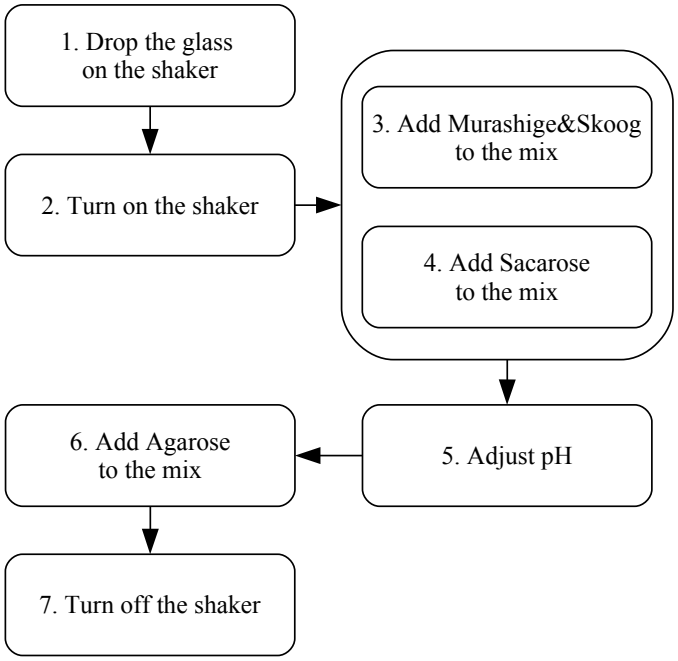

Fig. 4. Example of biotechnology lab process

The "do AC" event, which is not illustrated in Figure 4, represents the action of Adding Casein (a protein) to the mix. This action is considered incompatible with the other actions, but, for pedagogical reasons, the laboratory makes more chemical products available to students than required for the mix. Hence, as soon as the next chemical product is added to the mix (action 3 is performed), this event is validated as incorrect producing a "fail AC" event that leads to a red AC state. As Figure 3 shows, some students completed the remainder of the practical assignment correctly after performing "do AC" (path of orange states to orange state 7)

As regards relevant errors, red state 5 is caused by the fail event of not doing the right action 5 next, because some students performed action 6 or action 7 too early. For example, the automaton represents the fact that some students mistakenly performed action 6 instead of action 5 by the path: $do\ 6 \rightarrow fail\ 5$. As explained above, a wrong action can cause more than one error event at the same time in a student log. For example, if some students mistakenly perform action 7 instead of action 5, it will be represented by the path: $do\ 7 \rightarrow fail\ 5 \rightarrow fail\ 6$, because students have skipped two consecutive right actions.

The frequencies in Figure 3 show that many students forgot to turn on the shaker before adding substances to the mix (try 3 event labelled with 40%), or that most students forgot to adjust the mix pH (red state 5 labelled with 70%). In view of the error rates in these two cases, it might be a good idea to set hints at state 1 and/or the two states 4 if the instructor wants to stop most students from making these mistakes.

As explained previously, a different automaton is built for each cluster. Therefore, the frequencies of this example may not be the same as for the automaton of another cluster. Namely, red state 5 may not have such a high frequency or may not even exist. In this way, the tutoring feedback for students of different clusters may differ.

4.3 Formal Definition

Based on the definition of deterministic finite state automata, we define the extended automaton as follows:

$$EA = (S, S_C, S_{IE}, S_{RE}, S_{CE}, Z_C, Z_{IE}, Z_{RE}, s_0, T_N, T_V, \delta, \gamma, \phi_N, \phi_V) \tag{1}$$

$$\begin{aligned} Z_C &= S_C \\ Z_{IE} &= S_{IE} \\ Z_{RE} &= S_{RE} \cup S_{CE} \\ S &= S_C \cup S_{IE} \cup S_{RE} \cup S_{CE} \\ S_C \cap S_{IE} \cap S_{RE} \cap S_{CE} &= \emptyset \end{aligned}$$

In Equation 1 S represents the set of all states, and Z_C, Z_{IE} and Z_{RE} respectively include the states of the three zones defined above. s_0 is the initial state. T_N and T_V are sets of transitions between the states such that T_N includes normal transitions and T_V comprises vector transitions. δ is the transition function from one state to another, γ is a function that associates a state with its frequencies, and ϕ_N and ϕ_V are functions that associate a normal transition or a vector transition respectively with their frequencies.

Z_C defines the correct flow zone, which contains correct states (S_C - white states in Figure 3). Z_{IE} defines the irrelevant errors zone, which contains states of the same type (S_{IE} - yellow states in Figure 3). Z_{RE} is the relevant errors zone, which, in turn, contains two subsets of states S_{RE} (red states in Figure 3) and S_{CE} (orange states in Figure 3). S_{RE} is the set of states that are derived directly from relevant errors, whereas S_{CE} is the set of states that are the result of applying a right action to a state in this zone Z_{RE}; repair actions that do not correct all student mistakes belong to the S_{CE} group.

The function δ is defined in Equation 2, where s_i is the initial state and s_f is the final state of the given transition.

$$\begin{aligned} \delta &: T \rightarrow S \\ \delta(t) &= (s_i, s_f) \end{aligned} \tag{2}$$

Function γ is defined for each state in S by providing the frequency of the state, thereby extending the classic definition.

$$\begin{aligned} \gamma &: S \rightarrow \mathbb{N} \\ \gamma(s) &= f \end{aligned} \tag{3}$$

Like the γ function, we define two functions ϕ_N and ϕ_V, which provide the frequencies:

$$\begin{aligned} \phi_N &: T_N \rightarrow \mathbb{N} \\ \phi_N(t) &= f \end{aligned} \tag{4}$$

$$\begin{aligned} \phi_V &: T_V \rightarrow \mathbb{N}^n \\ \phi_V(t) &= f^n. \end{aligned} \tag{5}$$

4.4 Build Automaton Process

The build automaton process is detailed in Algorithm 1. The inputs of this algorithm are the student logs and a cluster, to which students belong, which it uses to create the automaton for this cluster.

This algorithm traverses each student log in chronological order and adds each student event in the log to the automaton whose input is the current state s_i.

The process of adding an event firstly checks whether there is already a state in the automaton that is the result of this event. If no such state exists, a new state resulting from this event is created, and the current state becomes the new state; otherwise, the frequency of the existing state is increased, and then the process looks for a transition from the current state to the existing state that matches the input event; if such a transition exists, then its frequency is incremented, else a new transition from the current state to the existing state is created. When creating a new transition or increasing its frequency, it is necessary to make a distinction between whether or not it is a vector transition.

To decide whether a state is a result of an event, or an existing transition matches an event, states and transitions should store some information on their source event. For the sake of clarity, however, we will skip these implementation details in the algorithm specification.

Algorithm 1 Build Automaton

Inputs
Input $student\ logs$
Input $cluster$
Outputs
Output EA (see Equation 1)

for each $student \in cluster$ **do**
  $s_i = s_0$
  Read *log* of *student* from *student logs*
  for each $event \in log$ **do**
    ADDEVENTLOG($event, s_i$)
  end for
end for

procedure ADDEVENTLOG($event, s_i$)
  if $\exists s_f \in S | s_f$ is result of $event$ **then**
    $\gamma(s_f) = \gamma(s_f) + 1$ (see Equation 3)
    if $\exists t \in model | \delta(t) = (s_i, s_f) \wedge$ t matches $event$ **then**
      if $t \in T_V$ **then**
        $count =$ number of consecutive events equal to *event* in *log*
        $\phi_V(t)_{[count]} = \phi_V(t)_{[count]} + 1$ (see Equation 5)
        Remove consecutive events equal to *event* from *log*
      else
        $\phi_N(t) = \phi_N(t) + 1$ (see Equation 4)
      end if
    else
      ADDTRANSITION(s_i, s_f)
    end if
    $s_i = s_f$
  else
    Add new state s to S (S is element of EA).
    ADDTRANSITION(s_i, s)
    $s_i = s$
  end if
end procedure

procedure ADDTRANSITION(s_i, s_f)
  if $s_i \in S_{IR}$ **then** (see Equation 2)
    Add a new t_V to T_V from s_i to s_f using $\delta(t)$ (T_V is element of EA)
  else
    Add a new t_N to T_N from s_i to s_f using $\delta(t)$ (T_N is element of EA)
  end if
end procedure

4.5 Application of the Collective Student Model in the Tutoring Process

As mentioned above, the proposed ITS relies on the collective student model to stop students from making mistakes or from floundering with the practical assignment. For this purpose, the ITS can use the collective student model at training time to detect a situation in which there is a significant probability of the student making a mistake or getting stuck. If it is pedagogically inconvenient for the student to be allowed to make the mistake, the ITS will have to be configured to warn the student by means of a hint.

The simplest case in which the model can indicate that the student is highly likely to make a mistake is when the confidence in a transition from the student's current state to an error state is high. For example, "try 3" from state 1 in Figure 3. The instructor responsible for the practical assignment will specify when a confidence or support is high.

However, the above is just one particular case of a more general situation where an error state is highly likely to be reached after several consecutive transitions from the current student state. This situation is referred to as an indirect transition. It is important to identify such situations when the support for a relevant error state is high. For example, if the current state of the student in Figure 3 is white 2, the error state AC could be reached through the sequence of transitions $do\ AC \rightarrow do\ 3 \rightarrow fail\ AC$. Besides, this is a highly likely sequence, because 60% of students who reach white state 2 (all of the students eventually reach white state 2) get the red state AC.

When working out the probability of reaching an error state through an indirect transition, we may take into account more than one alternative path in the automaton. For example, let us consider red state 5 as the error state in Figure 3, then it is clear that there are three paths to red state 5 from white state 3: $fail\ AC \rightarrow do\ 4 \rightarrow do\ 6 \rightarrow fail\ 5$, $do\ 4 \rightarrow do\ 6 \rightarrow fail\ 5$ and $do\ 4 \rightarrow do\ 7 \rightarrow fail\ 5$. Besides, as can be deduced from the automaton, 70% of students at white state 3 will eventually reach red state 5, because all the students reach white state 3 at some point (nobody forgot to

do action 3).

In order to speed up the real-time performance of the ITS, indirect transitions may be computed offline and included in the automaton as direct transitions.

A student is said to be floundering when he or she repeats the same wrong action many times in the practical assignment. To prevent students from getting stuck, and probably getting frustrated, we need to find out whether there is a high probability of the student proceeding from the current state to an irrelevant error state where he or she may repeat a wrong action several times. To do this, the irrelevant error state must have an outgoing vector transition with the right content, as applies in yellow state 2 with vector transition "do 2" in which 70% of students will exit that state after repeating the wrong action "try 3" three times.

4.5.1 Update model process

The collective student model is updated with each new event executed by each student at training time. The update process searches the state that is the result of the last event executed by a student. From this state, it creates a new or increments the frequency of an existing transition in a similar fashion to the build automaton process.

The process for updating the model is as follows:

1. Search for the current state s_i that represents the last event executed. If it is the beginning of the practical assignment, the current state will be state 0.
2. Execute the *AddEventLog* procedure from Algorithm 1, where $event$ is the next event to be processed and s_i is the current state.

4.5.2 Student reclassification

When a student has just started a practical assignment, he or she is assigned to a default cluster. The default cluster is the one that has the average values of the attributes used to generate the clusters.

As the student progresses in the practical assignment, however, the initial classification may no longer hold. It is the instructor that initially configures the checkpoints determining when to check whether a student no longer belongs to a cluster. These checkpoints may either be particular steps in the practical assignment protocol or be defined by percentages of the average total time taken by the students in the current cluster. Therefore, when the student reaches any of these checkpoints, he or she can be reclassified according to his or her log so far. To do this, we apply the following process:

1. Get the current cluster to which the student belongs.
2. Select the cluster that best matches the student log.
3. If the best match is the same cluster, do nothing.
4. Else
   - 4.1. Undo the application of the student log in the automaton of the former cluster by doing the opposite to what Algorithm 1 does (decreasing frequencies and/or removing states).
   - 4.2. Apply the student log to the automaton of the new cluster according to Algorithm 1.

In order to select the cluster that best matches the student log at a checkpoint, we can use the same method as for the initial clustering, albeit considering just the excerpts of the logs of students that have reached the same checkpoint as the log of the student under supervision. This selection consist of searching the cluster where its centroid is close to the value of the attributes calculated from the student log, technique similar to that used in incremental clustering [40].

5 Collective Student Model Validation

This validation has two main objectives. The first goal is to verify that the error in the predictions computed by the proposed model is acceptable for automatic tutoring. Roughly, the error associated with the prediction of a student action is equal to 1 minus the probability of the student performing that action according to the model. Hence, if the model predicts that there is a probability of 1 that the student will perform an action and the student eventually performs that action, the error will be 0. Otherwise, if the student does not perform that action, the error will be 1. The second goal is to check whether clustering methods can divide students into groups that require different tutoring feedback. To do this, we will compare the clusters by checking how many students in each cluster made a relevant error in the practical assignment.

In order to validate the proposed model, we will employ the student logs taken from a virtual laboratory for teaching biotechnology [35]. This laboratory is an educational 3D virtual environment for procedural training used by students to carry out a practical assignment composed of around 120 actions, such as add a chemical to a mix or turn on a machine. This virtual laboratory was implemented on the OpenSimulator platform.

Student logs were collected by a component inside this virtual laboratory, called automatic tutor. This component validates student actions and registers events like the ones outlined in Section 4.2.

5.1 Method

For the last three years (2013, 2014 and 2015), 85 students taking the Biochemistry and Biotechnology course taught as part of the BSc in Forestry Engineering at the UPM used the biotechnology virtual laboratory [35] to complete a practical assignment. The automatic tutor of this virtual lab collected and classified the events that these students generated during this practical assignment. These events may fall into any of the following categories:

- Correct Events
  - Corrective Events
  - Non-Corrective Events
- Error Events
  - Dependency Errors
  - Incompatibility Errors
  - World Errors
  - Other Errors.

Correct events are right actions. However, when a right action repairs an error made previously, the associated event

is considered to be a corrective event. Dependency or incompatibility errors depend on the configuration of the virtual laboratory (detailed in [35]) set up by the instructor. They are related to the right order in which to perform the actions in the practical assignment. World errors refer to failures in the handling of 3D objects, for example, if a student tries to drop an object where it should not be dropped. Finally, the other error events category represent errors that are not pedagogically relevant, for example, if the student tries to repeat an action that has already been performed.

5.1.1 Model mean error calculation

To calculate the model mean error, we proceeded as follows:

1. Select 90% of students at random.
2. Apply the process defined in Section 4.4 to build a model using the logs from the selected 90% of students.
3. Validate how well the model aligned with the remaining 10% of student logs.
4. Repeat step 1 to 3 n times to guarantee the representativeness of the randomly selected students.

The 10% of students used in step 3 constitute the *test set* for the model. This test set is used to estimate how accurately the model will perform in the practical assignment, like many validation algorithms in data mining or statistics.

The number n of times that steps 1 to 3 are repeated was 150 after checking that with $n = 75, 100$ and 150, results were equivalent.

The process for validating the alignment (step 3) of the test set with the model is defined in Algorithm 2. The inputs of this algorithm are the model, the confidence and the support that determine the submodel to be validated, and the test set of students. For example, if $confidence = support = 0$, then the entire model will be validated.

Temporary states in Algorithm 2 are states that are not in the model but have to be created temporarily for validation purposes. For instance, if a student in the validation test set generates an event, and the model does not contain any state that is the result of that event, the algorithm creates a new state which it links to the previous state (this new state is not added to the model). This operation is performed until the next event in the log matches a state in the model.

The alignment error of an event log with the model is calculated using a similar equation to Replay Fitness [39], which is employed to quantify the extent to which the model can reproduce the traces recorded in the log in process discovery. Equation 6 defines this calculation for transitions with normal frequencies ($\in T_N$), where t is the transition that matches the event performed by the student and s is the initial state of t (functions ϕ_N and γ are defined in equations 4 and 3 respectively).

$$E_{event} = 1 - \frac{\phi_N(t)}{\gamma(s)} \tag{6}$$

There is a slightly different equation for calculating this error in the case of a vector transition ($\in T_V$). If the student exits the loop the first time round, a similar equation to the above is used to calculate the error, where the transition frequency is replaced with the frequency stored in the first element of the vector (function ϕ_V is defined in equation 5).

Algorithm 2 Validation process

Inputs
Input $model$
Input $support$
Input $confidence$
Input $val\_example\_set$
Outputs
Output $\bar{E}$

for each $student\_log \in val\_example\_set$ **do**
  Get $cluster \in model | student\_log$ best fits $cluster$
  Get the $automaton \in cluster$
  Get $s_0 \in automaton$
  $previous\_state = s_0$
  for each $event \in student\_log$ **do**
    if $\exists state \in automaton | state$ is the result of $event$ **then**
      $exist(t) \equiv \exists t \in automaton | \delta(t) = (previous\_state, state)$
      if $exist(t)$ **then**
        if $state_{sup} \geq support \wedge t_{conf} \geq confidence$ **then**
          Calculate E_{event} according to equations 6, 7 or 8
        else
          $E_{event} = 0$
        end if
      else
        $E_{event} = 1$
      end if
    else
      $E_{event} = 1$.
      $state =$ new temporary state from $event$
    end if
    $previous\_state = state$
  end for
  Update $\bar{E}$ with E_{event} according to equation 9
end for

$$E_{event} = 1 - \frac{\phi_V(t)[1]}{\gamma(s)} \tag{7}$$

If the student does not exit the loop first time round, the transition frequency is replaced by the sum of all elements of the frequency vector, except from the first, i.e.,

$$E_{event} = 1 - \frac{\sum_{i=2}^{\phi_V(t)_{count}} \phi_V(t)[i]}{\gamma(s)}. \tag{8}$$

The reason for making a distinction between these two cases when calculating the vector transition error is that it makes more sense, when providing tutorial hints, to work with different predictions depending on whether a student makes a loop error once or more than once. For example, if a student is more likely to make a mistake only once rather than more than once, it will be less necessary to display a hint to avoid that mistake.

To calculate the mean event error, each event error E_i calculated by the above equations is multiplied by the number of students in the test set that generated the same

event n_i, and the sum of these products is divided by the sum of the number of students that generated each event.

$$\bar{E} = \frac{\sum_{i=1}^{n\ dif\ events} E_i n_i}{\sum_{i=1}^{n\ dif\ events} n_i} \quad (9)$$

5.1.2 Clustering methods

As selecting the clustering methods [40], [41], [42], [43], we only considered methods that do not need initially the k number of clusters. Thus, we selected the following methods:

1. XMeans
2. Expectation Maximization (EM)
3. Microsoft Sequence Clustering
4. No clustering.

The first two are partitioning methods, because they construct the groups based on the distances that exist between the objects. These methods use an iterative relocation technique, which moves each object into a group by analyzing how close is to other members of the group and how far is from objects in other groups. This distance measure is calculated by means of the Euclidean distance between two vectors.

The functions that calculate the vector associated with an object are as follows:

1. Clustering by errors
2. Clustering by errors and time
3. Clustering by events in each zone of the automaton.

The first function returns an error coefficient, which is calculated using the weighted sum of errors made by a student with a different weight depending on the pedagogical importance of the error. The second function returns an error coefficient-time pair, where the error coefficient is calculated as explained before, and time stands for the time that it took the student to complete the entire practical assignment. If the student did not complete the practical assignment, he or she is penalized with 24 hours. The clustering by events in zone function returns a triple with the number of events of each type existing in a student log.

The Microsoft Sequence Clustering algorithm [45] uses Markov chain analysis to identify ordered sequences. Then clusters are generated from these results using an Expectation Maximization algorithm applied to the ordered sequences. We ruled out the Weka Sequential Information Bottleneck algorithm (sIB) [46] because it requires the number of clusters as input.

For the first two methods (XMeans and EM), we used their implementation in the Weka framework [44].

5.2 Results

To get an idea of the data size, we built an automaton without clustering the logs of the 85 students (14943 log events). The result was an automaton with 532 states and 2778 transitions (1454 normal and 1324 vector). The number of states is detailed in Table 1 for each zone of this automaton.

TABLE 1
States by zone in a non-clustered model for validation

Zone	Number of States
Correct Flow	115
Irrelevant Errors	228
Relevant Errors	189

5.2.1 Model mean error

The support and confidence values 0, 0.1, 0.25, 0.5, 0.75, 0.9 were used as inputs for the validation process, since this set of values was considered representative and complete enough to illustrate the error evolution.

Table 2 shows the model mean error for the best method and function combination and all the confidence and support combinations. Model mean error was calculated using the process explained in Section 5.1.

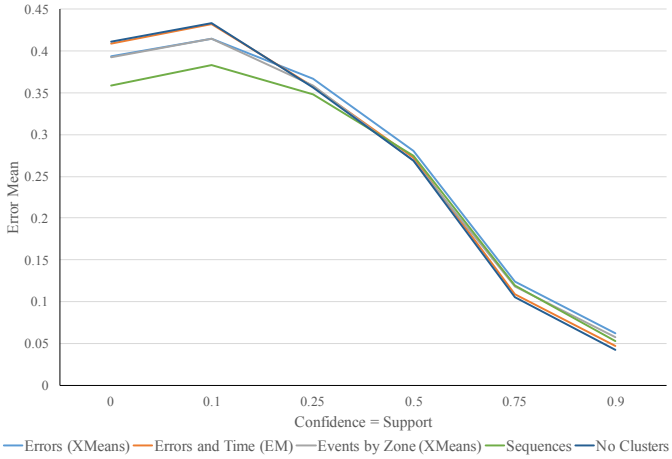

Fig. 5. Mean error when $confidence = support$

Figure 5 highlights that the differences between functions are very small (about 0.01) around 0.5. Also, at some point between 0.5 and 0.75, the no clustering line drops below all the others.

As explained in Section 4.5, we are interested for tutoring purposes in transitions that are highly likely to reach an error state. Conceptually, this is equivalent to working with a submodel with a high confidence (for example, $confidence = 0.5$) and $support = 0$. Figure 6 shows model mean error with a confidence of between 0 and 0.9 and $support = 0$.

As Figure 6 shows, all clustering functions, and especially the sequences function, have lower error values than the no clustering function. Figure 6 also shows that the error decreases rapidly when the confidence for a support of zero increases.

5.2.2 Errors by cluster

As explained above, the principal application of the collective student model is to generate anticipated feedback for students. Thus, let us analyze how well these methods group together students according to their performance in the practical assignment.

To do this, we considered the frequencies of the most common pedagogical errors in the clusters output by each

TABLE 2
Mean error by method and clustering function.

Function	Method	Num. of Clust.	Sup. Conf.	0	0.1	0.25	0.5	0.75	0.9
Errors	XMeans	2	0	0.394	0.358	0.346	0.308	0.238	0.217
			0.1	0.483	0.415	0.388	0.312	0.162	0.131
			0.25	0.463	0.389	0.366	0.285	0.126	0.069
			0.5	0.416	0.330	0.323	0.281	0.124	0.062
			0.75	0.361	0.213	0.208	0.198	0.124	0.062
			0.9	0.435	0.117	0.109	0.097	0.088	0.062
Errors and Time	EM	2	0	0.400	0.363	0.350	0.312	0.240	0.218
			0.1	0.493	0.423	0.394	0.312	0.159	0.145
			0.25	0.473	0.397	0.370	0.285	0.123	0.080
			0.5	0.422	0.331	0.323	0.277	0.120	0.070
			0.75	0.372	0.212	0.207	0.194	0.120	0.070
			0.9	0.452	0.117	0.106	0.092	0.083	0.070
Events by Zone	XMeans	2	0	0.393	0.354	0.340	0.303	0.236	0.210
			0.1	0.485	0.414	0.383	0.302	0.154	0.140
			0.25	0.465	0.389	0.358	0.274	0.119	0.059
			0.5	0.414	0.321	0.309	0.269	0.119	0.057
			0.75	0.367	0.207	0.202	0.190	0.119	0.057
			0.9	0.444	0.112	0.104	0.092	0.085	0.057
Sequences	Microsoft Sequence Clustering	6	0	0.358	0.323	0.307	0.264	0.189	0.153
			0.1	0.445	0.383	0.361	0.290	0.146	0.123
			0.25	0.427	0.365	0.349	0.280	0.136	0.105
			0.5	0.377	0.306	0.300	0.274	0.120	0.053
			0.75	0.324	0.201	0.197	0.192	0.120	0.053
			0.9	0.341	0.100	0.093	0.084	0.066	0.053
No Clustering			0	0.412	0.369	0.356	0.315	0.247	0.202
			0.1	0.522	0.433	0.400	0.312	0.146	0.107
			0.25	0.504	0.400	0.356	0.277	0.105	0.042
			0.5	0.444	0.312	0.298	0.269	0.105	0.042
			0.75	0.427	0.197	0.186	0.182	0.105	0.042
			0.9	0.574	0.105	0.085	0.077	0.070	0.042

Label: Min values with $sup. = 0$; Min values with $sup. = 0.1$; Min values with $sup. = 0.25$; Min values with $sup. = 0.5$; Min values with $sup. = 0.75$; Min values with $sup. = 0.9$

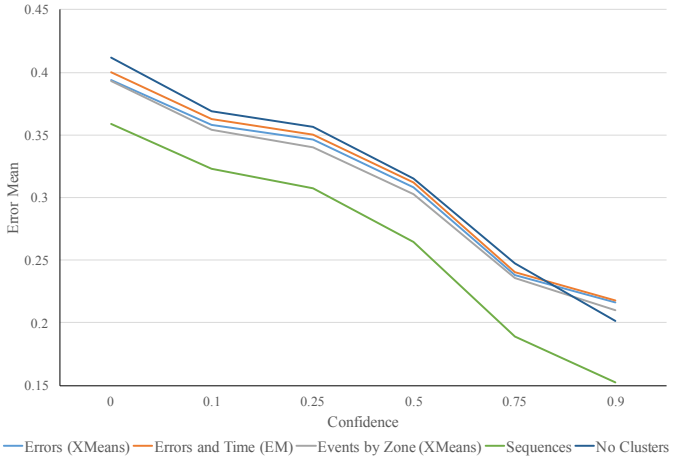

Fig. 6. Mean error when $support = 0$

clustering method. For this purpose, we took some error states located in the Relevant Errors Zone. Then, for each clustering method, we compared the frequencies of these errors in each cluster.

From this analysis, we selected results output by the events by zone clustering method (Table 3) and sequence clustering method (Table 4). The first column in Tables 3 and 4 represents the error code; the last column includes the variances of the error frequencies; and the intermediate columns contain the frequencies of the errors in each cluster.

Sequence clustering results were selected because this method was the best predictor of student behavior according to Table 2. The events by zone clustering method was selected because it proved to be the best at grouping students according to their performance in the practical

TABLE 3
Frequencies by cluster for events by zone clustering function (XMeans)

Error	Cluster 1	Cluster 2	Variance
1.20_1.16	0.545	1	0.103
1.20_1.14	0.364	0.716	0.062
2.52_2.40	0.182	0.649	0.109
2.52_2.38	0.091	0.581	0.120
2.52_2.37	0.091	0.581	0.120
2.52_2.39	0.091	0.568	0.114
2.52_2.35	0.091	0.554	0.107
3.52_3.34	0.182	0.527	0.059
2.52_2.41	0.091	0.527	0.095
2.52_2.34	0.091	0.459	0.068
Average	0.182	0.616	0.096

Label: Frequencies under 50%; Frequencies between 50% and 60%; Frequencies between 60% and 70%; Frequencies over 70%

TABLE 4
Frequencies by cluster for sequences clustering function

Error	Cl. 1	Cl. 2	Cl. 3	Cl. 4	Cl. 5	Cl. 6	Var.
1.20_1.16	0.688	0.833	0.857	0.6	0.5	0.778	0.020
1.20_1.14	0.750	0.667	0.5	0.7	0.750	0.556	0.011
2.52_2.40	0.563	0.750	0.571	0.3	1	0.444	0.060
2.52_2.38	0.469	0.583	0.571	0.5	0.625	0.444	0.005
2.52_2.37	0.531	0.583	0.429	0.5	0.750	0.333	0.020
2.52_2.39	0.5	0.583	0.5	0.3	0.750	0.444	0.022
2.52_2.35	0.469	0.583	0.5	0.4	0.625	0.444	0.007
3.52_3.34	0.531	0.417	0.429	0.6	0.375	0.444	0.007
2.52_2.41	0.375	0.583	0.5	0.4	0.750	0.444	0.020
2.52_2.34	0.344	0.583	0.5	0.3	0.625	0.222	0.027
Average	0.522	0.617	0.536	0.460	0.675	0.456	0.020

Label: Frequencies under 50%; Frequencies between 50% and 60%; Frequencies between 60% and 70%; Frequencies over 70%

assignment.

The above two tables show that the variances in Table 3 are greater than in Table 4. This difference is clearer in the last row of each table (average frequencies).

5.3 Discussion

As expected, Table 2 and Figure 5 show that errors tend to decrease when confidence and support increase. This is because the most frequent transitions and the most visited states reflect the most common behavior of the students in the practical assignment. Therefore, the predictions based on such transitions and states are more reliable.

We think that the min error (0.358 for the sequences method) yielded by $support = 0$ and $confidence = 0$ is reasonably good for a training environment like this, where students have a lot of freedom of action. Besides, we should take into account that the ITS will typically consider transitions that are likely to reach error states (with a confidence greater than 0.5, for example) for tutoring tasks. Therefore, prediction errors will be lower (less than 0.27 in this experiment) than when $confidence = support = 0$. In addition, as explained above, the model is updated with each new student action at training time. This should help to reduce prediction error because the model will tend to fit the students under supervision better.

Another conclusion that we can draw from the results in Table 2 is that the differences between clustering methods are small. For example, with $support = 0$ and $confidence = 0$ (i.e., the entire model), the min error is 0.358 (*Sequence clustering*) and the max error (*No clustering*) is 0.412. Although these differences are small, the best clustering method is clearly sequence clustering, mostly with $support = 0$, as shown in Figure 6. This supports the approach of using clustering, because, as mentioned above, tutoring tasks will consider transitions that are highly like to reach error states with any support.

Concerning results on errors by cluster, Table 3 shows that clustering methods, and particularly events by zone clustering, can successfully classify students into groups that will need different tutoring feedback throughout the practical assignment.

Let us take the example of the error in the second row (1.20_1.14). For this error, cluster 1 in Table 3 has a low frequency (36.4%), whereas cluster 2 has a very high frequency (71.6%). These figures imply that students in cluster 1 should not need any advice on this error because very few former students made that error. In contrast, most of the students in cluster 2 will need a hint to prevent them from making this error, as explained in Section 4.5.

Taking into account the results of the two studies (model mean error and errors by cluster), the best clustering method for this data set is events by zone, because it is the best at grouping by performance and the second best at predicting student errors with $support = 0$.

To sum up, as explained above, the best clustering method (events by zone) can manage prediction errors around 0.3 (with $support = 0$ and $confidence \approx 0.5$) at the beginning of a learning session and even better later. This is because as the learning session progresses, the model will tend to fit the students under supervision better. Hence, we can claim that the first goal posed at the beginning of section 5 has been reached, i.e., it was verified that the prediction error is acceptable for tutoring purposes. In addition, we can also claim that the second goal has also been reached, because events by zone clustering classifies students in groups that will require a different tutoring feedback.

6 Conclusions

This paper presents a model that can predict student actions in procedural training environments. Additionally, this paper explains how this model is integrated into an ITS architecture and how it can be used to improve the tutoring feedback by anticipating student errors as long as this is pedagogically convenient.

The collective student model is created from student logs by clustering logs and computing an extended automaton for each resulting cluster. We should highlight that there are few ITSs in the literature that rely on data mining techniques to enhance their tutoring feedback.

The proposed model has been validated using the student logs collected in a 3D virtual laboratory for teaching biotechnology. As a result of this validation, we concluded that the model can provide reasonably good predictions and support tutoring feedback that is more adapted to each student type.

An application that displays the collective student model would be very useful for facilitating the definition of the tutoring strategy. In this way, the instructor could visualize when students make more mistakes or which part of the practical assignment students find easier. Based on this information, the instructor could decide where and what tutoring feedback the ITS should provide. Additionally, this could also help the instructor to improve his or her own teaching.

Another line of future work will be to validate an ITS built upon the proposed model in order to evaluate the tutoring feedback induced by the proposed model.

Acknowledgments

Riofrío would like to acknowledge financial support from the Ecuadorian Secretariat of Higher Education, Science, Technology and Innovation (SENESCYT). Thanks to Rachel Elliot for her comments on the paper.

REFERENCES

[1] C. Romero and S. Ventura, "Educational data mining: A survey from 1995 to 2005," *Expert Systems with Applications*, vol. 33, no. 1, pp. 135–146, 2007.

[2] C. Romero and S. Ventura, "Educational Data Mining: A Review of the State of the Art," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 6, pp. 601–618, 2010.

[3] R. S. Baker, "Educational Data Mining: An Advance for Intelligent Systems in Education," *Intelligent Systems, IEEE*, vol. 29, no. 3, pp. 78–82, 2014.

[4] L. S. Vygotsky, *Mind in society: The development of higher psychological processes*. Harvard university press, 1978.

[5] A. M. Olney, "Scaffolding Made Visible," in *Design Recommendations for Intelligent Tutoring Systems*. Orlando: U.S. Army Research Laboratory, 2014, ch. 26, pp. 327–340.

[6] H. K. Holden and A. M. Sinatra, "A Guide to Scaffolding and Guided Instructional Strategies for ITSs," in *Design Recommendations for Intelligent Tutoring Systems*. Orlando: U.S. Army Research Laboratory, 2014, ch. 22, pp. 265–281.

[7] C. Romero, S. Ventura, M. Pechenizkiy, and R. S. Baker, *Handbook of Educational Data Mining*. CRC Press, 2010.

[8] D. Perera, J. Kay, I. Koprinska, K. Yacef, and O. R. Zaïane, "Clustering and Sequential Pattern Mining of Online Collaborative Learning Data," pp. 759–772, 2009.

[9] T. Y. Tang and G. McCalla, "Smart Recommendation for an Evolving E-Learning System: Architecture and Experiment," *International Journal on ELearning*, vol. 4, no. 1, pp. 105–129, 2005.

[10] D. Godoy and A. Amandi, "Link Recommendation in E-Learning Systems Based on Content-Based Student Profiles," in *Handbook of Educational Data Mining*. CRC Press, 2010, ch. 19, pp. 273–286.

[11] P. Fournier-Viger, R. Nkambou, E. M. Nguifo, A. Mayers, and U. Faghihi, "A Multiparadigm Intelligent Tutoring System for Robotic Arm Training," *Learning Technologies, IEEE Transactions on*, vol. 6, no. 4, pp. 364–377, 2013.

[12] G. W. Dekker, M. Pechenizkiy, and J. M. Vleeshouwers, "Predicting Students Drop Out: A Case Study." in *International Working Group on Educational Data Mining*. ERIC, 2009.

[13] C. Romero, P. G. Espejo, A. Zafra, J. R. Romero, and S. Ventura, "Web usage mining for predicting final marks of students that use Moodle courses," *Computer Applications in Engineering Education*, vol. 21, no. 1, pp. 135–146, mar 2013.

[14] J. A. Lara, D. Lizcano, M. A. Martínez, J. Pazos, and T. Riera, "A system for knowledge discovery in e-learning environments within the European Higher Education Area Application to student data from Open University of Madrid, UDIMA," *Computers & Education*, vol. 72, no. 0, pp. 23–36, mar 2014.

[15] C. Antunes, "Acquiring Background Knowledge for Intelligent Tutoring Systems." in *EDM*, 2008, pp. 18–27.

[16] A. Hershkovitz and R. Nachmias, "Learning about online learning processes and students' motivation through Web usage mining," *Interdisciplinary Journal of Knowledge and Learning Objects*, pp. 197–214, 2009.

[17] I. Arroyo, D. G. Cooper, W. Burleson, B. P. Woolf, K. Muldner, and R. Christopherson, "Emotion Sensors Go To School," in *Proceedings of the 2009 Conference on Artificial Intelligence in Education: Building Learning Systems That Care: From Knowledge Representation to Affective Modelling*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2009, pp. 17–24.

[18] T. Barnes and J. Stamper, "Automatic Hint Generation for Logic Proof Tutoring Using Historical Data," *Journal of Educational Technology & Society*, vol. 13, no. 1, pp. 3–12, 2010.

[19] M. Mavrikis, "Modelling student interactions in intelligent learning environments: constructing bayesian networks from data," *International Journal on Artificial Intelligence Tools*, vol. 19, no. 06, pp. 733–753, dec 2010.

[20] K. Porayska-Pomsta, M. Mavrikis, S. D'Mello, C. Conati, and R. S. J. d. Baker, "Knowledge Elicitation Methods for Affect Modelling in Education," *Int. J. Artif. Intell. Ed.*, vol. 22, no. 3, pp. 107–140, 2013.

[21] A. Bogarín, C. Romero, R. Cerezo, and M. Sánchez-Santillán, "Clustering for Improving Educational Process Mining," in *Proceedings of the Fourth International Conference on Learning Analytics And Knowledge*. New York, NY, USA: ACM, 2014, pp. 11–15.

[22] A. Peña-Ayala, "Educational data mining: A survey and a data mining-based analysis of recent works," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1432–1462, mar 2014.

[23] K. Sukhija, M. Jindal, and N. Aggarwal, "The recent state of educational data mining: A survey and future visions," pp. 354–359, 2015.

[24] L. Razzaq, M. Feng, G. Nuzzo-Jones, N. T. Heffernan, K. R. Koedinger, B. Junker, S. Ritter, A. Knight, C. Aniszczyk, and S. Choksey, "The Assistment project: Blending assessment and assisting," *Proceedings of the 12th Annual Conference on Artificial Intelligence in Education*, pp. 555–562, 2005.

[25] S. Ritter, J. Anderson, K. Koedinger, and A. Corbett, "Cognitive Tutor: Applied research in mathematics education," *Psychonomic Bulletin & Review*, vol. 14, no. 2, pp. 249–255, 2007.

[26] S. Ritter, R. Carlson, M. Sandbothe, and S. E. Fancsali, "Carnegie Learning's Adaptive Learning Products," in *Educational Data Mining 2015: 8th International Conference on Educational Data Mining*, Madrid, 2015.

[27] F. Vicente, S. Adjei, T. Colombo, and N. Heffernan, "Building Models to Predict Hint-or-Attempt Actions of Students," in *Educational Data Mining 2015: 8th International Conference on Educational Data Mining*, Madrid, 2015.

[28] R. S. Baker, S. M. Gowda, M. Wixon, J. Kalka, A. Z. Wagner, A. Salvi, V. Aleven, G. W. Kusbit, J. Ocumpaugh, and L. Rossi, "Towards Sensor-Free Affect Detection in Cognitive Tutor Algebra." *International Educational Data Mining Society*, 2012.

[29] C. Romero and S. Ventura, "Data mining in education," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 3, no. 1, pp. 12–27, 2013.

[30] A. de Antonio, J. Ramírez, R. Imbert, and G. Méndez, "Intelligent virtual environments for training: An agent-based approach," *Multi-Agent Systems and Applications IV*, pp. 82–91, 2005.

[31] R. Imbert, L. Sánchez, A. de Antonio, G. Méndez, and J. Ramírez, "A multiagent extension for virtual reality based intelligent tutoring systems," in *International Conference on Advanced Learning Technologies*. IEEE, 2007, pp. 82–84.

[32] J. Clemente, J. Ramírez, and A. de Antonio, "A proposal for student modeling based on ontologies and diagnosis rules," *Expert Systems with Applications*, vol. 38, no. 7, pp. 8066–8078, jul 2011.

[33] J. Clemente, "Una Propuesta de Modelado del Estudiante Basada en Ontologías y Diagnóstico Pedagógico-Cognitivo no Monótono," Ph.D. dissertation, Universidad Politécnica de Madrid, 2011.

[34] J. Clemente, J. Ramírez, and A. de Antonio, "Applying a student modeling with non-monotonic diagnosis to Intelligent Virtual Environment for Training/Instruction," *Expert Systems with Applications*, vol. 41, no. 2, pp. 508–520, feb 2014.

[35] M. Rico, J. Ramírez, D. Riofrío, M. Berrocal-Lobo, and A. De Antonio, "An architecture for virtual labs in engineering education," in *Global Engineering Education Conference (EDUCON), 2012 IEEE*, 2012, pp. 1–5.

[36] J. Mostow and J. Beck, "Some Useful Tactics to Modify, Map and Mine Data from Intelligent Tutors," *Nat. Lang. Eng.*, vol. 12, no. 2, pp. 195–208, 2006.

[37] H. Mannila and C. Meek, "Global Partial Orders from Sequential Data," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, New York, NY, USA: ACM, 2000, pp. 161–168.

[38] H. Mannila, H. Toivonen, and A. Inkeri Verkamo, "Discovery of Frequent Episodes in Event Sequences," *Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 259–289, 1997.

[39] J. C. Buijs, B. F. van Dongen, W. M. van der Aalst, and P, "Quality Dimensions in Process Discovery: The Importance of Fitness, Precision, Generalization and Simplicity," *International Journal of Cooperative Information Systems*, vol. 23, no. 01, p. 1440001, 2014.

[40] L. Rokach, and O. Maimon, "Clustering Methods," in *Data Mining and Knowledge Discovery Handbook*, Boston, MA: Springer US, pp. 321352, 2005

[41] J. Han, M. Kamber, and J. Pei, "Cluster Analysis: Basic Concepts and Methods," in *Data mining: concepts and techniques*, Elsevier Inc, 3rd ed., pp. 443494, 2012.

[42] J. Han, M. Kamber, and J. Pei, "Advanced Cluster Analysis," in *Data mining: concepts and techniques*, Elsevier Inc, 3rd ed., pp. 497542, 2012.

[43] P.-N. Tan, M. Steinbach, and V. Kumar, "Data mining cluster analysis: Basic concepts and algorithms," in *Introduction to data mining*, London: Pearson Education Limited, 1st ed., 2013.

[44] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009.

[45] Microsoft, "Microsoft Sequence Clustering Algorithm Technical Reference," 2016. [Online]. Available: https://msdn.microsoft.com/en-us/library/cc645866.aspx
[46] N. Slonim, N. Friedman, and N. Tishby, "Unsupervised Document Classification Using Sequential Information Maximization," in *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '02. New York, NY, USA: ACM, 2002, pp. 129–136.

Diego Riofrío is a PhD in Computer Science student at UPM. He holds a MSc in Computer Science (UPM 2012) and a BSc in Computer Engineering (Escuela Politécnica Nacional of Ecuador 2006). From 2004 to 2010 he was a R&D engineer at private companies in Ecuador. His main research interests include Educational and Training Virtual Environments, Intelligent Tutoring Systems and Virtual Worlds.

Jaime Ramirez is associate professor at the School of Computer Science, UPM. He holds a MSc in Computer Science (UPM, 1996) and a PhD in Computer Science (UPM 2002). His main research interests include Ontology Engineering, and Adaptive Systems with an special focus on Intelligent Tutoring Systems combined with 3D Virtual Environments.

Marta Berrocal received a BSc in Biological Sciences from the Universidad Complutense de Madrid (UCM) in 1996, and a PhD from the Universidad Politécnica de Madrid (UPM) in 2002. Since 2007 she has been assistant professor of Biochemistry and Biotechnology at the UPM's School of Forestry Engineering and project coordinator of the UPM Biotechnology Virtual Laboratory Creation and Development Educational Innovation Group.