A MIXED-INITIATIVE INTELLIGENT TUTORING SYSTEM
Based on Learning from Demonstration

Omar Alvarez-Xochihua, Riccardo Bettati
Department of Computer Science and Enginnering, Texas A&M University, College Station, Texas, U.S.A.

Lauren Cifuentes, Rene Mercer
Department of Educational Psychology, Texas A&M University, College Station, Texas, U.S.A.

Keywords: Intelligent tutoring systems, Mixed initiative, Learning from demonstration, Weighted Markov models.

Abstract: We present the design and evaluation of the framework of a *Mixed-Initiative Intelligent Tutoring System* that augments existing tutoring systems by integrating two interactive modes: instructor-student, and intelligent tutor-student. These interactive modes are intended to support students in well- and ill-defined problem solving. In this paper we discuss the use of the *Learning from Demonstration* approach to derive the solution paths and the appropriate tutorial actions in response to observed student behavior and instructor intervention in the *cybersecurity* domain. Our method aims to discover large portions of domain and tutoring knowledge from instructors' interactions with students at run time. We describe the use of a *Weighted Markov Model* approach for data representation for sequential data. Our experimental results indicate that the proposed technique is useful for data sets of sequences.

1 INTRODUCTION

Intelligent Tutoring Systems (ITSs) have helped students learn how to solve problems in various domains (e.g., medicine, physics, and mathematics) since the early 1980's. ITSs are computer-based experts that provide customized instruction to help students while solving a problem (Psotka, et al., 1998). A challenging goal for any ITS is to learn enough adequate domain-specific knowledge and tutorial actions from experts to support students in education. Developers of ITSs gather this knowledge by interviewing experts and/or tracking their problem solving steps primarily at design time.

For the ITS to properly scaffold students, developers must consider the problem-structure continuum. Some problems students encounter within a domain can be very well-defined. These types of problems are mainly characterized by a limited number of correct answers and a systematic solution path. Therefore, gathering experts' knowledge at ITS design time is a suitable approach. At the other end of the problem-structure continuum are ill-defined problems. Fields that exemplify these problem types are referred to as ill-defined domains (e.g., law and architecture). A compilation of problems found within ill-defined domains summarized by Lynch, et al. (2006) follows:

- initial steps to solve a problem can vary,
- multiple solutions and solution paths exist,
- right answers are context and time dependent,
- systematic solution methods do not exist, and

In these domains, acquiring relevant and sufficient expert knowledge to design ITSs is extremely costly. Fournier-Viger, et al. (2008, p.46) state that "for many ill-defined domains, the domain knowledge is hard to define explicitly" when developing an ITS. In fact, even when experts can provide a large amount of domain knowledge, the possibility of new points of view or evidence that may challenge previous conclusions exists. Hence, for an ITS to effectively aid students in ill-defined problem solving, gathering comprehensive domain knowledge during design time, or inferring new knowledge and the appropriate feedback at run time for novel situations is difficult. Therefore, participation of instructors during the tutoring process beyond design time should be considered.

Alvarez-Xochihua O., Bettati R., Cifuentes L. and Mercer R..
A MIXED-INITIATIVE INTELLIGENT TUTORING SYSTEM - Based on Learning from Demonstration.
DOI: 10.5220/0003347803330339
In *Proceedings of the 3rd International Conference on Computer Supported Education* (CSEDU-2011), pages 333-339
ISBN: 978-989-8425-49-2
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)

333

To develop and refine tutorial actions at system run time, two challenges must be addressed: (1) the ITS must learn comprehensive domain knowledge and tutorial actions from instructor-student interactions; and (2), the ITS must select the most effective tutorial action when requested to do so. The selection of the tutorial action depends on the state estimation by the ITS, which is subject to possible errors. Each selected tutorial action is therefore subject to a level of confidence of the ITS as to its appropriateness.

We present a Mixed-Initiative ITS framework where the intelligent tutor can either initiate interaction or request for instructor interaction with students. We describe how the ITS's knowledge-base and tutoring actions can be built at run time from instructor-student interactions. The ITS will observe the instructor, and make use of *Learning from Demonstration* (*LfD*) techniques to derive tutorial actions in response to observed student solution paths and instructor interventions. The ITS will automatically determine confidence levels based on predetermined thresholds before initiating tutoring actions. If the threshold has not been met, the ITS will forward the student history to the instructor and record the instructor's response as additional demonstration.

2 RELATED WORK

2.1 Data-Based Knowledge Learning

Recently, a number of ITS developers have been building ITS knowledge-bases (domain and tutoring) through observation of student activity. Nkambou, et al. (2007) describe an ITS that builds a behavior graph (BG) from student activity. BGs are used to represent all possible correct and incorrect paths a student can take while solving a problem. Bernardini and Conati (2010) identify common interaction behaviors from logged students' data within an exploratory learning environment.

Remarkably little attention has been given to the observation of the human tutor during the ITS tutoring process. We integrate observations from both students and instructors at run time. This is particularly important when the ITS's knowledge base is not good enough to associate current tutorial actions to new or unexpected student behavior.

2.2 Learning from Demonstration

Computer learning techniques based on demonstrations are identified using different terms. The most commonly used term within the ITS community is Programming by Demonstration (PbD). PbD refers to nonprogrammers developing computer applications from demonstrations of what actions are appropriate for the system.

SimStudent (Simulated Student) is a PbD-based ITS. This ITS allows an instructor to construct a graphic interface for a specific problem, and then use the interface to demonstrate successful problem solving. The ITS induces production rules (a set of conditions) from demonstrations that replicate the instructor's performance. Recently, *SimStudent* developers evaluated a new training method in which the author gives *SimStudent* problems to solve, and then it applies existing knowledge to solve the problem (Matsuda, et al., 2008). If *SimStudent* encounters knowledge gaps, it asks for instructor help. The instructor then teaches the ITS by demonstrating a correct step. Developing accurate student models is difficult and in many cases unreliable. Instead, we leverage existing usage data to identify more authentic student misconceptions and novel states.

Additional ITSs using PbD approaches exist (Aleven, et al., 2009). However, these demonstrations come only from the ITS's authors. Furthermore, problem solving demonstrations are mainly implemented at design time.

2.3 Mixed-Initiative Interaction

Mixed-Initiative interaction aims to provide an effective multi-agent (human or computer) collaboration to perform a task. Hearst, et al. (1999, p.14) describe Mixed-Initiative interaction as "a flexible interaction strategy, where each agent can contribute to the task what it does best."

Initially, ITSs using this approach were conversation-based systems that allowed students or ITSs to direct the conversation to perform a learning task (Freedman, 1997). There are also Mixed-Initiative ITSs within educational games. The ITS interacts with a student to elect who will direct gameplay (Caine and Cohen, 2007). Hubal and Guinn (2001) presented a Mixed-Initiative virtual training environment based on agents that simulate realistic interactions for students.

Even though their Mixed-Initiative is primarily based on agent-student interaction, these researchers also introduced the importance of instructor participation. They consider the intelligent virtual tutor as an instructor's helper, assisting students when the instructor is unavailable. While recent studies regarding ITSs using a Mixed-Initiative approach have focused on the ITS-student interaction, we expand this approach by integrating the instructor into the tutoring process.

3 MIXED-INITIATIVE ITS FRAMEWORK

We propose a more flexible and intelligent interaction framework, where the ITS and human-tutor interact contributing to the tutoring task what each does best. In this section we discuss the solutions used in order to implement our approach.

3.1 Building Comprehensive Domain Knowledge and Tutorial Actions

The ITS's domain knowledge and tutoring actions can be built at run time by using LfD techniques. LfD can be used to derive a policy from a set of labeled data to autonomously classify and respond with tutorial actions for students. Argall, et al. (2009) describe the formal LfD problem as a world consisting of states *S* (i.e., the student solution paths) and actions A (i.e., tutorial actions). The learning algorithm develops a policy $p : S \rightarrow A$ that selects the "appropriate" action in response to the observed world state.

The policy construction is defined by (a) how demonstrations are collected and when they are used, and (b) how the policy is derived from the observed demonstrations. The use of demonstrations can happen in *batch fashion*, where the policy is derived after all demonstrations have been collected, or in *on-line fashion*, where the policy is refined as more demonstrations are available (See Figure 1).

The policy is constructed as a function ($f():S \rightarrow A$) from the available demonstrations (e.g., a history of student activity and its required tutorial action.) The goal is to identify and generalize mappings from student activities to tutorial actions.

3.2 ITS Knowledge Confidence Level

Policy derivation relies greatly on classification of student activities and tutorial actions. Since such classifications are error prone, the effectiveness of the ITS relies heavily on its ability to assess and improve the confidence in individual classification results. We use confidence-driven LfD to trigger requests for classification from the instructor whenever the classification results indicate low confidence (Chernova and Veloso, 2007). Whenever the confidence level resulting from a classification falls below a given threshold, the ITS will contact the human instructor for a demonstrative tutorial action. This action is (a) forwarded to the student and (b) added to the ITS's knowledge base and used to further refine future classifications.

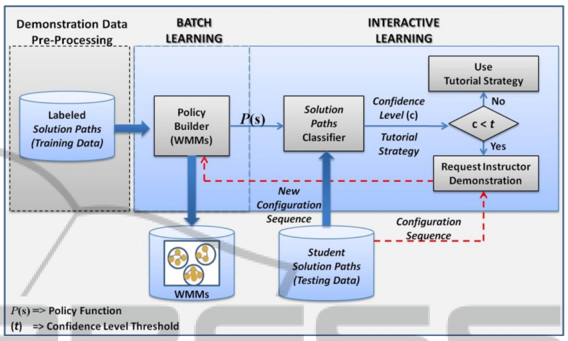

Demonstration Data Pre-Processing
BATCH LEARNING
INTERACTIVE LEARNING
Labeled Solution Paths (Training Data)
Policy Builder (WMMs)
P(s)
Solution Paths Classifier
Confidence Level (c)
Tutorial Strategy
Use Tutorial Strategy
No
c < t
Yes
Request Instructor Demonstration
New Configuration Sequence
Configuration Sequence
WMMs
Student Solution Paths (Testing Data)
P(s) => Policy Function
(t) => Confidence Level Threshold

Figure 1: Mixed-Initiative ITS Learning Process.

4 LEARNING DOMAIN

We tested the proposed framework in the Web Access Exercise System (WAES), a case-based instructional system that provides training in well- and ill-defined *cybersecurity* problems (Cifuentes, et al., 2009). In a typical real-world *cybersecurity* scenario, practitioners create an organizational security plan that includes a set of security requirements. Then, they select and configure a set of security services (e.g. firewalls, and VPN servers) that will protect the organization's information.

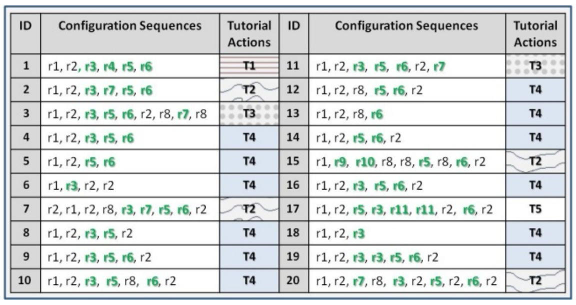
This domain exposes students to ill-defined situations. For example, one possible solution may use any number of the security services mentioned previously and then customize them to fit a specific organization's security requirements. In addition, as the number of security requirements increases, the possibility of incorrectly applying sequences of configuration rules increases; resulting in a multitude of correct, partially correct, or incorrect configuration paths.

4.1 Data Representation

Modeling dynamic behaviors characterized by ill-defined problems is not always feasible by using traditional machine learning approaches such as feature vectors. In particular, for our chosen domain, we encountered multiple versions and lengths of solution paths from each student. The conversion of this sequential data to vector structures would lead to loss of relevant information.

We implemented a Weighted Markov Model (WMM) approach as the primary clustering and classification method for the available sequential data. We define a WMM as a set of Markov Models (MMs) that, in addition to computing the transition probabilities, computes weight values in order to provide better classification predictions. In a regular Markov Model, there are *n* distinct states $S=\{s_1,s_2,\ldots,s_n\}$, a vector of initial state probabilities $B=\{b_1,b_2,\ldots,b_n\}$, and a transition probability matrix $A=\{a_{si},a_{sj}\}$. We add an initial-state-weights vector $WB=\{wb_1,wb_2,\ldots,\ wb_n\}$ and a transition-weights matrix $WA=\{wa_{si},\ wa_{sj}\}$.

We construct a set of MMs from student input as follows: We structure input into so-called configuration sequences (*CSs*), which in turn consist of individual configuration rules (*CRs*). An example illustrating *CSs* from three students is shown in Table 1: Student one has two *CSs* with three *CRs* respectively, while students two and three each only have one *CS*. We further parse a *CR* into a sequence of parameters. For instance, the rule "iptables -A FORWARD -j DROP" has three parameters ("iptables", "-A FORWARD", and "-j DROP"). Table 1 also displays examples of different configuration behaviors within each *CS* (e.g., order of parameters and rules, use of an alias, and/or use of abbreviations), semantic and syntactic misconceptions, and parameters functionality type (e.g. informative, resetting, and configuration). A vocabulary catalog is generated from the *CSs*. The vocabulary includes all of the configuration parameters used by students. Parameters are labeled with the corresponding functionality type. We automatically label the parameters by using the responses from network devices to the command entries. Finally, *CS*s are represented as string sequences to build the intended WMMs (See list of *CSs* in Figure 2).



ID	Configuration Sequences	Tutorial Actions	ID	Configuration Sequences	Tutorial Actions
1	r1, r2, **r3**, **r4**, **r5**, **r6**	T1	11	r1, r2, **r3**, **r5**, **r6**, r2, **r7**	T3
2	r1, r2, **r3**, **r7**, **r5**, **r6**	T2	12	r1, r2, r8, **r5**, **r6**, r2	T4
3	r1, r2, **r3**, **r5**, **r6**, r2, r8, **r7**, r8	T3	13	r1, r2, r8, **r6**	T4
4	r1, r2, **r3**, **r5**, **r6**	T4	14	r1, r2, **r5**, **r6**, r2	T4
5	r1, r2, **r5**, **r6**	T4	15	r1, **r9**, **r10**, r8, r8, **r5**, r8, **r6**, r2	T2
6	r1, **r3**, r2, r2	T4	16	r1, r2, **r3**, **r5**, **r6**, r2	T4
7	r2, r1, r2, r8, **r3**, **r7**, **r5**, **r6**, r2	T2	17	r1, r2, **r5**, **r3**, **r11**, **r11**, r2, **r6**, r2	T5
8	r1, r2, **r3**, **r5**, r2	T4	18	r1, r2, **r3**	T4
9	r1, r2, **r3**, **r5**, **r6**, r2	T4	19	r1, r2, **r3**, **r3**, **r5**, **r6**, r2	T4
10	r1, r2, **r3**, **r5**, r8, **r6**, r2	T4	20	r1, r2, **r7**, r8, **r3**, r2, **r5**, r2, **r6**, r2	T2

Figure 2: Configuration sequences and their tutorial actions. Rules in bold represent configuration rules.

4.2 Weighted Markov Models

Figure 2 shows a set of 20 *CSs* that have been reviewed by an expert and assigned a tutorial action. Only five different tutorial actions (e.g. T3 = "Deny-All rule must be the last rule in your configuration") were needed for the entire set of *CSs*. We expect this to be typical, with a small set of tutorial actions addressing multiple *CSs*. Figure 3 represents the set of MMs generated from data in Figure 2. Each model represents a tutorial action. Figure 4 shows the transition weights of the MMs based on the average occurrence within the entire dataset. Based on the expert's classification of the twenty *CSs,* five clusters were generated. Then we computed the associated matrices and vectors based on the *CSs*. The vector of initial state probabilities *(B)* and the transition probabilities matrix *(A)* for each model were computed using equations (1) and (2).

$$b\mathrm{si} = \frac{\beta(\mathrm{si})}{\sum_{m=1}^{k}\beta(\mathrm{sm})} \tag{1}$$

$$a\mathrm{si,sj} = \frac{\sigma(\mathrm{si,sj})}{\sum_{m=1}^{k}\sigma(\mathrm{si,sm})} \tag{2}$$

Table 1: Frequent behaviors and misconceptions faced in configuration-rules.

Student-ID	Configuration-Rules	Configuration behaviors
1	iptables –**flush**	reset configuration command
	iptables –**L**	Informative command
	iptables -A FORWARD -p tcp --dport http -j ACCEPT	
1	iptables –**F**	different/correct command
	iptables -A FORWARD -j DROP	different/incorrect rules order
	iptables -A FORWARD **--dport http –p tcp** -j ACCEPT	different parameters order
2	iptables -A FORWARD -p tcp **--dport httt** -j ACCEPT	incorrect parameter name
	iptables -A FORWARD -p tcp **--dport ftp** -j ACCEPT	incorrect parameter value
3	iptables –**F**	different/correct command
	iptables -A FORWARD -p tcp **--dport 80** -j ACCEPT	different/correct parameter value

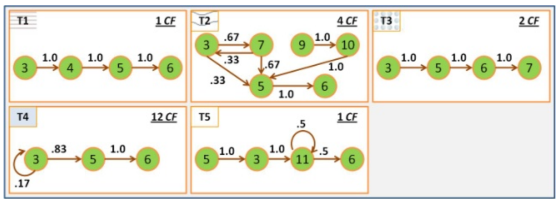


Figure 3: Set of Markov Models with Transition Probability Distributions.

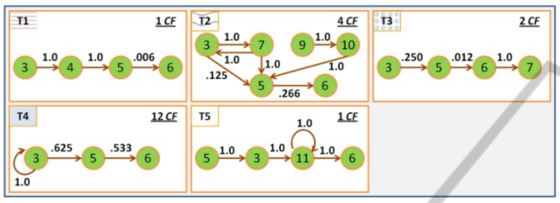


Figure 4: Set of Markov Models with Transition Weights based on State-Transition Frequency (e.g. State-Transition 3=>5 has 8 occurrences in the entire dataset. 1 in T2 (w=.125), 2 in T3 (w=.250), and 5 in T4 (w=.625).

We denote by $\beta(s_i)$ the number of initial states within a model that match the State s_i, and k is the number of states within the model. The term $\sigma(s_i,s_j)$ denotes the number of transitions from State i to State j. To compute the initial-state weights vector *(WB)* and the transition weights matrix *(WA)* for each of the models we used the following equations (l denotes the number of WMMs):

$$wb_{s_i} = \frac{\beta(s_i)}{\sum_{m=1}^{l} WMM^{(m)}\{\beta(s_i)\}} \qquad (3)$$

$$wa_{s_i,s_j} = \frac{\sigma(s_i,s_j)}{\sum_{m=1}^{l} WMM^{(m)}\{\sigma(s_i,s_j)\}} \qquad (4)$$

4.3 Knowledge Confidence Level

A new *CS* as entered by a student is classified by determining the greatest similarity between the new *CS* and the existing WMMs. We say that a WMM *represents* a *CS* when the model is able to generate that particular *CS*. Similarly, we say that the WMM partially represent a *CS* if it can only generate a sub-sequence of the *CS*. *Similarity* is determined by first identifying those WMMs that are able to completely or partially represent the new *CS,* and by then selecting the one with the highest likelihood. This measure is computed by multiplying the transition probability and positive weight values from each state transition within the new *CS* by using Equation (5). This probability value represents the confidence level returned by the classification policy.

When none of the current models is able to completely represent the new *CS*, those models that best represent the new *CS* are considered (partial representation). This partial classification approach is only considered when the number of supported rule-transitions in the new *CS* is greater than 50%; otherwise the ITS will automatically determine its tutoring confidence level to be 0.

$$\mathrm{WMM}_{n=1..l}^{(n)} = (b_{s1} * wb_{s1}) \prod_{i=1}^{k-1} (a_{s_i,s_{i+1}} * wa_{s_i,s_{i+1}}) \quad \forall\, wb_{s1}, wa_{s_i,s_{i+1}} > 0 \qquad (5)$$

In Equation (5), l represents the number of WMMs within the classification policy, s_i represents the observed states *(CR)* within the new *CS*, b_{s1} and wb_{s1} are the initial state probability and weight values for the first state in the *CS* respectively, and k is the number of observed states within the new *CS*.

5 FRAMEWORK EVALUATION

For the evaluation of our Mixed-Initiative ITS framework we used data collected from several *cybersecurity* classes that used WAES.

5.1 Participants and Implementation

Data was obtained from 20 community college students. Most of the students had medium to advanced computer skills and were asked to configure network devices. In addition, one expert in *cybersecurity* participated in the development of the tutorial actions and evaluation of the ITS framework performance.

Demonstration data consisted of students' *CSs* labeled with experts' tutoring actions. A total of 100 *CSs* consisting of more than 700 *CRs* were gathered. From these *CSs* we obtained more than 130 different *CRs*, and 100 different configuration parameters. About two-thirds of the available data was used for training. The remaining data was used for testing the framework. We used training data to build the set of MMs as a batch learning process; representing the initial learned classification policy. Then, the ITS framework used this policy to classify new *CSs* and to generate the appropriate tutorial action.

We implemented a non-intrusive evaluation of the ITS framework into the WAES architecture. Since the ITS framework is not tightly integrated into the WAES, these analyses can be performed off-line. We are able to "replay" student activities and associated ITS feedback and fine tune ITS framework elements to optimize effectiveness.

5.2 ITS Framework Effectiveness

To determine whether the ITS is "learning", we measured the *precision* and *accuracy* of the classification policy used to drive the feedback from the ITS. *Precision* refers to the capability of the ITS to classify students' configurations with similar misconceptions within the same cluster. *Accuracy* is the degree of veracity of the ITS's tutorial responses. (i.e., how close a recommended ITS tutorial action is to the one recommended by a human expert).

In our context, *precision* represents the ability of the ITS to learn comprehensive domain knowledge and tutorial actions from instructor-student interactions. The *accuracy,* on the other hand, captures the ITS's ability to use the learned domain. We measured the bias of the ITS by having a domain expert grade each ITS tutorial action selection based on correctness. In addition, when the ITS requested for expert help, we analyzed whether an existing tutorial action could have been used.

5.2.1 Precision of the Framework

From the training data we obtained eight different tutorial actions from the expert. A unique tutorial action was assigned to each *CS.* Based on this categorization we built eight WMMs. Equation (5) was used to compute the WMMs' probability values for each *CS* within the testing data. Then, *CSs* were classified within the WMMs by considering the highest probability value, this value represented the ITS confidence level.

In the classification process we observed that 45% of the new *CSs* were completely represented by one or more WMMs. 100% of the completely represented *CSs* were classified correctly. An additional 30% were not completely represented, but sufficiently supported (at least 50%) by one or more WMMs. The remaining 25% could not be classified with sufficiently high confidence by the ITS. Of the 30% that were sufficiently supported, 75% were correctly classified. Another 25% were incorrectly classified by the ITS.

The *precision* outcomes for each of the three identifications mentioned previously are shown in Figure 6. These outcomes demonstrate that the use of WMMs as the classification policy allows us to classify new *CSs* previously learned properly, even when we noticed that the sequence of *CRs* was not to identical to those in the training data. For *CSs* including partially new *CRs*, the *precision* of the classification was considered acceptable by the expert. The expert determined that 90% of the *CSs* with representation lower than 50% corresponded to *CSs* consisting of new completely correct or incorrect configurations. However, improvement of the classification algorithms for partially represented *CSs* is needed. We expect to reduce incorrect classifications of partially represented *CSs* by adjusting the estimation of the weight values within the WMMs in order to eliminate probable overweighting situations.

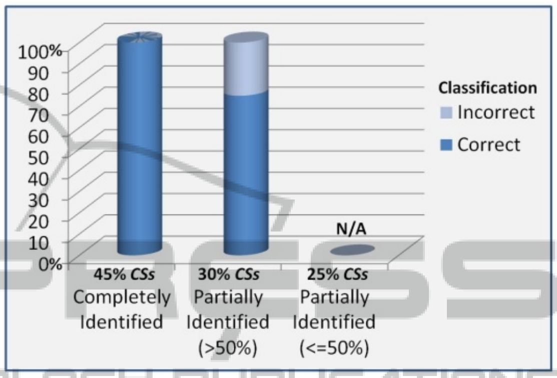


Figure 6: Precision outcomes per type of identification.

5.2.2 Accuracy of the Framework

Once a new *CS* has been classified in a specific WMM, the ITS determines its confidence level by comparing the probability value obtained from the selected WMM and the previously specified threshold. In this study we used a multiple-threshold approach. The number of *CSs* and rule-transitions within each WMM are different. Also, because most of the transition-probability and transition-weight values were lower than 1, the final estimation of the confidence level is a function of the length of the *CSs* based on the number of rule-transitions within a new *CS*. Therefore, we used a separate threshold value for each WMM. Using separate thresholds provided more precise control over each WMM.

For this experiment we used a straightforward threshold estimation approach. Initially, threshold values were computed by reclassifying *CSs* within the training data in their assigned WMMs. We selected the lowest probability-value returned by the reclassification process within each WMM as the threshold value. Then, we set stricter thresholds to decrease the classification errors for partially represented *CSs*. This allowed the ITS framework to add more *CSs* to the set of unknown classifications, rather than recommending a tutorial action for a new *CS* with a low confidence level. The above threshold approach gave us an increase of correct classification from 75% to 85% for partially represented *CSs*. However, this approach decreased correct classification by 12% for completely represented *CSs*. These results indicate that, different threshold values must be considered for classification of new *CSs* that are completely and partially represented. In addition, new methods for dynamic threshold estimations are going to be implemented in order to allow the ITS to adjust the threshold values at run time.

6 CONCLUSIONS

We have presented a novel Mixed-Initiative ITS framework using an *LfD* approach. We trained the ITS domain knowledge and tutoring actions from data of human instructor-students interaction. We tested the proposed framework using data from the *cybersecurity* domain. A WMM approach was used to represent sequential data. We determined that an ITS using the proposed framework can build comprehensive domain knowledge and appropriate tutorial actions based on human instructor-students interaction. We also found that the ITS can estimate its knowledge confidence level in order to initiate interaction with students and scaffold them based on learned knowledge, or submit a help request asking the instructor to lead the tutoring process.

Our Mixed-Initiative framework extends the knowledge base that currently exists in the ITS field by: presenting a way to integrate instructors into the tutoring loop; and, continuously improving an ITS's domain knowledge. By implementing these features we support developers of intelligent tutors in addressing ill-defined domains that are very dynamic. The use of students' data to generate the ITS's knowledge-base will help in the identification of unexpected situations, as well as contextualize the domain knowledge to specific audiences. By adding two interactive modes to support cognitive processes, we help to leave outliers and pedagogically interesting situations to the instructor to handle and routine situations to the ITS.

ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation under award number OCI-0753408. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the national Science Foundation.

REFERENCES

Aleven, V., McLaren, B. M. and Sewall, J., 2009. Scaling up programming by demonstration for ITS development: an open-access Web site for middle school mathematics learning, *IEEE Transactions on Learning Technologies*, 2(2), p. 64-78.

Argall, B., Chernova, S., Veloso, M. and Browning, B., 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5), p. 469-483.

Bernardini, A. and Conati, C., 2010. Discovering and recognizing student interaction patterns in exploratory learning environments. *International Conference on Intelligent Tutoring Systems,* (1), p. 125-134.

Caine, A. and Cohen, R., 2007. Tutoring an entire game with dynamic strategy graphs: the mixed-initiative sudoku tutor. *Journal of Computers*, 2(1), p. 20-32.

Chernova, S. and Veloso, M., 2007. Confidence-based policy learning from demonstration using gaussian mixture models, *6th International joint conference on autonomous agents and multiagent systems*. ACM, New York: USA.

Cifuentes, L., Marti, W., Alvarez, O., Mercer, R. and Scaparra, J., 2009. Systematic design of a case-based learning environment. *World conference on educational multimedia, hypermedia and telecommunications*, Honolulu, HI, USA. 22-26 June.

Fournier-Viger, P., Nkambou, R. and Mephu N. E., 2008. A sequential pattern mining algorithm for extracting partial problem spaces from logged user interactions. *3rd International workshop on ITS in ill-defined domain*. Montreal, Canada. 23-27 June.

Freedman, R., 1997. Degrees of mixed-initiative interaction in an intelligent tutoring system. *AAAI97 Spring symposium: computational models for mixed initiative interaction*, Stanford, CA. USA. p. 44-49.

Hearst, P., Allen, J. F., Guinn, C. I. and Horwitz, E., 1999. Trends and controversies: mixed-initiative interaction. *IEEE Intelligent Systems*, 14(5), p. 14-23.

Hubal, R. and Guinn, C., 2001. A mixed-initiative intelligent tutoring agent for interaction training. *Intelligent user interface conference*, Santa Fe, NM.

Lynch, C. F., Ashley, K. D., Aleven, V. and Pinkwart, N., 2006. Defining "ill-defined domains"; A literature survey. 2nd *International workshop on ITS in ill-defined domain*. Jhongli, Taiwan. 26-30 June.

Matsuda, N., Cohen, W., Sewall, J., Lacerda, G., and Koedinger, K., 2008. SimStudent: building an ITS by tutoring a synthetic student. *International journal of artificial intelligence in education (in preparation).*

Nkambou, R., Mephu N. E., Couturier, O., and Fournier-Viger, P., 2007. Problem-solving knowledge mining from users' actions in an ITS. *20th conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence. Springer-Verlag, Berlin, Heidelberg,* p. 393-404.

Psotka, J., Massey, L.D. and Mutter, S. A. 1988. *Intelligent Tutoring Systems: Lessons Learned.* Lawrence Erlbaum Associates.