An Intelligent Agent that Learns How to Tutor Students: Design and Results

Leen-Kiat Soh and Todd Blank
Department of Computer Science and Engineering
University of Nebraska, Lincoln, NE 68588-0115 USA
{ lksoh, tblank }@cse.unl.edu

Abstract. In this paper, we describe an intelligent agent that presents different learning content such as tutorials, examples, and problems adaptively to individual students and learns from its interaction with the students how to improve its performance. We have built an end-to-end intelligent tutoring system, premised on the above goal, with a graphical user interface (GUI) front-end, an agent powered by case-based reasoning (CBR), and a mySQL database backend. We use a casebase to store the pedagogical strategies, embedded in the individual cases and the similarity retrieval and adaptation heuristics. Each case has situation, solution and outcome parameters. The situation parameters include the students' static and dynamic profiles and the instructional content's characteristics while the solution parameters specify the characteristics of the example or problem to be delivered to the student. We developed a set of CS1 content that includes five topics and deployed our system in the laboratories. Our results show that when the machine learning mechanism is activated, our agent is able to learn to tutor students more efficiently.

Introduction

There exists no universal agreement on which abilities are needed or present in intelligent tutoring systems (ITSs) [12]. ITSs may possess a combination of the following attributes [12]: (1) *Generative* in which an ITS dynamically generates instructional material including problems, hints, and help on student performance; (2) *Able to model students* in which an ITS assesses the current state of a student's knowledge and does something instructionally useful based on the assessment; (3) *Able to model expert performance* in which an ITS detects expert performance and derive an instructionally useful model; (4) *Able to change pedagogical strategies* in which an ITS changes its instructional style based on the changing state of the student model, prescriptions of an expert model, or both; (5) *Mixed-initiative* in which an ITS changes its presentation schemes to improve human-computer interaction; and (6) *Self-improving* in which an ITS has the capacity to monitor, evaluate, and improve its own teaching performance as a function of experience.

Based on the above, a truly intelligent ITS should be adaptive, leading to customized tutoring and learning for individual students. Further, for any intelligent system to be adaptive, it must be able to (1) monitor and evaluate its own interaction with the students, (2) modify its *knowledge* if the evaluation result is below par, and (3) subsequently apply the modified knowledge. An ITS' knowledge generally consists of instructional content (tutorials, examples, problems, etc.), modeling of students, and pedagogical strategies.

Instructional content might not be always appropriate (for example, a particular problem may be appropriate for one student but too easy for another student), modeling of students might not be accurate (for example, a student who fails three problems in a row could be perceived as lacking understanding of the topics involved or lacking motivation to study the problems carefully), and in some instances, pedagogical strategies might not be applied correctly. Thus, when an ITS modifies its knowledge, it could modify its instructional content, its modeling of a particular student, or its confidence in a particular pedagogical strategy

In this paper, we propose an ITS with machine learning capabilities to refine its performance over time. The ITS, named Intelligent Learning Materials Delivery Agent (ILMDA), models students based on how they interact with the system (not just question-answer assessment), changes its pedagogical strategies, and self-improves. ILMDA's machine learning capabilities are powered by case-based reasoning (CBR) [8]. We use a casebase to store the pedagogical strategies, embedded in the individual cases and the similarity retrieval and adaptation heuristics.

In the following, we first discuss some related work in the area of intelligent tutoring systems. Then, we present our ILMDA project, its goals and framework. Subsequently, we outline our use of the CBR methodology. We then describe the implementation of ILMDA briefly. We have built an end-to-end ILMDA infrastructure, with an interactive GUI front-end, an agent powered by CBR and capable of learning, and a mySQL multi-database backend. We have deployed ILMDA in a CS1 course in the Fall semester of 2004. The results, though preliminary, are encouraging, indicating that ILMDA is able to learn to deliver more appropriate examples and problems based on student behavior.

1. Related Work

There have been successful ITSs such as PACT [7], ANDES [4], AutoTutor [5], and SAM [3]. Though ITSs have been tested on students and have been proven to facilitate learning, Graesser et al. [5] pointed out the weaknesses of the current state of tutoring systems: First, it is possible for students to guess and find an answer and such shallow learning will not be detected by the system. Second, ITSs do not involve students in conversations so students might not learn the domain's language. Third, to understand the students' thinking, the GUI of the ITSs encourages students to tend to focus on the details instead of the overall picture of a solution. Furthermore, these systems do not generally adapt to new circumstance, do not self-evaluate and self-configure their own strategies, and do not monitor the usage history of the instructional content being delivered or presented to the students.

A closely-related work to ILMDA is AnimalWatch. AnimalWatch [12] is an arithmetic tutor tailored for female elementary school students. It has a reinforcement learning component that automatically computes an optimal teaching policy, such as reducing the amount of mistakes made or time spent on each problem by using a "two-phase" learning algorithm: (1) model how a student interacted with the tutor, and (2) construct a tutoring policy. The reinforcement learning mechanism takes as input as set of abilities that describe the current tutoring situation and output a recommended action (e.g., which problems to generate and what type of hints to show). The tutor then observes whether the student's response is correct and how much it takes the student to generate this response. By keeping these observations over time, AnimalWatch is able to make predictions about "an average student with a proficiency of *x* who has made *y* mistakes so far on the problem." The authors report that students using the machine learning version averaged 27.7 seconds to solve a problem, while students using the classic version averaged 39.7 seconds. Our ILMDA design is similar to AnimalWatch in terms of the underlying concept—observing student interaction with the system and reinforcing decisions based on the observed outcomes. However, our machine learning component is comprehensive: it refines the existing pedagogical strategies that it has in its repository, learns new pedagogical strategies, learns how to apply them, learns when to use them; it also learns to quality-tag the system's instructional content.

2. Design

Our ILMDA system is based on a three-tier design. It consists of a graphical user interface (GUI) front-end application, a database backend, and the intelligent agent reasoning in between. A student user accesses the content set via the GUI. The agent captures the student's interactions with the GUI and provides the ILMDA reasoning module with a parametric profile of the student and environment. The ILMDA reasoning module performs CBR to obtain a search query (a vector of search keys), and then uses the query to retrieve the most appropriate example or problem from the database. The agent then delivers the example or problem in real-time back to the user through the interface.

2.1 Case

Each instructional content set consists of three parts: (1) a tutorial, (2) a set of related examples, and (3) a set of exercise problems to assess the student's understanding of the topic. Based on how a student progresses through the content set and based on his or her profile, ILMDA chooses the appropriate examples and exercise problems for the student. How to choose the appropriate examples and exercise problems is based on case-based reasoning (CBR). ILMDA has a casebase of cases. Each case is composed of three parts: situation, solution, and outcome parameters.

The *situation parameters* include the student static and dynamic profiles and the instructional content's characteristics. The student static and dynamic profiles form the basis for the ILMDA's learner model. A learner model tells us the metacognitive level of a student by looking at the student's behavior as he or she interacts with the instructional content set. We achieve this by profiling a learner/student along two dimensions: student static background and dynamic student activity. The background of a student stays relatively static and consists of the student's last name, first name, major, GPA, goals, affiliations, aptitudes, and competencies. It also includes self-reported self-efficacy and motivation, based on a survey taken before a student processes a content set. The dynamic student profile captures the student's real-time behavior and patterns. It consists of the student's online interactions with the GUI module of ILMDA including the number of attempts on the same item, number of different modules taken so far, average number of mouse clicks during the tutorial, average number of mouse clicks viewing the examples, average length of time spent during the tutorial, number of quits after tutorial, number of successes, and so on. For details on these parameters, please refer to [1].

The *solution parameters* specify the characteristics of the example or problem to be delivered to the student. Each example or problem is meta-tagged with a set of attributes: length, interest, Bloom's taxonomy, level of difficulty, degree of scaffolding, the number of times viewed, the average time per use, average number of clicks per use, and so on.

Bloom *et al.* [2] identified three domains of educational activities. The three domains are cognitive, affective, and psychomotor. Cognitive is for mental skills (Knowledge), affective is for growth in feelings or emotional areas (Attitude), while psychomotor is for manual or physical skills (Skills). There are six major categories, starting from the simplest behavior to the most complex: knowledge, comprehension, application, analysis, synthesis, and evaluation. Currently, most of our examples and problems are at first four of Bloom's levels.

Instructional scaffolding [11] is support for learning, and the level of scaffolding varies for different students and scenarios. In ILMDA, we use scaffolding similar those proposed in [6]. Specifically, we use cues, hints, references, and elaborations. Cues are highlighted phrases. A hint poses “what if” and “think about this” statements. A reference points the student to a particular item in the content set. An elaboration explains the steps or partial solutions. We use highlighted phrases, diagrams, and figures as cues; “What if” and “Think about this” questions as hints and prompts; and stepwise, spelled-out solutions as partial solutions. In addition, references will be used to point students back to certain pages of a tutorial or an example.

The *outcome parameters* are based on the usage history of a case, which includes the number of times the case has been used, the number of times the case has been used successfully, whether the student quit the example or problem, whether the student answers the problem correctly, and so on.

2.2. Case-Based Reasoning (CBR) and Learning

Briefly, a CBR system works as follows. When a new situation arises, the system searches its casebase for cases whose situations are similar to the new situation. This is termed similarity-based retrieval. Usually, the system retrieves the most similar case as the best case. Given the best case, the system now has in its reasoning process a solution. However, this solution is one that corresponds to the best case’ situation. Thus, the system has to adapt the solution to match the new situation. Usually, the system modifies the solution based on the differences between the best case’s situation and the new case's situation. After adaptation, the solution is applied. After application, the new situation and the adapted solution, together with the outcome, become a new case. If this new case is sufficiently different from all existing cases in the casebase, the system adds it to the casebase. The addition of a new case is case-based learning (CBL).

Each case documents a unique pedagogical strategy: given the situation, what is the appropriate solution? ILMDA forms the situation based on its observation of the student. For example, for students who have higher motivation, self-efficacy and aptitude, ILMDA will give lower level of scaffolding such as cues; and vice versa. With this strategy, ILMDA will gradually withdraw the scaffolds as a student is observed to have improved his/her performance.

In our agent design, we have three case-based learning modules. First, there is a *case learning module* that learns about good and bad cases. Secondly, we have a *similarity learning module* to learn about the significance of situation parameters such as GPA and the time spent in the tutorial. This allows us to retrieve most weighted similar cases as best cases, taking into consideration that some parameters are more important than the others. Finally, we have an *adaptation learning module* that learns about the importance or quality of the adaptation heuristics. The case learning module uses traditional CBL and a small amount of reinforcement learning. The other two modules meta-learn how to better retrieve from the database, evaluate the similarity between two cases, and adapt the solution of the best case to the current problem. For details on these modules, please refer to [1]. Briefly, by recording the success rate of each case, ILMDA refines its existing pedagogical strategies. Through the case learning module, ILMDA learns new pedagogical strategies. Through the similarity learning module, ILMDA learns when to use them. Through the adaptation learning module, ILMDA learns how to apply them. For detailed assumptions behind our agent learning and case adaptation, please refer to [9].

3. Implementation

We have implemented an end-to-end prototype ILMDA system with a front-end, applet-based Graphical User Interface (GUI) that captures all user mouse activities, a CBR-powered agent, and a backend database system containing a set of learning materials. The ILMDA system is written in Java (SDK version 1.4.2). Both integrated development environments were run under Windows 2000/XP operating systems. The backend database is a MySQL database (Version 3.23). The system connects to the database using the JDBC drivers for Java. We have also built a comprehensive simulator [10] to provide us with *virtual students* to use ILMDA. This simulator allows us to test the correctness and feasibility of the ILDMA before deployment.

4. Results

In Fall 2004, we deployed ILMDA to CSCE155 at the Department of Computer Science and Engineering at the University of Nebraska. CSCE155 is the first core course for the Computer Science majors. Typically, it has about 150 students per year, with a diverse group of students from majors such as CS, Math, Engineering, and so on. Further, the programming background of these students is highly diverse. Some incoming freshmen have had some programming in their high schools; some have had none. Thus, it is important for the course to be able to adapt to the different student aptitude levels and motivations. This course is thus well-suited for evaluating ILMDA.

The course has a 2-hour structured lab component once a week. We used ILMDA in four CS1 labs: (1) File I/O, (2) Exceptions, (3) Inheritance and Polymorphism, and (4) Recursion. For each topic, the content set had a tutorial, a set of 3-4 examples, and a set of 20-25 problems. Students reviewed these materials through ILMDA prior to their labs.

For our evaluation, we used two types of agents: learning and non-learning. Our non-learning ILMDA did not refine its adaptation heuristics, did not learn new cases, and did not adjust it similarity weights. It basically disabled all its learning capabilities. The learning ILMDA enables all its machine learning mechanisms. We set up the ILMDA system such that it toggled on and off the learning mechanisms alternatively for each login.

4.1 Case-Based Learning Results: Impact on Pedagogical Strategies

Initially, at the beginning of the Fall 2004 semester, we set the weights on the situation parameters to 1.0; that is, all situation parameters are considered equally important. At the end of the semester, however, ILMDA modified these weights based on its observation of how many times a retrieved case was useful. Briefly, we found that GPA was not that important (dropping from 1.0 to 0.46), quitting a problem before answering it was important (increasing from 1.0 to 2.54), the number of times a student went back-and-forth between an example and the tutorial was quite important (0.80) but not as important as the back-and-forth between a problem and the tutorial (0.99), and so on. These results are empirical data, key to the self-evaluation capabilities of ILMDA to meta-learn about when to use which pedagogical strategies.

We also observe learning results in adaptation heuristics—there are 9 sets of heuristics, and each heuristic is triggered by a subset of about 20 situation parameters. An example of adaptation heuristics is as follows: To adapt the solution parameter BLOOM (i.e., the Bloom's taxonomy), we look at the differences between the new situation and the best case' situation. Initially, at the beginning of the semester, we set the weights of each situation parameters on the adaptation heuristics to be 1.0. Take the number of questions answered correctly by a student (extracted from the student's profile) as "successes". If "successes" of the new situation is greater than "successes" of the best case' situation, then increase the Bloom's taxonomy level of the problem to be selected next. At the end of the semester, this number increased to 9.0. We also observe a significant change in the weight of "exmpQuits" (the number of quits during an example), going from 1.0 to -11.76. This indicates that the more often a student has quit during the viewing of an example, the lower the Bloom's taxonomy level is for the problem to be selected.

Another example is to adapt the solution parameter SCAFFOLDING (i.e., the degree of scaffolding). Initially, we expected that if a student is observed to have spent more time on a tutorial, example, or problem, or if a student is observed to go back-and-forth between example and tutorial, or problem and example, then ILMDA should provide a higher degree of scaffolding. However, after a semester of using ILMDA, our expectation was not met for "aveExmpTime" (average time spent on an example), "expToTtrl" (number of times going from example to tutorial, and "probToExmp" (number of times going from problem to example). We suspect that this was due to the following: When students were found to spend more time on examples, or were found to refer back to the examples, they were more likely to quit ILMDA before answering questions correctly. As a result, ILMDA adjusted its adaptation heuristics to reduce the degree of scaffolding, thinking that this would improve its performance.

4.2 A Case Study: Recursion

Here we report a case study on the Recursion topic:

- The average numbers of examples and problems that a student read when interacting with the learning ILMDA were 2.216, and 10.946, respectively and that with the non-learning ILMDA were 3.047 and 15.116, respectively. This indicates that the learning ILMDA system was able to deliver fewer examples and problems.
- The average percentage of problems answered correctly by a student indicates how well a student did as part of our assessment. When interacting with the learning ILMDA system, the percentage was 66.2%. When interacting with the non-learning ILMDA system, it was 60.2%. This is encouraging. This hints that the learning ILMDA was able deliver more appropriate examples and problems. Combining these two observations, we see that the learning ILMDA seems to be effective (delivering more appropriate examples and problems) and efficient (delivering fewer examples and problems).
- Table 1 shows the average number of seconds spent in each section. Students interacting with the learning ILMDA spent more time than students interacting with the non-learning ILMDA. This could be due to the learning ILMDA providing more interesting and appropriate examples and problems, engaging the students to invest more time and effort into reading the materials.

Section	**Tutorial**	**Per Example**	**Per Problem**
Learning	457	64	74
Non-Learning	276	61	30

Table 1: Average time spent on each section in seconds for learning and non-learning ILMDA.

4.3 Efficiency and Effectiveness

To further measure the efficiency and effectiveness of ILMDA's machine learning capabilities, we devise a scheme as follows. We first identify a level of topical comprehension for students to attain: answering one of the most difficult problems correctly. To be effective, ILMDA should learn to guide students to reach that level. To be efficient, ILMDA should learn to guide students to reach that level with a small number of problems. Tables 2 and 3 show the results. From Table 2, for File I/O, Exceptions, and Inheritance, the learning ILMDA consistently used fewer problems to bring a student to the level of topical understanding. However, the learning ILMDA did more poorly for the Recursion topic—requiring more than 2 additional problems on average to achieve a level of topical understanding. Comparing this observation with that in the above case study, we conclude this: Although the learning ILMDA increased the percentage of correct answers from the students, it did not bring them towards answering difficult problems efficiently. We suspect that the reason for this learning failure is due to the recursion topic. Unlike the other three topics which are closely related to programming, recursion is more closely related to problem solving. Upon closer analysis, we realized that ILMDA gave students difficult questions more often than easy questions in the recursion topic, with a correlation of 0.11 between the number of times a question is given and its difficulty level, whereas ILMDA gave students easy questions more often in other topics (correlations = 0.2–0.56). We will investigate this further in our next round of evaluation.

	File I/O		**Exceptions**		**Inheritance**		**Recursion**	
Deg. Diff.	**No Learning**	**Learning**	**No Learning**	**Learning**	**No Learning**	**Learning**	**No Learning**	**Learning**
5.5	8.78	6.50	8.75	4.10	3.57	2.50	4.34	6.67
6.0	8.78	6.50	8.75	4.10	3.57	2.50	4.34	6.67
6.5	12.00	11.36	15.28	15.80	4.00	2.50	7.44	10.33
7.0	12.00	11.36	25.66	14.50	6.60	2.75	7.44	10.33
7.5					8.2	7.25	8.00	11.28
8.0					8.2	7.25	8.08	11.28
8.5							13.13	15.10
9.0							13.13	15.10

Table 2: Average number of problems given by ILMDA to a student to eventually correctly answer a problem with a certain degree of difficulty. Empty cells are due to that there are not problems of that degree of difficulty. Also, we did not include the data for problems with degree of difficulty smaller than 5.5.

Table 3 shows the percentages of failed sessions. A failed session is one in which a student quits before reaching one of the most difficult problems. We see that the results are not conclusive. That means we do not have evidence that learning ILMDA is more effective than the non-learning ILMDA. Upon closer analysis, we realize that the cases in our casebase were initially tuned to minimize the number of problems given to the students, thus prompting ILMDA to be more aggressive in giving more difficult problems but more conservative in giving easier problems. Further, in a student profile, a student usually answers questions correctly more often than not since there are more easy questions. Thus, an incorrect answer might not cause ILMDA to change its perception of the student drastically. Combining the results of Table 3 with that of the case study on recursion, we conclude this: Even though the learning ILMDA was able to increase the percentage of correct answers from the students, it was not able to engage students consistently longer for them to stay on through the set of problems.

5. Conclusions and Future Work

We have described an intelligent tutoring system that learns how to tutor students. The system, called ILMDA, uses case-based reasoning to apply pedagogical strategies, and learn new ones, learn how and when to apply them. By storing the pedagogical strategies in cases, our ILMDA could be applied to different student groups and content topics by replacing the cases. This “plug-and-play” feature makes ILMDA a potentially useful testbed and knowledge-discovery platform. Results from our initial deployment in Fall 2004 are mixed. There are encouraging indicators that the learning ILMDA is efficient. Yet, there lacks evidence for the learning ILMDA being effective. Our current and future work include the continuous deployment in Spring 2005 and more comprehensive analyses.

	File I/O		**Exceptions**		**Inheritance**		**Recursion**	
Deg. Diff.	**No Learning**	**Learning**	**No Learning**	**Learning**	**No Learning**	**Learning**	**No Learning**	**Learning**
5.5	41.94%	31.43%	66.67%	40.00%	53.33%	69.23%	40.91%	51.35%
6.0	41.94%	31.43%	66.67%	40.00%	53.33%	69.23%	40.91%	51.35%
6.5	45.16%	37.14%	70.83%	66.67%	60.00%	69.23%	43.18%	59.46%
7.0	45.16%	37.14%	75.00%	66.67%	66.67%	69.23%	43.18%	59.46%
7.5					66.67%	69.23%	45.45%	62.16%
8.0					66.67%	69.23%	45.45%	62.16%
8.5							50.00%	72.97%
9.0							50.00%	72.97%

Table 3: Percentage of failed sessions at various degrees of difficulty for the four topics. For the Inheritance-Learning column, students who quit all quit at level 5.5.

6. Acknowledgement

This work is supported in part by CSE Department, the Great Plains Software Technology Initiative, and an Enhancing Teaching and Learning Grant, all at the University of Nebraska. The authors thank L.D. Miller for his contribution to the project.

References

[1] Blank, T., Miller, L.D., Soh, L.-K., & Person, S. (2004). Case-Based Learning Mechanisms to Deliver Learning Materials. *Proc. ICMLA’2004*, Louisville, KY, USA, December 16-18, pp. 423-428.
[2] Bloom, B. S., Mesia, B. B., & Krathwohl, D. R. (1964). *Taxonomy of Educational Objectives* (two vols: *The Affective Domain & The Cognitive Domain*). New York: David McKay.
[3] Cassell, J., Annany, M., Basur, N., Bickmore, T., Chong, P., Mellis, D., Ryokai, K., Smith, J., Vilhjálmsson, H., & Yan, H. (2000). Shared Reality: Physical Collaboration with a Virtual Peer, *ACM SIG-CHI Conf. on Human Factors in Computer Systems*, April 1-6, The Hague, The Netherlands.
[4] Gertner, A. S. & VanLehn, K. (2000). ANDES: A Coached Problem-Solving Environment for Physics. *Proc. ITS’2000*, pp. 133-142.
[5] Graesser, A. C., VanLehn, K., Rosé, C. P., Jordan, P. W., and Harter, D. (2001). Intelligent Tutoring Systems with Conversational Dialogue, *AI Magazine*, 22(4):39-51.
[6] Hartman, H. (2002). Scaffolding and Cooperative Learning, *Human Learning and Instruction*, pp. 23-69. New York: City College of City University of New York.
[7] Koedinger, K. R., Anderson, J. R., Hadley, W. H., and Mark, M. A. (1997). Intelligent Tutoring Goes to School in the Big City, *Journal of Artificial Intelligence in Education,* 8(1):30-43.
[8] Kolodner, J. (1993). *Case-Based Reasoning.* Morgan Kaufmann.
[9] Soh, L.-K., Blank, T., and Miller, L. D. (forthcoming). Intelligent Agents that Learn to Deliver Online Materials to Students Better: Agent Design, Simulation, and Assumptions. In C. Chaoui, M. Jain, V. Bannore, and L. C. Jain (eds.) *Studies in Fuzziness and Soft Computing: Knowledge-Based Virtual Education,* Chapter 3, Volume 178, May 2005, pp. 49-80.
[10] Soh, L.-K., Blank, T., Miller, L. D., & Person, S. (2005). ILMDA: An Intelligent Learning Materials Delivery Agent and Simulation. *Proc. IEEE Int. EIT. Conf.*, Lincoln, NE, USA, May 22-25.
[11] Vygotsky, L. S. (1978). *Mind in Society*, Cambridge, MA: Harvard University Press.
[12] Woolf, B. P., Beck, J., Eliot, C., & Stern, M. (2002). Growth and Maturity of Intelligent Tutoring Systems: A Status Report. In Forbus, K. D. and P. J. Feltovich (2002). (Eds.) *Smart Machines in Education*, Menlo Park, CA: AAAI Press, pp. 99-144.