AIED 2013 Workshops Proceedings
Volume 4

AIED Workshop on Simulated Learners

Workshop Co-Chairs:

Gord McCalla
Department of Computer Science, University of Saskatchewan

John Champaign
RELATE, Massachusetts Institute of Technology

https://sites.google.com/site/aiedwsl/

Preface

This workshop is intended to bring together researchers who are interested in **simulated learners**, whatever their role in the design, development, deployment, or evaluation of learning systems. Its novel aspect is that it isn't just a workshop about pedagogical agents, but is also concerned about other roles for simulated learners in helping system designers, teachers, instructional designers, etc. As learning environments become increasingly complex and are used by growing numbers of learners (sometimes in the hundreds of thousands) and apply to a larger range of domains, the need for simulated learners (and simulation more generally) is compelling, not only to enhance these environments with artificial agents, but also to explore design issues using simulation that would be otherwise too expensive, too time consuming, or even impossible using human subjects. The workshop aims to be broadly integrative across all possible roles for simulated learners.

July, 2013
Gord McCalla & John Champaign.

Program Committee

Table of Contents

Using Simulated Learners and Simulated Learning Environments within a Special Education Context

Carrie Demmans Epp[1] and Alexandra Makos[2]

[1] Technologies for Aging Gracefully Laboratory (TAGlab), Dept. of Computer Science
University of Toronto, Toronto, Canada
carrie@taglab.ca
[2] Ontario Institute for Studies in Education (OISE)
University of Toronto, Toronto, Canada
alexandra.makos@mail.utoronto.ca

Abstract. The needs of special education populations require specific support to scaffold learning. The design and use of intelligent tutoring systems (ITS) has the potential to meet these needs. Difficulty in the development of these systems lies in their validation due to the ethics associated in studying learners from this population as well as the difficulty associated with accessing members of this learner group. This paper explores the use of simulated learners as a potential avenue for validating ITS designed for a special education population. The needs of special education learners are discussed. Potential avenues for employing simulated learners and simulated learning environments to test ITS, instructional materials, and instructional methods are presented. Lastly, the expansion of an educational game designed to develop emotion recognition skills in children with autism spectrum disorder is used to illustrate how simulated learning environments can be used to support the learning of these students.

Keywords: Special Education, Ethics, Simulated Learners, Simulated Learning Environments

1 Introduction

Many intelligent learning environments have been shown to help learners who belong to the general population, but few existing systems have been shown to meet the needs of those who fall under the umbrella of special education [1]. Learners in this category have highly differentiated needs that are specified in an individual education plan (IEP) [2]. Their increased need for personalization and continuous reinforcement makes the argument for augmenting their education with intelligent tutoring systems (ITS) even stronger. However, this has not been done widely.

Several factors may contribute to the lack of ITS use within special education. The lack of validation that has been performed on the systems for special education populations [1], the difficulty of integrating ITS into special education settings [3], and the difficulty of designing activities that ensure deep understanding may contribute to the

lack of ITS that support this population. The variability of learner needs presents additional challenges for system designers with respect to content development [3]. Furthermore, challenges that relate to the motivation, attitude, and social vulnerability of members of this population make it more difficult to design and validate systems. Developing systems for the special education population as a whole is difficult [4].

In addition to the above challenges, it may be difficult for designers to obtain access to a sufficiently large sample of the population to ensure that their ITS is beneficial in special education contexts. This is where the use of simulated learners and simulated learning environments can be advantageous since their use can mitigate the challenges presented by limited access to this vulnerable population and reduce the negative ethical implications of testing these systems on members of this population.

It is important to look at the research on situated learning in order to understand the achievements in best practices and lessons from research on simulated learning. Critical to this research is the combination of immersion and well-designed guidance that supports the situated understanding of learners whereby they not only have a deep understanding of the particular concepts that are being targeted, but the learners are able to then generalize and apply these learned concepts to other contexts [5]. Research shows that game-like learning through digital technologies is a viable tool across disciplines [6] and suggests that elements of game-like learning scaffold and guide learners towards a deep understanding of concepts. The on demand instruction of information that is vital to progress in the game is also important [5] and can be exploited to encourage learning. Simulations can include these elements and use stimuli to which special education populations react positively. Some stimuli that have been shown to increase student engagement include music, visual cues, and social stories [7]. Not only do these "strategies…help teachers increase engagement [but they] are vital for promoting positive outcomes for students" [7].

To support the argument for the use of simulated learners in this educational context, we first describe the characteristics and needs of this population as well as the learning environments in which they can be found. Following this, we discuss the use of ITS by special education students, which includes student interactions with agents. After laying this groundwork, we discuss the ethical implications and potential benefits to using simulated learners for validating ITS for use by special education populations. We then describe the potential uses of simulated learners and learning environments. This includes the description of an educational game, called EYEdentify, which was designed to develop emotion recognition skills in children with autism spectrum disorder (ASD). A discussion of how gaming principles and simulated environments can be further employed to expand EYEdentify for the purposes of helping scaffold learners' social interactions is provided.

2 Special Education

An introduction to the learning environments that exist in schools and the needs of learners who are classified as special education is presented. The use of agents and other forms of intelligent tutoring, within special education contexts, is then provided.

2.1 Learners and Learning Environments

These learners are either segregated into dedicated special education classrooms or integrated into classrooms whose majority population consists of learners from the general student body. Research has explored the design and integration of ubiquitous technology into special education classrooms [8], but few e-learning environments have been created to specifically support these students.

The needs and abilities of this population are highly variable, which can make generalizability hard [9]. This variability can be used to argue for the importance of personalizing students' learning materials, environments, and experiences, which is evidenced by the existence of IEP that detail the learner's specific needs and the accommodations that can be used to help the learner succeed [2]. Some of these accommodations include providing learners with additional time in order to complete tasks [1] or allowing learners to perform tasks using different modalities (e.g., oral responses rather than written ones) [2]. While these accommodations are necessary to ensuring the learner's success, it can be difficult to provide the necessary support, especially in integrated classrooms. The use of ITS that better support the individual needs of these learners could help alleviate the teacher's need to provide these supports.

2.2 Simulated Learner and Agent Use

While the use of agents within ITS used by special education populations has been studied, it appears that researchers and system developers are not simulating learners who have special needs. Nilsson and Pareto have instead used teachable agents within a special education context to help learners improve their math skills [3]. However, they experienced difficulty integrating the ITS into the classroom. Whereas, Woolf et al. were able to integrate their ITS into a classroom that had a mixed demographic: the class consisted of both low and high performing students, and of those who were low-performing, one third had a learning disability [10]. In this case, students interacted with an agent who played the role of a learning companion in order to support the learner's affective needs. It was found that this approach was especially beneficial to the low-performing students in the study, which may indicate the potential that this system holds for helping many of the learners who fall under the special education umbrella. Other work has also shown that interactions with agents within an ITS can improve or maintain learner interest and motivation [1].

3 Ethics

Given the vulnerable nature of this population, it is important that we not increase the risk that they are exposed to by introducing them to ITS or other learning techniques that have not been properly vetted since these could threaten the emotional well-being of learners or their learning success [11]. The use of simulated learners can help ensure that these systems are properly tested before we expose special education learners to them. Simulated learners can help teachers, instructional designers, and system developers meet the ethical guidelines of professional bodies by providing evidence

of the limitations and appropriateness of the instructional methods used by systems or of the system itself [12].

4 Potential for Simulated Learner Use

We foresee two potential uses for simulated learners within a special education context both of which have been explored within other contexts. The first is during the development and testing of ITS [13, 14], and the second is for teacher training [13]. Using simulated learners in these ways provides developers and instructors with access to learners in this population and prevents any potential harm that could result from experimenting with members of this population. However, it may create a false sense of the validity and usefulness of different systems and instructional techniques, especially when we lack a full understanding of the abilities and symptomology of some members of this population (e.g., those with Phelan-McDermid Syndrome).

Generalizability is difficult to perform with this population [9], but some level of generalizability is required if a system is to be used by many people. Unfortunately, current design methods, such as participatory design, fail to address how the system's use and design should change over time. Furthermore, most users are unable to predict how they will use a system until they have integrated that system into their environment [15]. Carrying these challenges into the special education domain increases their severity because of the additional communication barriers that may exist between system designers and learners with special needs [4]. While observation is a component of many design methods, the lack of access to this population when combined with the communication challenges that exist reduces the feasibility of employing many of the more traditional user-centered design techniques.

Using simulated learners could benefit system designers and developers by allowing them to evaluate a system with various members of the special education population. This could reduce demands on a vulnerable population while allowing for some level of system validation to be performed. Furthermore, the use of simulated learners would allow systems to be tested with a far greater variety of learner types in order to identify where the system may or may not be beneficial. If the system were web-based, the simulated learners could be implemented using a Selenium test suite based on behavioural models of the system's target learners.

To effectively use simulated learners in this context, it is important to create these learners using different and competing theoretical models of their behaviours and abilities. This also alleviates some of the concerns that have been expressed over the use of simulated users when testing adaptive systems [16]. The source of these models can be teachers or special education experts since their mental models might inform good stereotype-based models of learners that capture general behaviours which are grounded in the expert's classroom experience. For example, haptic feedback can be used to reinforce certain behaviours (e.g., pressing a button) in children with ASD.

However, we would argue for also including models from other sources since the above experts are in short supply and cannot provide sufficient diversity in the models to ensure that systems are adequately tested for a general special education popula-

tion. Simulated learners can be created from the cognitive models that are currently described in the educational psychology literature or through the application of educational data mining and learning analytics techniques to the logs of ITS usage where low performing and special education students were included in the classroom intervention. An example from the educational psychology literature could consider models of attention deficit hyperactivity disorder (ADHD), which include the amount of hyperactivity and inattention that a learner has, to create simulated students that behave in a way that is consistent with both the inattention that is known to affect individual outcomes and the hyperactivity that can affect the classroom environment for all students. Thus, allowing teachers to explore strategies that minimize the impact of both of the behaviours that characterize students with ADHD [17].

The diversity of models on which the simulated learners are based may help compensate for the inaccuracies that are inherent to modeling techniques, therefore, reducing the need for simulated learners to have high-fidelity cognitive models. Especially, since there is an incomplete understanding of the cognitive processes of all those who fall under the umbrella of special education, as is demonstrated by research in mathematics and learning disabilities [18].

That said, simulated learners that are based on these models could be used to validate the design of learning materials and to ensure their effectiveness or comprehension [13, 14]. Teachers could use simulated learners to test learning materials for their ability to increase learner engagement across a variety of contexts [7] before trying the materials on learners in their class. This would give teachers the opportunity to refine their teaching materials and confirm their suitability for students in the class.

Simulated learners can also be used to help prepare teachers either during preservice training or before a new school year begins when the teacher is preparing for his/her incoming students [13]. The use of agents who play different types of special education learners reduces the need to worry about the possible negative consequences that mistakes would have on learners [19]. This use of simulated learners also holds the potential to reduce teacher errors since teachers can try new techniques with the simulated learners and learn from those experiences, which may reduce the risk of their committing errors with live learners.

5 Potential for Simulated Learning Environment Use

While simulated learning environments can pose a threat to learning because of the complexity of the learning experience [20], they still hold the potential to benefit learners with special needs. Simulated environments allow learners to take risks in order to develop a deeper understanding of the situations they encounter [5]. This can increase learner awareness of potential situations that could be encountered when interacting with others. Ideally, simulated learning environments would be used to help the learner develop and transfer skills into the real world by gradually increasing the external validity of the tasks being performed.

Simulations allow system designers to ensure that the problems or activities being studied resemble those that learners experience outside of the simulation [1] and they

allow for the gradual increase in the complexity and ecological validity of tasks [21]. This means that learners can begin their learning activities in a simpler environment that is safe and progress towards more realistic situations, enabling the use of van Dam's spiral approach, where learners encounter a topic multiple times at increasing levels of sophistication [22]. This can help learners transfer their developing skills into the real world. Additionally, the use of simulations accessible on different technologies can shift learner dependence on experts to technology whereby learner use of the technology can help learners gain a sense of independence and begin to develop the skills required to expand and extend their interactions to the real world [23]. We illustrate this trajectory through a discussion of a mobile game that was designed to help children with autism spectrum disorder learn to recognize emotions.

5.1 EYEdentify: An Educational Game for Emotion Recognition

EYEdentify is a mobile application for the Android platform that is designed to develop the emotion recognition skills of children with ASD since these are lacking. Previous technologies that have tried to teach this skill to children with ASD have primarily focused on the use of videos to model emotions for the learner [24]. Current research focuses on social skill development through the use of interventions that use a video series to develop social skills by exploiting the relationship between facial expressions and emotion [4, 25]. Emotion recognition research suggests the most important features of the face necessary to correctly identify emotions are the eyes and the mouth [26]. Considering research on social skill development and advancements in portable technology, a mobile application that can support anytime-anywhere support to children with this deficit is timely.

EYEdentify is a game that uses a basic learner model to provide a flexible intervention in the form of an engaging game. It has an open learner model that can show the child's progress to parents, caregivers, teachers, and specialists. The first version of this application incorporates four emotions (i.e., happy, sad, frustrated, and confused) into a matching game that progresses through different levels (Fig. 1). There are three types of images that are used in this game to help scaffold the child's learning: cartoon robot faces, real faces that are superimposed on robot faces, and photographs of actual faces. The cartoon robot faces are designed to emphasize the eyes and mouth. The superimposed faces are designed to activate the child's knowledge of focusing on the eyes and mouth to correctly identify the displayed emotions while maintaining the scaffold of the robot head. The photograph of an individual making a particular expression is used to activate the knowledge from the previously superimposed images to correctly identify the emotions. Difficulty increases with respect to the type of emotion that is incorporated into game play and the types of images that are used. Positive feedback is provided to the child throughout the game to encourage continuous play. The game also has a calming event that is triggered by the accelerometer when the mobile device is shaken aggressively. The calming event increases the volume of the music that is being played and prompts the child to count to ten. The child is then asked whether or not s/he wants to continue playing the game.

Fig. 1. The gameplay screen with the correct responses identified (surrounded in green).

The mobile application provides the ability to customize game play by incorporating personalized feedback and images. Users can customize feedback by typing a comment and recording an audio message before adding this feedback to the schedule. Image customization uses the front camera of the device to capture individuals parroting the facial expression represented on the robot prompt. As children progress through the levels, they are rewarded with parts to assemble their own robot.

The current version focuses on developing emotion recognition skills for four of the fifteen basic emotions identified by Golan et al. [25]. The addition of the remaining eleven emotions could be used to extend game play. Currently, the mobile application is functional; however, more emotions are being incorporated and iOS versions are being developed before releasing EYEdentify on Google Play and the App Store.

5.2 Expanding EYEdentify to Include a Simulated Learning Environment

The expansion of EYEdentify to include a simulated learning environment draws on Csikszentmihalyi's definition of flow and research on gaming. Flow is described as the experience of being fully engaged in an activity where an individual is "so involved…that nothing else seems to matter" [27]. This is derived from activities where a person's skills are matched to the challenges encountered [27]. For learners, this means that they will be in a mental state that keeps them motivated to stay involved in a particular activity. Research in gaming and game design incorporates these psychological underpinnings whereby elements of a game seek to cultivate and support the player's active engagement and enhanced motivation [28]. In educational games, these elements are employed to scaffold learning just-in-time and provide instructors with the ability to adapt the system to the specific needs of the learner [29].

EYEdentify currently provides a matching game with rewards that are self-contained within the mobile application. Preliminary trials indicate that it keeps learners involved in the activity of identifying emotions for long periods of time. These

trials parallel the findings of research that used a video intervention program known as “The Transporters” to develop the social skills of children with ASD [30].

EYEdentify’s game play can be expanded into simulated learning environments to move players beyond the acquisition of emotion recognition skills toward the development of social skills. In creating game-based simulations for learners to use, the capacity to scaffold their learning within game play and support the development of transferable skills to the real-world increases.

There are several ways to expand game play into a simulated learning environment. All possibilities would require the mastery of basic emotion recognition and could involve levels of progressive difficulty that incorporates these emotions into depictions of social situations. The front camera of the mobile device could be used to scaffold the recognition of emotions by way of augmented reality, as could the recent introduction of Google glass. Avatars that represent individuals from the learner’s day-to-day life could be used by learners to practice particular social situations. Additionally, game play could incorporate depictions of situations that model different social interactions. This could then be incorporated with a Sims-like environment where learners would have to identify the emotion of the character that they are interacting with and demonstrate the appropriate behaviour or emotional response. Specific to keeping learners engaged, the addition of an emotion recognition system that can detect the learner’s emotion from the front camera and keep track of their emotion when playing the game to determine that learner’s level of engagement would be useful. Through the development of these possibilities, EYEdentify has the potential to enhance learners’ emotion recognition and social skill development in a way that enables the learner to transfer these skills to their day-to-day encounters.

6 Conclusion

The use of simulated learners and learning environments within special education contexts holds great potential for improving the quality and applicability of ITS use by members of this population. Simulated learners can be used to test learning materials, learning methods, and ITS to ensure their appropriateness for the members of this population, who have highly variable needs. The use of simulated learners and learning environments can be further exploited for teacher training. In addition to this use, simulated learning environments can be used to help learners who have been classified as having special needs to transfer their knowledge and skills to their everyday lives. The potential for members of this population to use simulated learning environments was illustrated through an example of an educational game, EYEdentify, that is used to help children with autism spectrum disorder improve their ability to recognize emotions. The described potential expansions of this game show how different approaches to simulated learning environments and the use of augmented reality can be used to help learners transition between the simulated world and the one they encounter every day.

References

1. Bruno, A., Gonzalez, C., Moreno, L., Noda, M., Aguilar, R., Munoz, V.: Teaching mathematics in children with Down's syndrome. In: Artificial Intelligence in Education (AIED). Sydney, Australia (2003).
2. Government of Ontario: Individual Education Plans Standards for Development, Program Planning, and Implementation. Ontario Ministry of Education (2000).
3. Nilsson, A., Pareto, L.: The complexity of integrating technology enhanced learning in special math education - a case study. In: 5th European Conference on Technology Enhanced Learning on Sustaining TEL: from Innovation to Learning and Practice. pp. 638–643. Springer-Verlag, Berlin, Heidelberg (2010).
4. Wainer, A.L., Ingersoll, B.R.: The use of innovative computer technology for teaching social communication to individuals with autism spectrum disorder. Research in Autism Spectrum Disorders. 5, 96–107 (2011).
5. Assessment, equity, and opportunity to learn. Cambridge University Press, Cambridge ; New York (2008).
6. Jackson, L.A., Witt, E.A., Games, A.I., Fitzgerald, H.E., von Eye, A., Zhao, Y.: Information technology use and creativity: Findings from the Children and Technology Project. Computers in Human Behavior. 28, 370–376 (2012).
7. Carnahan, C., Basham, J., Musti-Rao, S.: A Low-Technology Strategy for Increasing Engagement of Students with Autism and Significant Learning Needs. Exceptionality. 17, 76–87 (2009).
8. Tentori, M., Hayes, G.: Designing for Interaction Immediacy to Enhance Social Skills of Children with Autism. Ubiquitous Computing (Ubicomp). pp. 51–60. ACM, Copenhagen, Denmark (2010).
9. Moffatt, K., Findlater, L., Allen, M.: Generalizability in Research with Cognitively Impaired Individuals. In: Workshop on Designing for People with Cognitive Impairments, ACM Conference on Human Factors in Computing Systems (CHI). ACM, Montreal, Canada (2006).
10. Woolf, B.P., Arroyo, I., Muldner, K., Burleson, W., Cooper, D.G., Dolan, R., Christopherson, R.M.: The Effect of Motivational Learning Companions on Low Achieving Students and Students with Disabilities. In: Aleven, V., Kay, J., and Mostow, J. (eds.) Intelligent Tutoring Systems (ITS). pp. 327–337. Springer Berlin Heidelberg, Berlin, Heidelberg (2010).
11. Cardon, T.A., Wilcox, M.J., Campbell, P.H.: Caregiver Perspectives About Assistive Technology Use With Their Young Children With Autism Spectrum Disorders. Infants & Young Children. 24, 153–173 (2011).
12. Code of Fair Testing Practices in Education. Washington, D.C.: Joint Committee on Testing Practices, American Psychological Association (1988).
13. VanLehn, K., Ohlsson, S., Nason, R.: Applications of Simulated Students: An Exploration. International Journal of Artificial Intelligence in Education (IJAIED). 5, 135–175 (1996).
14. Mertz, J.S.: Using A Simulated Student for Instructional Design. International Journal of Artificial Intelligence in Education (IJAIED). 8, 116–141 (1997).

15. Dawe, M.: Design Methods to Engage Individuals with Cognitive Disabilities and their Families. In: the Science of Design Workshop, ACM Conference on Human Factors in Computing Systems (CHI) (2007).
16. Paramythis, A., Weibelzahl, S., Masthoff, J.: Layered Evaluation of Interactive Adaptive Systems: Framework and Formative Methods. User Modeling and User-Adapted Interaction (UMUAI). 20, 383–453 (2010).
17. Rogers, M., Hwang, H., Toplak, M., Weiss, M., Tannock, R.: Inattention, working memory, and academic achievement in adolescents referred for attention deficit/hyperactivity disorder (ADHD). Child Neuropsychology. 17, 444–458 (2011).
18. Geary, D.C.: Mathematics and Learning Disabilities. J Learn Disabil. 37, 4–15 (2004).
19. Ogan, A., Finkelstein, S., Mayfield, E., D'Adamo, C., Matsuda, N., Cassell, J.: "Oh dear stacy!": social interaction, elaboration, and learning with teachable agents. In: ACM Conference on Human Factors in Computing Systems (CHI). pp. 39–48. ACM, New York, NY, USA (2012).
20. Moreno, R., Mayer, R., Lester, J.: Life-Like Pedagogical Agents in Constructivist Multimedia Environments: Cognitive Consequences of their Interaction. In: World Conference on Educational Multimedia, Hypermedia and Telecommunications (EDMEDIA). pp. 776–781 (2000).
21. Henderson-Summet, V., Clawson, J.: Usability at the Edges: Bringing the Lab into the Real World and the Real World into the Lab. In: Workshop on Usability in the Wild, International Conference on Human-Computer Interaction (INTERACT) (2007).
22. Van Dam, A., Becker, S., Simpson, R.M.: Next-generation educational software: why we need it & a research agenda for getting it. EDUCAUSE Review. 40, 26–43 (2007).
23. Stromer, R., Kimball, J.W., Kinney, E.M., Taylor, B.A.: Activity schedules, computer technology, and teaching children with autism spectrum disorders. Focus on Autism and Other Developmental Disabilities. 21, 14–24 (2006).
24. DiGennaro Reed, F.D., Hyman, S.R., Hirst, J.M.: Applications of technology to teach social skills to children with autism. Research in Autism Spectrum Disorders. 5, 1003–1010 (2011).
25. Golan, O., Ashwin, E., Granader, Y., McClintock, S., Day, K., Leggett, V., Baron-Cohen, S.: Enhancing Emotion Recognition in Children with Autism Spectrum Conditions: An Intervention Using Animated Vehicles with Real Emotional Faces. Journal of Autism and Developmental Disorders. 40, 269–279 (2009).
26. Erickson, K., Schulkin, J.: Facial expressions of emotion: A cognitive neuroscience perspective. Brain and Cognition. 52, 52–60 (2003).
27. Csikszentmihalyi, M.: Flow: The Psychology of Optimal Experience. Harper Perennial Modern Classics (2008).
28. Tom Chatfield: 7 ways games reward the brain | Video on TED.com. (2010).
29. Fernández López, Á., Rodríguez Fórtiz, M.J., Noguera García, M.: Designing and Supporting Cooperative and Ubiquitous Learning Systems for People with Special Needs. In: Confederated International Workshops and Posters on the Move to Meaningful Internet Systems: ADI, CAMS, EI2N, ISDE, IWSSA, MONET, OnToContent, ODIS, ORM, OTM Academy, SWWS, SEMELS, Beyond SAWSDL, and COMBEK. pp. 423–432. Springer-Verlag, Berlin, Heidelberg (2009).
30. The Transporters. Changing Media Development Ltd (2006).

Simulated Students, Mastery Learning, and Improved Learning Curves for Real-World Cognitive Tutors

Stephen E. Fancsali, Tristan Nixon, Annalies Vuong, and Steven Ritter

Carnegie Learning, Inc.
Frick Building, Suite 918
437 Grant Street, Pittsburgh, PA 15219

{sfancsali, tnixon, avuong, sritter}
@carnegielearning.com

Abstract. We briefly describe three approaches to simulating students to develop and improve intelligent tutoring systems. We review recent work with simulated student data based on simple probabilistic models that provides important insight into practical decisions made in the deployment of Cognitive Tutor software, focusing specifically on aspects of mastery learning in Bayesian Knowledge Tracing and learning curve analysis to improve cognitive (skill) models. We provide a new simulation approach that builds on earlier efforts to better visualize aggregate learning curves.

Keywords: Knowledge tracing, learning curves, student modeling, Cognitive Tutor, simulation, simulated students, mastery learning

1 Introduction

There are at least three general approaches to simulating students for the purposes of improving cognitive (skill) models and other features of intelligent tutoring systems (ITSs). One approach, generally connoted in discussions of "simulated" students or learners, employs aspects of cognitive theory to simulate students' learning and progression through ITS problems (e.g., via machine learning or computational agents like SimStudent [2]). Another class of simulations makes use of relatively simple probabilistic models to generate response data (i.e., Bayesian Knowledge Tracing [BKT] [1]) intended to represent a (simulated) student's evolving performance over many practice attempts. Third, there are data-driven approaches that do not easily fit into either of the first two categories.

In this work, we explicate and provide examples of each approach and briefly describe Carnegie Learning's Cognitive Tutors (CTs) [3]. We then focus on the second approach and review recent work on simulations of student learning with simple probabilistic models. These simulation studies provide novel insights into a variety of features of CTs and their practical deployment.

CTs implement mastery learning; mathematics content is adaptively presented to students based upon whether the tutor has judged that a student has mastered particular skills. Mastery is assessed according to whether the tutor judges that the probability that a student has mastered a particular skill exceeds a set threshold. We review a simulation study that provides for best and worst-case analyses (when "ground truth" characteristics of simulated learner populations are known) of tutor skill mastery judgment and efficient student practice (i.e., adaptively providing students with opportunities to practice only those skills they have not mastered). This study not only provides justification for the traditionally used 95% probability threshold, but it also illuminates how the threshold for skill mastery can function as a "tunable" parameter, demonstrating the practical import of such simulation studies.

Finally, learning curves provide a visual representation of student performance on opportunities to practice purported skills in an ITS. These representations can be used to analyze whether a domain has been appropriately atomized into skills. If opportunities correspond to practice for a single skill, we expect to see a gradual increase in the proportion of correct responses as students get more practice opportunities. If, for example, the proportion of students responding correctly to an opportunity drastically decreases after three practice opportunities, it seems unlikely that the opportunities genuinely correspond to one particular skill. Turning to the third, data-driven approach to simulating students, we provide a new method to visualize aggregate learning curves to better drive improvements in cognitive (skill) models used in CTs, This approach extends recent work that explores several problems for utilizing learning curves aggregated over many students to determine whether practice opportunities correspond to a single skill.

2 Cognitive Tutors

CTs are ITSs for mathematics curricula used by hundreds of thousands of K-12 and undergraduate students every year. Based on cognitive models that decompose problem solving into constituent knowledge components (KCs) or skills, CT implements BKT to track student skill knowledge. When the system's estimate of a student's knowledge of any particular skill exceeds a set threshold, the student is judged to have mastered that skill. Based on the CT's judgment of skill mastery, problems that emphasize different skills are adaptively presented so that the student may focus on those skills most in need of practice.

3 Three Approaches to Simulating Learners

There are at least three general simulation methods used to model student or learner performance. One simulation strategy, based on cognitive theories such as ACT-R [4], explicitly models cognitive problem-solving processes to produce rich agent-based simulated students. The SimStudent project ([2], [5]), for example, has been developed as a part of a suite of authoring tools to develop curricula for CTs, called Cognitive Tutor Authoring Tools (CTAT) [6]. SimStudent learns production rules

from problem-solving demonstrations (e.g., an author providing simple demonstrations of problem solutions or via ITS log data). These human-interpretable production rules correspond to KCs that comprise cognitive models vital to CTs. SimStudent aims to simplify development of new CT material by automating the discovery of KC models in new domains via a bottom-up search for skills that potentially explain the demonstrations.

Second, there are numerous probabilistic methods that model task performance as a function of practice, according to various task and learner-specific parameters. One may instantiate numerous such models, with varying parameters, and sample from the resulting probability distributions to obtain simulated performance data for an entire hypothetical learner population.

One common example is a Hidden Markov Model (HMM) with two latent and two observable states, that can serve as a generative BKT model, using parameters specified according to expert knowledge or inferred by a data-driven estimation procedure. Two hidden nodes in the HMM represent "known" and "unknown" student knowledge states. In practice, of course, student knowledge is latent. Simulated students are assigned to a knowledge state according to BKT's parameter for the probability of initial knowledge, $P(L_0)$, and those in the "unknown" state transition to the "known" state according to the BKT parameter for the probability of learning or transfer, P(T). Simulated, observed responses are then sampled according to BKT parameters that represent the probability of student guessing, P(G) (i.e., responding correctly when in the unknown state) and slipping, P(S) (i.e., responding incorrectly when in the known state), depending upon the state of student knowledge at each practice opportunity.

Contrary to her real-world epistemological position, simulations generally allow an investigator to access the student's knowledge state at each simulated practice opportunity. This allows for comparisons between the "ground truth" of skill mastery and any estimate derived from resulting simulated behavior. Clearly, richer cognitive agents, such as SimStudent, provide a more complete picture of the student's cognitive state at any point.

Simpler probabilistic models represent student knowledge of a skill with a single state variable, so they correspondingly scale better to larger scale simulations of whole populations. While a probabilistic model only requires a reasonable distribution over initial parameters, richer cognitive models may require training on a great deal of detailed, behavioral or demonstration data. Nevertheless, cognitive model-based simulations allow us to investigate issues like timing (i.e., response latency), sensitivity to input characteristics, and error patterns in learner responses.

There are many cases in which a relatively simple probabilistic model may be of utility, despite its impoverished nature. A simplistic representation of student knowledge provides an ideal situation to test the performance and characteristics of inference methods using data from a known generating process and parameters. One might, for example, compare the point at which simulated students acquire knowledge of a skill to the point at which the CT judges the student to have mastered the skill. The approach thus allows for students of "best" and "worst" case scenarios with respect to the relationship between how the CT models students and the actual make up

of (simulated) student populations. We can better understand the dynamics of the student sub-populations we inevitably face in practice by simulating data from diverse sub-populations, the make up of which we can specify or randomize in various ways. Furthermore, we can simulate student performance (sometimes augmenting available empirical data) both with and without mastery learning (i.e., students being removed from a population because they have mastered a skill) on learning curves constructed from aggregate data.

Previous work [7] explored a third, data-driven simulation method that "replays" empirical student performance data through CT in order to estimate the impact of a change in BKT parameters in a more substantive way. For each KC that occurred in a given problem, we sampled the next observed response on that KC from the sequence actually observed from a real student. These responses would then drive updates to CT's cognitive model, knowledge tracing, and the problem-selection mechanism. If more data were required than were observed for a given student, further observations were sampled from a BKT model initialized to the state inferred from the student's actions thus far. By repeating this process for all students in the observed data set, we could obtain estimates of the number of problems students would be expected to complete if a change to the cognitive model were implemented.

This method has the advantage of preserving characteristics of real student data rather than resorting to a theoretical model of student performance. However, it does make several assumptions about the reproducibility of that behavior under the hypothesized changes. Specifically, it assumes that the observed sequence of correct/incorrect responses would not change even given a different selection of problems, potentially emphasizing different KCs. This assumption may be justified if we believe we have complete coverage of all KCs relevant to the task in question in the cognitive model and that all KCs are truly independent of each other.

While simulation methods based on rich cognitive theory and data-driven re-play of empirical data provide many opportunities for future research, we focus in this paper on simple, probabilistic simulations in the context of the BKT framework.

4 Substantive Measures of Efficient Student Practice

Before we discuss how the BKT mastery threshold probability functions as a "tunable" parameter in an ITS like the CT, we provide "substantive" quantification of goodness of fit of cognitive/skill models for CTs beyond mere RMSE of prediction (i.e., beyond the extent to which models can predict whether students will respond correctly to particular practice opportunities) [8-11]. New error or goodness of fit measures are countenanced in terms of efficient student practice, based on the number of practice opportunities (i.e., "over-practice" or "under-practice") we might expect a student to experience in a CT. Over-practice refers to the continued presentation of new practice opportunities, despite the student's mastery or knowledge of the relevant KC.[1] Student "under-practice" instances are those in which a student has yet to

[1] One exception is an experimental study [11] that reports increased efficiency by deploying parameters estimated using a data mining method called Learning Factors Analysis (LFA).

achieve knowledge of a KC, and yet the mastery learning system has judged the student as having mastered it, ending the presentation of further learning opportunities. From estimates of expected under- and over-practice, one can calculate other meaningful measures of students gains and losses, such as time saved or wasted.

Some of this work [8, 9] uses empirical data to estimate the extent of under-practice and over-practice we might expect students to experience. Specifically, the expected numbers of practice opportunities it takes a student to reach mastery when parameters are individualized per student are compared to the expected practice when a single (population) set of parameters is used to assess all students. One individualization scheme used to study under and over-practice estimates all four BKT parameters, per student, from response data over all relevant skills (i.e., each student receives one individualized quadruple of BKT parameters for all KCs) [8]. Another approach [9] only individualizes P(T) for each student based on both per-student and per-skill components estimated from observed data [12]. Both individualization schemes provide for substantive gains (compared to using a set of population parameters to assess all students' progress to mastery) in the efficiency of practice (i.e., fewer expected under and over-practice opportunities) as well as better prediction performance judged, in the standard way, by a metric like RMSE.

5 Idealized Performance of Mastery Learning Assessment

Now we address how BKT performs with respect to efficiency of practice in idealized cases in which the composition of student (sub-) populations is known. Simulation studies can shed light on how BKT performs when mastery learning parameters used by the CT run-time system exactly match those of the generating model (i.e., the best case), and in worst cases in which student parameters either maximally differ from mastery learning parameters or vary at random for each student.

Recent work addresses these issues by adopting a probabilistic simulation regime [10]. Since we can track the point at which a simulated student acquires knowledge of a skill, we are able to compare this to the opportunity at which the mastery learning system first judges it to be acquired. Simulations were run for fourteen skills, a subset of those found by [13] to be representative of a substantial portion of skills in deployed CT curricula, across thousands of virtual students.

Even in idealized, best case scenarios (i.e., when parameters used to assess skill mastery perfectly match simulated student data-generating parameters), for most skills and a large number of students, we expect there to be one to four "lagged" practice opportunities between the point at which simulated students transition to mastery and the point at which the BKT run-time system judges mastery. That is, in general, even when a student population is modeled "perfectly," and given the traditional setting of the probability threshold for mastery at 95%, most students should be expected to see at least a few opportunities beyond the point of skill acquisition. That some "over-practice" may be inevitable provides a relevant context within which to consid-

---

Efficiency is operationalized as decreased time required to work through material in the Geometry CT without decreasing overall learning.

er empirically driven results of [8, 9]. Although a certain amount of lag may be inherent in the nature of BKT, we seek to establish a range for the "acceptable" lag, and to better appraise efficiency of practice [10].

6 Mastery Learning Threshold as a "Tunable" Parameter

In addition to lagged opportunities and over-practice, situations in which students under-practice skills are important to consider. Given the possibly inevitable lag between skill acquisition and mastery judgment, simulations [10] have also been used to explore how the mastery probability threshold might be "tuned" to influence the trade-off of over-practice and under-practice experienced by students in mastery learning systems like CTs.

Pre-mature mastery judgments can lead, for example, to students being moved along by the CT to problems that emphasize new KCs without having mastered prerequisite KCs. Other things held equal, simulations in [10] provide that pre-mature mastery judgment is more likely to occur in worst-case scenarios, when mastery-learning parameters do not match parameters for sub-populations of simulated students.

Simulations in [10] also establish that the mastery-learning threshold can function as a tuning parameter, partially governing the trade-off between the expected proportion of students pre-maturely judged to have reached skill mastery and the number of over-practice opportunities they are likely to experience. As the threshold probability is increased, the proportion of students assessed as having pre-maturely mastered skills decreases while the proportion of those that are exposed to practice opportunities after skill acquisition increases (along with the number of lagged and over-practice opportunities, i.e., those beyond a calculated acceptable lag they experience).

The results of [10] show that the traditionally used 95% threshold seems to provide for a "conservative" tutor that is more likely to present opportunities after skill acquisition rather than under-practice. Depending upon course design and practice regimes, the mastery-learning threshold might be manipulated to important, practical effect. For example, pre-mature mastery judgments might be acceptable in larger numbers when there is a mixed-practice regime that will allow students to practice KCs later in the curriculum.

7 Using Simulations to Illuminate Learning in Learning Curves

Learning curves provide a visual representation of student performance over opportunities to practice skills. For each (purported) skill, we construct a learning curve by plotting opportunities (i.e., 1st, opportunity, 2nd opportunity, and so on) on the x-axis and the proportion of students that provide correct responses at each opportunity on the y-axis. Aggregated over real-world student practice opportunity data, such

curves provide means by which to visually[2] inspect whether opportunities genuinely correspond to practice of one particular skill. If opportunities correspond to one particular skill, we expect a gradual increase in the proportion of students that respond correctly with increasing practice. Generally, for well-modeled skills (and a variety of other cognitive tasks), it is thought that such a plot should correspond roughly to a power law function (i.e., the power law of practice [14]), though this point is not without controversy [15]. Recent research [16-17] demonstrates how some aggregate learning curves can distort the picture of student learning. Aggregate learning curves may, for example, appear to show no learning, when, in fact all students are learning at different rates. Others may provide for a small rise in probability of correct response initially but then "drop," as if students were forgetting, even when individual students are consistently mastering their skills.

The learning curve of Fig. 1 illustrates aspects of both problems, with a relatively flat portion, followed by a drop, after a small increase in probability correct from its initial value. The red line, representing the size of the student population at each opportunity, illustrates that BKT is determining that students are mastering the skill relatively quickly.

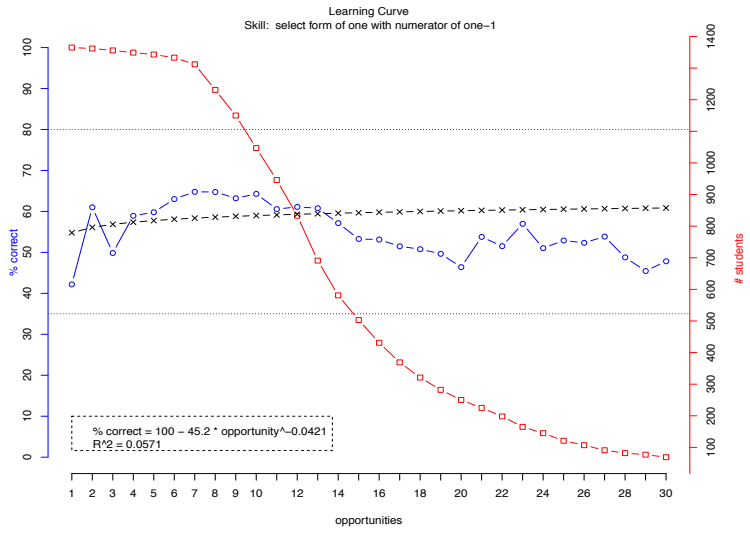


Fig. 1. Empirical Learning Curve for Skill "Select form of one with numerator of one"; the blue line represents empirical data plotted as percentage of correct responses, and the black line represents a fitted power function. The red line provides the size of the student population.

Two ways to re-visualize problematic, aggregated learning curves have been suggested [16]. One is to provide multiple learning curves (on the same plot) for individual

[2] Developers at Carnegie Learning also deploy several data-driven heuristics (that correspond to various visual features of learning curves) to analyze our large portfolio of KCs (i.e., several thousand KCs over several mathematics CT curricula) and observed student data to draw attention to those KCs that may require revision in our deployed cognitive models.

"segments" of students based upon how many opportunities students, in observed data, take to reach the mastery learning threshold for a skill. Such segmented learning curves are provided with the same x-axis and y-axis as standard learning curves (i.e., practice opportunity count on the x-axis and, e.g., percentage of student correct response on the y-axis).

The second approach suggested by [16] has the analyst plot "mastery-aligned" learning curves. In such learning curves, students are also segmented according to the number of opportunities required to reach mastery, but the end-point of the x-axis corresponds to the opportunity at which students' reach mastery (*m*) and moving left along the x-axis corresponds to the opportunity before mastery (*m-1*), the second to last opportunity before mastery (*m-2*), and so on.

Further work [17] provides a mathematical explanation, along with proof-of-concept simulation studies based on HMMs, for the dynamics of aggregate learning curves to explain how both mastery learning itself and differing student subpopulations, when aggregated, can contribute to learning curves that do not show learning (or manifest other peculiar, possible deceptive, phenomena like "negative" learning).

We illustrate an alternative to [16] by providing a method that relies on probabilistic simulation to construct aggregate learning curves that better represent learning in empirical student data. Specifically, we "pad" empirical data for student skill opportunities with simulated data to mask the effects of attrition due to mastery learning and possibly "reveal" student learning. Student opportunity data are generated with the same parameters used to track student progress and the probability of student knowledge estimated at the point at which the student crossed the mastery threshold. Such simulations provide us data after a student no longer receives practice opportunities for a particular skill because they have been judged as having achieved mastery.

For the aggregate learning curve of Fig. 1, the "padded" learning curve is Fig. 2. The fitted power-law slope parameter decreases from -0.042 to -0.363 (indicating more learning), and the goodness-of-fit of the power law function (R^2) increases from 0.0571 to 0.875. We apply the method to 166 skills identified[3] by [16] as possibly problematic in the Cognitive Tutor Algebra I (CTAI) curriculum. We find an improvement (i.e., power-fit parameter decreases from above -0.1 to below -0.1, a criterion deployed by [16]) for 98 skills (59%). While this method provides an improved visualization and understanding of fewer skills than the disaggregation procedures suggested by [16], this seems to provide evidence of the great extent to which mastery learning attrition obfuscates evidence for student learning.

Importantly, our simulation method does not eliminate the early dip in the learning curve at opportunity 3 when little attrition has yet to take place, but only masks the effects of attrition due to mastery learning. Such an approach focuses largely on a better representation or visualization of the "tail" of aggregate learning curves. This

[3] These skills were chosen because the over-whelming majority of students are judged to eventually master them (i.e., CT "thinks" the students are learning); they are not pre-mastered (i.e., $P(L_0) < 0.95$); they do not show learning in their aggregate learning curve (i.e., power-law fit parameter > -0.1); aggregate learning curves for these skills do not have multiple maxima; and we have data for at least 250 students for these skills [16].

allows us to focus on other features of the learning curve that may indicate ill-modeled KCs in a cognitive model, software bugs, and other possible problems.

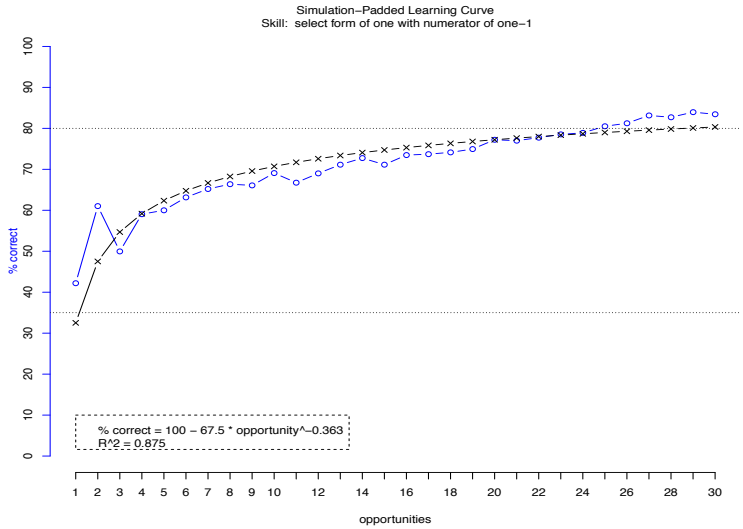


Fig. 2. Simulation-Padded Learning Curve for Skill "Select form of one with numerator of one"

8 Summary

We briefly reviewed several methods for simulating learners. We focused on ways in which simple probabilistic models, in contrast to methods that rely on rich cognitive theory, can be used to generate student performance data to help drive practical decision-making about CT deployment, focusing first on the mastery threshold probability of BKT as a tunable parameter to determine aspects of efficient practice. Then we introduced a new method for visualizing aggregate learning curves that relies on both empirical and simulated data that helps to mask the bias introduced by mastery learning attrition. Future work will further explore these methods, new simulation regimes, and their practical import.

References

1. Corbett, A.T., Anderson, J.R.: Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. User-Modeling and User-Adapted Interaction 4, 253–278 (1995)
2. Matsuda, N., Cohen, W.W., Sewall, J., Koedinger, K.R.: Applying Machine Learning to Cognitive Modeling for Cognitive Tutors. Human-Computer Interaction Institute, Carnegie Mellon University. Paper 248 (CMU-ML-06-105) (2006)

3. Ritter, S., Anderson, J.R., Koedinger, K.R., Corbett, A.T.: Cognitive Tutors: Applied Research in Mathematics Education. Psychonomic Bulletin & Review 14, 249–255 (2007)
4. Anderson, J.R.: Rules of the Mind. Erlbuam, Hillsdale, NJ (1993)
5. Matsuda, N., Cohen, W.W., Sewall, J., Lacerda, G., Koedinger, K.R.: Evaluating a Simulated Student Using Real Students Data for Training and Testing. In: Proceedings of the International Conference on User Modeling (LNAI 4511), pp. 107–116 (2007)
6. Aleven, V., McLaren, B.M., Sewall, J., Koedinger, K.R.: The Cognitive Tutor Authoring Tool (CTAT): Preliminary Evaluation of Efficiency Gains. In: Proceedings of the 8th International Conference on Intelligent Tutoring Systems, pp. 61-70 (2006)
7. Dickison, D., Ritter, S., Nixon, T., Harris, T., Towle, B., Murray, R.C., Hausmann, R.G.M.: Predicting the Effects of Skill Model Changes on Student Progress. In: Proceedings of the 10th International Conference on Intelligent Tutoring Systems (Part II), pp. 300-302 (2010)
8. Lee, J.I., Brunskill, E.: The Impact of Individualizing Student Models on Necessary Practice Opportunities. In: Proceedings of the 5th International Conference on Educational Data Mining, pp. 118–125 (2012)
9. Yudelson, M.V., Koedinger, K.R.: Estimating the Benefits of Student Model Improvements on a Substantive Scale. In: Proceedings of the 6th International Conference on Educational Data Mining (2013)
10. Fancsali, S., Nixon, T., Ritter, S.: Optimal and Worst-Case Performance of Mastery Learning Assessment with Bayesian Knowledge Tracing. In: Proceedings of the 6th International Conference on Educational Data Mining (2013)
11. Cen, H., Koedinger, K., Junker, B.: Is Over-Practice Necessary? – Improving Learning Efficiency with the Cognitive Tutor through Educational Data Mining. In: Proceedings of the 13th International Conference on Artificial Intelligence in Education, pp. 511–518 (2007)
12. Yudelson, M.V., Koedinger, K.R., Gordon, G.J.: Individualized Bayesian Knowledge Tracing Models. In: Proceedings of the 16th International Conference on Artificial Intelligence in Education (2013)
13. Ritter, S., Harris, T.K., Nixon, T., Dickison, D., Murray, R.C., Towle, B.: Reducing the Knowledge Tracing Space. In: Proceedings of the 2nd International Conference on Educational Data Mining, pp. 151-160 (2009)
14. Newell, A., Rosenbloom, P.S.: Mechanisms of Skill Acquisition and the Law of Practice. In: Anderson, J.R. (ed.) Cognitive Skills and Their Acquisition, pp. 1-55. Erlbaum, Hillsdale, NJ (1981)
15. Heathcote, A., Brown, S.: The Power Law Repealed: The Case for an Exponential Law of Practice. Psychonomic Bulletin & Review 7, 185-207 (2000)
16. Murray, R.C., Ritter, S., Nixon, T., Schwiebert, R., Hausmann, R.G.M., Towle, B., Fancsali, S., Vuong, A.: Revealing the Learning in Learning Curves. In: Proceedings of the 16th International Conference on Artificial Intelligence in Education, pp. 473-482 (2013)
17. Nixon, T., Fancsali, S., Ritter, S.: The Complex Dynamics of Aggregate Learning Curves. In: Proceedings of the 6th International Conference on Educational Data Mining (2013)

Exploring through Simulation the Effects of Peer Impact on Learning

Stephanie Frost[1] and Gord McCalla[1]

ARIES Lab, Dept. of Computer Science, U. of Saskatchewan, Saskatoon, Canada
stephanie.frost@usask.ca, mccalla@cs.usask.ca

Abstract. Simulation modelling helps designers to keep track of many possible behaviours in a complex environment. Having a technique to simulate the effect of peer impact on learning allows designers to test the social effects of their educational software. We implement an agent-based simulation model based on the ecological approach (EA) architecture [9]. The model considers learner attributes, learning object attributes and two styles of peer impact to explore the effects when learners are either positively or negatively impacted by high achieving peers. In this study, we observe different patterns of behaviour based on the style of peer impact and by limiting simulated learners' access to information (the EA metadata). Gaining understanding of these patterns will inform our future work on recommending sequences of learning objects (LOs).

Keywords: simulated learning environments, simulated learners, ecological approach, instructional planning

1 Introduction

Before taking an action in a learning environment, it is important for an intelligent tutoring system (ITS) to have some way of estimating the likelihood that the action will be successful, i.e. that it will benefit the learner(s) involved. To compute such an estimate, there are many dimensions to consider such as: the nature of the content being learned, the pedagogical style of the environment, learning goals, individual learner characteristics, and social factors such as how a learner's own performance can be influenced by knowledge of peer performance. Such complexity is often managed through the use of models.

Simulation modelling can be used by instructional developers for testing their systems; this was identified by VanLehn, Ohlsson and Nason [11] in a survey of possible uses of simulated students. One example is SimStudent by Matsuda et al. [8] which can be used by designers to explore through simulation the effects of various decisions on cognitive tutor design. Whether a model is used "internally" (by an ITS to compute the next action) or "externally" (to evaluate a system design), a challenge remains: How does the model estimate the amount of learning that occurs when a learner interacts with a Learning Object (LO)? In particular, we wanted to explore the impact on learning when learner performance is influenced by the performance of peers. Some learners may become encouraged

when observing high peer achievement and perform even better than they would have otherwise. Other learners might become discouraged in the same situation and perform even worse. Having a technique to simulate the effects of peer performance would allow instructional developers to test social effects of their designs. In this paper, we use simulation to explore the behaviours exhibited by two different reactions to peer impact.

We describe our approach in Section 2, followed by the simulation study in Section 3. It is possible to simulate many different kinds of educational software in the ecological approach (EA) architecture [5], and then test the simulation under various conditions to get insight into issues the designer is interested in. Because our model is implemented in the EA architecture, our approach for modelling peer impact can be used across many different styles of learning systems. The data to feed our simulation is synthetic, but could, itself, be modelled on data extracted from actual learner behaviour [5]. We follow with a description of ongoing research that uses simulation for testing and developing a method for recommending sequences of LOs, and conclude with a discussion of our findings.

2 Model Structure

In another paper [5], we have argued that it is not necessary to model every detail of the learning process, but that systems can be tested in a simulation that captures only the most relevant characteristics for a given purpose. Therefore, we take an approach that lets an instructional developer choose different dimensions – such as attributes of the learning objects, aspects of the pedagogical environment, attributes of the learner – and assign weights to each dimension according to the priorities of the developer. This section describes the structure of the simulation model so as to provide background for the experiment around peer impact, described in Section 3.

The EA architecture [9] provides a way to record metadata about learner interactions with LOs. As learners interact with LOs, any information that is known about the learner at the time of the interaction can be saved as metadata and associated with the LO. The EA assumes that each learner is represented by a learner model that contains static attributes (*characteristics*) as well as other data gathered as they interact with the LOs (*episodic*).

We developed an agent-based simulation model with very simple abstractions of learners and LOs. Each learner agent has an attribute, *aptitude-of-learner*, a number between (0,1), which we use to model the range of aptitudes (low to high) different learners have for a given subject matter. In our model, this attribute is assigned at the start of the simulation and does not change, but in future work we plan to create more sophisticated simulations where this attribute is not static. The simulated LOs have an attribute to represent *difficulty level*, which is also a number between (0,1) where higher values represent more difficult material. The simulated LOs are arranged into a random directed acyclic graph to represent prerequisite relationships between the LOs.

The model execution revolves around an atomic action: the learner's interaction with a LO. This action might occur hundreds or thousands of times during a simulation run, thus creating a multitude of EA metadata from which measurements can be taken. In related work [5], we introduce the term *evaluation function* to describe the function that computes the degree of success as result of an interaction between a learner and a LO. We will use the term *P[learned]* to describe the value that is generated by the evaluation function, i.e. the "probability that the learner learned the LO", or the "system's belief that the learner knows the LO". The P[learned] value is included as part of the EA metadata that is associated with LOs after learners interact with them.

Our evaluation function is a weighted sum, where each term deals with a dimension of learning to be considered. Each dimension of learning is calculated with a mini function. For example, suppose Learner_A were a novice with *aptitude-of-learner*=0.1. Next, suppose LO_X were a fairly easy LO, which implies a high probability of success. We use a mini function, *difficulty-of-LO*, to translate the LO difficulty attribute into a high probability value, giving *difficulty-of-LO*=0.8. Suppose we also wish to take into account that the likelihood of the learner learning the LO is higher if the learner has already viewed prerequisite LOs. Prerequisite information is given in the LO attributes. Our simulation model has a function for *hasPrerequisites* which searches through the EA metadata to discover whether the learner has indeed viewed the prerequisites and returns 1.0 if the answer is yes and 0.0 otherwise. If we want these dimensions to have approximately equal weights, then we can define the evaluation function below and obtain P[learned] as follows:

$$\begin{aligned}&\text{(w)(aptitude-of-learner)} + \text{(w)(difficulty-of-LO)} + \text{(w)(hasPrerequisites)}\\ &= (0.33)(0.1) + (0.33)(0.8) + (0.34)(1.0) = 0.637\end{aligned}$$

If, on the other hand, we wish to give the aptitude a higher weight, such as 60%, then the new value could be (0.6)(0.1) + (0.2)(0.8) + (0.2)(1.0), or 0.42. As expected, giving greater weight to this learner's low aptitude decreases the P[learned] somewhat. More dimensions can be incorporated so long as the weights sum to 1.0. The evaluation function, implemented as a weighted sum, will provide an estimated likelihood the LO has been learned between (0,1), making it easy to compare averages of such P[learned] values between simulation runs. However, we caution against comparing two simulation runs with different evaluation functions (i.e. different weights or dimensions) because that would be like comparing two numbers with different units of measure.

The independent variables in our experiment are the *aptitude-of-learner* values, the *difficulty level* values, the directed acyclic graph giving prerequisite relationships between LOs, as well as a dimension called *peer-impact*, which is explained in the next section.

3 Experiment

Our experiment is intended to explore through simulation the effects of peer impact on learning. We motivate the experiment by visiting literature around how peers can impact each other's scores.

Students are impacted by their peers even in their ordinary lives. A study was performed by Hanushek et al. [6] to clarify the impacts of peer group characteristics on achievement in the context of family and school factors, race and socio-economic status. Results suggested that students benefitted from higher achieving schoolmates. In contrast, the American Academy of Pediatrics warned that Facebook pages can make some children feel badly because they see themselves as being inferior to their peers [10]. This effect is due to the nature of Facebook, where most users will censor their posts and only share the most positive information about themselves, skewing the view of reality. Along the same lines, Daniel et al. [3] found in a study that learners will usually only participate in online learning activities if they have trust in their peers or some degree of self confidence.

Others have used simulations to study peer effects. Mao et al. [7] used a simulation model to study the impact of social factors in a course where students shared learning materials with each other. The output of Mao et al.'s model was a comparison of the amount of sharing connected to status levels: gold, silver, bronze, common. Populations fluctuated as users began at the common status and gradually transitioned between levels. The paper concluded that simulation models can be useful for developing and improving incentive mechanisms in virtual communities. In a different study, Zhang et al. [12] studied the fluctuation of a population of learners through various activities: registration, activation, action and adaptation. The authors found that learners who participated the most were also the ones most sensitive to changes in the community and had the most fluctuations.

This research, and other research, shows that a learner's score can be impacted by peer performance. We decided to explore this issue by creating a notion of "peer impact", where learners respond differently from one another according to how well other learners are doing in mastering the LOs. This takes the form of a new dimension in our evaluation function called *peer-impact*. Like the other dimensions we discussed in Section 2 (*aptitude-of-learner*, *difficulty-of-LO*, *hasPrerequisites*), this is a function that produces a value between (0,1) to represent a positive or negative impact on P[learned]. In our experiment, we use the following Equation 1 to compute P[learned] each time a learner visits a LO.

$$.25(\text{apt-of-learner}) + .25(\text{diff-of-LO}) + .25(\text{hasPrereq}) + .25(\text{peer-impact}) \quad (1)$$

We created two styles of peer impact called *reinforcing* and *balancing* which refer to a comparison between an individual learner's average P[learned] on the LOs they have viewed so far, compared to the average P[learned] of all learner agents, which we call "class average". The information to compute these

P[learned] averages is obtained from the EA metadata. Each learner is given one of these styles at the start of the simulation and it remains fixed. Future work could explore more sophisticated learner agents where this attribute is not static.

The reinforcing style means that the learner's score is "attracted" to the class average P[learned]. That is, when the class average is higher than their own, the peer impact function for a reinforcing learner produces a value close to 1; thus the learner will perform even better than they would have otherwise. This is a positive feedback loop, because as the learner performs better so does the class average thus further encouraging the learner to do better. If the class average is lower than their own, then the *peer-impact* function gives a value close to zero; thus the learner will do even worse than they would have otherwise.

Balancing is the opposite. In this case, a learner's score is "repelled" from the class average P[learned]. That is, when the class average is higher than the individual's average P[learned], then their score will be pulled down lower than it would have been otherwise. This is a negative feedback loop because when the class average is high, the learner's average goes in the other direction. When the class average is low, then the learner's score will be boosted higher than it would have otherwise. In Figure 1, we show the *peer-impact* function (the values 0.2 and 0.8 were chosen as thresholds to allow clear effects of the two types of learner to emerge).

```
if currentLearner BALANCING
   if class average is HIGHER than mine
      set peerImpact == randomNumBetween(0.0,0.2)
   if class average is LOWER than mine
      set peerImpact == randomNumBetween(0.8,1.0)
if currentLearner REINFORCING
   if class average is HIGHER than mine
      set peerImpact == randomNumBetween(0.8,1.0)
   if class average is LOWER than mine
      set peerImpact == randomNumBetween(0.0,0.2)
```

Fig. 1. Function to generate *peer-impact* for a given learner at a given time in the simulation

The dependent variable in our experiment is the P[learned] values generated by the simulation; we gain insight into whether the peer impact has a positive or negative effect by observing the relative P[learned] values. We varied this experiment under six conditions. We varied the proportions of balancing and reinforcing styles: mostly balancing, mostly reinforcing, and fifty-fifty. For instance, if the model is set to mostly balancing, when new learners are initialized, they have a high chance of being assigned the balancing personality and a low chance of being assigned the reinforcing personality. These three propor-

tions were each run under two difficulty levels: one with mostly easy LOs and high aptitude learners, and the other with mostly difficult LOs and low aptitude learners. These six conditions were hand picked to be representative samples on a curve of possible population mixes that should provide some insight about the effect of these two kinds of personality on the learning environment. We ran each of the six conditions 5 times because our model is stochastic; it produces slightly different results each time even under the same starting conditions.

A typical result is shown in Figure 2 (fifty-fifty, high difficulty with low aptitude learners). Each line represents the average P[learned] of different portions of the simulated learner population: the lightest thin line for all learners, black thin line for the learners who were assigned the reinforcing personality, and the dark grey thin line for the learners who were assigned the balancing personality. Normally, our simulation model would be used to evaluate a particular instructional planning technique, but because this experiment is intended to illuminate peer impact, the order in which LOs are consumed isn't important. Therefore, the simulated learners, of which there are 80, visited random LOs, of which there are 100.

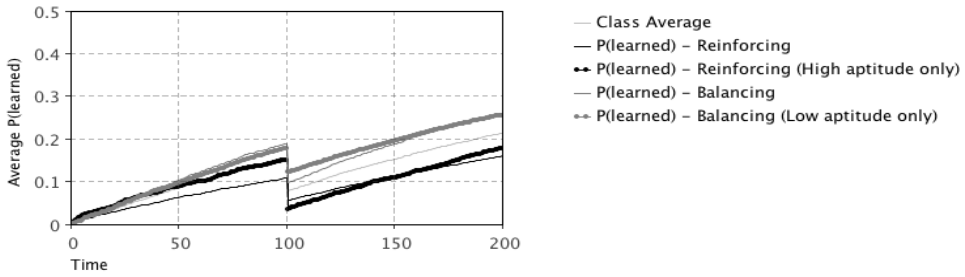


Fig. 2. Typical result

At the start of the simulations, the class average starts at zero. The balancing simulated learners had higher scores in this state because this is the behaviour defined in the evaluation function – that balancing learners do well when the class average is lower than their individual average. The learning gradually increases for both groups as the simulated learners visit more and more LOs. Although the results seem low overall – P[learned] only reaching short of 0.3 – this is due to the number of LOs (100) created in the simulation and the time it would take for learners to visit them all. We ran the simulation again with only 30 LOs and observed the same patterns, but with a steeper slope; the average P[learned] reached around 0.5. This raises interesting questions about whether the amount of time required to learn a set of LOs should actually be represented with a linear function. In reality, learners would get tired or lose interest or change their learning goals. Future work could compare instructional plans with learners having different levels of stamina.

The thick lines in Figures 2 and 3 represent subsets of the balancing and reinforcing personalities whose behaviour we wish to discuss in this experiment. Simulated learners do not have access to the actual class average, but compute the average based on what other simulated learners have allowed them to perceive about their performance. Based on Daniel et al.'s [3] results that confident learners are more likely to share their success, simulated learners with high P[learned] values shared their EA metadata, while those with lower P[learned] values did not. This creates a *suppression effect*, where each simulated learner has access to different information in the computation of how others are doing, depending on which other learners have suppressed information at the time they are computing the average.

The thick grey line shows only the balancing learners with low aptitudes while the thick black line shows only the reinforcing learners with high aptitudes. At the start of the simulation, the thick black line is below the thick grey line: it is perhaps surprising that a group of simulated learners with high aptitudes would have overall lower scores than a group of simulated learners with low aptitudes. We highlight this because it shows that different parts of the evaluation function – *peer-impact*, *aptitude-of-learner* etc. – can dominate at different times. In this case, high aptitude can be dominated by peer impact for reinforcing personalities when the class average is low.

In Figure 3, we observe another interesting phenomenon by injecting 80 more simulated learners halfway though the experiment, a somewhat contrived situation, although one that might happen in the real world if, say, two classes merged partway through a course, or if two study groups in an online course were mashed together, or due to the openness of many online courses (e.g. MOOCs) when new learners can join any time. Under most of the experimental conditions we tried, such as the typical result in Figure 2, although the influx of new learners caused the class average to drop (as expected, because each new learner starts with an average P[learned] of zero), there was no apparent change in the relative ranking of the groups of learners being measured. That is, if the balancing learners had the highest average before the influx, this continued afterward. However, in about a third of the runs with low difficulty LOs and high aptitude learners, the influx of learners caused a *phase shift*: now the thick black line jumps above the thick grey line (see Figure 3). This makes sense: the balancing learners who tend to do more poorly when the class average drops, do just that. The influx also creates a situation where there are now learners with high averages intermingled with learners with zero averages; this creates a different environment than the starting condition where everyone started at zero. Different environmental conditions cause the model to exhibit different behaviour. With the suppression effect deactivated, all learners have access to the same information. In this condition, we observed that the thick grey line overlapped with the thick black line and there was no apparent phase shift (i.e. no lines crossing over).

Even though the observed patterns are merely a result of the evaluation function implementation – that is, the model is simply doing what it was programmed to do – it helps system designers to keep track of the different possible

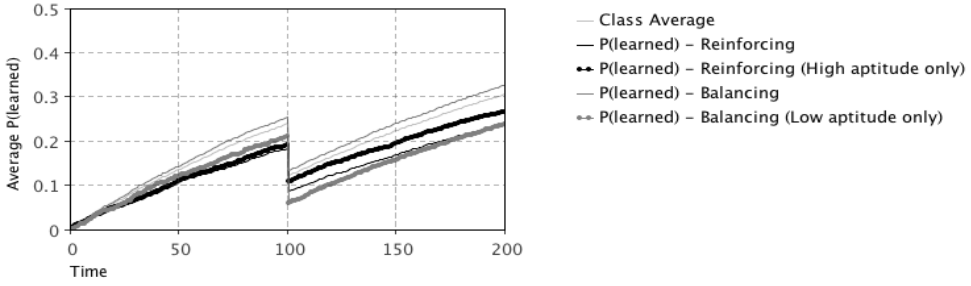


Fig. 3. Condition showing phase shift

behaviours as they try to design systems to support learning in all of these conditions: low or high aptitude learners, easy or difficult material, peer effects, prerequisites and many other possible dimensions, with each behaving differently in different situations. Without simulation, it is unlikely we would have made our observations about the phase shift as well as the observation about the high aptitude reinforcing learners having lower scores than low aptitude balancing learners. These observations reveal the specific circumstances that instructional developers should address in order to maximize the expected learning. For example, the system could be programmed to intervene when it detects that the current class average will push a learner's expected outcome in an undesirable direction. When the class average is higher than an individual's average, the scores of other learners should be displayed more prominently for the balancing learners but not for the reinforcing learners.

Through this experiment, we have also shown that simulations can be used to test unexpected situations. Future experiments could test for influxes of new LOs instead of new learners. Other variations could look at adding or removing LOs to impact the difficulty level of the course or the level of expertise of peer learners. When we injected a herd of simulated learners, we observed some surprising results. But, by examining the underlying dynamic behaviour as the simulation proceeded, we could actually explain why these results happened, thus gaining more intuition about learning that would help to better inform an experiment that might be carried out with real learners.

4 Other Research Directions

In ongoing work, we are also developing a technique for recommending sequences of LOs. Instructional planners have been built that explore different kinds of sequencing such as sequencing things of the same type, like "lessons" or even sequencing several types of activities, like presentations and assessments [1]. Our method involves using the EA metadata to identify "trails" of LOs. We are investigating the use of user-based and item-based approaches to generate recom-

mendations of these trails using Apache Mahout [1]. Using information captured in the EA metadata, we create metrics for giving sequences a score to reflect the quality of the sequence, for example does P[learned] increase or decrease over the sequence. We are also exploring changes to the evaluation function to favour sequences that suggest coherence, such as trails that give learners a view of the big picture before going into the details. Sequences with high scores are then used as a basis for recommending sequences to other learners. Our study will examine whether learners receiving sequence recommendations see any improvement over learners receiving one LO recommendation at a time.

Other work in simulating recommender systems for learning systems has been done by Drachsler et al. [4]; but the main difference is that this work did not involve sequences, peer impact or the EA architecture. Champaign [2] uses the ecological approach architecture to use the experiences of past learners to suggest sequences of LOs for future learners while also studying the impact of peer ratings, which are not the same as our peer impact because our peer impact is linked to the evaluation function.

Even with the simplistic models of learners and LOs we have presented so far, the peer impact experiment demonstrates the combinatorics of the various features is already becoming too complex to rely on human intuition; this is one of the main reasons for simulation modelling.

5 Conclusion

We created simulated learners whose overall learning was influenced by one of two styles of peer impact. Our study demonstrated that different patterns emerge when when simulated learners change their own behaviour based on the behaviour of the group and when these learners have limited access to information due to others' ability to suppress their EA metadata. In some conditions, a phase shift occurred from the initial situation where the class average is zero to a new situation with some learners having relatively high averages. The simulated learners prior to the influx had higher averages because they had the opportunity to visit LOs before the arrival of the new simulated learners. One style of peer impact is not universally better or worse than another, but each has advantages in different circumstances. It is important for instructional developers to understand such patterns. In future work, the use of simulations with the EA architecture will shed more light on peer impact and will allow us to also factor in the effects of different kinds of sequence recommendations.

The EA metadata make it easy to look deeply into the underlying dynamics and identify the conditions that create such behaviours. The EA metadata also allow us to change the inputs of the simulation and take measurements, as we did to compare the P[learned] averages between learners with different styles of peer impact. By using the EA architecture for the simulation studies, the later construction of a real learning system is made easier if the real system also uses

[1] http://mahout.apache.org/

the EA architecture. That is, if the real system also stores information about a learner's interaction with a LO as metadata associated with the LO, then estimating the likelihood of success for a real learner follows the same methods used by developers to estimate the success of simulated learners.

Acknowledgements
We would like to thank Dr. Julita Vassileva for discussions with the first author about using simulations for instructional planning and for insights on applying social concepts in the simulation model; Graham Erickson for his ideas about the evaluation function; and Dr. Nathaniel Osgood for discussions about cause loop diagrams and agent-based modelling in AnyLogic. We also wish to acknowledge the Natural Sciences and Engineering Research Council of Canada for funding some aspects of this research through a Discovery Grant to the last author.

References

[1] Brusilovsky, P. and Vassileva, J.: Course sequencing techniques for large-scale web-based education. International Journal of Continuing Engineering Education and Lifelong Learning, 13, 75-94 (2003).
[2] Champaign, J.: Peer-based intelligent tutoring systems: a corpus-oriented approach. Ph.D. Thesis, University of Waterloo, Waterloo, Canada (2012)
[3] Daniel, B., McCalla, G., Schwier, R.: Social Network Analysis techniques: implications for information and knowledge sharing in virtual learning communities. Int. J. of Interactive Media in Education, 2(1), 20-34 (2008)
[4] Drachsler, H., Hummel, H., and Koper, R.: Using simulations to evaluate the effects of recommender systems for learners in informal learning networks. In Learning, 3 CEUR Workshop Proc., 404-423 (2008).
[5] Erickson, G., Frost, S., Bateman, S., and McCalla, G.: Using the ecological approach to create simulations of learning environments. To appear in Lane, H.C., Yacef, K., Graesser, A., Mostow, J. (eds.), Proc. 16th Int. Conf. on AIED, Memphis (2013)
[6] Hanushek, E., Kain, J., Markman, J., Rivkin, S.: Does peer ability affect student achievement? Journal of Applied Economics, 18, 527-544 (2003)
[7] Mao, Y., Vassileva, J., and Grassmann, W.: A system dynamics approach to study virtual communities. In Proc. 40th Annual Hawaii Int. Conf. on System Sciences, HICSS '07, pp.178a-, Washington, DC, USA, IEEE Computer Society (2007)
[8] Matsuda, N., Cohen, W.W., Sewall, J., Lacerda, G. and Koedinger, K.R.: Predicting students performance with SimStudent that learns cognitive skills from observation. In Luckin, R., Koedinger, K.R., and Greer, J. (eds.), Proc. 12th Int. Conf. on AIED, Marina del Rey, 467-476 (2007)
[9] McCalla, G: The ecological approach to the design of e-learning environments: purpose-based capture and use of information about learners. Journal of Interactive Media in Education.
[10] Tanner, L: Docs warn about teens and 'Facebook depression'. Associated Press. http://www.msnbc.msn.com/id/42298789/ns/health-mental_health/t/docs-warn-about-teens-facebook-depression/ Accessed April 13, 2013.
[11] VanLehn, K., Ohlsson, S., and Nason, R.: Applications of simulated students: an exploration. Int. J. Artificial Intelligence in Education, 5, 135-175 (1996)
[12] Zhang, Y., and Tanniru, M.: An agent-based approach to study virtual learning communities. Hawaii International Conference on System Sciences, 1(11c) (2005)

Using HCI Task Modeling Techniques to Measure How Deeply Students Model

Sylvie Girard, Lishan Zhang, Yoalli Hidalgo-Pontet, Kurt VanLehn,
Winslow Burleson, Maria Elena Chavez-Echeagary, Javier Gonzalez-Sanchez

Arizona State University, Computing, Informatics, and Decision Systems Engineering, Tempe, AZ, 85281, U.S.A.

{sylvie.girard, lzhang90, yhidalgo, kurt.vanlehn, winslow.burleson, helenchavez, javiergs}@asu.edu

Abstract: User modeling in AIED has been extended in the past decades to include affective and motivational aspects of learner's interaction in intelligent tutoring systems. An issue in such systems is researchers' ability to understand and detect students' cognitive and meta-cognitive processes while they learn. In order to study those factors, various detectors have been created that classify episodes in log data as gaming, high/low effort on task, robust learning, etc. When simulating students' learning processes in an ITS, a question remains as to how to create those detectors, and how reliable their simulation of the user's learning processes can be. In this article, we present our method for creating a detector of shallow modeling practices within a meta-tutor instructional system. The detector was defined using HCI (human-computer interaction) task modeling as well as a coding scheme defined by human coders from past users' screen recordings of software use. The detector produced classifications of student behavior that were highly similar to classifications produced by human coders with a kappa of .925.

Keywords: intelligent tutoring system, shallow learning, robust learning, human-computer interaction, task modeling

1 Introduction

Advances in student modeling in the past two decades enabled the detection of various cognitive [3, 4, 8, 11, 13, 16, 17], meta-cognitive [1,6], and affective [2, 9] processes during learning based on classification of episodes in log data. Steps have been taken toward detecting when learning occurs [4] and to predict how much of the acquired knowledge students can apply to other situations [5, 6]. However, an obstacle in such research is how to gain an understanding of the user's cognitive or meta-cognitive processes while learning. While some of the indicators used in the literature

are common to any intelligent tutoring system, others are closely linked to the activities and pedagogical goals of a specific application. The adaptation of such indicators to the design of a new system often necessitates a detailed analysis of the new domain and how the tutoring system guides learners to acquire its skills and knowledge. In particular, an issue within this process is the ability to reach common ground between learner scientists that perform an analysis of learners (meta-)cognitive actions at a high level - via video or log analysis of student's past actions for example – and the definition of the indicators by software engineers, related to how the system was implemented, that can be used to simulate such processes in agreement with the constraints and functionalities of software. We view the specificity of detectors as unavoidable, so the best solution is to develop good methods for analyzing the new tutoring system and designing the detectors. This short article describes our method and its application to out project, AMT. In the AMT project, a choice was made to use HCI (human computer interaction) task modeling - a method for formally representing human activity, and by extension, the behavior of an interactive system -, as well as video coding schemes from human coders, to develop the detectors. The detectors aim to evaluate student's use of shallow and deep modeling practices with and without being guided by a meta-tutor, on the domain of dynamic systems modeling.

In Section 2, the AMT learning environment, for which the detectors were created, is introduced. In a third section, the task model of the user's activity in AMT is described. Next, the process of defining a coding scheme for the detector with human coders is presented, followed by the definition of the different classifications that define the value, the implementation and empirical evaluation of the detector. The final section summarizes the uses of task modeling within this work, and how it could be applied in future to other applications.

2 AMT software: a meta-tutor to teach deep modeling of dynamic systems.

AMT software teaches students how to create and test a model of a dynamic system. In our modeling language, a model is a directed graph with one type of link, as illustrated in Figure 1. Each node represents both a variable and the computation that determines the variable's value. There are three types of nodes.

- A *fixed value* node represents a constant value that is directly specified in the problem. A fixed value node has a diamond shape and never contains incoming links.
- An *accumulator* node accumulates the values of its inputs. That is, its current value is the sum of its previous value plus or minus its inputs. An accumulator node has a rectangular shape and always has at least one incoming link.
- A *function* node's value is an algebraic function of its inputs. A function node has a circular shape and at least one incoming link.

The students' learning objective is to draw a model representing a situation that is described in the form of a relatively short text. In the example of Figure 1, the description of the problem was " *Rust destroys steel and can spread quickly. Suppose you take a large sheet of steel, such as one that might be used as the roof of the box-car on a train, and you put it outside in the weather. Suppose it starts with a spot of rust that is 10 square inches in area. However, each week the rust spot gets bigger, as it grows by 30%. Therefore at the end of the first week, the rust spot is 13 square inches in area.*" and the objective of the problem was to "Graph the size of the rust spot over 10 weeks."

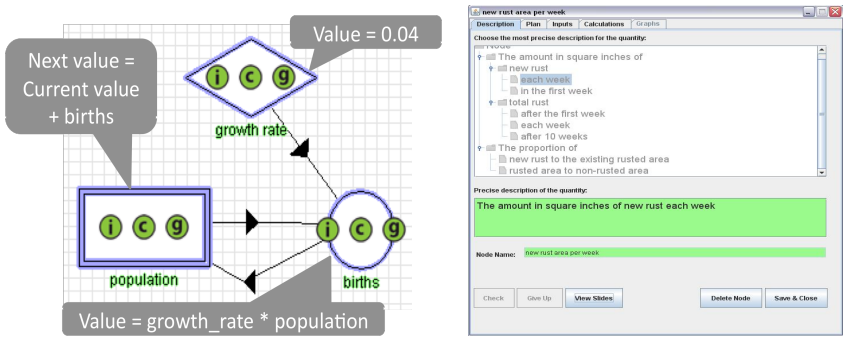


Fig. 1. The left image is the example of model, with gray callouts added to explain the contents of nodes. The right image is the example of a node editor.

The student constructs the model node by node, by filling in all information within each node in the form of four interactive tabs (description, plan, inputs, and calculations). During construction, students can use the *Check* button to evaluate the correctness of the current tab, or the *Solve it for me* button to ask the system to fill out the tab automatically.

The instruction is divided into three phases: (1) an introduction phase where students learn basic concepts of dynamic system model construction and how to use the interface; (2) a training phase where students are guided by a tutor and a meta-tutor to create several models; and (3) a transfer phase where all scaffolding is removed from soft-ware and students are free to model as they wish. The tutor gives feedback and corrections on domain mistakes.

The meta-tutor requires students to follow a goal-reduction problem solving strategy, the Target Node Strategy [18]. The basic idea is to focus on one node at a time (the target node) and completely define it before working on any other node. This process decomposes the whole problem of modeling a system into a series of atomic modeling problems, one per node. Like *Pyrenees* [2], it teaches students that if they just master this one difficult but small skill, then the rest of the problem solving will be straight-forward. In addition, the meta-tutor complains if students appear to be guessing too much or giving up too early, just as the *Help Tutor* did [3].

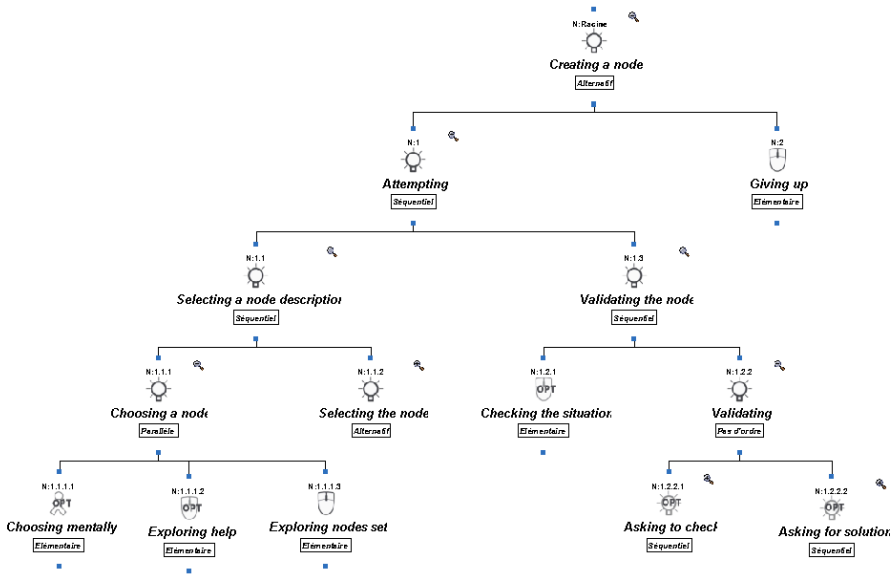
While students learn, their motivation, attention to details, and modeling depth can fluctuate. To assess students, the project needed detectors that detect shallow and deep modeling practices both with and without the meta-tutor. The measure should be usable in the transfer phase of the experiment as a dependent variable, because deep

modeling is the skill/knowledge that AMT teaches. The depth measure should also apply to student's behavior during the training phase so that we can check whether the instructional manipulations done during that phase have their intended effects (i.e., the measure serves as a manipulation check). The detector should further operate in real time (i.e., it doesn't require to know future actions or states in order to interpret the current action) so that it can be eventually be used by the system itself to condition its behavior.

3 Task Modeling: analysis of user's actions on software

A task model is a formal representation of the user's activity. It is represented by a hierarchical task tree to express all sub-activity that enables the user to perform the planned activity. The tasks need to be achieved in a specific order, defined in the task tree by the ordering operators. In AMT, every modeling activity follows the same procedure involving the same help features, task flow, and meta-tutor interventions. With a single task model of a prototypical modeling task, it is therefore possible to account for all of the user's activity in software. Due to the complexity of the final model, only one sub-activity will be described in this paper, illustrated in Figure 2. Only part of the model is deployed in the figure, and some subtasks will not be detailed here. In this part of the model the sub-activity the learner wishes to perform is to create a new node for the dynamic system s/he is currently modeling. We will first describe the task tree, and then insert the iterations and conditions that enable a formal verification of the flow of the task within the task model.

Figure 2: Sub-task "Creating a Node" in the AMT activity task model using K-MADe



Short description of the sub-task to model:

In order for a node to be created, the description tab of the node editor needs to be completed by selecting a node description, which corresponds to a valid quantity in the system to model. Each node is unique and cannot be created more than once. The user can engage in the task only if at least one node still needs to be created for the model to be complete.

Task tree and order of the tasks:

At the top level of the task tree "Creating a node", the learner can either attempt to create the node (task 1) or give up on the creation (task 2). The second task is represented in software by the user closing the node editor window, and can be done at any time during the task. The task "Creating a node" is over when a good description has been found and validated. The system can then try to initialize the selection and create the node.

In the first level of the task "Attempting", the learner first needs to select a node description (task 1.1), i.e.: what quantity the node will represent. S/he is then allowed to finish the creation of the node by validating the selection (task 1.2).

In order to select a node description, the user first needs to choose a node description (task 1.1.1) among the set of node descriptions offered by the system. This process involves the user choosing mentally one description (task 1.1.1.1), exploring the help features offered by software (task 1.1.1.2) and exploring the set of node descriptions displayed (task 1.1.1.3). S/he can then select the node (task 1.1.2). This subtask is not described in Figure 1 for a lack of space.

In order to validate the selection, the learner can choose to go back to the description of the problem to verify the correctness of his solution according to the problem to be simulated (task 1.2.1), and then has to validate the selection (task 1.2.1.2). When the user checks the validity of the selection, it can either be performed by checking the solution against the set of nodes still remaining to be modeled (task 1.2.1.2.1) or asking software to produce the solution (task 1.2.1.2.2). The user is allowed to ask for the solution only when a description has been checked at least once.

Now that the different actions of the learner are defined, the iterations and conditions will help represent the flow of the activity on the subtask "Selecting a node description" (task 1.1).

Iterative and Optional tasks

- Task 1.1 is iterative: it is possible to make several selections before trying to finish the description by validating.
- Task 1.1.1.2 is optional: The learner is not forced to explore the help features to choose a description, this is merely a choice on the learner's part.
- The main task, "creating a node", is iterative until the node is created or the activity is abandoned. The later is represented in the task model by an interruptible task: the learner can stop his/her creation of node activity any time by choosing to close the node editor window.

Conditions on tasks:

- Main task 1 has a pre-condition attached to it: the software only allows the user to engage in a creation of a new node if there is at least one node re-

lated to the modeling of the dynamic system that still remains to be created.

A first task model was created to represent learner's activity on software without the presence of the meta-tutor. This corresponds to the first version of software, which was evaluated against the interface including the meta-tutor in [18]. This second software interface includes a text-based agent that intervenes as the students engage in modeling to help them achieve deeper modeling behaviors, by applying constraints to the user's actions and giving meta-cognitive feedback. The meta-tutor was therefore added to the task model under the type "system" and the model was completed to include the constraints and interventions of the meta-tutor.

The final task model produced represented all possible actions of the learner on software in order to model a dynamic system. Next, a study of these actions, which led to the definition of the depth detectors, is detailed.

4 Detecting when students are modeling using shallow practices

The task model developed with K-MADe was used to define the episode structure. The first step in creating a coding scheme is to define a unit of measurement for the user's modeling actions. The task model clearly highlighted the different sub-activities the learner could engage in, referred to as goals. All goals are interruptible tasks in favor to accessing the help features[1] or abandoning the completion of the current goal for a new one. After a brainstorming session where researchers studied how students' actions fell in line with those goals, the following unit of depth, called "segment", was defined. This established the unit of coding to be used in the next phase.

Screen videos representing the learners' use of the AMT software with and without the meta-tutor were recorded during an experimental study described in [6]. These videos were studied to determine how much shallow vs. deep modeling occurred and the contexts, which tended to produce each type. A coding system was then created for video recordings of the learners' behavior. Three iterations of design for this coding scheme were performed, ending with a coding scheme that reached a multi-rater pairwise kappa of .902. The final coding scheme mapped learners' behavior to six classifications, which were implemented as the following depth detectors[AIED short paper]

- GOOD_METHOD: The students followed a deep method in their modeling. They used the help tools appropriately, including the one for planning each part of the model.
- VERIFY_INFO: Before checking their step for correctness, students looked back at the problem description, the information provided by the instruction slides, or the meta-tutor agent.

[1] It is to be noted that two help systems are available to users: (1) referring back to the instructions always available for viewing, and (2) looking at the problem situation where all details of the dynamic system to model are described.

- SINGLE_ANSWER: The student's initial response for this step was correct, and the student did not change it.
- SEVERAL_ANSWERS: The student made more than one attempt at completing the step. This includes guessing and gaming the system:
  - The user guessed the answer, either by clicking on the correct answer by mistake or luck, or by entering a loop of click and guessing to find the answer.
  - The user "games the system" by using the immediate feedback given to guess the answer: series of checks on wrong answers that help deduce the right answer.
- UNDO_GOOD_WORK: This action suggests a modeling misconception on the students' part. One example is when students try to run the model when not all of the nodes are fully defined.
- GIVEUP: The student gave up on finding how to do a step and clicked on the "give up" button.

Another detector was defined as a linear function of the six episode detectors. It was intended to measure the overall depth of the students' modeling, therefore providing an outcome measure in the transfer phase in future experimental studies. It considered two measures (GOOD_ANSWER, VERIFY_INFO) to indicate deep modeling, one measure (SINGLE_ANSWER) to be neutral, and three measures (SEVERAL_ANSWERS, UNDO_GOOD_WORK, and GIVE_UP) to indicate shallow modeling.

Once the coding scheme reached a sufficient level of agreement between coders, the task model was used to adapt the coding to students' actions on the software. The episodes that were coded for depth by human analysts in the sample video were analyzed by creating scenarios from the task model within K-MADe. The validation of six detectors' implementation involved three human coders, who watched a sample of 50 episodes, paying attention to the depth of modeling exhibited by the student's actions, and chose the classification that best represented the depth of the learner modeling at the time of the detected value. A multi-rater and pairwise kappa was then performed, reaching a level of inter-reliance of .925.

5 The different uses of the Task Model

The task modeling language K-MAD and its task model creation and simulation environment, K-MADe [7] were chosen for the following reasons: the environment enables the creation and replay of scenarios of student's actions, a set of functionalities not described here enable a formal verification of the model. Additionally the associated simulation environment ProtoTask [14] allows non-specialists in task modeling to visualize the flow of the task model, via scenarios in a clear and simple manner.

The use of K-MAD helped in the creation of the detectors and are a first step in offering an alternative technique to simulated learners, by tackling the following problems:

- *Breaching the gap between learner scientists' understanding of how the learning process works and programmers' definition of the application flow, functionalities, and indicators.*
- *Enabling a formal validation of software flow, understandable by all.*
- *Using simulated learners scenarios to define the detectors.*

A researcher in educational technology - expert in teaching modeling and part of the AMT project - and an HCI practitioner, realized the task model. The former was an expert on how AMT software was designed in terms of pedagogical content and task flow. His expertise focused in particular on the actions the students were allowed/incited/forbidden to do within software at each moment of the modeling task. The HCI practitioner was not familiar with intelligent tutoring systems or meta-tutors. She was involved in the creation of the task model in a consulting capacity, in regards to her expertise in task modeling of interactive systems.

The task model could be defined at the level of the user's planning of actions and system flow, with iterations and conditions alone. However, the objects in K-MADe enable us to represent the constraints of the learner's actions concretely and to apply a formal verification of task flow. It was therefore possible to represent the set of descriptions as either valid or invalid, to detect when a node has been checked and the result of that check, and to add constraints on the checking procedure such as to avoid node duplication. This enabled a formal verification of software flow prior to validate its fidelity to learner scientists' ideas about possible actions on software and the underlying processes involved.

Once the model was constructed, the use of ProtoTask to visualize software flow and follow learners' possible sets of actions allowed by software enabled the ability to simulate learners by creating scenarios of use that could be played and replayed at will, focusing on the cognitive and meta-cognitive levels of learner's experience on software. In the process of creating our detectors, a video analysis of learner's past actions was performed. The model could be used to check the possible actions of users with what the designer of the system wanted to offer as functionalities and software flow. During this analysis, the task model could be used once again to define scenarios that simulated learner's pertinent behaviors using ProtoTask. Once those scenarios were formed, the task analyst came back to the original K-MAD modeling language and studied the similarities and contrasts between scenarios to define the rules that govern the detection of shallow and deep modeling practices within AMT. Once the task model identified points of detection of such practices, it became easy for programmers to go back to software and implement the rules.

6 Conclusion and Future Work

In this paper, a method to create a detector of deep modeling within a meta-tutor using HCI task modeling and video coding schemes was described. The main outcome of this process was the creation of detectors inferring the depth of students' modeling practices while they learn on a meta-tutoring system, reaching a multi-rater and pairwise kappa score of .925. We believe the use of the task model to define shal-

low and deep modeling practices by helping to create the detectors to be of value for any simulated learning environments, in particular for indicators that a common to all learning tasks present in a tutoring system.

In interdisciplinary teams, the design of indicators can lead to communication issues due to misunderstandings and a lack of common ground between analysis made at a high level of learners' cognitive and meta-cognitive processes, and the representation of those behaviors within software. In particular, video-coding processes can become costly when the coders' understanding of the details of how the system works differs from how the system actually works. Our experience using K-MADe and ProtoTask highlighted an ease in this project in gaining a better view of the tutoring system and the detection of deep modeling within the interface. In particular, the use of ProtoTask by the non-specialists in task modeling helped clarify issues of task flow and the definition of the set of user's actions at each moment of interaction.

A limitation of the method is the applicability to different types of tutoring systems. In AMT, a single task model was able to represent the entirety of a users' learning activity. In tutoring systems that teach a set of skills through different pedagogical approaches for diverse types of learning tasks, the creation of such task models might prove more costly and may not be completely adapted to the creation of detectors that need to be adapted to each task specifically.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 0910221. We would like to thank Sybille Caffiau for consulting in the project and sharing her expertise in task modeling of interactive systems.

References

1. Aleven, V., McLaren, B.M., Roll, I., Koedinger, K.R.(2006): Toward meta-cognitive tutoring: A model of help seeking with a Cognitive Tutor. International Journal of Artificial Intelligence and Education 16, 101–128
2. Arroyo, I., and Woolf, B.P., 2005. Inferring learning and attitudes from a Bayesian Network of log file data. In Proceedings of the 2005 conference on Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology, Chee-Kit Looi, Gord McCalla, Bert Bredeweg, and Joost Breuker (Eds.). IOS Press, Amsterdam, The Netherlands, The Netherlands, 33-40.
3. Baker, R. S. J. d., Corbett, A. T., Koedinger, K. R., Evenson, S., Roll, I., Wagner, A. Z., … Beck, J. E. (2006). Adapting to when students game an intelligent tutoring system, Proceedings of the 8th international conference on Intelligent Tutoring Systems, Jhongli, Taiwan Berlin, Heidelberg.
4. Baker, R.S.J.d., Goldstein, A.B., Heffernan, N.T.: Detecting the Moment of Learning. In: Aleven, V., Kay, J., Mostow, J. (eds.) ITS 2010. LNCS, vol. 6094, pp. 25–34. Springer, Heidelberg (2010)
5. Baker, R. S. J. D., Gowda, S. M., & Corbett, A. T. (2011). Towards predicting future transfer of learning, Proceedings of the 15th international conference on Artificial intelli-

gence in education. Proceedings from AIED'11, Auckland, New Zealand Berlin, Heidelberg.

6. Baker, R. S. J. D., Gowda, S. M., Corbett, A. T., & Ocumpaugh, J. (2012). Towards automatically detecting whether student learning is shallow., Proceedings of the 11th international conference on Intelligent Tutoring Systems, Chania, Crete, Greece Berlin, Heidelberg.
7. Caffiau, S., Scapin, D., Girard, P., Baron, M., & Jambon, F. (2010). Increasing the expressive power of task analysis: Systematic comparison and empirical assessment of tool-supported task models. Interacting with Computers, 22(6), 569–593. doi:10.1016/j.intcom.2010.06.003
8. Corbett, A.T., MacLaren, B., Kauffman, L., Wagner, A., Jones, E.A.: Cognitive Tutor for Genetics Problem Solving: Learning Gains and Student Modeling. Journal of Educational Computing Research 42(2), 219–239 (2010)
9. D'Mello, S. K., Lehman, B., & Person, N. (2010). Monitoring affect states during effortful problem solving activities. International Journal of Artificial Intelligence in Education, 20(4), 361–389., doi:10.3233/JAI-2010-012
10. Girard, S., Chavez-Echeagary, H., Gonzalez-Sanchez, J., Hildalgo-Pontet, Y., Zhang, L., Burleson, W., and VanLehn, K., (2013), Defining the behavior of an affective learning companion in the affective meta-tutor project, in K. Yacef et al. (Eds.): Proceedings of the 16th international conference on Artificial Intelligence in EDucation (AIED'13), LNAI 7926, pp. 21--30. Springer-Verlag, Berlin, Heidelberg.
11. Gowda, S.M., Pardos, Z.A., and Baker, R. S. J. D. 2012. Content learning analysis using the moment-by-moment learning detector. In Proceedings of the 11th international conference on Intelligent Tutoring Systems (ITS'12), Stefano A. Cerri, William J. Clancey, Giorgos Papadourakis, and Kitty Panourgia (Eds.). Springer-Verlag, Berlin, Heidelberg, 434-443. DOI=10.1007/978-3-642-30950-2_56
12. Koedinger, K.R., Corbett, A.T., Perfetti, C. (2010): The Knowledge-Learning-Instruction (KLI) Framework: Toward Bridging the Science-Practice Chasm to Enhance Robust Student Learning. Carnegie Mellon University Technical Report, June, 2010
13. Martin, J., VanLehn, K.: Student assessment using Bayesian nets. International Journal of Human-Computer Studies 42, 575–591 (1995)
14. Lachaume, T., Girard, P., Guittet, L., & Fousse, A. (2012). ProtoTask, new task model simulator. In M. Winckler, P. Forbrig, & R. Bernhaupt (Eds.), Human-Centered Software Engineering (Vol. 7623, pp. 323– 330). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-34347-6
15. Muldner, K., Burleson, W., Van, D. S., Brett, & Vanlehn, K. (2011). An analysis of students' gaming behaviors in an intelligent tutoring system: predictors and impacts. User Modeling and User-Adapted Interaction, April 2011, 21(1-2), 99–135m doi:10.1007/s11257-010-9086-0
16. Shih, B., Koedinger, K.R., Scheines, R.: A response time model for bottom-out hints as worked examples. In: Proceedings of the 1st International Conference on Educational Data Mining, pp. 117–126 (2008)
17. Walonoski, J. A., & Heffernan, N. T. (2006). Prevention of off-task gaming behavior in intelligent tutoring systems Proceedings of the 8th international conference on Intelligent Tutoring Systems. Jhongli, Taiwan Berlin, Heidelberg.
18. Zhang, L., Burleson, W., Chavez-Echeagary, H., Girard, S., Gonzalez-Sanchez, J., Hildalgo-Pontet, Y., and VanLehn, K., (2013), Evaluation of a meta-tutor for constructing models of dynamic systems, in K. Yacef et al. (Eds.): AIED'13, LNAI 7926, pp. 666--669. Springer-Verlag, Berlin, Heidelberg.

Validating Item Response Theory Models in Simulated Environments

Manuel Hernando, Eduardo Guzmán and Ricardo Conejo

E.T.S. Informática. Universidad de Málaga,
Bulevar Louis Pasteur, 35. 29071
Málaga, Spain
{mhernando, guzman, conejo}@lcc.uma.es

Abstract. The Item Response Theory is a successful technique generally used in testing systems. Its application in problem solving environments requires the collection of large amount of data. That issue is stressed with ill-defined domains in which the actions that a student could accomplish are difficult to predict. Known IRT models could not be as appropriate as it is desired to that application and we have to explore new alternatives. One of these alternatives is a new family of models called quasipolytomous models of IRT. These models are halfway between dichotomous and polytomous models and require less data than polytomous models being more informative than dichotomous ones. Validating these new models is a very difficult issue in a real environment since student knowledge level is not observable. A simulation environment could help us to verify new models of IRT. Besides, with a simulator we can study different scenarios and observe how our model behaves in them.

Keywords: Problem Solving Environments, Student Modeling, Procedural Knowledge Estimate, Item Response Theory.

1 Introduction

Student modeling in problem solving environment is an important issue in the AIED field. Constraint-Based Modeling (CBM) [1] and Cognitive Tutors (CT) [2] are the outstanding approaches in that matter. CBM models are a set of constraints associated to principles in the domain that could be either violated or not, those constraints are related to declarative principles of the domain. CT inferred procedural student knowledge directly from student interactions through a technique called Knowledge Tracing [2] which is based on Bayesian procedures and estimates the probability that a student has learned a certain rule of the domain given his/her actions.

Procedural knowledge could be also inferred by other techniques such as the Item Response Theory (IRT) [3], which is one of the most important strategies of declarative knowledge assessment in testing systems. Our proposal of applying IRT to problem solving environment sets a connection between problem solving environment and

testing, that is, if we make a matching between the elements of problem solving and the elements of testing we can, directly, apply IRT to infer procedural knowledge. In this sense, we model the solution of each problem as a directed graph where nodes are states of the solution path of the problem and edges are transitions between states. Using that representation each node could be understood as a question and each edge as an option in the question.

Our challenge is also to develop an automatic (or semiautomatic) procedure for mining the problem solving path from the logs of students' performance while solving it. This mining process would lead us as well to infer the IRT components which will be used for diagnosing the procedural knowledge. Furthermore, we also want this procedure to be dynamic, that is, the solution path and the inference of IRT components have to be updated dynamically when new logs will be available. Accordingly, the problem graph will include all possible student actions, so when a student completes an action that never was completed by another student it has to be included in the problem graph. These new actions have to be taken into account when the procedural knowledge is assessed, that is, the calibration procedure of the IRT has to be done when new actions are incorporated to the problem graph.

There are, mainly, two families of IRT-based models according to how they update the estimated student knowledge in terms of the student's response: dichotomous and polytomous models. Dichotomous models consider each response as either correct or incorrect whereas polytomous models consider each response individually. These traditional IRT models could not be as good as it is expected dealing with this type of calibrations since dichotomous models are not as informative as we need in this kind of problems and there will be actions with little evidence (maybe actions followed by 1 or 2 students) to polytomous calibration, especially in ill-defined domains where the set of possible actions is very large.

In order to explore new IRT models that fit better with our challenges we have developed a simulation environment in which virtual students (with a known real procedural knowledge of the domain) solve virtual problems (simulating their behavior according to their prior knowledge) and we have compared their estimated knowledge with their real knowledge. In this sense, we have developed a new family of IRT-based models called *quasipolytomous models* which are halfway of dichotomous and polytomous models considering not all possible responses but a subset of them.

2 Item Response Theory Models

The IRT is one of the most successful and well-founded strategies for knowledge inference in testing systems [3]. IRT infers and models student performance by means of some probabilistic functions called characteristic curves, the idea is that student's results could be explained by a set of non-observable factors (for instance, the knowledge level).

There are a lot of IRT models, based on how the models update the estimated student knowledge in terms of his/her response they could be [4]:

- Dichotomous models: Each response is considered as either correct or incorrect. When a student selects an option in a question test, his/her estimated knowledge is updated according to whether the option selected is the correct one or if that is other. These models require only a characteristic curve per item that represents the probability that a student with a certain knowledge level answers it correctly. This characteristic curve is called item characteristic curve (ICC).
- Polytomous models: Each possible response has its own characteristic curve called operating characteristic curve (OCC) [5], which expresses the probability that a student select that answer [6]. The student estimated knowledge is updated by means of the OCC related to the selected response.

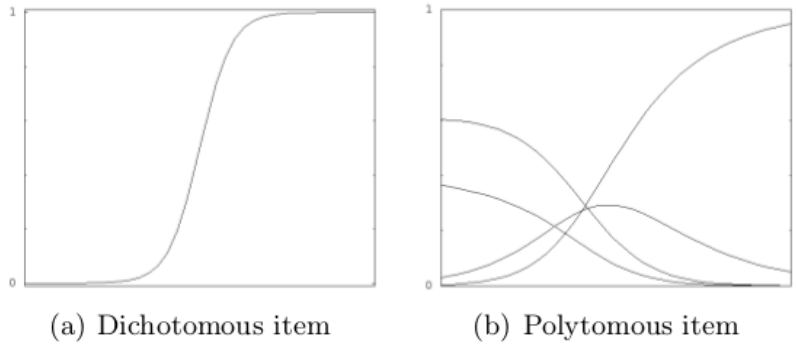


Fig. 1.Characteristic curves of an item under dichotomous and polytomous models

Polytomous models are more informative than dichotomous ones since they take into account each possible response independently instead of considering each answer as correct or incorrect.

Figure 1 shows the curves of an item of both the dichotomous and polytomous models of IRT. The dichotomous model is shown in Figure 1(a) and has only the ICC which is the probability of answering correctly this item (y-axis), given a certain knowledge level (x-axis). The polytomous model is presented in Figure 1(b).Each item choice has its own characteristic curve which is the probability of choosing this choice (y-axis), giving a certain knowledge level (x-axis).

The most popular proposals for modeling the dichotomous characteristic curves are the logistic models, which use logistic functions. These models could be classified, considering the number of parameters that the function has. According to this classification there are 3 kinds of logistic models: 1PL, 2PL, and 3PL, with one, two and three parameters, respectively. A generic 3PL ICC of an item X_i is defined as follows:

$$P(X_i|\theta) = c_i + (1 - c_i)\frac{1}{1+e^{-Da_i(\theta - b_i)}} \quad (1)$$

Where D is a parameter introduced to fit the curve similar to the normal curve, parameter a_i is the discrimination, parameter b_i is the difficulty, parameter c_i is the guessing parameters of the item X_i, and θ is the knowledge level.

There are a lot of polytomous models of IRT. In our work we have considered the IRT-model proposed by Thissen and Steinberg for multiple-choice items [7]. In this model besides the observable categories (selectable choices) there is another non-observable and latent category called "*don't know*" (DK) that expresses the probability that a student does not know how to answer the item. Each observable category has a portion of the category DK included since students who do not know will select an observable category. The formula of each observable category is exposed below, X_i represents the response to the item i and h is the category selected:

$$P(X_i = h|\theta) = \frac{e^{a_h\theta+c_h}}{\sum_{k=0}^{m_i} e^{a_k\theta+c_k}} + d_h \frac{e^{a_0\theta+c_0}}{\sum_{k=0}^{m_i} e^{a_k\theta+c_k}} \quad (2)$$

category 0 is the non-observable category DK and d_h is the portion of that category included in each observable category h. The parameters denoted by a reflects the order, as well as discrimination, for the categories, and the parameters denoted by c reflect the relative frequency of the selection of each alternative.

3. Introducing IRT in problem solving

The application of IRT to problem solving environments requires a polytomous model since each student action should be taken into account; a dichotomous model only would be able to establish if an action is correct or incorrect. However, some actions could have little evidence and IRT calibration could be not as accurate at it is expected. For that reason, we have developed a new family of models of IRT called quasipolytomous models of IRT which are on the halfway between dichotomous and polytomous ones.

Quasipolytomous models consider not all choices as independent but only those that have enough evidence. For instance, let us consider an item witch 20 choices (what is usual in problem solving environment if we include all student actions), if 8 of them have been selected only by 1 or 2 students they do not offer us enough evidence to do a polytomous calibration. Instead of doing it, we consider these 8 choices as a simple choice reducing the number of OCCs from 20 (one per choice) to 13 (12 individual choices and an extra choice that group the other 8) that have, all of them, evidence enough to do an IRT calibration.

In testing systems there could be items with a lot of choices too, let us consider figure 2 in which the number of choices is 120 since we have to take into account the permutation between these 5 elements.

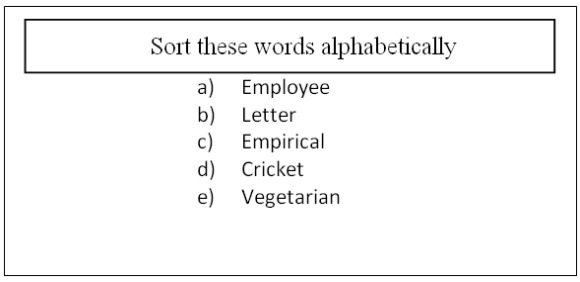


Fig. 2. An item with 120 choices

3 Simulation environment

In order to verify quasipolytomous models we have developed a simulator in which virtual students have a real (prior) knowledge assigned and they have to solve some virtual problems according to their knowledge level. Once the students solve the proposed problems, their knowledge is estimated by means of a quasipolytomous model of IRT and then, these estimations are compared with the students' real knowledge.

3.1 Virtual problems

In this simulation environment, a virtual problem is represented as a collection of items; each item is a state of problem solving path with a certain number of possible transitions to other states. These transitions are the choices of the item.

While students are solving problems, new students' actions could appear in the system and they need to be included in the model. For that reason, virtual problems are not static entities but they can change during students' interactions.

At the beginning, each problem has only the ideal solution path, that is, those nodes that are part of the ideal solution. Those nodes are modeled by dichotomous items according to the equation 1.A characteristic curve, the ICC, is enough to model this kind of items. Students who do not answer correctly the item could, with some probability, make a new action at this step of problem solving. Then the opposite curve (i.e. 1-ICC) is branched into two curves changing the original dichotomous item to a polytomous one. According to the former explanation, students' actions could provoke the addition of new nodes to the problem graph.

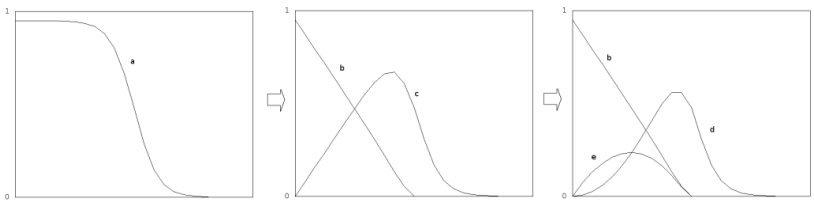


Fig. 3. Addition of new curves to an item

Figure 3 shows how new curves are added to an item. Firstly we have the opposite ICC curve which expresses the probability that a student, given his/her knowledge level, will answer the item incorrectly. This is curve marked as *a* in the figure. Curve *a* is branched into curves *b* and *c* when the action represented by curve *c* is added to the model; finally, curve *c* is branched again and then curves *d* and *e* are added to the item. The item was, at the beginning, a dichotomous item with two possible responses; then a new action was included in it and, as a consequence, the wrong curve was converted in other two curves. Again a new action was added to the model and the item changes to a 4-choices item.

3.2 Virtual students

A virtual student is an entity that has a real (prior) knowledge associated and is able to solve problems according to it. The idea is that, depending on its knowledge level, the student could go on through the problem graph by completing an action or other. Further, they could accomplish new actions with a certain probability.

A student will select an action of the state according to his/her knowledge level and the characteristic curve since these curves are probabilistic functions of the knowledge level. Figure 4 shows an example of a virtual student selecting a choice in a 4-choice item. That student will select each choice with a probability of 0.2, 0.38, 0.18 and. 0.24 respectively. When a non-correct action is selected, there will be a probability that the student will accomplish a new action and, therefore, this curve will be branched into other two.

Since real knowledge of virtual students is assigned before simulation, we can choose what population distribution we want in each experiment. It is an important feature of the simulation since we can study the impact of different kind of population in order to validate new models in all cases or only in some of them.

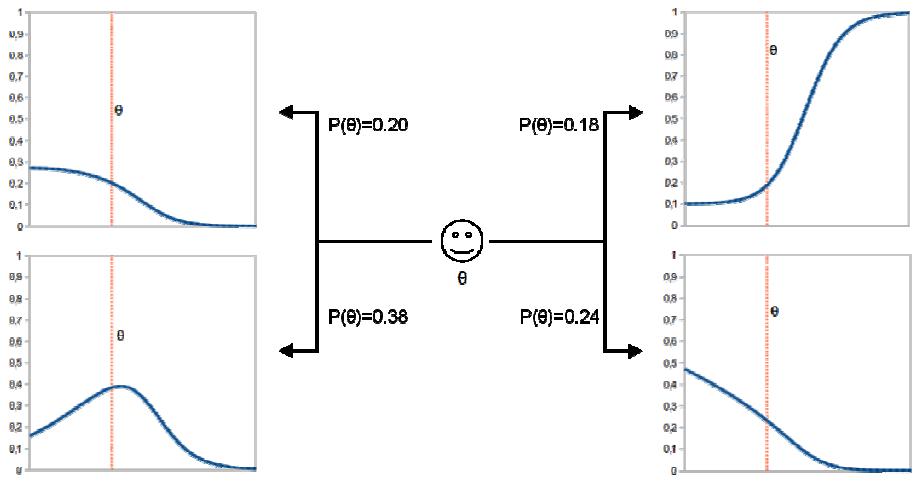


Fig. 4. Virtual student selecting a choice

4 Experimentation

In order to verify our model in a simulated environment, we have conducted some experiments with virtual students and virtual problems. The experiments have been accomplished to verify the model with different population distributions as well as different models of IRT and different number of problems.

4.1 Experiment description

The experiments accomplished with our simulator were able to compare scores offered by quasipolytomous models of IRT in different situations. Each experiment has been accomplished 30 times in order to reduce the impact of anomalous data. The number of virtual students in each experiment was 1000.

In order to get confidence results we have not done the calibration IRT phase, instead of that, we have estimated knowledge level with the known curves of each model. Polytomous models could not be as accurate as expected including the calibration stage since some item choices could have not enough evidence to be calibrated properly. The student estimated knowledge level is calculated using the formula of the equation 3, the probability of having a specific knowledge level given the steps followed by the student solving the problem is calculated multiplying the probability of selecting each step given the knowledge level.

$$P(\theta|s_1, s_2, \cdots, s_n) = \prod_{i=1}^{n} P(s_i|\theta) \quad (3)$$

In our experiments, virtual students have solved two problems with 10 items. We have accomplished mainly three experiments varying the population distribution, the probability of generating new actions and the item difficulty respectively. In our experiments, we have compared accuracy of knowledge level estimation obtained by a quasipolytomous model with those obtained by polytomous and dichotomous models with the same data.

The accuracy of knowledge level estimation was calculated using the next formula, where θ is the real knowledge, θ^* is the estimated knowledge, and N the number of students:

$$d(\theta, \theta^*) = \frac{\sum_{i=1}^{N}(\theta-\theta^*)^2}{N} \quad (4)$$

We have chosen four types of population distribution to accomplish the experiment. First, a normal distribution of population was selected; the probability of having a knowledge level was centering in the middle of knowledge range, since that range was [-3, 3]; the distribution was centered in 0. The second population was a uniform one, in which the probability of having a knowledge level is equally distributed. The other two populations were a low-level and a high-level population, which were normal distributions centered in small and high values respectively. Figure 5 shows the

knowledge level distributions used in our experiments. From left to right they are the normal, the uniform, the low-level, and the high-level distribution respectively.

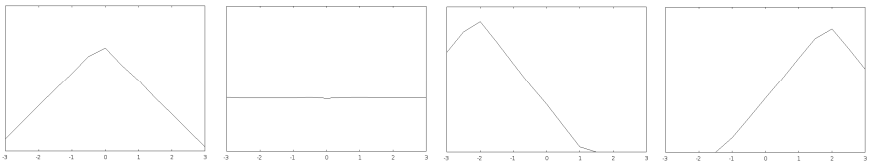


Fig. 5. Students' knowledge level distributions used in the experiments

The impact of the generation of new actions was also studied in our experiments. To this end, we have conducted experiments varying the percentage of adding a new action giving it values of 1%, 1.5%, 2%, and 2.5%. More than 2.5% of adding new actions could lead to a very large number of curves in the model.

Finally, we have also considered the difficulty of the items. We have conducted an experiment changing the value of this parameter. The difficulty value is calculated according to a normal distribution centered in a certain value, which is the difficulty average. We have done our experiments with values of -1, 0, and 1, respectively.

4.2 Experiment results

Experiments conducted suggest that polytomous models of IRT perform more accurately than dichotomous ones. This result is not surprising, since polytomous models are more informative. Experiments also show that quasipolytomous models of IRT are not as accurate as polytomous ones but more accurate than dichotomous ones. Besides, results obtained by quasipolytomous models are very similar to those obtained by polytomous ones.

Table 1 shows results obtained with different population distribution. In all cases accuracy obtained by polytomous models are better than obtained by dichotomous and quasipolytomous models. These differences are higher in the normal distribution and lower with a low-level population.

Table 1. Accuracy of IRT models changing the population distribution

Distribution	dichotomous	quasipolytomous	polytomous
Normal	4.153118	0.7957639	0.6714305
Uniform	3.841433	0.6284451	0.5397757
Low-level	2.290098	0.4186174	0.2723583
High-level	3.784738	0.8671618	0.8108576

We can see in the former table that differences between results obtained by quasipolytomous models and polytomous ones are not significant.

Our second experiment conducted studied how affects the probability of adding a new action. To this end, we changed the percentage of adding a new action from 1%

to 2.5%. Table 2 shows those results, when we increase the probability of adding a new action the accuracy of dichotomous and quasipolytomous models gets worse since but the accuracy of the polytomous models gets better since they obtain more precise information.

Table 2. Accuracy of IRT models changing the percentage of adding a new action

% new action	dichotomous	quasipolytomous	polytomous
1.0%	3.291380	0.6609889	0.6126646
1.5%	3.489956	0.6695576	0.5845097
2.0%	3.613464	0.6778118	0.5593333
2.5%	3.674588	0.7016298	0.5379146

Finally, we have compared results obtained by these models varying item (solving path step) difficulty average. The student knowledge level used in our experiment is in the range [-3, 3]. We have considered average item difficulty of -1, 0, and 1. An item with difficulty of b_i will be answered correctly by a student with knowledge level of b_i with a probability of 0.5. Table 3 shows the results of this experiment, the dichotomous models have a better behavior when the difficulty is below 0, however quasipolytomous and polytomous models get better results when the difficulty increases. Polytomous and quasipolytomous models accurate better when the item difficulty is higher since it allows the model to have more curves to estimate the student knowledge level.

Table 3. Accuracy of IRT models changing the item difficulty average

Difficulty	dichotomous	quasipolytomous	polytomous
b_i**= -1**	2.364223	1.2011291	1.0558448
b_i **= 0**	3.131082	0.5410807	0.4450589
b_i **= 1**	5.056735	0.2902812	0.2199130

5 Conclusion

In this paper, we have validated a new approach that uses the Item Response Theory, a well-founded theory generally used for declarative knowledge estimation in testing systems, to infer procedural skills in problem solving environments. To do that, we have developed a new model of IRT, the quasipolytomous model. This model is halfway between dichotomous and polytomous models being more informative than dichotomous models and needing less amount of data than polytomous ones.

This verification could be difficulty accomplished in a real environment, since we need to know the prior knowledge level of students to measure the estimation accuracy. This knowledge level, however, is a latent trait that is not observable. In addition,

we needed a controlled environment where the students' performance was not biased by external factors. For all these reasons we have developed a simulation environment.

Using a simulation environment we can choose the nature of the population in order to study how well the model performs in different situations and with different students' samples. We also can decide the difficulty of the steps of the problem (i.e. items): before any calibration, the simulator is able to decide what items are more difficult and what are easier.

Other advantage of using a simulation environment is that we can repeat each experiment in order to reduce the impact of anomalous data. Furthermore, we can present the problems to a large number of students, which could be difficult in a real environment.

Regarding the quasipolytomous model of IRT, our experiments show that its application is useful in problem solving environment (besides any kind of procedural task, and in declarative domains for inferring declarative knowledge), especially if we work with ill-defined domains in which the amount of possible new actions is very large. Quasipolytomous models of IRT offer similar estimations as polytomous models but need less data. Besides, quasipolytomous models of IRT are more informative than dichotomous ones since they collect data from correct and incorrect responses.

Acknowledgements: This work is part of DEDALO project which is financed by the Andalusian Regional Ministry of Science, Innovation and Enterprise (P09-TIC-5105).

6 References

1. Mitrovic, A., Martin, B., Suraweera, P.: Intelligent tutors for all: The constraint-based approach. IEEE Intelligent Systems 22 (2007) 38-45
2. Anderson, J.R., Corbett, A.T., Koedinger, K.R., Pelletier, R.: Cognitive Tutors: Lessons learned. The journal of the Learning Sciences 4 (1995) 167-207
3. Embretson, S.E., Reise, S.P.: Item response theory for psychologists. Lawrence Erlbaum, Mahwah (2000)
4. Guzmán, E., Conejo, R., de-la Cruz, J.L.P.: Adaptive testing for hierarchical student models. User Model. User-Adapt. Interact. 17 (2007) 119-157
5. Dodd, B.G., De Ayala, R.J., Koch, W.R.: Computerized adaptive testing with polytomous items. Applied Psychological Measurement 19 (1995) 5-22
6. Guzmán, E., Conejo, R.: A model for student knowledge diagnosis through adaptive testing. In: In Proceedings of 7th International Conference Intelligent Tutoring Systems, ITS2004 Brazil, Springer-Verlag (2004) 12-21
7. Thissen, D., Steinberg, L.: A response model for multiple choice items. Psychometrika 49 (1984) 501-519

Toward a reflective SimStudent: Using experience to avoid generalization errors

Christopher J. MacLellan, Noboru Matsuda, and Kenneth R. Koedinger

Human-Computer Interaction Institute
Carnegie Mellon University
Pittsburgh PA 15213, USA
cmaclell@cs.cmu.edu, mazda@cs.cmu.edu, and koedinger@cmu.edu

Abstract. Simulated learner systems are used for many purposes ranging from computational models of learning to teachable agents. To support these varying applications, some simulated learner systems have relied heavily on machine learning to achieve the necessary generality. However, these efforts have resulted in simulated learners that sometimes make generalization errors that the humans they model never make. In this paper, we discuss an approach to reducing these kinds of generalization errors by having the simulated learner system reflect before acting. During these reflections, the system uses background knowledge to recognize implausible actions as incorrect without having to receive external feedback. The result of this metacognitive approach is a system that avoids implausible errors and requires less instruction. We discuss this approach in the context of SimStudent, a computational model of human learning that acquires a production rule model from demonstrations.

Keywords: simulated learners, metacognition, cognitive modeling, representation learning, grammar induction, generalization error

1 Introduction

Simulated learning systems can be used for a wide range of tasks, such as modeling how humans learn, as teachable agents, and as a means to automate the construction of models that can be used in cognitive tutors. In an effort to reduce the amount of developer effort needed to deploy simulated learners for these tasks, researchers have been relying increasingly on the use of machine learning algorithms. However, by increasing the generality of these systems through machine learning approaches, these systems become more susceptible to making unrealistic generalization errors.

When using simulated learners to model human learning, we desire systems that predict student's errors as well as their correct behavior. Unrealistic generalization errors, in the context of these systems, are errors that the system predicts humans will make, but that they never actually make. If a system is prone to making these kinds of errors, then it becomes difficult to draw conclusions from the predictions the simulated learners makes for novel tasks.

These generalization errors also complicate the use of simulated learners as teachable agents because they result in a system that produces non-human behavior. When human students are teaching a simulated learner in a peer-tutoring scenario and it makes errors that humans never make, then it decreases the authenticity of the experience. This inauthenticity might effect the social dynamics of the learning-by-teaching scenario possibly making the teachable agent less effective.

Finally, generalization errors also have negative effects when using simulated learners to automatically build cognitive tutors. For this purpose, simulated learners have been used to author production rule models via interactive demonstrations of the solutions to the problems the system will tutor. This approach may decrease the amount of work required to build a cognitive tutor and allow subject-matter experts to author tutors directly, without an AI developer. In this paradigm, SimStudent's errors are useful to the extent that they correspond with typical student errors; in these cases, the resulting production rules can be added to the tutor's bug library. However, if the errors are unrealistic, the author must waste time identifying and deleting these nonsensical production rules.

In this paper, we propose an approach that uses background knowledge to mitigate unrealistic generalization errors with no changes to the underlying algorithms and which should increase the effectiveness of the underlying learning mechanisms. Before presenting this approach in section 4, we first review SimStudent, the simulated learning system that provides the context for this work (section 2) and introduce a motivating example of a nonsensical generalization error SimStudent currently makes (section 3). After presenting this approach, we present some initial results and discuss conclusions and future work.

2 The SimStudent Architecture

The simulated learner system that we focus on in this paper is SimStudent, a system that induces production rule models from demonstration and problem solving. The SimStudent system is used primarily for three tasks: to model and predict human learning, to author cognitive tutors, and to function as a teachable peer-agent.

In order to understand how SimStudent works and the situations in which it makes unrealistic generalization errors, we will review the types of knowledge used by SimStudent, how this knowledge is represented, and the learning mechanisms SimStudent uses to acquire this knowledge from experience.

2.1 Knowledge and Representation

There are three kinds of knowledge in SimStudent: primitive operator function knowledge, conceptual knowledge, and procedural knowledge. The first kind of knowledge is hand-constructed and consists of the low-level functions for manipulating data available to the system (i.e., adding two values, appending two

strings together, etc.). One example of a low-level function is SkillAdd, which accepts two arguments, each of type arithmetic expression, and returns the sum of these two expressions as a single arithmetic expression. These functions constitute SimStudent's background knowledge. Depending on the task SimStudent is being used for, different kinds of background knowledge may be appropriate.

Head	Body	Prob
Expression	← Number Variable	0.95
Expression	← Minus Variable	0.05
Variable	← x	1.0
Minus	← -	1.0
Number	← 0	0.1
	...	
Number	← 9	0.1

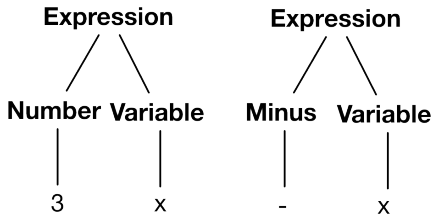


Fig. 1. A simple probabilistic context-free grammar and example parses of two expressions using this grammar.

The second kind of knowledge is conceptual, or representational, knowledge, which is encoded as a probabilistic context-free grammar. It is automatically acquired by SimStudent and is used to interpret the interface and information in it. Figure 1 shows a simple example of the conceptual knowledge SimStudent might possess about expressions for an algebra domain. This knowledge enables SimStudent to automatically extract plausible "chunks" from the input, such as the coefficient or term in an equation, which can subsequently be manipulated by primitive operator functions or procedural rules. Furthermore, this knowledge can be used to determine the likelihood that a given example was produced by the grammar.

If (current-row 'output-cell 'row)
(cell-in-row 'row 1 'left-side)
(is-left-child-of 'left-side 'coefficient)
→ **then** (write-text 'output-cell
(append "divide" 'coefficient)).

Fig. 2. An example production rule for division.

The final kind of knowledge is procedural knowledge, which represents the skills that we desire students to learn. This knowledge is encoded as production rules, which contain conditions under which the rules apply and what to do under those conditions. Figure 2 shows an example of a production rule signifying that when the left side of the equation's parse tree has a left child (here called coefficient), then enter "divide <the coefficient>" into the output cell.

2.2 Learning Mechanisms

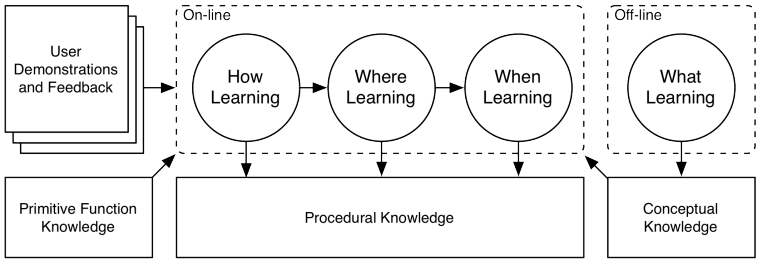


Fig. 3. A diagram of the SimStudent learning mechanisms and how they interact.

Of the three kinds of knowledge manipulated by the SimStudent system, two are learned automatically: the conceptual and procedural knowledge. To acquire these two kinds of knowledge the system employs four learning mechanisms: what learning, where learning, when learning, and how learning. The what learning is used to acquire the conceptual knowledge whereas the where, when, and how learning are used to acquire the procedural knowledge. Figure 3 shows how these four learning mechanisms interact. Before SimStudent is used, the what learning is run to acquire the conceptual knowledge. When SimStudent encounters a situation where it does not know how to act, which is common initially, it requests a demonstration from the author (the tutor developer or student tutor). This demonstration is comprised of four parts:

- *Focus of attention:* the set of relevant interface elements (e.g., the left and right hand sides of an equation);
- *Selection:* the interface element to manipulate (e.g., the output cell);
- *Action:* the action taken in the selection (e.g., update the text value); and,
- *Input:* the argument to the action (e.g, the text string used to update the selection).

Every time the system sees a new demonstration or gets corrective feedback on its performance, it learns or modifies a production rule. Production rule learning is done in three parts: 1) how learning attempts to explain the demonstration and produce the shortest sequence of primitive operator functions that replicates the demonstrated steps and ones like it, 2) where learning identifies a generalized path to relevant elements in the tutor interface that can be used as arguments to the function sequence, and 3) when learning identifies the conditions under which the learned production rule produces correct actions. We will now review each of these learning mechanisms.

What This mechanism operates off-line to acquire a probabilistic context-free grammar from only positive examples. This task can be defined as:

- *Given:* a set of examples of correct input;
- *Find:* a probabilistic context-free grammar with the maximal likelihood of producing the examples.

This task is performed using a grammar induction approach outlined by Li et al. [1], which uses a greedy approach to hypothesize the grammar structure and Expectation Maximization to estimate the grammar parameters.

Whenever a demonstration is given to SimStudent, it augments the provided information with the most likely parse trees of the content of each element in the focus of attention. This additional information is used by SimStudent in the subsequent learning mechanisms to extract deep feature knowledge from the content (e.g., to recognize and extract the coefficient of a term in an equation). The parse trees make this deep feature information directly accessible to SimStudent through the nodes in the parse tree (e.g., the left child of the parse tree for "$3x$" in Figure 1 corresponds to the coefficient).

How This is the first of three mechanisms executed in response to a demonstration. The how learning task can be defined as:

- *Given:* a set of demonstrations consisting of the state of the relevant interface elements and the parse trees of the contents of these elements as well as the resulting input for each state;
- *Find:* a sequence of primitive functions that when applied to each state produces the corresponding input.

This task is performed by exhaustively applying the primitive operator functions over all nodes in the focus of attention parse trees until the input is produced. The iterative-deepening depth-first search strategy is used to find the shortest sequence of functions that explains the data [1]. If no sequence exists, then a special functions is created that takes the states and produces the corresponding inputs.

Where This learning mechanism identifies the path to the relevant tutor interface elements. The tutor interface elements are specified by a hierarchical tree structure (a table is comprised of rows which each contain cells). During interactive instruction, the relevant interface elements are specified by the author teaching SimStudent. For each relevant element, SimStudent generates a parse tree for the contents. The relevant portions of these parse trees are defined as those that are utilized by the operator function sequence acquired through the how learning. The task of learning a general path to this relevant information can be defined as:

- *Given:* a hierarchical representations of the interface elements and their parse trees, the function sequence from the how learner, and a set of elements that have been identified as relevant;

- *Find:* a list of paths through the representation hierarchy to all of the relevant elements and the relevant portions of their parse trees.

The SimStudent approach to this task is to conduct specific-to-general learning over the set of relevant interface elements and parse trees [1]. Returning to the table examples, if the first cell in the first row of the table is always relevant, then a path to that specific cell will be returned. However, if all of the elements in the row are specified as relevant, then the entire row will be returned. After the location to the relevant elements has been identified, the system utilizes the function sequence to identify the relevant portions of the parse trees for each element. This same specific-to-general learning is then conducted over these relevant parse trees (within each element).

When This final mechanism identifies the conditions when the learned production rule is applicable. This task is defined as:

- *Given:* a set of positive and negative examples, each consisting of a set of features and their associated label;
- *Find:* a set of conditions over the features that separate the positive and negative examples.

As specified, this is a supervised learning task. The features used by SimStudent to represent each example are predicates that are automatically generated from the relevant portions of the parse trees. For example, there exists an "is-left-child-of" predicate, which says that a particular argument is the left child of a given node in one of the parse trees. This type of feature enables the retrieval of equations, terms, coefficients, and variables. Given the feature descriptions of each example, the positive and negative labels come from the user instructing the SimStudent system. The first positive example is the initial demonstration. Subsequent examples are generated when SimStudent tries to use the learned rules to solve novel problems and receives yes/no feedback from the author. To derive the set of conditions given the examples, SimStudent uses the FOIL algorithm [2], which uses information theory to perform a general-to-specific exploration of the space of hypothetical conditions.

These four learning mechanisms result in a simulated learning system that accepts user demonstrations and feedback and automatically acquires probabilistic context-free grammar rules and production rules. The system requires little background knowledge; for each task only the primitive functions need to be defined by the developer. However, the cost of this generality is a system that sometimes makes unrealistic generalization errors.

3 An example of an unrealistic generalization error

To explore the types of generalization errors that SimStudent makes, we turn to the algebra domain. One of the skills that students learn in this domain is how to proceed when given a problem of the form $< Symbol >< Variable >=<$

Symbol > (e.g., $3x = 6$). The skill that we desire the student to learn in this situation is to specify that their next step is to divide both sides by the coefficient of the term on the left side of the equation (the production rule from Figure 2).

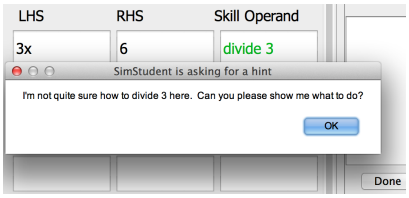


Fig. 4. SimStudent requesting a demonstration in an algebra tutor interface after the author has just entered "divide 3."

When SimStudent is first presented with a problem of this form, such as $3x = 6$, it will inform the author that it does not know how to proceed and ask for a demonstration. The author might demonstrate to SimStudent that the cells containing the left and right hand sides of the equation are relevant to the problem (by double-clicking on these cells) and update the next step interface element with "divide 3" (see Figure 4).

After receiving this demonstration, SimStudent parses the contents of the focus of attention (The first parse tree in Figure 1 shows an example of what the left hand of the equation might look like). Next, it employs the how learning mechanism, which searches for a sequence of functions that when applied to the nodes in the parse tree produce the input. In this example, it might learn to append the left child of the parse tree (for the left side of the equation) to the word "divide" and place it into the tutor interface (the then part of the production rule in Figure 2). Using the locations of the relevant elements (the left child of the parse tree), SimStudent then learns a general path through the representation hierarchy to the relevant elements and the relevant portions of the parse trees for these elements. Finally, SimStudent runs FOIL over the relevant information to learn the conditions under which the learned behavior is applicable. This results in the if portion of the production rule in Figure 2.

The learned production rule is more general than the single demonstration it was learned from; it is applicable for many equations, such as $4x = 12$ or $2x = 8$. However, when SimStudent is presented with a subtly different example that utilizes the same skill, $-x = 2$, it results in the mistaken generation of the input "divide -" (instead of "divide -1"). This is because in this situation the left child of the parse tree on the left hand side of the equation is a minus sign instead of the coefficient (see the second parse tree in Figure 1). In a review of problems of the form $-x =< Constant >$ in the 'Self Explanation CWCTC Winter 2008 (CL)' dataset accessed via DataShop [3], none of the human student made this error– therefore it is an example of unrealistic generalization error.

4 Reflecting before Acting

One reason that humans do not make this error is that they have a "sense" for what are reasonable output actions and they (subconsciously) reflect on actions before taking them. When a student is faced with the problem $-x = 2$ they may mentally produce the output "divide -," but realize that a "-" by itself is not mathematically grammatical because they have never seen an instance where this has occurred. This might lead them to consider a different action or to ask for help.

To reproduce this type of behavior, we modified SimStudent to utilize its conceptual knowledge, the probabilistic context-free grammar trained on example inputs (described as "what" learning in section 2). The acquired grammar is used to recognize when a potential output is not grammatical (when it cannot be parsed) and automatically flag the situation as a negative example. In other words, the system supervises itself and provides negative feedback (which the when learner uses) to improve its learning.

Now, when SimStudent is presented with a problem and finds an applicable rule, it simulates the execution of the rule and constructs a probabilistic parse of the value generated by the rule. If the value cannot be parsed by the current grammar (there is a 0% probability that the grammar produced the value), then SimStudent flags the trace as a negative instance and re-runs the when learning, which refines the conditions of the rule so that it no longer applies in the erroneous situation. If SimStudent has no other applicable rules, then it request a demonstration from the author, exactly like a human student.

5 Initial Results

To evaluate the effectiveness of this metacognitive loop, we have tested the probabilistic parser's ability to separate correct from incorrect actions based on the parse probability defined by the probabilistic context-free grammar. Table 1 shows five problems where SimStudent might make unrealistic errors. The first three are problems where SimStudent might induce a rule for dividing by the symbol before the variable instead of the coefficient. The last two problems correspond to inducing a rule retrieving the symbol after the variable and division sign instead of the entire denominator. On all five problems, the probabilistic grammar was capable of identifying the correct from the incorrect actions.

These results suggest that this approach is capable of identifying these kinds of errors. In general, this approach will be effective at identifying errors that result in non-grammatical output, where grammatical is defined by the probabilistic context-free grammar. This is effective because the rules are learned specific-to-general on a substantial amount of positive example inputs. By bringing this previous experience to bare, SimStudent can avoid nonsensical generalization errors and produce its own negative feedback, which enhances the effectiveness of its other learning mechanisms (more self-labeled examples for the when learning). Furthermore, this requires no additional work from an author and should reduce the amount of required author feedback.

Table 1. Five examples of problems where SimStudent might make the generalization error of retrieving the character before the variable or after the variable and the division sign, the corresponding correct and incorrect actions, the validity of these actions, and the parse probability of the actions.

Example	Possible Action	Valid	Parse Probability
$-x = 2$	divide $-$	No	0.00%
	divide -1	Yes	19.64%
$(-2)x = 6$	divide)	No	0.00%
	divide (-2)	Yes	0.09%
$3(x + 1) = 6$	divide (	No	0.00%
	divide 3	Yes	27.90%
$x/(-3) = 3$	multiply (	No	0.00%
	multiply (-3)	Yes	0.09%
$x/-5 = 1$	multiply $-$	No	0.00%
	multiply -5	Yes	19.64%

This task of verifying the output could alternatively be viewed as applying constraints to SimStudent's output and learning from constraint violations. Viewed this way, our work is related to the work on constraint-based tutoring systems [4]. In our case, there is only one constraint, "the output must be grammatical" where grammatical is defined as the probability of the output being produced by the grammar must be greater than 0%. We use a threshold of greater than 0% to signify grammatical, but one could imagine using a different threshold (e.g., greater than 0.05%). Thus, this constraint could be viewed as a probabilistic constraint that is automatically acquired from positive training examples.

6 Conclusion and Future work

In this paper, we outlined a novel approach to detecting and learning from unrealistic generalization errors that can be employed by simulated learner systems. The implications of this approach are threefold: (1) its use will result in models of learning that more closely aligns with human data, (2) teachable agents using this approach will be more realistic for the students using them, and (3) developers can produce cognitive tutor models with less work.

While this approach shows promise, it clearly has some shortcomings that should be remedied in future work. First, a more in-depth analysis of the alignment between SimStudent and human students is necessary. Previous work [5, 6] has looked at the human errors that SimStudent is capable of predicting, but a more detailed analysis of the unrealistic generalization errors, or errors that SimStudent makes that human students do not, would be useful. This would serve as a baseline to evaluate the SimStudent model and to evaluate the effectiveness of this approach.

A second direction for future work is to compare this approach to other approaches that might reduce these errors. We could imagine a system that has additional condition knowledge for the operator functions so that it would not generalize to situations where the function sequence would not be applicable (such as trying to divide by a symbol instead of a number). It would also be interesting to explore how reflection might facilitate the acquisition of this additional condition knowledge for the operator functions.

Finally, we are interested in applying this approach in other more complex and open-ended domains such as in RumbleBlocks, an educational game that teaches K-3 children about the relationships between the concepts of stability, low center of mass, wide base, and symmetry. We have been exploring how probabilistic grammars can be used to learn conceptual knowledge in RumbleBlocks [7] and we believe that this approach should scale up to this more complex domain.

References

1. Li, N., Schreiber, A.J., Cohen, W.W., Koedinger, K.R.: Efficient Complex Skill Acquisition Through Representation Learning. Advances in intelligent tutoring systems **2** (2012) 149–166
2. Quinlan, J.R.: Learning Logical Definitions from Relations. Machine Learning **5** (1990) 239–266
3. Koedinger, K.R., Baker, R.S.J.d., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J.: A Data Repository for the EDM community: The PSLC DataShop. In Romero, C., Ventura, S., Pechenizkiy, M., Baker, R.S.J.d., eds.: Handbook of Educational Data Mining. CRC Press (2010)
4. Mitrovic, A., Ohlsson, S.: Evaluation of a constraint-based tutor for a database language. International journal of artificial intelligence in Education **10** (1999) 238–256
5. Lee, A., Cohen, W.W., Koedinger, K.R.: A Computational Model of How Learner Errors Arise from Weak Prior Knowledge. In Taatgen, N., van Rijn, H., eds.: Proceedings of the Annual Conference of the Cognitive Science Society, Austin, TX (2009) 1288–1293
6. Matsuda, N., Cohen, W., Sewall, J., Lacerda, G., Koedinger, K.R.: Evaluating a Simulated Student using Real Students Data for Training and Testing. In Conati, C., McCoy, K., Paliouras, G., eds.: Proceedings of the International Conference on User Modeling. (2007) 107–116
7. Harpstead, E., MacLellan, C., Koedinger, K.R., Aleven, V., Dow, S.P., Myers, B.: Investigating the Solution Space of an Open-Ended Educational Game Using Conceptual Feature Extraction. In: Proceedings of the International Conference on Educational Data Mining. (2013)

Towards Moment of Learning Accuracy

Zachary A. Pardos† and Michael V. Yudelson‡

†Massachusetts Institute of Technology
77 Massachusetts Ave., Cambridge, MA 02139
‡Carnegie Learning, Inc.
437 Grant St., Pittsburgh, PA 15219, USA
zp@csail.mit.edu,yudelson@carnegielearning.com

1 Introduction

Models of student knowledge have occupied a significant portion of the literature in the area of Educational Data Mining[1]. In the context of Intelligent Tutoring Systems, these models are designed for the purpose of improving prediction of student knowledge and improving prediction of skill mastery. New models or model modifications need to be justified by marked improvement in evaluation results compared to prior-art. The standard evaluation has been to forecast student responses with an N-fold student level cross-validation and compare the results of prediction to the prior-art model using a chosen error or accuracy metric. The hypothesis of this often employed methodology is that improved performance prediction, given a chosen evaluation metric, translates to improved knowledge and mastery prediction. Since knowledge is a latent, the estimation of knowledge cannot be validated directly. If knowledge were directly observable, would we find that models with better prediction of performance also estimate knowledge more accurately? Which evaluation metrics of performance would best correlate with improvements in knowledge estimation? In this paper we investigate the relationship between performance prediction and knowledge estimation with a series of simulation studies. The studies allow for observation of the ground truth knowledge states of simulated students. With this information we correlate the accuracy of estimating the moment of learning (mastery) with a host of error metrics calculated based on performance.

2 Bayesian Knowledge Tracing

Among the various models of knowledge, a model called Bayesian Knowledge Tracing [2] has been a central focus among many investigators. The focus on this model has been in part motivated by its use in practice in the Cognitive Tutors [4], used by over 600,000 students, and by its grounding in widely adopted cognitive science frameworks for knowledge acquisition. For our experiments we will be employing the most frequently used basic Bayesian Knowledge Tracing

[1] A session during the main proceedings of EDM 2012 was dedicated to papers on Knowledge Tracing, a frequently used approach to modeling student knowledge.

model for both simulation and evaluation; however, there are implications beyond BKT models. Knowledge Tracing is a simple Hidden Markov Model of Knowledge defined by four parameters; two performance parameters and two knowledge parameters. The performance parameters, guess and slip, are the emission parameters in an HMM which respectively correspond to the probability that a student answers correct even if she is in the negative knowledge state (guess) and the probability that she answers incorrectly even if she is in the positive knowledge state (slip). The knowledge parameters, prior and learn rate, are the probability that a student knows the skill before answering any questions and the probability that, if the student is in the negative knowledge state, she will transition to the positive state at any given opportunity.

3 Related Work

There has been a limited amount of prior work focusing on detecting the moment of learning. We were able to track one relevant publication by Baker and colleagues [1]. They investigated detection of moment of learning in student data by modifying BKT structure. Another relevant result was published by [5]. They looked at scoring student model fits on simulated data and found a disparity between rankings of two frequently used metrics: root mean squared error and area under ROC curve. In this work we would like to address the question of the quality of detecting the moment of learning and investigate the problem of choosing a goodness-of-fit metric for that purpose.

4 Data

Our simulation dataset consisted of 1,000 simulated students and 100 skills with 30 questions per skill. Every student answered all 30 questions for each of the 100 skills. In the BKT simulation model we included no dependencies between skills and also no student specific parameters; therefore, the data can be thought of as either being produced by 1,000 students total or a new 1,000 students for every skill. Programmatically, data for each skill is stored in a separate file. Each row in each file represents one students data for that skill. The data stored from the simulation contains the students ground truth binary state of knowledge (mastered or not) at each of the 30 opportunities to answer (first 30 columns) and also the students correctness of responses to the 30 questions (stored in the second set of 30 columns).

In addition to the simulated data files containing student knowledge states and observed responses, we had corresponding files containing inferences of knowledge states and predictions of responses made with 16 different parameter sets resulting in 1,600 prediction files. Details of the parameter selection for simulation and prediction are discussed in the next section.

5 Methodology

5.1 Simulation

We generated 1,000 students knowledge and performance for 100 skills. Skills are defined by a set of four knowledge tracing parameters which the skill data is generated from. The 100 sets of four parameters were selected at random, uniformly sampling from the following constrained ranges for the parameters; prior between 0.01-0.80, learn rate between 0.01-0.60, and guess and slip between 0.05-0.40. After the 100 sets of parameters were selected, simulated data was produced by specifying a Dynamic Bayesian Network representation of Knowledge Tracing with a time slice length of 30. This representation, defined in Kevin Murphys Bayes Net Toolbox, with a particular parameter set fixed in the conditional probability tables, was then sampled 1,000 times, representing each simulated student. The sample Dynamic Belief Network function in BNT for simulation is a simple one; a random number between 0 and 1 is generated, if the number is equal to or lower than the prior parameter, the simulated student begins in the negative (not learned) state at time slice 1. To generate the observed response at this time slice, another random number is generated, if that number is greater than the guess parameter, the observed response is incorrect. To determine if the students knowledge state is positive (learned) at the next time slice; a random number is generated, if that number is less than or equal to the learning rate, then the students state is positive. With a positive state, the new random number needs to be greater than the slip parameter in order to produce a correct response. This is repeated for 30 times to simulate 30 knowledge states and observed responses per student.

5.2 Prediction

Typically, to predict student data, a hold-out strategy is used whereby a fraction of the students and their data is used to find a good fitting set of parameters. That good fitting set is then used to predict the fraction of students not used in training. The research question of this paper did not involve parameter fitting but rather required us to evaluate various models and observe how the models prediction of performance corresponded to its inference of knowledge. To do this we needed variation in models which we accomplished by choosing 16 candidate parameter sets with which to predict student data from each of the 100 skills. Since no training was involved, all data served as the test set. The top five sets of parameters used in the Cognitive Tutors was used, as well as 10 randomly generated parameters sets using the the same parameter constraints as the simulation, and, lastly, the ground truth parameter set for the skill was used to predict. The the same 15 parameter sets were used to predict the 100 skills, only the ground truth parameter set changed.

The prediction procedure is the same one used in all papers that use Knowledge Tracing; the prior, guess and slip parameters dictate the probability of correct on the first question. After the prediction is made, the correctness of

Table 1: Confusion Table

		Actual	
		Correct	Incorrect
Predicted	Correct	True Topisitve(TP)	False Positive (FP)
	Incorrect	False Negative (FN)	True Negative (TN)

the first question is revealed to the KT algorithm, which incorporates this observation using Bayes Theorem to infer the likelihood that the knowledge was known at that time. A learning rate transition function is applied and the processes is repeated 30 times in total to create 30 predictions of knowledge and 30 predictions of correctness per student for a skill.

6 Metrics

The most common metrics used to evaluate prediction performance in the EDM literature has been Area Under the Receiver Operator Curve (AUC) and Root Mean Squared Error (RMSE). One of the goals of our experiment is to reveal how indicative these measures are of the models accuracy in inferring knowledge. While these are the most common metrics, many others have been used in machine learning to evaluate predictions. We utilize a suite of metrics to investigate which metric is best at forecasting knowledge inference accuracy.

6.1 Model Performance

We selected a set of metrics in wide use today to score models when predicting student performance and knowledge state. Below is a short description of them.

Confusion Table Metrics Confusion table (rf. Table 1) is a table widely used in information retrieval and is a basis for a set of metrics capturing correctness of a retrieval or classification algorithm. Rows and columns of the confusion table denote the predicted and actual classes respectively and the cells in the intersection contain the counts of cases. Refer to Table 1 for an illustration. Here we illustrate a case for binary classification akin to the problem of binary classification of student performance (correct or incorrect) and state of knowledge (known or unknown).

If prediction is not categorical, say a probability from [0, 1], it is customary to round it: probabilities of 0.5 and greater become 1. For example, the cases when prediction matches the reality are captured in True Positive cell and the cases when the actually incorrect responses are marked as correct are captured in False Positive cell. We will use the confusion table metrics below.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1a}$$

$$precision = \frac{TP}{TP + FP} \tag{1b}$$

$$recall = \frac{TP}{TP + FN} \tag{1c}$$

$$F - measure = 2\frac{precision*recall}{precision + recall} \tag{1d}$$

As opposed to the so-called point measures described above, there is also a frequently used Area Under Receiver Operating Characteristic curve (AUROC), which is a curve measure. The curve is produced by varying the rounding threshold (0.5 for point measures) from 0 to 1 and computing and plotting False Positive Rate (FPR) vs. True Positive Rate (TPR) (see below).

$$TPR = \frac{TP}{TP + FN} \tag{2a}$$

$$FPR = \frac{FP}{FP + FN} \tag{2b}$$

An area under resulting curve is the sought metric. An area of 0.5 is equivalent to random chance for a binary classifier. An area greater than 0.5 is, thus, better than chance. An exact AUC calculation can also be derived by enumerating all possible pairs of predictions. The percentage of the pairs in which the true positive prediction is higher is the AUC. This is the ability of the predictor to discriminate between true and false.

Pseudo R^2 R^2 or percent variance explained is often used as a goodness of fit metric in linear regression analysis. For with binary classification, there exist several versions of R^2 called pseudo R^2. Applicable to our situation is Efrons pseudo R2 (refer to Equation below).

$$R^2 = 1 - \frac{\sum_{i=1}^{N} y_i - \hat{y}_i}{\sum_{i=1}^{N} y_i - \bar{y}} \tag{3}$$

Where N is the number of data points, y_i is the i-th component of the observed variable, $\bar{y_i}$ is the mean observed value, and $\hat{y_i}$ the prediction of i-th component of the observed variable.

Metrics Based on Log-Likelihood Likelihood functions are widely used in machine learning and classification. Likelihood captures the probability of the observing data given parameters of the model. In binary classification a natural log transformation of the likelihood function is often used (see below). Here

N is the total number of datapoints, y_i is the i-th component of the dependent variable, $\hat{y}_i$ is the predicted value of the i-th component of the dependent variable.

$$loglikelihood = \sum_{i=1}^{N} y_i \ln(\hat{y}_i) + (1 - y_i) \ln(1 - \hat{y}_i) \tag{4}$$

In addition to log-likelihood itself, there are several metrics that use log-likelihood as kernel component. For example, Akaike Information Criterion (AIC), Akaike Information Criterion with correction for finite sample size (AICc), Bayesian Information Criterion (BIC), and several others. These metrics introduce various forms of penalty for the size of the model (number of parameters) and number of datapoints in the sample in order to put overfitting models at disadvantage when performing model selection. Here k is the number of model parameters, N is the number of datapoints.

$$AIC = -2loglikelihood + 2k \tag{5a}$$

$$AICc = AIC + \frac{2k(k+1)}{N-k-1} \tag{5b}$$

$$BIC = -2loglikelihood + k \ln(N) \tag{5c}$$

Since we are comparing models that are only different in the parameter values and are doing so on the same dataset, we will not see difference in ranks assigned by log-likelihood, AIC, AICc, and BIC metrics.

Capped Binomial Deviance In addition to log-likelihood and log-likelihood-based metrics, we include the Capped Binomial Deviance (CBD). Capped binomial deviance is a version of the log-likelihood where prediction values are mandated to be at least away from 0 and 1 values and uses a logarithm with base 10 instead of natural logarithm. The is usually set to a small value of 0.001.

6.2 Moment of Learning

To capture the quality of detecting the moment of learning we devised a metric based on mean absolute deviation (MAD). Namely, moment of learning MAD is the average absolute difference of number of skill application opportunities between the moment when the internal state of the generating skill model switched to learned state and the moment when the probability of the skill being in a learned state reaches 0.95 (a traditionally used threshold in the area of intelligent tutoring systems). A perfect model would have a moment of learning MAD of 0. The larger the moment of learning MAD is the worse the model prediction of model of learning is.

7 Experiments and Results

7.1 Experiment 1

Research question: Among accuracy metrics used for ranking various parameter sets (models), which ones correlate best with accuracy of moment of learning prediction?

7.2 Results

The Table 2 below contains the correlations of performance prediction value, knowledge prediction value for all metrics, and moment of learning mean absolute error. Since prediction of performance is most widely adopted as a standard approach and the fact that we are trying to contrast it to the moment of learning mean absolute error, we sorted the rows corresponding to various statistical metrics by the respective column. The first column lists the metric used to evaluate the goodness of performance and knowledge prediction. The second column is the correlation between knowledge and performance prediction using the particular metric on both (this is the column the table is sorted by). The third column is the correlation between the particular metric used to evaluate performance and Mean Absolute Deviation (MAD) of Moment of Learning prediction. This is the column which tells us if the metrics used to evaluate performance are correlated with error in mastery / Moment of Learning prediction. The fourth column gives correlations of Moment of Learning MAD and metric values for predicting internal knowledge state. This correlation captures agreement between identifying the moment student learned a skill (this happens once per student-skill tuple) and the correctness of identifying the skills knowledge state for the student across all skill attempts.

7.3 Experiment 2

Hypothetically, the ground truth parameter sets should be the best at both making predictions of performance and estimating knowledge. A good metric should favor the ground truth parameters, therefore we ask: How often is the ground truth model the best at prediction performance according to the various metrics?

7.4 Results

The correlations of the performance and knowledge state prediction metrics from prior section targeted the 15 model parameter combinations that were different from the generating ground truth model parameters. Now, let us look at how the ground truth model compares to the other 15 we tested with respect to the statistical metrics we chose. Table 3, for each metric, gives the number of times a ground truth model parameter set is the best with respect to a given metric, and an average rank of the ground model parameter set as compared to the

Table 2: Metric correlations

Metric	Correlation of performance and knowledge metric	Correlation of performance metric and Moment of Learning MAD	Correlation of knowledge metric and Moment of Learning MAD
recall	0.878 ***	-0.954 ***	-0.819 ***
F-measure	0.561 ***	-0.839 ***	-0.792 ***
accuracy	0.522 ***	-0.802 ***	-0.822 ***
precision	0.334 ***	-0.797 ***	-0.628 ***
RMSE	0.470 ***	0.754 ***	0.828 ***
AIC	0.375 ***	0.751 ***	0.702 ***
AICc	0.375 ***	0.751 ***	0.702 ***
BIC	0.375 ***	0.751 ***	0.702 ***
CBD	0.409 ***	0.751 ***	0.762 ***
log-likelihood	0.375 ***	0.751 ***	0.702 ***
pseudo R^2	0.592 ***	-0.236 *	-0.296 **
AU ROC	0.335 ***	-0.119	-0.652 ***

Note: with respect to correlations with moment of learning MAD, in some cases a negative correlation is desirable (e.g., for accuracy), and for some cases a positive correlation is desirable (e.g., for RMSE). This is due to the fact that the smaller the moment of learning MAD the better, which is true for some metrics and the inverse is true for others. The table is sorted while observing this phenomenon (effectively sorting by the absolute value of the correlation coefficient).

Table 3: ground truth model rank vs. the other 15 models

Metric	Ground truth model has rank of 1	Mean rank of ground truth model
AIC	88/100	1.82/16
AICc	88/100	1.82/16
BIC	88/100	1.82/16
CBD	88/100	1.82/16
log-likelihood	88/100	1.82/16
RMSE	88/100	1.82/16
pseudo R^2	88/100	1.83/16
accuracy	33/100	2.52/16
F-measure	12/100	4.27/16
AU ROC	26/100	4.35/16
recall	0/100	6.65/16
precision	5/100	9.71/16

other 15 model. In each case we are aggregating across 100 different sets of 15 models plus one ground truth model. As we can see log-likelihood based models and RMSE form a group of metrics that gives ground truth models a large edge over the 15 reference models. Confusion table metrics, Area under ROC curve and the pseudo R2 gibe a drastically smaller support for it.

7.5 Experiment 3

Ground truth parameters do not always predict the data the best, but often do when using metrics like RMSE or log-likelihood. Do the parameter sets that are not predicted well by ground truth share a common pattern? Does the relative performance of ground truth correlate with high or low values of prior, learn, guess or slip in the generating parameters?

7.6 Results

Seeing log-likelihood based and RMSE metrics score the ground truth model at the same level of mean rank, we are wondering whether, across all 100 of generating parameter sets, the data produced by the same sets of parameters is equally hard to predict with ground truth model. For that we looked at whether the BKT parameter values correlate with ranks ground truth model receives on the moment of learning MAD metric.

First of all, moment of learning MAD metric ranked ground truth as best only 33/100 times with an average rank of 2.53/16. Correlations of moment of learning MAD ranks for ground truth models showed that theres a small marginally significant effect of pInit probability on the moment of learning MAD score ($r = 0.18$, $p - val = 0.07$). Guessing probability does not correlates with moment of learning MAD ($r = -.06$, $p - val = 0.55$).

Probability of learning and slip probability, however, are very strongly related to the moment of learning metric. The larger the learning rate of a simulated skill is, the higher the rank of the ground truth model is ($r = 0.68$, $p - val < 0.001$). Namely, the faster the skill is learned, the worse job ground truth model is doing. In the case of pSlip, the relation is the opposite: the higher the guess rate is, the higher rank moment of learning MAD assigns to the ground truth model ($r = -0.52$, $p - val < 0.001$).

Both the pLearn and pSlip parameters are controlling the process of skills transitioning into the learned state. Strong negative correlation of moment of learning MAD and pSlip is quite logical. Higher pSlip results in more errors even when the skill is mastered, as a result the transition to the learned state becomes more blurred. In this situation the ground truth model has an edge over other models. However, it is high to explain a high positive correlation of moment of learning MAD and pLearn. Higher pLearn means more correct responses overall, this should put ground truth model at an advantage. Additional investigation is necessary to address this phenomenon.

8 Discussion

In our first experiment we found that three less commonly used accuracy metrics showed the best correspondence to accuracy of moment of learning estimation. These metrics were: recall, F-measure, and accuracy, with recall giving a very high correlation of 0.954. Also noteworthy was the poor performance of AUC

with a correlation of -0.119. This was the worst correlation and suggests that AUC should not be used to determine the relative goodness of models based on prediction performance if the underlying goal is to rank models based on knowledge estimation goodness. Metrics like recall and F-measure ought to be adopted in place of AUC for these purposes.

We also found that ground truth model parameters did not always perform the best and that RMSE and log-likelihood based metrics tended to predicted ground truth being the best parameter set more than the others. AUC, recall, F-measure, and precision, however, were among the worst. Therefore, if the underlying goal of an analysis is to recover ground truth parameters (such as with inferring pedagogical efficacy), RMSE and log-likelihood measures should be used and the aforementioned accuracy metrics should be avoided. The experiments 2 raised the question of why ground truth may not always predict the best experiment 3 indicated that high learning rate and low slip in the generating parameters can prove difficult for mastery prediction.

Overall detecting the moment of learning in the generated data by observing a switch from a string of all 0s (unknown state) to the string of all 1s (known state) is often not easy even when ground truth parameters are used. Especially if guess and slip parameters are larger, several back-and-forths between known and unknown state are common. In the area of ITS it is customary to wait till three correct attempts in a row to be sure student has mastered the underlying skill. In our case, when we assumed the moment of learning is the first time when probability of knowing the skill crosses the 0.95 threshold. Following from recent results on the lag with detecting the moment of learning that occurs in the Bayesian Knowledge Tracing [3], in future, we will experiment with adjustments to our computation of the moment of learning to compensate for this.

References

1. Baker, R.S.J.d., Goldstein, A.B., Heffernan, N.T. (2010) Detecting the Moment of Learning. Proceedings of the 10th Annual Conference on Intelligent Tutoring Systems, 25-34.
2. Corbett, A. T. and Anderson, J. R.: Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction, 4(4), 253-278. (1995)
3. Fancsali, S.E., Nixon, T., Ritter, S. (2013) Optimal and Worst-Case Performance of Mastery Learning Assessment with Bayesian Knowledge Tracing. In: Proceedings of the 6th International Conference on Educational Data Mining.
4. Koedinger, K. R., Anderson, J. R., Hadley, W. H., and Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. International Journal of Artificial Intelligence in Education, 8, 3043.
5. Pardos, Z. A., Wang, Q. Y., Trivedi, S. (2012) The real world significance of performance prediction. In Proceedings of the 5th International Conference on Educational Data Mining. Crete, Greece. pp 192-195.

Impact of Prior Knowledge and Teaching Strategies on Learning by Teaching

Ma. Mercedes T. RODRIGO[1], Aaron ONG[1], Rex BRINGULA[2], Roselle S. BASA[2], Cecilo DELA CRUZ[2], Noboru MATSUDA[3]

[1]Ateneo Laboratory for the Learning Sciences, Department of Information Systems and Computer Science, Ateneo de Manila University, Loyola Heights, Quezon City, Philippines
[2]University of the East, Manila, Philippines
[3]Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA, USA
mrodrigo@ateneo.edu, icemanfresh@yahoo.com
rexbringula@gmail.com, roselle_basa@yahoo.com
noboru.matsuda@cs.cmu.edu

Abstract. We investigate cognitive factors that are predictive of learning gains when students learn to solve equations by teaching a synthetic peer, called SimStudent. Previous empirical studies showed that prior knowledge is strongly predictive of post-test scores. However, in a recent study in the Philippines that replicated our previous study in the USA, there were students with low prior-knowledge who tutored their SimStudent better than other equally low prior students. In this paper, we analyze both process data (tutoring interactions) and outcome data (test scores) to understand what makes learning by teaching more effective. The results imply a presence of individual behavioral differences beyond the difference in the prior knowledge that might have affected SimStudent's learning, which in turn had non-trivial influence on tutor learning.
Keywords. Learning by teaching, teachable agent, SimStudent, Algebra equations, prior knowledge

1. Introduction

Since the late 1990s, researchers have investigated intelligent tutoring systems with intelligent pedagogical agents (often called *teachable agents*) to study a promising type of learning where students learn by teaching [1-3]. These technologies allow researchers to conduct tightly controlled experiments and to collect detailed *process data* representing interactions between students and teachable agents that together provide empirical evidence for the benefit of learning by teaching [4].

Matsuda *et al*. (in print), for example, showed that students' learning significantly correlated with the learning of teachable agents. Biswas *et al*. [5]

studied whether students could learn to self-regulate their teaching activities and how the ability of self-regulation affects the tutor learning. It is therefore of intellectual interest to uncover how the tutoring interaction affects students' learning by teaching.

In the current study, we use SimStudent, which is a teachable agent that helps students learn problem-solving skills by teaching [6]. It has been tested and redesigned several times, resulting in insights regarding the effects of learning by teaching and related cognitive theories to explain when and how students learn by teaching. Previous studies showed that pre-test score were highly predictive of post-test scores when students learn equation solving by teaching SimStudent [7]. In general, when students do not have sufficient prior knowledge on the subject to teach, they are not able to teach correctly and appropriately hence the benefit of learning by teaching would be arguably decreased.

Nonetheless, there are some students with low prior knowledge who learned more than others by teaching SimStudent. Among equally low-prior students, those who showed better performance on the post-test actually tutored their SimStudent better as well. The difference in the learning gain among students with comparable prior-knowledge indicates a presence of effective interaction for learning by teaching that might bootstrap tutor learning even with insufficient prior knowledge.

The goal of this paper is to investigate cognitive factors that affect tutor learning. The central research question is why some students (even with low prior knowledge) learned more than other students with comparable prior knowledge. To address this research question, the current paper analyzes data from two classroom (in-vivo) studies conducted in the USA and the Philippines. The Philippines study was a replication of the USA study reported earlier [8].

In the rest of the paper, we first introduce a learning environment in which students learn to solve linear equations by teaching SimStudent. We will then introduce two classroom studies conducted in the USA and the Philippines followed by the results and discussions.

2. Online Learning Environment with SimStudent

This section provides a brief overview of SimStudent and the online learning environment, Artificial Peer Learning environment using SimStudent (APLUS), in which students learn to solve algebra equations by interactively teach SimStudent. Technical details about SimStudent and APLUS can be found elsewhere [7]

2.1. SimStudent

SimStudent is a synthetic pedagogical agent that acts as a peer learner. It learns procedural skills from examples. That is, a student gives SimStudent

a problem to solve. SimStudent then attempts to solve the problem one step at a time, occasionally asking the student about the correctness of each step. If SimStudent cannot perform a step correctly, it asks the student for a hint. To respond to this request, the student has to demonstrate the step.

Students may not be able to provide the correct feedback and hints. As SimStudent is unable to distinguish correct from incorrect feedback, it continues to try to generalize examples and generate production rules that represent the skills learned. SimStudent is also capable of making incorrect inductions that would allow SimStudent to learn incorrect productions *even when students teach SimStudent correctly*. SimStudent's ability to model students' incorrect learning is one of the unique characteristics of SimStudent as a teachable agent.

2.2. APLUS: Artificial Peer Learning Environment using SimStudent

Figure 1 shows an example screen shot of APLUS. In APLUS, students act as a tutor to teach SimStudent how to solve equations. SimStudent is named Stacy and visualized at the lower left corner of APLUS. The tutoring interface allows the student and Stacy to solve problems collaboratively. In the figure, a student poses the problem 3x+6=15 for Stacy to solve. Stacy enters "divide 3" and asks the student whether this is correct. The student responds by clicking on the [Yes/No] button. If the student gets stuck, she can consult the examples tabbed at the top of the screen.

The student has the option of gauging how much Stacy has learned with the use of a quiz. The student chooses when and how often to administer

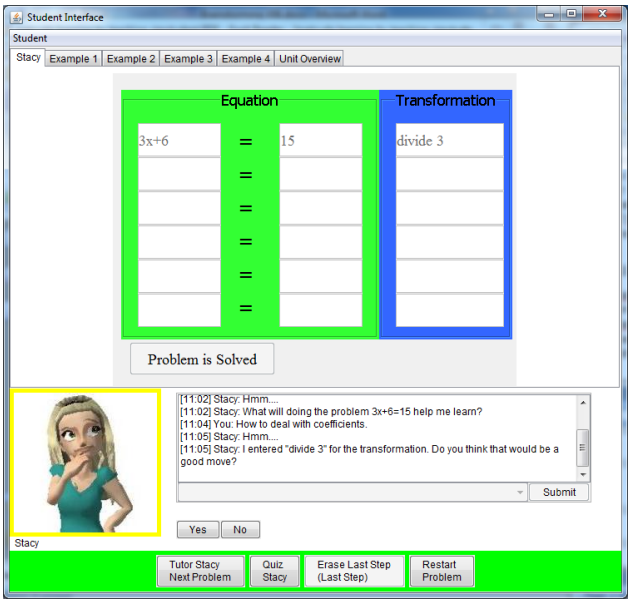


Fig 1. A screen shot of APLUS. SimStudent is visualzed with an avatar image and names Stacy.

the quiz by clicking a button at the bottom of the interface. The quiz interface looks like the tutoring interface, however, when Stacy takes the quiz, she does so independently, without any feedback or intervention from the student. At the end of the quiz, the student is presented with a quiz result.

The quiz is divided into 4 sections, each with two equation problems. The quiz items were created from the mix of one-step, two-step, and target equations (i.e., the equations with variables on both sides).

Stacy cannot progress to a section until she passes the previous section. The students were asked to tutor Stacy to be able to solve equations with variables on both sides. In the classroom studies, the students were informed that their goal was to help Stacy pass all four (4) sections of the quiz.

3. Methods

3.1. Participants

The USA study took place in one high school in Pittsburgh, PA, under the supervision of the Pittsburgh Science of Learning Center [8]. There were eight Algebra I classes with an average of 20 students per class. A total of 160 students with ages ranging from 14 to 15 participated in the study.

The Philippines study took place in one high school in Manila, Philippines, under the supervision of the co-authors from the University of the East and the Ateneo de Manila University. We enlisted participation from five first year high school sections with an average of 40 students per class. There were 201 study participants in all with ages ranging from 11 to 15. The average age of the participants was 12.5 years.

3.2. Structure of the study

In both the USA and the Philippine studies, each participant was randomly assigned to one of two versions of SimStudent: an experimental condition in which Stacy prompted the participants to self-explain their tutoring decisions and a control condition with no self-explanation prompts. The study was designed this way to investigate a particular research question on the effect of self-explanation for tutor learning [8], which is beyond the scope of the current paper. For three consecutive days, participants used their assigned version of SimStudent for one classroom period per day (42 minutes for the USA and 60 minutes for the Philippines study).

3.3. Measures

Students took pre- and post-test before and after the intervention. The students also took a delayed-test two weeks after the post-test was administered. Three versions of isomorphic tests were randomly used for pre-, post-, and delayed-tests to counterbalance the test differences. Students had the entire class period to finish the tests.

The tests are divided into five parts. Of these five parts, three parts are to test procedural knowledge on how to solve equations (the Procedural Skill Test, or PST), whereas other two parts are to test conceptual knowledge about algebra equations (the Conceptual Knowledge Test, or CKT). 102 out of 160 USA participants took all three tests, whereas in the Philippines 146 out of 201 participants took all three tests. In the following analyses, unless otherwise indicated, only those students who took all three tests are included.

The system automatically logged all of the participants' activities including problems tutored, feedback provided, steps performed, examples reviewed, hints requested, and quiz attempts. In the following analysis, we use these factors as process data.

4. Results

4.1. Overall Test Scores

Table 1 shows mean test scores plus or minus SD for the pre, post, and delayed Procedural Skill Tests from two studies. To see how students' test scores varied before and after teaching SimStudent, we conducted a two-way repeated-measures ANOVA with condition as a between-subjects variable and test-time (pre, post, and delayed) as a within-subjects variable. For the USA study, the repeated measure analysis revealed a weak trend for the main effect for test-time. A post-hoc analysis detected a difference from pre-test to post-test [8]. In the Philippines study, the test-time was also the main effect, and the post-hoc analysis detected that delayed-test was significantly higher than pre-test; $t(247.1) = 2.457$, $p < 0.05$. This difference, however, was likely due to the classroom instruction that students were taking during the two-week interval between the intervention and the delayed test.

Both in the USA and the Philippine studies, condition was not the main effect—the presence of self-explanation did not affect tutor learning with the version of APLUS and SimStudent used in two studies.

Table 1: Mean test scores ± SD for pre, post, delayed procedural skill test for each study.

	Pre-test	Post-test	Delayed-test
Philippines (PH)	0.21±0.01	0.22±0.02	0.25±0.03
USA (US)	0.68±0.04	0.71±0.05	0.69±0.06

4.2. Impact of prior knowledge

As shown in Table 1, there was a notable difference in the pre-test scores suggesting that USA students had higher level prior knowledge than Philippine students; $t(142.4) = -22.25$, $p < 0.001$.

To see how prior knowledge affected learning and if the impact of prior knowledge differ between two studies, we ran a regression analysis with post-test score as a dependent variable and study (US vs. PH) as a fixed factor using pre-test score as a covariate. The results showed that pre-test is a strong predictor of post-test; $t(244) = 2.80$, $p < 0.01$. There was also a strong interaction between pre-test and study; the regression coefficient (slope) differed significantly between two studies; $b_{PH} = 0.32$ vs. $b_{US} = 0.76$; $F(1,244) = 11.24$, $p < 0.001$—suggesting that, in general, USA students gained (from pre- to post-test) more than Philippine students. Figure 2 shows the scatter plot for pre-test (x-axis) and post-test (y-axis) scores. USA students (red triangles) had steeper regression line than Philippine students.

4.3. Quiz Results

In the USA study, 36 out of 102(35%) students made their SimStudents pass all four quiz sections. In the Philippines study, no students passed all four sections. At the best, only 7 out of 146 (5%) of Philippine students had their SimStudents pass quiz section 2.

In the Philippines study, there were 73 students who solved quiz item #1 correctly. Of those 73 students, 68 students solved quiz item #2 correctly (hence by definition passing quiz section 1). Of those 68, only 11 students passed quiz section 2 (i.e., solving the first four quiz items correctly).

One possible explanation for the Philippine students' poor performance on the quiz is that Philippine students have insufficient prior knowledge, as indicated by the low pre-test scores and the weak regression slope. A number of factors may account for the difference prior knowledge, including curricular and age differences.

Still, some Philippine students managed to solve the first four quiz items (i.e., passing the quiz section 2), while others did not. Why might this be so? The next section addresses this issue.

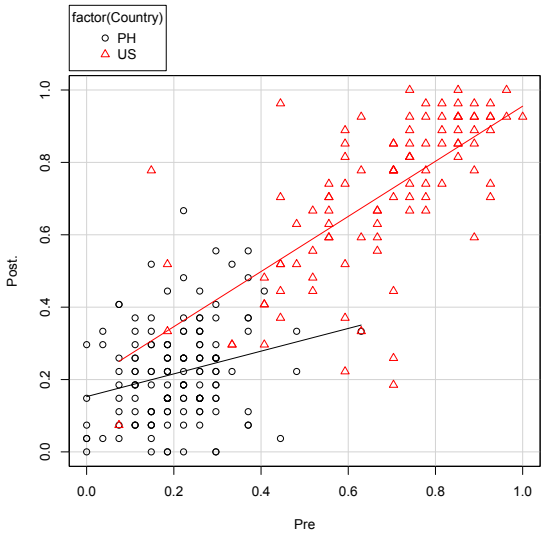


Fig. 2: Scatter plot of pre-test (x-axis) and post-test (y-axis) scores. US students had larger regression slope (0.76) than the PH students (0.32).

4.4. What makes learning by teaching more effective?

To understand why some SimStudents performed better on the quiz than others, we have analyzed the process data. In this analysis, we grouped students depending on the quiz sections their SimStudents passed. We call students whose SimStudents passed and failed quiz section *x* the "passing S*x*" and "failing S*x*" students, respectively. By definition, there were no passing S3 students in the Philippines study.

Our focus in this particular analysis is to understand how some students managed to pass quiz sections in the Philippines study. Therefore, we only included Philippine students for this analysis unless otherwise noted.

4.4.1. Accuracy of tutoring

One cognitive factor that had a significant contribution to tutor learning in the past studies is the accuracy of tutoring—i.e., the accuracy of recognizing correct and incorrect steps made by SimStudent as well as the accuracy the steps demonstrated as hint.

We thus compared the mean accuracy of passing/failing S1 and S2 students. The result suggested that the accuracy of tutoring is a key for success on the quiz in the Philippines study as well. For S1: $M_{Passing}$ = .70 (SD = .14) vs. $M_{Failing}$ = .52 (SD = 0.16); $t(119.3)=-6.89$, $p < 0.001$. For S2: $M_{Passing}$ = .75 (SD = 0.09) vs. $M_{Failing}$ = .59 (SD = 0.18); $t(8.7)=-4.39$, $p < 0.01$.

Students' prior knowledge should have affected tutoring accuracy. There was actually a strong correlation between the prior knowledge (measured as the pre-test score on the Procedural Skill Test) and the accuracy of tutoring. There was also a study difference—USA students tutored more accurately than Philippine students. The centered polynomial regression with the centered pre-test score (i.e., the difference from the mean) as the covariate (C.Pre) and the study (US vs. PH) as a fixed factor predicting the accuracy of tutoring (AT) revealed the following regression coefficients: AT = 0.62 + 0.16*C.Pre + 0.18[if US]; r^2=0.42, $F(2, 235)=88.31$, $p<0.001$; meaning that *Philippine students at the average procedural skill pre-test tutored with a 62% accuracy rate. USA students tutored 18% more accurately than Philippine students in general.* There was no study difference for the regression slope—suggesting that the prior knowledge affected the accuracy of tutoring equally in two studies.

A further analysis that compared passing and failing S1 students revealed that the prior knowledge was not the dominant factor that affected the accuracy of tutoring. In the Philippines study, the average pre-test score of the Procedural Skill Test for passing S1 students (*M*=.21, *SD*=0.10) was not higher than failing S1 students (*M*=.20, *SD*=0.09). However, the average accuracy of tutoring was higher for passing S1 students (*M*=.70, *SD*=.14) than failing S1 students (*M*=.52, SD=0.17).

As for the students' learning, there was a weak trend on the average normalized gain from pre- to post- favorable to passing S1 students (*M*=.05, *SD*=0.22) than failing S1 students (*M*=.01, *SD*=0.18); $t(92.3)=-0.46$, $p=0.65$.

This indicates that *the passing S1 students in the Philippines study learned more by teaching than the failing S1 students although where was no significant difference of the prior knowledge among them*. There might have been difference in the way passing and failing S1 students tutored SimStudent. The next section shows the results on analyzing process data.

4.4.2. Tutoring strategies

Since quiz items were fixed, using quiz items for tutoring could be a good strategy to help SimStudent pass the quiz. Actually, in the USA study, passing S4 students showed a higher percentage of using quiz problems for tutoring (M_{US} = .95, SD = .11) than failing S4 students (M_{PH} = .59, SD = .42); $t(28) = -4.08$, $p < 0.001$.

Thus, we first investigated whether passing S1 and S2 students in the Philippines study used more quiz items for tutoring than failing S2 students. We found that only 47% (1826 out of 3898) problems tutored in the Philippines study were the quiz items. Philippine students did not copy quiz items for tutoring as often as the successful (i.e., passing S4) USA students.

If time on task were a crucial factor for learning by teaching, then students who tutored on more problems should have learn more than those who tutored on fewer problems. To test this hypothesis, we first analyzed if passing S1 students simply tutored more problems than failing S1 students. The average number of problems tutored was 28.9±14.6 for passing S1 students and 20.9±12.2 for failing S1 students. The difference was not statistically significant. There was no notable difference in the number of problems tutored between passing and failing S1 students.

4.4.3. Resource usage

Did passing S1 students self-learn the materials by using resources more than failing S1 students? When counting the number of times students referred to worked-out examples, there was actually a notable difference. The passing S1 students referred to worked-out examples more than failing S1 students; $M_{Passing\ S1}$ (N=52) = 164±116 vs. $M_{Failing\ S1}$ (N=79) = 106±94; $t(93.19) = -3.00$, $p < 0.01$.

Furthermore, passing S1 students copied more example problems for tutoring than failing S1 students; $M_{Passing\ S1}$ = 2.2 vs. $M_{Failing\ S1}$ = 1.4; $t(111.16) = -3.62$, $p < 0.001$. Even when students did not actually understand how to solve equations, they could simply copy worked-out examples line by line to tutor SimStudent, which should have certainly affected SimStudent's ability to pass the quiz.

There was also a significant correlation between the number of example problems tutored and number of times example tab were clicked; r^2=0.36, $t(133)$=8.67, $p < 0.001$—suggesting that *Philippine students were actually switching between tutoring interface and example tabs frequently when they were copying example problems and their solutions for tutoring*.

4.4.4. Predictor of learning

Since there were several factors that contributed SimStudent's and students' learning found in the data, we conducted a regression analysis to see how certain factors contributed to the post-test score on the procedural skill test. The following variables were entered in the regression model: pre-test score on the Procedural Skill Test, total number of problems tutored, total number of quiz items tutored, total number of examples viewed, total number of example problems tutored, accuracy of tutoring, and study.

The result showed that pre-test score, accuracy of tutoring (AT), and study were significant predictors of post-test score (PTS) on the Procedural Skill Test. When pre-test score was centered (C.Pre), the following regression coefficients were revealed: PST = 0.21 + 0.61*C.Pre + 0.23*AT + 0.14[if US]; $r^2 = 0.77$, $F(3, 234)=267.7$, $p < 0.001$. Since pre-test and accuracy of tutoring are highly correlated, dropping accuracy of tutoring from the model also showed an equally good fit: PST = 0.34 + 0.63*C.Pre + 0.34[if US]; $r^2 = 0.76$, $F(2, 245) = 399.3$, $p < 0.001$.

5. Discussions and Concluding Remarks

We found that the prior knowledge had a strong influence on tutor learning—if students do not have sufficient prior knowledge for tutoring, they would not benefit from tutoring as much as students who have appropriate prior knowledge. The regression model mentioned in the results section shows that prior knowledge is the dominating predictor of post-test score for the Procedural Skill Test.

Nonetheless, in the Philippines study, students who managed to have their SimStudent pass the first quiz section (i.e., the first two quiz problems) outperformed those who failed to do so on the post-test of the Procedural Skill Test (albeit the small effect size) even when there was no pre-test difference between passing and failing students. Students who tutored SimStudent better learned more. The same correlation between SimStudent's and students' learning was observed in previous studies [7].

These results indicate that some students had actually learned how to tutor better SimStudent via the actual tutoring interaction. We found that, in the Philippines study, students who managed their SimStudent to pass the first two sections of the quiz copied worked-out examples more often than those who failed to pass the quiz. Furthermore, those passing students reviewed the worked-out examples more often than failing students. Further investigation would be necessary to understand how to better assist students with low prior knowledge to learn by teaching.

Learning by teaching is a promising type of learning especially when combined with an advanced agent technologies. Yet, there are many to understand when and how students learn by teaching and how to best facilitate their learning with various individual differences.

6. Acknowledgements

The authors thank the Ateneo Laboratory for the Learning Sciences, Marc Lester Armenta, Regina Ira Antonette M. Geli, Victoria Keiser, Gabriel Jose G. Vitug, and Evelyn Yarzebinski. We thank the Department of Science and Technology Philippine Council for Industry, Energy, and Emerging Technology Research and Development (PCIEERD) for the grant entitled, "Development of Affect-Sensitive Interfaces" and the Engineering Research and Development for Technology (ERDT) program for the grant entitled, "Development of an Educational Data Mining Workbench."

7. References

1. Chin, D., et al., *Preparing students for future learning with Teachable Agents.* Educational Technology Research and Development, 2010. **58**(6): p. 649-669.
2. Pareto, L., et al., *A Teachable-Agent Arithmetic Game's Effects on Mathematics Understanding, Attitude and Self-efficacy*, in *Proceedings of the International Conference on Artificial Intelligence in Education*, G. Biswas, et al., Editors. 2011, Springer: Heidelberg, Berlin. p. 247-255.
3. Uresti, J.A.R. and B. du Boulay, *Expertise, Motivation and Teaching in Learning Companion Systems.* International Journal of Artificial Intelligence in Education, 2004. **14**(2): p. 193-231.
4. Roscoe, R.D. and M.T.H. Chi, *Understanding tutor learning: Knowledge-building and knowledge-telling in peer tutors' explanations and questions.* Review of Educational Research, 2007. **77**(4): p. 534-574.
5. Biswas, G., et al., *Measuring Self-Regulated Learning Skills through Social Interactions in a teachable Agent Environment.* Research and Practice in Technology Enhanced Learning, 2010: p. 123-152.
6. Matsuda, N., et al., *Learning by Teaching SimStudent – An Initial Classroom Baseline Study comparing with Cognitive Tutor*, in *Proceedings of the International Conference on Artificial Intelligence in Education*, G. Biswas and S. Bull, Editors. 2011, Springer: Berlin, Heidelberg. p. 213-221.
7. Matsuda, N., et al., *Cognitive anatomy of tutor learning: Lessons learned with SimStudent.* Journal of Educational Psychology, in print.
8. Matsuda, N., et al., *Studying the Effect of Tutor Learning using a Teachable Agent that asks the Student Tutor for Explanations*, in *Proceedings of the International Conference on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL 2012)*, M. Sugimoto, et al., Editors. 2012, IEEE Computer Society: Los Alamitos, CA. p. 25-32.