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-Echeagaray, and 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. 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. 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 [1], meta-cognitive [2], and affective [4] processes during learning based on classification of episodes in log data. Steps have been taken toward detecting when learning occurs [1] and to predict how much of the acquired knowledge students can apply to other situations [2]. However, an obstacle in such research is the lack of generality of the detectors for tutoring systems involving problem solving tasks, especially when trying 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 from one application to another often necessitates a detailed analysis of the new domain and how the tutoring system guides learners to acquire its skills and knowledge. 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 AMT.

K. Yacef et al. (Eds.): AIED 2013, LNAI 7926, pp. 766–769, 2013.

AMT teaches students how to create and test a model of a dynamic system. 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 software 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 [6], which decomposes the overall modeling problem into a series of "atomic" modeling problems whose small scope encourages students to engage in deep modeling rather than shallow guess-based modeling strategies. To assess students, the project needed detectors that detect shallow and deep modeling practices both with and without the meta-tutor.

2 Task Modeling: Analysis of User's Actions on Software

A task model is a formal representation of the user's activity in an interactive system. 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 order 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. The task modeling language K-MAD and its task model creation and simulation environment, K-MADe [3] were chosen because they enable the creation and replay of scenarios of student's actions and they enable a formal verification of the model.

The task model developed with K-MADe was used to define the episode structure. 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:

- 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.
- 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.

3 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. One use of the detectors was to consider the proportion of shallow versus deep learning as an outcome measure in the transfer phase. This was used as a dependent measure of shallow learning in an experimental study investigating the effectiveness of the meta-tutor versus the original interface, described in [6]. The second use of the detectors was to help drive the behavior of an affective learning companion in the current phase of the AMT project [5]. A limitation of the method however 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. 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, Part I. LNCS, vol. 6094, pp. 25–34. Springer, Heidelberg (2010)
2. Baker, R.S.J.d., Gowda, S.M., Corbett, A.T., Ocumpaugh, J.: Towards automatically detecting whether student learning Is shallow. In: Cerri, S.A., Clancey, W.J., Papadourakis, G., Panourgia, K. (eds.) ITS 2012. LNCS, vol. 7315, pp. 444–453. Springer, Heidelberg (2012)
3. Caffiau, S., Scapin, D., Girard, P., Baron, M., Jambon, F.: Increasing the expressive power of task analysis: Systematic comparison and empirical assessment of tool-supported task models. Interacting with Computers 22(6), 569–593 (2010)
4. D'Mello, S.K., Lehman, B., Person, N.: Monitoring affect states during effortful problem solving activities. International Journal of Artificial Intelligence in Education 20(4), 361–389 (2010)
5. Girard, S., Chavez-Echeagaray, M.E., Gonzalez-Sanchez, J., Hidalgo-Pontet, Y., Zhang, L., Burleson, W., VanLehn, K.: Defining the behavior of an affective learning companion in the affective meta-tutor project. In: Chad Lane, H., Yacef, K., Mostow, J., Pavlik, P. (eds.) AIED 2013. LNCS (LNAI), vol. 7926, pp. 21–30. Springer, Heidelberg (2013)
6. Zhang, L., Burleson, W., Chavez-Echeagaray, M.E., Girard, S., Gonzalez-Sanchez, J., Hidalgo-Pontet, Y., VanLehn, K.: Evaluation of a meta-tutor for constructing models of dynamic systems. In: Chad Lane, H., Yacef, K., Mostow, J., Pavlik, P. (eds.) AIED 2013. LNCS (LNAI), vol. 7926, pp. 666–669. Springer, Heidelberg (2013)