Experimental Evaluation of Automatic Hint Generation for a Logic Tutor

John C. Stamper[1], Michael Eagle[2], Tiffany Barnes[2], Marvin Croy[3]

[1]Human-Computer Institute, Carnegie Mellon University
[2]Department of Computer Science, University of North Carolina at Charlotte
[3]Department of Philosophy, University of North Carolina at Charlotte
[1]john@stamper.org, [2]{maikuusa, tiffany.barnes}@gmail.com, [3]mjcroy@uncc.edu

Abstract. In our prior work we showed it was feasible to augment a logic tutor with a data-driven Hint Factory that uses data to automatically generate context-specific hints for an existing computer aided instructional tool. Here we investigate the impact of automatically generated hints on educational outcomes in a robust experiment that shows that hints help students persist in deductive logic courses. Three instructors taught two semester-long courses, each teaching one semester using a logic tutor with hints, and one semester using the tutor without hints, controlling for the impact of different instructors on course outcomes. Our results show that students in the courses using a logic tutor augmented with automatically generated hints attempted and completed significantly more logic proof problems, were less likely to abandon the tutor, and performed significantly better on a post-test implemented within the tutor.

Keywords: data mining, machine learning, logic tutor

1 Introduction

In our previous work, we added the Hint Factory, an automatic hint generator, to the Deep Thought logic proofs tutor to automatically deliver context specific hints to students solving logic proofs, demonstrated its feasibility on historical data [2], and ran a pilot study to ensure hints were delivered correctly and appropriately [3]. We now evaluate its impact on educational outcomes, by adding hints to eight of eleven logic proof problems and examining their use in six deductive logic classes across two semesters. To control for the impact of different instructors, each of the three college philosophy instructors taught one semester of Deductive Logic with the Deep Thought tutor augmented with our data-derived hints, and each taught a semester using the tutor without hints. We hypothesized that providing our context specific hints automatically generated from data would improve students' ability to solve the given proof problems, and that having these hints available while students are solving practice problems would improve overall learning of the material.

We tested our first hypothesis on the impact of hints by examining the attempt and completion rates between students with hints and those without on three levels of problems. We tested the second hypothesis on overall learning by testing the learning on two post-test problems where no hints were available for any students. The results show that students in the hint group attempt and complete significantly more problems than students in the no-hint group. Further, students who were given hints on early problems outperformed students without hints on the post-test.

2 Background and Related Work

Marking student work as right or wrong is a simple form of feedback that can often be automated, but automatically generating effective formative feedback is a much more complex problem. Shute's review of the literature suggests that effective formative feedback be multidimensional and credible, specific but not evaluative, and infrequent but timely [14]. Determining the timing and frequency of hints is a particular challenge, but studies suggest that offering hints on demand, instead of proactively, can have positive effects on learning [11]. While some studies have suggested as much as 72% of help-seeking behaviors can be unproductive [1], Shih's work suggests that some of these behaviors are in fact helpful [13]. Shih argues that using help to achieve a bottom-out hint can be seen as looking for a worked example, an effective learning strategy [13].

Based on the hint and help literature, we devised a strategy of automatically generating hints to be as specific as possible, derived on-demand, and directed to the student's problem-solving goal, to provide the right type of help at the right time. Based on our experience in teaching logic for many years, we have observed that students often know how to execute the steps needed to solve logic proof problems but may have trouble choosing what to do next. These observations confirm that our on-demand, context-specific system could address the needs of students solving logic proof problems, but the research in our current study was needed to evaluate whether our implemented system achieved that goal.

Historically, the research and development of intelligent tutoring systems (ITS) have relied on subject area experts to provide the background knowledge to give hints and feedback. Two classes of effective tutors, cognitive tutors and constraint based tutors, rely on "rules" that experts create in a time-intensive process [8]. While this expertise and time are limited, the amount of data being collected from computer aided instruction continues to grow at an exponential rate. Data-driven methods applied to large data repositories like the PSLC DataShop [7] can enable the rapid creation of new intelligent tutoring systems, making them accessible for many more students.

As with RomanTutor, an ITS that uses sequential pattern mining over collected data to recommend actions to astronauts learning to operate a robot arm [10], we provide direct, data-driven feedback in an environment where students can choose from a large space of actions to perform and many are correct. We construct Markov Decision Processes (MDPs) that represent all student approaches to a particular problem, and use these MDPs directly to generate hints with the Hint Factory [3]. Barnes and Stamper demonstrated the feasibility of this approach on historical data, showing that extracted MDPs with our proposed hint-generating functions could provide hints over 80% of the time [2]. Fossati and colleagues have used our MDP method in the iList tutor used to teach linked lists and deliver “proactive feedback” based on previous student attempts [6]. In a pilot study, we augmented Deep Thought with the Hint Factory and showed that students were able to solve more logic proof problems when hints were included [3]. Almost the opposite of Bootstrap Novice Device (BND), which bootstraps example-based tutors with student data [9], we create a data-driven tutor that can be bootstrapped with expert solutions [12], providing at least some automatically generated hints initially and improving as additional student problem attempts are added to the model.

3 Hint Factory & Deep Thought Tutor

The Hint Factory consists of the MDP generator and the hint provider. The MDP generator is created through an offline process that assigns values to states in student problem attempt data. The hint provider uses these values to select the next “best” state at any point in the problem space.

The MDP Generator uses historical student data to generate a Markov Decision Process (MDP) that represents a student model, containing all previously seen problem states and student actions. A Markov decision process (MDP) is defined by its state set S, action set A, transition probabilities T, and a reward function R [15]. The goal of using an MDP is to determine the best policy, or set of actions students have taken at each state *s* that maximize its expected cumulative utility (V-value) which corresponds to solving the given problem. The expected cumulative value function can be calculated recursively using equation (1). For a particular point in a student’s logic proof, a state consists of the list of statements generated so far, and actions are the rules used at each step. Actions are directed arcs that connect consecutive states. Therefore, each proof attempt can be seen as a graph with a sequence of states connected by actions.

We combine all student solution graphs into a single graph, representing all of the paths students have taken in working a proof. Next, value iteration is used to find an optimal solution to the MDP. For our experiments, we set a large reward for the goal state (100) and penalties for incorrect states (10) and a cost for taking each action (1), resulting in a bias toward short, correct solutions such as those an expert might derive. We apply value iteration using a Bellman backup to iteratively assign values $V(s)$ to all states in the MDP until the values on the left and right sides of equation (1) converge [15]. The equation for calculating the expected reward values $V(s)$ for following an optimal policy from state *s* is given in equation (1), where $R(s,a)$ is the reward for taking action *a* from state *s*, and $P_a(s, s')$ is the probability that action *a* will take state *s* to state s'. $P_a(s, s')$ is calculated by dividing the number of times action *a* is taken from state *s* to s' by the total number of actions leaving state *s*.

$$V(s) := \max_a \left(R(s,a) + \sum_{s'} P_a(s,s')\ V(s') \right) \qquad (1)$$

Once value iteration is complete, the optimal solution in the MDP corresponds to taking an expert-like approach to solving the given problem, where from each state the best action to take is the one that leads to the next state with the highest expected reward value [3]. The Hint Factory uses these values when a student is in a particular state to choose the next "best" state from which to generate a hint. When the hint button is pressed, the hint provider searches for the current state in the MDP and checks that a successor state exists. If it does, the successor state with the highest value is used to generate a hint sequence.

We have augmented Deep Thought, a custom online tool implemented as a Java applet, whose graphical interface allows students to visually connect premises and apply logic rules to solve logic proof problems [5], with the Hint Factory. For students with hints, a hint button appears, as shown at the lower right in Figure 1, when a student loads a problem. The button is bright yellow to make it more visible. When a new problem with hints is selected, the hint provider loads the entire hint file into memory. The Hint Factory for Deep Thought generates four types of hints: 1) indicate a goal expression to derive, 2) indicate the rule to apply next, 3) indicate the premises where the rule can be used, and 4) a bottom-out hint combining 1-3 [see 3 for more details].

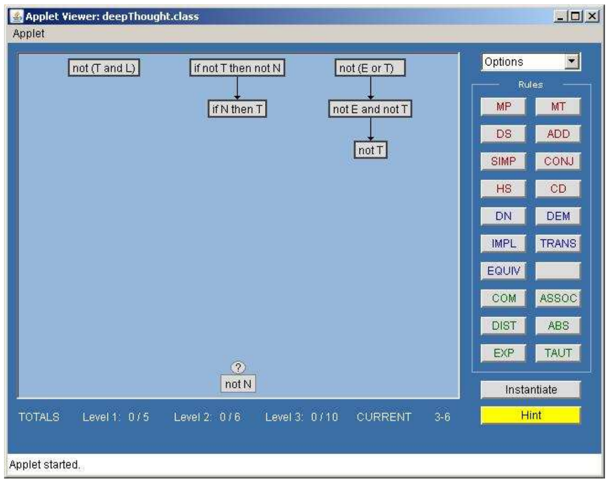

Fig. 1. The Deep Thought tutor showing a partially completed solution to problem 3-6. The student can select statements on the left side and apply rules from the buttons on the right. The added hint button in the lower right was only visible to students in the Hint group.

4 Experiment

Students from six different sections of a deductive logic course used the Deep Thought tutor. The sections included three sections in the Spring 2009 semester and three sections in the Fall 2009 semester. Each of three college philosophy professors taught one section each semester, one semester using Deep Thought with the Hint Factory and one semester using Deep Thought with no hints. This controlled for effects from different instructors by switching between the experimental and control conditions between the two semesters. Students in the Hint group classes could receive unlimited hints on the eight problems that had them and the Control group received no hints throughout. Students generally completed the problems, in order, over the course of the semester, but could access the problems at any time. Students accessed the Deep Thought tutor via the Moodle learning management system used to administer the course. All three classes used the same learning management system and were assigned the same problems. In the first semester there were 82 students in the Hint group and 37 students in the Control group; in the second semester there were 39 students in the Hint group and 83 in the Control group. Students with no log-data were dropped from the study; resulting in 68 and 37 students in the Hint group, and 28 and 70 students in the Control group for the spring and fall semesters respectively. This results in a total of 105 students in the Hint group and 98 students in the Control group.

Students from the six classes were assigned 13 logic proofs in the Deep Thought tutor. We have organized these problems into three constructs: level one (L1) consisting of the first 6 problems assigned, which use only inference rules; level two (L2) consisting of 5 problems using replacement and inference rules; and the post test (L3) consisting of the last two problems assigned. We use L3 as a post test measure since there were no hints for these two problems for either group.

5 Results and Discussion

We tested Hypothesis 1 by measuring the average percentage of completed problems for each of the three levels for each group, as shown in Table 1. Students were given credit if and only if they found a solution to the problem. To investigate the differences in performance between the groups, we submitted the results of L1, L2, and L3 to between-subjects two-tailed tests with an alpha of .05. The Hint group performed significantly better on all three levels, as shown in Table 2.

Table 1. Percent problem completion rates for each level. Students in the Hint group completed significantly more problems in each of the 3 levels. The L3 level had no hints for either condition and also acted as a post test measure.

Group	N	L1 Mean*	L1 SD	L2 Mean*	L2 SD	L3 Mean*	L3 SD
Hint	105	78.17	31.17	67.4	37.8	59.0	45.0
Control	98	61.17	36.17	42.4	40.8	41.5	46.0

Next we examined the effects on student motivation by comparing the number of problems attempted for each group. Table 3 shows the average percent of problems attempted and standard deviation for each of the three levels. Each of these percentages is from a total of six, five, and two problems in L1, L2, and L3 respectively. Students were given credit if they attempted to solve the problem, even if they did not find a solution (logs were flagged as solved when students solved a problem but data was collected even if a solution was not found). To investigate the differences in student motivation we submitted the attempt rates for L1, L2, and L3 to between-subjects two-tailed tests with an alpha of .05. There was no significant difference between the attempt rates for L1 with d = .27. The Hint group attempted significantly more L2 and L3 problems, with t (186.94) = -3.07, p =.002, d = .44, for L2, and t (195.33) = -2.32, p =.021, d = .33 for L3. This suggests that students begin the first level with the same motivation, but as the class continues, having hints available keeps students engaged so they attempt more problems in the later levels.

Table 2. Two-tailed t-test results comparing performance between the Hint and Control groups. All results are significant.

Level	t-test results	P level	Effect size
L1	t (191.82) = -3.58	p <.001	d = .51
L2	t (196.71) = -4.52	p <.001	d = .64
L3	t (201) = -2.78	p =.006	d = .39

Table 3. Percent problem attempt rates for each level. Students in both group attempt roughly the same number in L1, but the Hint group attempts significantly more problems in L2 and L3.

Group	**N**	**L1 Mean**	**L1 SD**	**L2 Mean***	**L2 SD**	**L3 Mean***	**L3 SD**
Hint	105	83.17	27.17	73.2	35.0	71.5	41.5
Control	98	75.33	31.00	56.0	43.2	57.0	46.0

To further test the effects of hints on student performance and persistence we looked at the overall rate in which students abandoned the Deep Thought tutor after the first level (L1) as seen in Table 4. Twenty-eight percent of the Control group abandoned the tutor after L1 (classified as Dropped below), while only 10% of the Hint group stopped attempting Deep Thought problems. A chi-square test of the relationship between group (Hint, Control) and Dropout (Continued, Dropped) produced $\chi^2(1) = 11.05$, which is statistically significant at p = .001. This is associated with an odds ratio of 3.62, indicating that the odds of dropping after the first level are more than 3.6 times higher when the students are not provided hints. This is a meaningful difference that suggests that online computer-aided instruction tools could benefit greatly from being augmented with automatically generated hints using the Hint Factory.

Table 4. Number of students that continued or dropped out of the tutor after L1.

Group	**Continued**	**Dropped**	**Total**
Hint	95	10	105
Control	71	27	98

These results suggest that automatically generated hints keep students engaged and motivated to continue through the tutor. The mechanism for this effect may lie in the ability of students who are frustrated to ask for more help in Deep Thought plus the Hint Factory, while students in the Control group had no such alternative.

Interestingly, we have observed students using the Hint Button for more than just asking for hints. The Hint Button is visible whenever a problem has hints, and is enabled when a specific hint is available (which is about 80% of the time). When the button is disabled, its label is grey and clicking on it has no effect. When students perform a step that does not already exist in our MDP, the Hint Button is disabled, and some students have observed this. After several steps, if the student gets stuck the student may want a hint but know that none are available. We have observed that, rather than give up, students will delete their steps until the Hint Button becomes available again. In cases where steps do not exist in our MDPs, it is often the case that the student has tried something correct, but unusual, that may not lead to a solution. We believe that some students may have learned this through experience with the Hint Button and are thus receiving some tacit hints that particular steps in their work are unusual or unorthodox. Since students often generate several steps when they don't know a good strategy for moving ahead in a logic proof problem, this tacit help, while not evaluative, may be adding up to gentle nudges in the right direction. This can be thought of like seeing a subtle worn path while walking in the forest – you can go another way, but some who've gone before have taken the path.

6 Conclusions and Future Work

The main contribution of this work was to show that adding automatically generated hints to Deep Thought increased the attempt and completion rate for students solving logic proofs, independent of instructor or semester. Our results showed that the Hint group had significantly higher completion rates for all three levels of problems. We also showed that, while the two groups were equally motivated to attempt solving problems in level L1, the Hint group attempted significantly more problems in levels L2 and L3. Furthermore, students without hints were 3.6 times more likely to quit using the tutor ***altogether*** after the first level of problems.

When we consider L3 as a post-test measure that both groups completed without hints after levels L1 and L2, we see an overall learning effect for students in the Hint group, who were able to complete significantly more problems. This means that having hints early on helps students later when hints are not available, and suggests that having hints improves overall learning of logic proof solving.

In our future work, we plan to further analyze data for these six courses to more fully understand how students used hints. For example, we have collected but were unable to analyze pre and post test data for this research because it includes data for other topics in deductive logic. We plan to partition out the questions related to logic proofs and use these to group students by ability to investigate the hint usage patterns for each ability level. We are also inspired by Beck's excellent work on when to provide help based on his modification of Bayesian knowledge tracing [4]. The MDP method and Hint Factory serve as a model tracer that provide hints for step-based problem solving but does not use any information about student knowledge because that is not explicitly modeled. In the future we hope to compare methods for modeling knowledge in the logic domain using Bayes Nets and comparing these with methods of creating MDPs tailored to students at different knowledge levels. We also plan to extend our methods to create effective data-driven intelligent tutors for other step-based problem solving domains.

References

1. Aleven, V., McLaren, B., Roll, I., & Koedinger, K. (2004). Toward tutoring help seeking: Applying cognitive modeling to meta-cognitive skills. *Proceedings of the Seventh International Conference on Intelligent Tutoring Systems.*
2. Barnes, T., Stamper, J. (2008). Toward Automatic Hint Generation for Logic Proof Tutoring Using Historical Student Data. *In E. Aimeur, & B. Woolf (Eds.) Intelligent Tutoring Systems (ITS 2008)*, pp. 373-382. Berlin, Germany: Springer Verlag.
3. Barnes, T., Stamper, J., Lehmann, L., Croy, M.: A Pilot Study on Logic Proof Tutoring Using Hints Generated from Historical Student Data. *In R. Baker, T. Barnes, J. Beck (Eds.) Educational Data Mining (EDM 2008)*, pp. 197—201. Montreal, Canada. (2008)
4. Joseph E. Beck, Kai-min Chang, Jack Mostow, Albert T. Corbett: Does Help Help? Introducing the Bayesian Evaluation and Assessment Methodology. Intelligent Tutoring Systems 2008: 383-394
5. Croy, M., Barnes, T. & Stamper, J.: Towards an Intelligent Tutoring System for propositional proof construction. *In Brey, P., Briggle, A. & Waelbers, K. (eds.), European Computing and Philosophy Conference*, pp. 145—155, Amsterdam, Netherlands: IOS Publishers (2007)
6. Fossati, D., Di Eugenio, B., Ohlsson, S., Brown, c., Chen, L., Cosejo, D. I learn from you, you learn from me: How to make iList learn from students*. In V. Dimitrova, R. Mizoguchi, B. Du Boulay and A. Graesser (Eds.), Proc. 14th Intl. Conf. on Artificial Intelligence in Education, AIED 2009*, pp. 186—195., Brighton, UK. IOS Press (2009)
7. Koedinger, K.R., Baker, R.S.J.d., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J. (2010) A Data Repository for the EDM commuity: The PSLC DataShop. *In Romero, C., Ventura, S., Pechenizkiy, M., Baker, R.S.J.d. (Eds.) Handbook of Educational Data Mining*. Boca Raton, FL: CRC Press.
8. Mitrovic, A., Koedinger, K. & Martin, B.: A comparative analysis of cognitive tutoring and constraint-based modeling. *User Modeling*: 313—322. (2003)
9. McLaren, B., Koedinger, K., Schneider, M., Harrer, A., & Bollen, L.: Bootstrapping Novice Data: Semi-automated tutor authoring using student log files, *In Proc. Workshop on Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes, 7th Intl. Conf. Intelligent Tutoring Systems (IT-2004)*, Maceió, Brazil (2004)
10. Nkambou, R., Mephu Nguifo, E., Fournier-Viger, P.: Using Knowledge Discovery Techniques to Support Tutoring in an Ill-Defined Domain. *In E. Aimeur, & B. Woolf (Eds.) Intelligent Tutoring Systems (ITS 2008)*, pp. 395—405. Berlin: Springer Verlag. (2008)
11. Razzaq, L. & Heffernan, N. (2010). Hints: Is It Better to Give or Wait to be Asked? In Aleven, V., Kay, J & Mostow, J. (Eds). *Proc. 10th Intelligent Tutoring Systems (ITS2010)* Part 1. Springer. Pages 349-358.
12. Stamper, J., Barnes, T. (2010). Enhancing the Automatic Generation of Hints with Expert Seeding. *In J. Mostow, V. Aleven & J. Kay (Eds.) Intelligent Tutoring Systems (ITS 2010)*, vol II pp. 31-40. Berlin, Germany: Springer.
13. Shih, B., Kenneth R. Koedinger, and Richard Scheines. "A Response Time Model For Bottom-Out Hints as Worked Examples." In *Educational Data Mining*, 2008, p. 117-126.
14. Shute, V.J., Focus on formative feedback*. Review of Educational Research*, 2008. 78(1): p. 153-189.
15. Sutton, R. & A. Barto. Reinforcement Learning: An Introduction, 1998, The MIT Press, Cambridge, MA.