Individual Differences and Predictive Validity in Student Modeling

Albert T. Corbett
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
ac21 @andrew.cmu.edu

John R. Anderson
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213
ja0s @andrew.cmu.edu

Valerie H. Carver
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213
vc0e @andrew.cmu.edu

Scott A. Brancolini
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213
sb5p @andrew.cmu.edu

Abstract

This paper evaluates the student modeling procedure in the ACT Programming Tutor (APT). APT is a practice environment that provides assistance to students as they write short programs. The tutor is constructed around a set of several hundred programming rules called the *ideal student model*, that allows the program to solve exercises along with the student. As the student works the tutor maintains an estimate of the probability that the student has learned the rules in the ideal model, in a process we call *knowledge tracing*. The cognitive model, and the learning and performance assumptions that underlie knowledge tracing are described. The assumptions that underlie knowledge tracing also yield performance predictions. These predictions provide a good fit to students' performance in completing tutor exercises, but a more important issue is how well the model predicts students' performance outside the tutor environment. A previous study showed that the model provides a good fit to average posttest performance across students, but is less sensitive to individual differences. This paper describes a method of individualizing learning and performance estimates on-line in the tutor and assesses the validity of the resulting performance predictions.

The ACT Programming Tutor (APT)

This paper describes efforts to model students' changing knowledge state as they complete exercises with the ACT Programming Tutor (APT). The project assumes that a cognitive skill can be represented as a set of production rules and our diagnostic goal is to model changes in the students' knowledge of these rules over time. The ultimate purpose is to predict students' performance levels when working on their own outside the tutor environment and to customize the tutor exercise sequence so that students achieve a satisfactory performance level as rapidly as possible. In the following sections we describe the tutor, the curriculum, the cognitive model, the learning and performance assumptions, individual differences and an evaluation of the model.

APT is a practice environment for students learning to program in Lisp, Prolog or Pascal (Anderson, Corbett, Koedinger & Pelletier, in press). The Lisp and Prolog modules are presently employed to teach a self-paced introductory programming course at Carnegie Mellon University and the Pascal module is in use in an introductory programming course in a Pittsburgh High School. The tutor presents exercises that require students to write short programs and provides assistance as the students code their solutions. This report focuses on the Lisp curriculum and Figure 1 depicts the computer screen at the beginning of the first Lisp exercise. The exercise description appears in the window at the upper left and the student enters a solution in the code window immediately below. This exercise involves just two coding goals or steps. In the figure, the student has just completed the first step by entering the Lisp operator *car*. In the remaining step the student will enter the literal list argument to *car*, *'(c d e)*. The tutor provides immediate feedback on a step-by-step basis and requires immediate error correction, so the student always remains on a recognized solution path.

The skill meter in the upper right corner of Figure 1 represents the focus of this study. It displays the tutor's model of the student's knowledge state. This model consists of a set of production rules for writing Lisp programs. Each rule that has been introduced in the curriculum is represented by a bar graph in the skill meter depicting the probability that the student has learned the rule. In the figure, this probability is .50 for three of the rules, but has risen to .80 for the rule the student has just exercised. In this study we evaluate this student modeling process and how well it predicts student performance outside the tutor.

The Cognitive Model

A central assumption underlying the tutor is that a cognitive skill such as programming can be modeled as a set of independent production rules. Each step in performing a

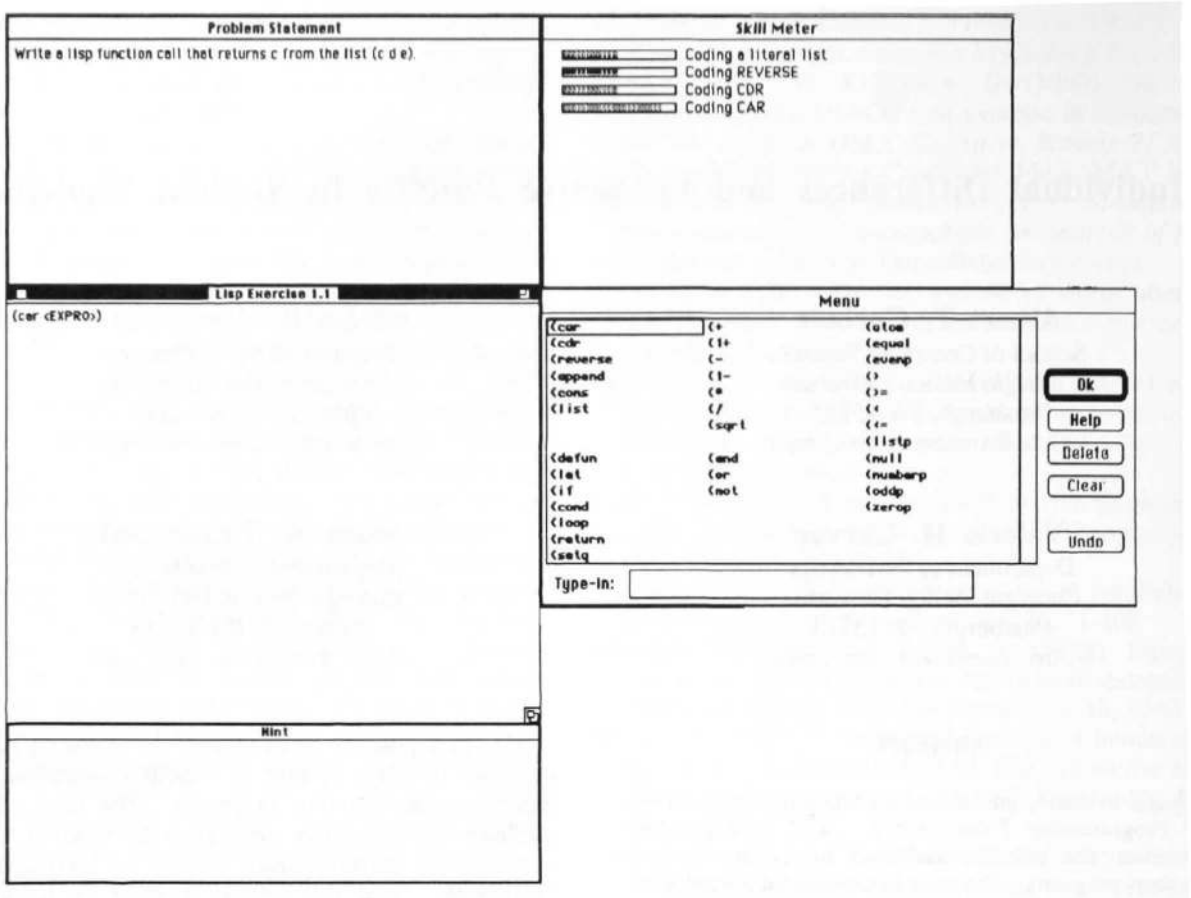

Figure 1. The APT Programming Tutor interface at the beginning of a Lisp exercise.

skill is governed by one of these if-then rules, relating the current goal and problem state to an appropriate action. The tutor is constructed around a set of several hundred language-specific rules for writing programs called the *ideal student model*. The tutor attempts to match the student's action at each step in solving an exercise to an applicable rule in the ideal student model in a process we call *model tracing*. If a match is found, the assumption is made that the student has applied an analogous cognitive rule and the tutor's internal representation of the problem state is updated accordingly. If no match is found, the tutor notifies the student of the error and allows the student to try again.

The Learning Model

The ideal model also serves as an overlay model of the individual student's knowledge state (Goldstein, 1982). In *knowledge tracing*, the tutor maintains a probability estimate that the student has learned each rule in the ideal model. At each opportunity to apply a rule, the probability that the student knows the rule is updated, contingent on whether the student's action was correct or not. The Bayesian computational procedure is a variation of one described by Atkinson (1972). It assumes a simple two-state learning model with no forgetting. Each rule is either in the learned or unlearned state. A rule can make the transition from the unlearned to the learned state at each opportunity to apply the rule, but rules do not make the transition in the other direction. The model employs two learning parameters and two performance parameters, displayed in Table 1. Parameter values are estimated empirically from earlier tutor data. Best fitting estimates have been shown to vary across cognitive rules (Corbett & Anderson, 1992a), so each of the four learning and performance parameters is estimated separately for each production rule in the cognitive model.

Mastery Learning

The tutor uses the knowledge tracing mechanism in an attempt to implement mastery learning. Each lesson in the tutor is divided into sections in which a small set of programming rules is introduced. The student completes a fixed set of required exercises in each section that cover the rules being introduced, then continues working on remedial exercises in the section until the probability that the student has learned each rule in the section has reached a criterion value, 0.95. The knowledge tracing procedure passed a minimum validity test when it was introduced. Posttest scores were higher when the remediation algorithm was in operation (Anderson, Conrad & Corbett, 1989). However, we need to examine the validity of the model in predicting performance to determine the feasibility of genuine mastery learning in which all students are assisted in reaching a high level of performance.

Table 1. The learning and performance parameters employed in knowledge tracing.

$P(L_0)$	The probability a rule is in the learned state prior to the first opportunity to apply the rule (i.e., from reading the text).
P(T)	The probability a rule will make the transition from the unlearned to the learned state following an opportunity to apply the rule
P(G)	The probability a student will guess correctly if the applicable rule is in the unlearned state
P(S)	The probability the student will slip and make an error when the applicable rule is in the learned state

Predictive Validity and Individual Differences in Parameter Estimates

The learning and performance assumptions in the model allow us to predict the probability of correct coding actions. The probability that a student *s* will respond correctly at each goal *g* in problem solving is given by the following equation:

$$P(C_{gs}) = P(L_{rs}) * (1 - P(S_r)) + (1 - P(L_{rs})) * P(G_r)$$

That is, the probability that student *s* will successfully apply an appropriate rule *r* at goal *g* is the sum of two products: (1) the probability that rule *r* is in the learned state for student *s* times the probability of a correct response if the rule is in the learned state, and (2) the probability that rule *r* is not in the learned state for student *s* times the probability of a correct guess if the rule is not in the learned state. These predictions have been shown to fit the tutor data very well (Corbett & Anderson, 1992b; Corbett, Anderson & O'Brien, 1993).

It is important that the model accurately predicts posttest performance outside the tutor environment. Students can deviate from correct solution paths in a posttest, so predictions at the level of individual goals are problematic, however, we can derive performance predictions at the level of whole exercises. If production rules are independent as the model assumes, then the probability that a student will complete an exercise correctly is given by:

$$\Pi P(C_{gs})$$

the product of the probabilities that the student will respond correctly at each successive goal in solving the exercise.

In a previous study, the model proved quite accurate in predicting average posttest performance levels across students, regardless of whether students worked to mastery in the tutor curriculum or just completed a fixed set of exercises (Corbett, Anderson & O'Brien, in press). When the students worked through a fixed exercise set the model was also sensitive to individual differences among students. Actual and expected accuracy was strongly correlated, r = 0.69. When students worked through the remediation process to "mastery" in the tutor, however, the model was insensitive to individual differences among students in posttest performance.

The insensitivity to individual differences when students have nominally reach mastery is not surprising. By definition, when students reach mastery the estimated probability that each student has learned each cognitive rule exceeds 0.95 and there is little variability in predicted test performance across students. If students genuinely master the material, there should also be little variability in actual test performance, or at least little that can be predicted from tutor performance. Instead, test performance correlated reliably a raw measure of tutor performance, overall error rate, r = -0.64. The more errors students made in completing a common set of tutor exercises in working to mastery, the lower they tended to score on the posttest.

This pattern of results, in which test performance correlates inversely with error rates in the tutor, can arise because the model fails to capture individual differences among students in learning. The learning and performance parameter estimates for each cognitive rule are group averages. The model will underestimate the learning state (probability the student has learned the rules) for students who learn quickly (i.e., students whose "true" $p(L_0)$ and p(T) are above average). These students, who will tend to make fewer errors in practice, will get more remediation than necessary and tend to overlearn the material. Conversely, the tutor will overestimate the learning state for students who are learning slowly ("true" $p(L_0)$ and p(T) below average). These students will tend to make more mistakes and get more remediation, but nevertheless will get less remediation than is necessary. This will result in a negative correlation of tutor errors and posttest accuracy. (Individual differences in the performance parameter have opposing effects. Individual differences in the slip parameter will also lead the model to underestimate the learning state of those students making fewer errors and vice versa. Individual differences in the guess parameter work in the opposite direction; the model will overestimate the learning state of students making few errors and vice versa).

Modeling Individual Differences

We need to model differences in learning and performance parameters across both cognitive rules and students. However, a single student does not provide sufficient data to generate student-specific parameter estimates for each rule. Instead, we can start with the best fitting rule-specific parameter estimates across a group of subjects and incorporate individual differences in the form of four subject-specific weights, one weight for each parameter type, wL_0, wT, wG and wS.

We applied this model to the tutor data in our previous study. In fitting each student's data, we converted each of the rule-specific group parameter estimates to odds form, multiplied by the corresponding subject-specific weight, and converted the odds back to a probability as shown here:

$$\frac{pi_r * wi_s}{(1 - pi_r) + pi_r * wi_s}$$

In this formula, *i* indicates the parameter type ($P(L_0)$, P(T), P(G), P(S)), *r* the rule and *s* the student. We generated a set of four best fitting weights across all rules for each subject with a curve-fitting program. The resulting set of weighted rule-specific parameters yields a revised estimate of each student's knowledge state and a revised set of test performance predictions. These revised predictions were more sensitive to individual differences; the correlation of actual and predicted performance was 0.51 (Corbett, Anderson & O'Brien, in press).

While this model is promising, there is a practical problem in deploying this model. The subject weights, like the learning and performance parameters must be estimated empirically. While the rule-specific performance parameters can be estimated from previous tutor data, the same is not true of the subject-specific weights. In applying the model retroactively to the earlier study, we used the student's entire tutor data set to estimate weights. In practice, we need to employ the weights in the tutor when there is only partial data on a student's performance and when there is insufficient time to employ a curve-fitting algorithm in any case. The present study evaluates an attempt to estimate student-specific weights on-line.

The Study

Students in this study worked through the first six sections of the APT Lisp curriculum and completed three posttests. The study is designed to assess the posttest predictive validity of the revised knowledge tracing procedure with individualized learning and performance weights.

Students

Twenty college students were recruited for pay. The students' mean Quantitative and Verbal SAT scores were 628 and 577 respectively. These students had taken an average of 0.6 programming courses previously, but this was their first exposure to Lisp.

The Curriculum

The curriculum in this study introduces two data structures, *atoms* and *lists*, along with *function calls* (operations) and *function definitions*. The first section introduces three extractor functions, *car*, *cdr* and *reverse*, that return components of or a transformation of a list. The second and third sections introduce three constructor functions, *append*, *cons* and *list* that build new lists. In the fourth section nested extractor function calls are introduced. In the fifth section students learn to define new functions that employ these extractor algorithms. In the sixth section students define functions that employ both constructor and extractor functions. This curriculum is covered by 54 rules in the ideal student model and requires a minimum of 64 exercises to complete.

Procedure

Students worked through the curriculum at their own pace. In each section students read text describing Lisp, then completed a fixed set of required exercises that cover the programming rules being introduced. At that point, regression equations, as described below, were employed to generate four learning and performance weights for each student. In the first five sections, students then completed remedial exercises as needed to bring all production rules introduced in the section to a mastery criterion (minimum learning probability of 0.95). In the final section, students completed a fixed set of exercises with no remediation.

Following the first, fourth and sixth sections students completed a cumulative posttest. The quiz interface was a structure editor, identical to the tutor interface, except that (1) students can freely edit their code and (2) no help is available. The posttests contained six, twelve and fifteen programming exercises respectively. These exercises are similar to the ones completed with the tutor. Since students do not work to mastery in the last curriculum section, the final test allows us to assess the predictive validity of the model and its sensitivity to individual differences when there is considerable variability in both the model predictions and actual test scores. The first two tests allow us to assess whether the model is sensitive to any residual variability in actual test performance after all students have nominally reached mastery.

Estimating Weights. When best fitting student weights are derived, as in the previous study, the logarithms of the weights are strongly correlated with the total number of errors students make in the fixed set of required tutor exercises. When a student completed each set of required exercises in the current study, four updated weights were derived based on cumulative errors and the regression equations derived in the earlier study.

Results

Students completed an average of 11 remedial exercises in addition to the 64 required exercises. While the number of remedial exercises ranged from 0 to 26 for nineteen students, the remaining student had substantial difficulty with

constructor functions in the second and third sections and ultimately completed 98 remedial exercises. Nevertheless, this student's quiz performance fell within the range of the other nineteen students and the model's predictive validity for this student was characteristic of the rest of the group.

Internal Validity. We can use the knowledge tracing model to predict accuracy at each goal in the tutor exercises, as described above. To assess the internal validity of the knowledge tracing process we examined the tutor protocol files and traced each student's performance goal-by-goal through the curriculum. At each of the 345 goals in the required exercises, we first predicted the probability of a correct response given the applicable rule, then applied the the knowledge tracing procedure to update the learning probability for the applicable rule. We did not fit (predict accuracy for) the remedial exercises, however we did update learning probabilities at these goals, just as the tutor would. The model yielded a good fit to the data across the 345 goals. The actual probability of a correct response and the predicted probability both averaged 0.90 across goals. Actual and expected accuracy rates were also highly correlated across goals, $r=0.79$, $p < 0.0001$.

Posttest Predictive Validity. As described above, the knowledge tracing model allows us to predict the probability that a student will complete a posttest exercise correctly. For each student and each posttest we computed (1) the proportion of exercises actually completed correctly and (2) the mean expected probability of completing the exercises correctly. The average of these statistics across subjects is displayed in Table 2. As can be seen the model accurately predicts average posttest performance across subjects in the first test, but overestimates performance on the last two tests by roughly 15%.

Table 2 includes two measures concerning individual differences in posttest performance. The correlation of actual posttest performance accuracy and predicted accuracy, r_{AP}, is reported. In addition, the correlation of actual posttest accuracy and raw error rate in the tutor, r_{AE}, is displayed. Results for the first and third posttests replicate those obtained previously with the standard knowledge tracing model. The current model and the standard model are insensitive to individual differences on the first test, but there are no discernible individual differences on this relatively easy test. On the other hand, the current model is highly sensitive to individual differences on the third posttest, $r = 0.81$, $p < 0.0001$. However, this test covers the final tutor section in which students completed a fixed curriculum rather than working to mastery and the standard model is also highly sensitive to individual differences under these circumstances.

The second posttest represents the crucial test for sensitivity to individual differences in the current individualized knowledge tracing model. Students work to mastery in all curriculum sections on this test and in the preceding study the standard knowledge tracing model proved completely insensitive to substantial individual differences in performance that correlated strongly with raw tutor error rate. In the present study, there is again a strong correlation of Posttest 2 performance with raw error rate, $r = -0.63$. Unlike the standard model, however, the individualized knowledge tracing model in this study shows at least moderate sensitivity to these individual differences, $r = 0.31$, $p < 0.10$.

Discussion

The new individualized knowledge tracing model is reasonably successful in predicting average group performance, although it overpredicts posttest performance somewhat. It is more effective than the standard knowledge tracing model in identifying individual differences. Raw tutor error rate remains as good or better a predictor of individual differences, although the ultimate goal of the knowledge tracing model is to structure practice so that this correlation of raw tutor error rate and posttest performance is substantially reduced or eliminated.

There are a variety of possible reasons that the model predicts higher posttest accuracy than is actually obtained: (1) while the model fits the tutor data in general, it may be systematically overpredicting performance on students' final tutor exercises as well as on the posttest; (2) students motivation may drop off for the posttest, resulting in a

Table 2. Actual and expected proportion of exercises completed correctly across subjects in each of the three posttests.

	Mean Proportion Correct			
	Actual	Predicted	r_{AP}	r_{AE}
Posttest 1	.92	.91	0.01	0.05
Posttest 2	.76	.87	0.31	-0.63
Posttest 3	.66	.75	0.81	-0.78

performance decline; (3) students may have forgotten some knowledge acquired with the tutor; and (4) students may acquire suboptimal rules with the tutor that do not fully generalize to the test.

We can largely rule out the first three possibilities in this study, based on some additional data analyses. First, there is no evidence that the model is systematically overpredicting tutor performance toward the end of each section. The exercises can be categorized according to the functions and algorithms employed, and we computed the actual and predicted performance for the final exercise of each type in the tutor. The model actually *under*predicts performance on these exercises by about 5%. Second, we measured total time spent on each quiz as an indirect measure of motivation. Posttest time correlates inversely with posttest performance. That is, students who perform worse on the test are taking longer to complete it, as would be expected if they are struggling. Finally, the decline in students' performance from the tutor to posttest does not appear to reflect forgetting. Each quiz is cumulative, and there are four sections in common to both quizzes 2 and 3. The relative retention interval for rules introduced in each of these four sections is not correlated with performance on the rules either within or across the two tests.

The likely explanation for the model's 15% overprediction of posttest performance is that the rules students are forming do not fully generalize to the posttest environment. Even the early portions of the Lisp curriculum is conceptually challenging and students may form suboptimal rules during practice that are only partially correlated with correct rules (Corbett & Anderson, 1992b, 1993). However, the tutor uses a "dropout" remediation algorithm that may contribute to this problem. In each section of the curriculum a few rules are introduced. As the student "masters" some of these rules they are dropped from the practice sequence and students complete exercises that employ the remaining rules. As practice progresses and the set of remaining rules and corresponding actions declines, there is a growing potential to generate spurious rules based on superficial characteristics of the exercises, or even to simply guess correctly. This pattern would simultaneously lead to a strong inverse relationship between tutor errors and posttest performance, and weaken the predictive validity of the knowledge tracing model. This suggests the remedial algorithm should retain programming rules in the sequence of practice exercises even after the student has "mastered" the rules. While this increases total practice time, it may be necessary for students to genuinely master the material.

Acknowledgement

This research was sponsored by the Office of Naval Research under grant N00014-91-J-1597.

References

Anderson, J.R., Conrad, F. and Corbett, A.T. (1989) Skill acquisition and the LISP Tutor. *Cognitive Science, 13*, 467-505.

Anderson, J.R., Corbett, A.T. Koedinger, K.R. and Pelletier, R. (in press). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences.*

Atkinson, R.C. (1972). Optimizing the learning of a second-language vocabulary. *Journal of Experimental Psychology, 96*, 124-129.

Corbett, A.T. and Anderson, J.R. (1992a) Knowledge tracing in the ACT programming tutor. In *The Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society* (pp. 623-628). Hillsdale, NJ: Lawrence Erlbaum.

Corbett, A.T. and Anderson, J.R. (1992b) Student modeling and mastery learning in a computer-based programming tutor In C. Frasson, G. Gauthier, G. McCalla, (Eds.) *Intelligent tutoring systems: Second international conference on intelligent tutoring systems* (pp. 413-420). New York: Springer-Verlag.

Corbett, A.T. and Anderson, J.R. (1993) Student modeling in an intelligent programming tutor. In E. Lemut, B. du Boulay & G. Dettori (Eds.) *Cognitive models and intelligent environments for learning programming* (pp. 135-144). New York: Springer-Verlag.

Corbett, A.T,, Anderson, J.R. and O'Brien, A.T. (1993). The predictive validity of student modeling in the ACT Programming Tutor. In P. Brna, S. Ohlsson & H. Pain (Eds.) *Artificial Intelligence and Education, 1993: The Proceedings of AI-ED 93* (pp. 457-464). Charlottesville, VA: AACE.

Corbett, A.T. and Anderson, J.R. and O'Brien, A.T. (in press). Student modeling in the ACT Programming Tutor. In P. Nichols, S. Chipman and R. Brennan (Eds.) *Cognitively Diagnostic Assessment..* Hillsdale, NJ: Lawrence Erlbaum.

Goldstein, I.P (1982). The genetic graph: A representation for the evolution of procedural knowledge. In D. Sleeman and J.S.Brown (Eds.) *Intelligent tutoring systems.* New York: Academic.