Evaluation of the q-matrix Method in Understanding Student Logic Proofs.

Conference Paper · January 2006

Source: DBLP

CITATIONS
6

READS
206

1 author:

Tiffany Michelle Barnes
North Carolina State University
281 PUBLICATIONS **2,575** CITATIONS

SEE PROFILE

Evaluation of the q-matrix method in understanding student logic proofs

Tiffany Barnes

Department of Computer Science
University of North Carolina at Charlotte
9201 University City Blvd, Charlotte, NC 28223
tbarnes2@uncc.edu

Abstract

In building intelligent tutoring systems, it is critical to be able to understand and diagnose student responses in interactive problem solving. We present a novel application of the q-matrix method, an educational data mining technique, to the problem of analyzing formal proofs. Our results indicate that automated analysis of formal proof data can provide an intelligent tutoring system with useful diagnostic information for generating feedback and guiding ITS design.

Introduction

According to the ACM computing curriculum, discrete mathematics is a core course in computer science, and an important topic in this course is solving formal logic proofs. However, this topic is of particular difficulty for students, who are unfamiliar with logic rules and manipulating symbols. To allow students extra practice and help in writing logic proofs, we are building an intelligent tutoring system on top of our existing Proof Verifier program. Our experience in teaching discrete math, and in surveys conducted on our Proof Verifier, indicate that students particularly need feedback when they get stuck. An intelligent tutoring system, such as those in (Conati, et al., 2002; Heffernan & Koedinger, 2002; Van Lehn & Martin 1998) may provide the individualized feedback that students need. However, these systems are quite costly to develop (Murray, 1999), and many systems, including REDEEM (Ainsworth, et al., 2003), ASSERT (Baffes & Mooney, 1996), and CTAT (Koedinger, et al. 2004), have been exploring ways to reduce the time needed to develop ITSs without sacrificing the ability to adapt to students. In our past work, we have applied the q-matrix method to quickly modify an existing computer-based training tool to be a simple ITS, using knowledge discovery and data mining techniques to model student knowledge and direct knowledge remediation (Barnes, 2005a & 2005b; Barnes, et al. 2005).

We propose a few ways that educational data mining could be applied to reduce ITS design time and to focus that design on areas where students need the most help. Namely, we demonstrate the novel use of the q-matrix method in determining proof strategies and suggest some ways this method can be used to generate feedback.

Background

The original inspiration for the q-matrix method came from Tatsuoka, et al.'s rule space research, which showed that it was possible to automate the diagnosis of student knowledge states, based solely on student item-response patterns (Tatsuoka, 1983). The rule-space idea evolved into a q-matrix, a binary matrix showing the relationship between test items and latent or underlying attributes, or concepts (Birenbaum, et al., 1993), and they are now used in intelligent tutoring systems to relate underlying concepts to problems (e.g. Van Lehn, et. al, 1998). Students were assigned knowledge states based on their test answers and the constructed q-matrix.

An example binary q-matrix is given in Table 1. A q-matrix, or "attribute-by item incidence matrix", contains a one if a question is related to the concept, and a zero if not. For example, in this q-matrix, questions q1, q6, and q7 are all related by concept con1, while q1 is also related to q2, q4, and q7 by concept con2. In contrast, question q5 can be answered correctly without understanding any of the concepts c1-c3. This indicates that prerequisite knowledge not represented in the q-matrix affects the answer to q5. Brewer extended q-matrices to include values ranging from zero to one, representing a probability that a student will answer a question incorrectly if he does not understand the concept (1996).

In 1996, Brewer created a method to extract a q-matrix from student data, and found that the method could be used to recover knowledge states of simulated students more accurately than by using factor analysis. In (Barnes, et al., 2005), we applied the method to large groups of students, and found that experts disagreed on q-matrices, and found Brewer's extraction method comparable to standard knowledge discovery techniques for grouping student data. In particular, the method outperformed factor analysis in modeling student data and resulted in much more understandable q-matrices, but had higher error than k-means cluster analysis on the data. However, cluster analysis is not as suitable for automated direction of

student learning as the q-matrix method, because human intervention would usually be required to create behaviors to associate with each cluster.

Table 1. Example q-matrix

	q1	q2	q3	q4	q5	q6	q7
c1	1	0	0	0	0	1	1
c2	1	1	0	1	0	0	1
c3	1	1	1	0	0	0	0

Each cluster in the q-matrix method is represented by its concept state, a vector of bits where the kth bit is 0 if the students do not understand concept k, and a 1 if they do. Each concept state also has associated with it an ideal response vector (IDR), or representative vector, that is determined using the concept state and the q-matrix. For each question q in the q-matrix we examine the concepts needed to answer that question. If the concept state contains all those needed for q, we set bit q in the IDR to 1, and otherwise to 0. There are 2^{NumCon} concept states for a q-matrix with NumCon concepts. A concept state's ideal response vector (IDR) is the answer we predict a student in that state would give under ideal conditions (e.g. he does not make any slips or guesses).

Mathematically, given a q-matrix Q with NumCon concepts, and a concept state Con, where Con(k) denotes whether concept k is understood or not, we calculate the ideal response (IDR) for each question Ques as:

$$\text{IDR(Ques)}= \prod_{k=1}^{NumCon} \begin{Bmatrix} 1 & Con(k)=1 \\ 1-Q(k,Ques) & Con(k)=0 \end{Bmatrix} \quad (1)$$

Table 2 lists the IDRs for all the possible concept states for the q-matrix given in Table 1. The all-zero concept state 000 describes the "default" knowledge not accounted for in the model, while the all-one concept state 111 describes full understanding of all concepts. Concept state 011's IDR corresponds to a binary OR of the IDRs for concept states 001 and 010, plus the addition of a 1 for q2, which requires both concepts c2 and c3 for a correct outcome.

Table 2. Ideal Response Vectors for each Concept State

Concept State	IDR	Concept State	IDR
000	0000100	100	0000110
001	0010100	101	0010110
010	0001100	110	0001111
011	0111100	111	1111111

Q-Matrix Model Evaluation

To evaluate the fit of a given q-matrix to a data set, we compute its concept states and IDRs as in Table 2 and assign each data point to the concept state whose IDR is the closest match. Each data point's associated error is the L_1 (Hamming) distance between the concept's IDR and the data point, yielding a direct count in the number of bit differences between the data point and IDR. In other words, the distance d(p,IDR) between data point p and its IDR is summed over all questions q:

$$d(p,IDR) = \sum_q |p(q) - IDR(q)| \quad (2)$$

For example, for a data vector 0111110 and the q-matrix given in Table 1, the nearest IDR would be 0111100 in concept state 011, and the error associated with this assignment is 1, since there is only one difference between the data point and its nearest IDR. The total error for a q-matrix on a given data set is the sum of the errors over all data points.

Q-Matrix Model Extraction

The q-matrix algorithm, as devised by Brewer in 1996, is a simple hill-climbing algorithm that creates a matrix representing relationships between concepts and questions directly from student response data. The algorithm varies c, the number of concepts, and the values in the q-matrix, minimizing the total error for all students for a given set of n questions. To avoid of local minima, each hill-climbing search is seeded with different random q-matrices and the best of these is kept.

First, c, the number of concepts, is set to one, and a random q-matrix of concepts versus questions is generated with values ranging from zero to one. Also, 2^c concept states are generated. A binary string of length c represents each state where a zero in position k represents that a student in this state does not understand concept k, and a one represents that he does. For each concept state, its corresponding ideal response vector is generated based on the q-matrix.

We then compute the total error for the q-matrix computed over all students as described in the previous section. After the error has been computed for a q-matrix each value in the q-matrix is changed by a small amount, and if the overall q-matrix error is improved, the change is saved. This process is repeated for all the values in the q-matrix several times, until the error in the q-matrix is not changing significantly.

After a q-matrix is computed in this fashion, the algorithm is run again with a new random starting point several times, and the q-matrix with minimum error is saved, to avoid falling into a local minimum. It is not guaranteed to be the absolute minimum, but provides an acceptable q-matrix for a given number of c concepts.

To determine the best number of concepts to use in the q-matrix, this algorithm is repeated for increasing values of

c. The final q-matrix is selected when 1) a pre-set stopping criterion has been met (such as less than 1 error per student) or when adding an additional concept does not decrease the overall q-matrix error significantly, and 2) the number of concepts is significantly smaller than the number of questions. Based on our work with the q-matrix, we have determined a rule of thumb for the number of concepts, as the log (base = number of variables) of the number of responses. This heuristic, which needs experimental validation, may reduce the q-matrix method run time by providing a target number of concepts.

Modification for Proofs

Solutions to formal logic proofs are not well-suited to a traditional application of the q-matrix method, where we analyze student responses as answers to questions common to all students. In solving proofs, instead of answering a single question, students type in consecutive lines of a proof, which consist of 4 parts: the statement, reference lines, the axiom used, and the substitutions which allow the axiom to be applied in this particular case. After the student enters these 4 parts to a line, the program verifies that: 1) the statement is true, based on the previous lines of the proof, and 2) the axiom and appropriate substitutions listed correspond to the reference lines and statement. The program prompts the student to continue if the proof line is valid, or warns the student that there was a problem with the line if it is not. In each case, the student is returned to entering his proof.

There are several reasons that a straightforward q-matrix analysis could not be used for proofs. These are:

1. The program does not accept "wrong" proof lines. Therefore, there is no data to make a model that discriminates based on right versus wrong answers.
2. A proof can be incomplete for many reasons, including lack of time or the inability to continue.
3. Statements in one student's proof may or may not appear in another's.
4. Students solve proofs with different methods and in different orders.
5. There are too many valid proofs to anticipate every one.
6. If we did anticipate all possible valid proofs, considering each different answer as a "question" would require considerably more student data. Also, many proofs are the same approach executed in different orders.

Although the program does not display invalid proof statements, these data were collected and saved along with each student proof. This type of data, though not used here, can later be analyzed to determine the types of errors students make in proof writing.

Method

In this research, we apply the q-matrix method to proofs written by a large group of students and analyze the results to determine if we can understand student proofs using the q-matrix alone. Our hypothesis was that concept states resulting from extracted q-matrices would reflect fundamental methods for solving proofs, and that this automatic analysis would provide meaningful information for generating feedback in our planned ITS.

The Proof Verifier, written on the NovaNET educational system (see http://www.pearsondigital.com/novanet/), was administered to approximately 200 students in the Fall 2002 Discrete Mathematics course, CSC 226, at NC State University. Using the verifier was required for credit in the class, but students were allowed to attempt each problem until it was solved. As students work each of the 10 proofs, their work is saved. We later process their proofs to extract the rules used by each student in a particular proof, creating "answer vectors" for each student – where the ith bit of the answer vector contains a one if the student used the ith rule, and a zero if not.

The steps in this experiment were to: 1) determine a mapping from student responses to answer vectors to be used as inputs to the q-matrix method, 2) remove variables that might not be interesting or useful in the analysis of student proofs, 3) derive a q-matrix to explain the student data, 4) interpret the resulting q-matrix in the context of the given proof to solve.

Mapping Responses to Answer Vectors

The first step in analyzing student proofs was to determine a way to map proofs to answer vectors. For each proof, there are many possible solutions. However, our conjecture was that many student proofs, though appearing different, might be using the same underlying approach. Fundamentally, these approaches should overlap in the axioms that students used in the process of finding a solution. Therefore, our goal was to find groups of rules that, when used together, would devise all or part of the most crucial steps in solving a particular proof. Since there are 36 rules on our Axiom List, we generate a bit string of length 36 for each student proof that describes the use of axioms, by placing a 1 in the ith position if the ith axiom were used, and a 0 otherwise.

Variable Selection

In this step, we used two criteria to determine which variables to consider in our q-matrix analysis. First, since we were looking for general relationships among axioms, and classes of problem solutions, axioms used by only one student solving a proof were not considered for analysis. Second, we narrowed the list of axioms to those we would consider as those most important in solving proofs. As in a

general data mining process, this step was necessary to analyze only those items we felt were most important.

The most important rules in writing proofs are those that result in a "change in the knowledge base" of a proof. In other words, the structure of the knowledge in the proof undergoes a change when these rules are applied. Specifically, the use of a logical implication (one of the first 10 axioms on our list), or Negation of Conclusion, results in more than a simple rearrangement of the variables. For example, Modus Ponens combines a → b with a to find b is true. In a way, we have eliminated the variable a. In our analysis, we also considered rules that would be crucial steps in being able to apply logical implication rules, including the logical equivalences DeMorgan's, Implication, and Contrapositive.

Rules that do not result in a "change in the knowledge base" of the proof include Simplification, Conjunction, and the remaining logical equivalences, including rules such as Double Negation, Commutative and Associative rules. An additional reason for eliminating these rules from consideration was that the Proof Verifier often lets students skip these steps, so we'd have no way of knowing whether some students applied, but did not enter, these steps.

After eliminating all logical equivalence rules other than DeMorgan's, Implication, and Contrapositive, we then grouped rules that were only slightly different ways of writing the same approach. For example, we grouped DeMorgan's for AND and for OR together as one rule. We also grouped Constructive Dilemma for AND and OR together as one rule, and Modus Ponens and Modus Tollens together as another. Though Modus Ponens and Modus Tollens may seem dissimilar, in teaching since 1996, we have observed that students often demonstrate a preference for using just one of these rules, versus approaches using Disjunctive Syllogism.

The final list of rules selected for analysis, after eliminating rules not generally used by students and those considered redundant or less important, is: Hypothetical Syllogism (HS), Disjunctive Syllogism (DS), Modus Ponens and Modus Tollens (MP/MT), Constructive Dilemma (CD), DeMorgan's (DM), Contrapositive (CP), Implication (IMP), and Negation of Conclusion (NC).

Q-matrix Extraction and Analysis

In a q-matrix, items that relate to all concepts (i.e. with 1's for every concept) are the least frequently answered items. Therefore, these items will have zeroes in most ideal response vectors. On the other hand, items related to no concepts (i.e. with 0's for all concepts) are the most frequently used items, and all ideal response vectors will predict a 1 for these items. These values have the least change over all students. In a q-matrix, we can then interpret the rows as ideal response vectors if we first reverse the bits for these two types of questions. (Note that, in cases where there are a large number of concepts, we should also reverse the bits for questions related to "almost all" or "almost no" concepts.)

Results and Discussion

Although we analyzed data from all 10 proofs offered in the Proof Verifier, for conciseness we present data only from Proof 1, listed in Table 3, to illustrate our results. In our experiment, 189 students completed Proof 1 in Fall 2002. For each proof, students were allowed to solve the proof by direct or indirect proof, as shown in Tables 4-5.

Table 3. Proof 1 problem description

Given: $a \rightarrow b,\ \ c \rightarrow d,\ \neg(a \rightarrow d)$ *Prove:* $b \wedge \neg c$

Table 4. Proof 1 Example of Direct Proof

	Statement	Line	Reason
1.	$a \rightarrow b$		Given
2.	$c \rightarrow d$		Given
3.	$\neg (a \rightarrow d)$		Given
4.	$\neg (\neg a \vee d)$	3	Implication
5.	$a \wedge \neg d$	4	Demorgan's
6.	a	5	Simplification
7.	$\neg d$	5	Simplification
8.	b	1,6	Modus Ponens
9.	$\neg c$	2,7	Modus Tollens
10.	$b \wedge \neg c$	8,9	Conjunction

Table 5. Proof 1 Example of Indirect Proof

	Statement	Line	Reason
1.	$a \rightarrow b$		Given
2.	$c \rightarrow d$		Given
3.	$\neg (a \rightarrow d)$		Given
4.	$\neg (b \wedge \neg c)$		Neg of Conclusion
5.	$b \rightarrow c$	4	Implication
6.	$a \rightarrow c$	1,5	Hyp. syllogism
7.	$a \rightarrow d$	2,6	Hyp. syllogism
8.	$(a \rightarrow d) \wedge \neg (a \rightarrow d)$	3,8	Conjunction
9.	contradiction		

Table 6 lists the q-matrix extracted from Fall 2002 data, and Table 7 summarizes ideal and student response vectors.

Table 6. Proof 1 q-matrix

	HS	DS	MP/ MT	CD	DM	CP	IMP	NC
Con 1	1	1	0	1	0	1	0	1
Con 2	0	1	1	1	0	1	0	0

By analyzing the q-matrix given in Table 6, we can draw several conclusions about student proofs without seeing them. First, since their columns are all ones, we know that Disjunctive Syllogism (DS), Constructive Dilemma (CD), and Contrapositive (CP) are not often used for Proof 1. Likewise, since their columns are all zeroes, DeMorgan's (DM) and Implication (IMP) are often used for Proof 1.

From Con 1, we see that Hypothetical Syllogism (HS) and Negation of Conclusion (NC) are used together, and from Con 2, Modus Ponens/Tollens (MP/MT) distinguishes some student responses from others.

Table 7. Proof 1 Ideal and Student Response Vectors

Row	Response	# Students	Error	Err * # Stud
	00001010	**State 00: Baseline**		
1	00000010	1	1	1
2	01000010	1	2	2
3	01001010	1	1	1
4	01001110	2	2	4
5	00000011	1	2	2
	10001011	**State 01: Concept 1**		
6	10000011	3	1	3
7	10001011	23	0	0
8	01001011	2	2	4
9	10101011	8	1	8
10	01101011	1	3	3
11	00001111	1	2	2
12	10001111	2	1	2
	00101010	**State 10: Concept 2**		
13	00100010	11	1	11
14	01100010	1	2	2
15	00110010	1	2	2
16	00101010	40	0	0
17	10101010	2	1	2
18	01101010	24	1	24
19	11101010	1	2	2
20	00111010	13	1	13
21	00100110	1	2	2
22	10100110	1	3	3
23	00110110	1	2	2
24	00101110	13	1	13
25	00111110	18	2	36
26	00100011	2	2	4
27	00101011	3	1	3
	11111111	**State 11: Concepts 1 and 2**		
28	10110110	1	3	3
29	01111110	8	2	16
30	11101011	1	2	2
31	01100111	1	3	3
	TOTAL:	**189**		**175**

Table 7 provides a summary of Proof 1 responses, and their corresponding concept states. In this table, responses are grouped by concept state and their corresponding ideal response vectors (IDRs). We then list actual student responses in each of these states and the number of students responding with this vector. Finally, we list the error associated with assigning each actual response to a concept state (by counting the number of different bits in the IDR and actual response) and the total error for all the students with a given actual response vector. The last row of the table sums the student and total error columns.

State 00 represents the use of neither concept 1 or 2. There are 6 student proofs in this state, whose IDR is 00001010, representing the use of DeMorgan's and Implication rules. One student (row 1) used only Implication, which is not sufficient for solving Proof 1, which revealed an error in our Proof Verifier. The remaining solutions in State 00 all use Disjunctive Syllogism (DS), or Negation of Conclusion in their solutions. State 11 corresponds to an all-ones IDR, meaning that solutions combined both concepts.

State 01 solutions, with the IDR 10001011, use Hypothetical Syllogism, DeMorgan's, Implication, and Negation of Conclusion. This set of rules corresponds to 23 of the 40 responses in state 10 (as in Table 7). Eight responses in this state used all these rules except for DeMorgan's, indicating that this proof can be solved using Negation of Conclusion, Hypothetical Syllogism and Implication, as shown in Table 5.

Most students are in State 10, with 132/189 responses. State 10 solutions, with the IDR 00101010, use Modus Ponens and/or Modus Tollens, DeMorgan's, and Implication. An example student proof using these rules is given in Table 5. As shown in Table 7, 40 students used these same rules for their proofs, making this answer vector the most common. Five students in this state also used Negation of Conclusion (NC) (see Table 7, rows 26-27). This seems to show that there are 2 distinct ways to use Negation of Conclusion in this proof, one using Hypothetical Syllogism, as in state 01, and the other using Modus Ponens/Tollens, as in state 10. This rich understanding of solutions to Proof 1 resulted solely in examining the q-matrix, and verifying our conclusions with actual student data.

Generating Feedback

Based on this analysis, we could construct several different feedback scenarios for students solving Proof 1. We could automatically generate hints that suggest common strategies, such as: 1. Try Modus Ponens/Tollens, 2. Most solutions to Proof 1 use both DeMorgan's and Implication. Based on the current student solution, we could also offer tailored hints. If the student has already used either Hypothetical Syllogism or Negation of Conclusion, we may suggest the use of the other to "complete" the use of Concept 1. We can also use the q-matrix analysis to guide the choice of where to generate the most-needed hints for adaptation in the Proofs ITS. For example, we can create hints for each concept or state, greatly reducing the space of solutions that needs hint generation.

Conclusion

The main contribution of this work is the novel application of the q-matrix method, a tool used for educational assessment and remediation, as a more generalized data mining tool. In this application, we have demonstrated the use of the q-matrix method in both clustering and understanding groups of student responses. Using the extracted q-matrices, we were able to make predictions about student responses, as supported by student data, and also find errors in our program. We were able to create proofs that used only the predicted responses in the solution, confirming our hypothesis that q-matrices would extract groups of rules that were fundamental to solving the given problem.

Our analysis indicates that large clusters of similar student responses skew the q-matrix model, but that a balance could still be maintained by keeping the number of concepts extracted low. Another interesting finding, as reported in (Barnes, 2005a), that was replicated in this experiment, is each of the 10 extracted proof q-matrices to converge on binary values, even though the q-matrices were allowed to vary as probabilities between 0 and 1. This finding leads us to believe that, given binary input values, good q-matrix models will usually converge to binary values, making the q-matrix method very valuable in understanding a data set. With binary values, q-matrix results may be easily applied to understand student responses and generate helpful feedback for our future intelligent tutoring system. In our future work, we plan to investigate the properties of the q-matrix method that may be causing this binary convergence, and compare this algorithm with methods that restrict the search space to only binary values in the q-matrix.

We have also suggested some ways that a q-matrix extracted from student data can be used to automatically generate hints or guide the creation of relevant feedback. We suggest that this combination of data-driven and traditional ITS methods may be particularly useful in reducing the overall cost of creating new ITSs.

References

Ainsworth, S.E., Major, N., Grimshaw, S.K., Hayes, M., Underwood, J.D., Williams, B. & Wood, D.J. 2003. REDEEM: Simple intelligent tutoring systems from usable tools, in T. Murray, S. Blessing & S.E. Ainsworth (eds). *Adv. Tools for Adv. Technology Learning Environments*. pp. 205-232. Amsterdam: Kluwer Academic Publishers.

Baffes, P. & R.J. Mooney. 1996. A novel application of theory refinement to student modeling. *Proceedings of AAAI-96*, pp. 403-408, Portland, OR, August 1996.

Barnes, T. 2005a. Experimental analysis of the q-matrix method in automated knowledge assessment. Proceedings of *IASTED Computers and Adv. Technology for Education (CATE 2005),* Oranjestad, Aruba, August 29-31, 2005.

Barnes, T. 2005b. The q-matrix method: Mining student response data for knowledge. Proceedings of *AAAI 2005,* Pittsburgh, PA, July 9-13, 2005.

Barnes, T., D. Bitzer & M. Vouk. 2005. Experimental analysis of the q-matrix method in knowledge discovery. *Proceedings of the 15th International Symposium on Methodologies for Intelligent Systems*, Saratoga Springs, NY, May 25-29, 2005.

Birenbaum, M., Kelly, A., & Tatsuoka, K. 1993. Diagnosing knowledge state in algebra using the rule-space model. *Journal for Research in Mathematics Education, 24*(5), 442-459.

Brewer, P. 1996. Methods for concept mapping in computer based education. Computer Science Masters Thesis, North Carolina State University.

Birenbaum, M., Kelly, A., & Tatsuoka, K. 1993. Diagnosing knowledge states in algebra using the rule-space model. *Journal for Research in Mathematics Education*, 24(5), 442-459.

Conati, C., A.S. Gertner, & K. VanLehn. 2002. Using Bayesian networks to manage uncertainty in student modeling. *User Model. User-Adapt. Interact.* 12(4): 371-417.

Heffernan, N.T. & K.R. Koedinger. 2002. An intelligent tutoring system incorporating a model of an experienced human tutor. *Intelligent Tutoring Systems,* pp. 596-608.

Koedinger, K. R., Aleven, V., Heffernan. T., McLaren, B. & Hockenberry, M. 2004. Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. Proceedings of the *7th Annual Intelligent Tutoring Systems Conference*, Maceio, Brazil, pp. 162-173.

Murray, Tom. 1999. Authoring Intelligent Tutoring Systems: An analysis of the state of the art. *International Journal of AI in Education* (1999), 10: 98-129.

Tatsuoka, K. 1983. Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of Educational Measurement*, 20(4), 345-354.

VanLehn, K., Niu, Z., Siler, S., & Gertner, A. 1998. Student modeling from conventional test data: A Bayesian approach without priors. *Intelligent Tutoring Systems*, pp. 434-443.