Educational Multimedia

Toward Next-Generation, Intelligent Tutors: Adding Natural Handwriting Input

Lisa Anthony, Jie Yang, and Kenneth R. Koedinger
Carnegie Mellon University

This article explores handwriting recognition-based interfaces in intelligent tutoring systems for students learning algebra equations.

Mathematic skills are essential not only for those wanting successful careers in sciences or engineering but also for nonscientists using equations in daily life. For example, a stay-at-home dad might need to use simple algebra skills to figure out his monthly budget. Despite this clear need, algebra is typically one of the subjects where students struggle the most. Our project aims to improve student performance and engagement in algebra via intelligent tutoring systems (ITSs) that accept natural handwriting input (see Figure 1).

Figure 1. A screenshot of our current prototype system illustrating the use of worked-examples-based instruction and handwriting input.

Many schools throughout the US now incorporate computers, including PDAs or tablet PCs, and ITSs as a regular part of classroom instruction.[1] An ITS is educational software that can monitor the student as he or she works at his or her own pace, and tailor feedback, step-by-step hints, and even the curriculum to address a student's particular needs. This self-pacing provides an opportunity for teachers to give more individual attention to students who need it most. One particular type of ITS, Cognitive Tutors, is quite successful in the classroom. When compared to traditional classroom instruction, Cognitive Tutors raise student achievement one standard deviation,[2] turning C students into B students. Our goal is to improve mathematics learning in Cognitive Tutors even further via multimodal- and multimedia-interface technologies, turning those B students into A students.

A user interface bridges information exchanges between a student and an ITS. Different modalities in interface input and output can significantly affect ITS efficiency. Input modality here refers to the modality of generation by the student, while the output modality is the modality presented to the student by the system. Although output modality has been studied with respect to learning, including the use of animations, diagrams, and talking heads,[3] little attention has been paid to the effect of input modality on learning. Most ITSs rely on standard menu-based GUIs. We believe that the input modality is extraneous to the problem-solving process, as it is not relevant to the math concept being practiced. However, the input modality can interfere with learning by imposing an extraneous cognitive load on the student—that is, mental effort not directly related to the learning process. Text- and menu-based interfaces are especially guilty of putting an additional burden on the student because they provide only cumbersome support for representing and manipulating math. An interface that can more directly support the standard notations for mathematics that the student is learning could reduce extraneous cognitive load and increase learning.

We have developed and tested a prototype ITS that uses handwriting input for teaching algebra equations. Our latest results show that handwriting input has benefits both for general usability and for learning. Although this work is in the domain of high school algebra learning, it's likely to generalize to other types of math and to other levels of students.

Motivation and approach

In many courses, teachers supplement traditional education strategies with software tutors. Math is one domain in which tutoring systems can help; a recent survey found that high school students in the US have a poor mastery of basic math concepts compared to their counterparts in other industrialized nations.[4]

There is evidence that using handwriting interfaces could offer benefits in math learning environments.[5,6] One benefit might come from more direct manipulations of pen-based input. Another factor is the improved support for meaningful 2D spatial information in math. For example, the placement of the x in $2x$ versus 2^x significantly changes the meaning of the expression. Handwriting is more flexible and robust than typing or pointing for representing and manipulating such spatial relationships.[5]

Because handwriting recognition is not perfectly accurate, there is a trade-off between improving recognition accuracy and the need for detailed instructional feedback. One strategy is to modify the instruction paradigm. For example, during worked-examples-based instruction, a common instructional method, students study example problems with the solutions provided before solving their own problems.[7] Advance studying gives the student a low-risk opportunity to practice the problem-solving concepts he or she is about to solve on his or her own. Such a feed-forward approach perhaps would reduce the need for detailed instructional feedback at each step, allowing the system to delay providing a recognition hypothesis until more information is available. Figure 1 shows the type of annotated worked examples we use in our tutor.

A second strategy is to improve handwriting recognition accuracy. Our current prototype achieves a representative best a priori recognition accuracy by training the recognizer on a large corpus of handwriting samples from different users from the target population (middle and high school math learners). In addition, due to the special nature of a learning task and the needs of a learner, students might not require the system to immediately output a recognition hypothesis. Students are not so much interested in what the tutor thinks they wrote as in whether what they wrote was the correct answer. If the students must spend time correcting the system's recognition, we have merely exchanged one type of cognitive load for another, and might not see any differences in learning. The tutoring system can provide the student with other forms of information (such as worked examples) to alleviate reliance on this type of recognition feedback and provide the recognition engine with more information—such as the problem's context or knowledge about the student's skills—prior to the end of the problem).

A fundamental goal of this project is to determine how the use of handwriting input will help student learning. As part of our affiliation with the Pittsburgh Science of Learning Center (PSLC), we began to emphasize in vivo studies after some preliminary laboratory explorations. In vivo studies take place in a real-world classroom setting in which the experimental system is compared to an authentic control condition. The participating classrooms use Cognitive Tutor Algebra all year; only some classes or students switch to the experimental system during the study. This approach represents an emerging trend in educational research.[8,9] Being part of the PSLC provides us with the unique opportunity of conducting real-world, applied studies and seeing the experimental software and technology used in the field.

Current prototype

We have developed a prototype system that uses a foundation of state-of-the-art ITS and handwriting recognition components. Cognitive Tutors are a class of ITS that pose authentic problems to students and emphasize learning-by-doing.[10] In Cognitive Tutor Algebra, students represent a given scenario algebraically in a spreadsheet or with graph functions and solve equations with a symbol-manipulation tool. The tutor can provide step-by-step feedback and help. Our prototype adds a handwriting interface to already-existing Cognitive Tutoring Algebra lessons that have been field-tested extensively.

Figure 1 shows a screenshot of our system in which the student is solving the problem $-6.72 = -0.25y + 0.23$ by referring to the worked example on the left side of the screen. The student enters his or her solution process in handwriting and types his or her final answer in the text field at the bottom of the screen. The Cognitive Tutor Algebra curriculum has several equation-solving units. We can deploy our prototype for any of these units. Some problem types we use include $ax + b = c$, $x/a + b = c$, $a/x = c$, $ax + bx = c$, $ax + b = cx + d$, and so on.

The recognizer we use is the Freehand Formula Entry System (FFES); rather than developing a robust recognizer from scratch, our approach can blaze a trail for using handwriting engines in more real-world applications without requiring expert skill in recognition algorithms.[11] We chose FFES because it was open source and had the highest accuracy rates in training experiments on our corpus when compared with several other state-of-the-art recognizers, such as JMathNotes[12] and Microsoft's tablet PC software developer's kit (see http://msdn2.microsoft.com/en-us/developercenters/default.aspx). FFES uses the CIT (California Interface Tools) character recognizer, a nearest-neighbor classifier based on a 48-dimensional feature space. FFES also uses a mathematical expression parser (DRACULAE).[13] The main advantage of this handwriting system over other engines is that this parser is designed to be effective for the spatial relationships between mathematical symbols and numbers.

In recognition terminology, writer-independent means that the system or engine has been trained on samples from several different users, in contrast to writer-dependent, which means the training samples come from the same person who will be using or testing the system. In general, writer-dependent accuracy rates are higher than writer-independent rates because differences in handwriting (or speech) vary more widely across users. But a writer-dependent system requires more training samples from each individual user. In learning environments, it's a hard sell to teachers to take time for students to spend time training the system with no learning objectives. On the other hand, it's difficult to embed the handwriting-training task into learning-oriented tasks because the system cannot provide adequate feedback on the learning aspects without good a priori recognition accuracy. Therefore, we attempt to improve recognition without up-front training for each user.

To train the recognizer and achieve the best a priori accuracy before incorporating it into the tutoring system, we collected a corpus of data from over 40 high school and middle school algebra students. The corpus contains 16,191 characters grouped into 1,738 equations. The symbol set includes 21 symbols: the digits, common variable letters, and simple algebraic operators. We have, on average, 404 samples per user (an average of 17 samples per symbol, per user). In addition to collecting this corpus, we studied strategies for training the recognizer. Although the best accuracies obtained, about 91 percent, were writer-dependent, when working in a classroom, we must bring in the best system possible out of the box, which means training in advance on other students' handwriting samples. Our experiment results indicate that ensuring an equal representation of each training user's handwriting will prevent the system from having a bias toward a particular style of writing when a new student uses the system. Each of 40 users had to write just two samples per symbol to converge to best a priori accuracy, averaging 83 percent on individual symbols or 71 percent on full equations (accuracy on full equations was determined via Levenshtein's string distance formula).[14] Very few samples per symbol, per user, are needed in this training method to achieve reasonable out-of-the-box performance.

User evaluations

Our first study explored what advantages, if any, handwriting-based input offered for math input on the computer.[5] Forty-eight college-level students entered given algebra and calculus equations on the computer over the course of 45 minutes in several modalities, including typing via Microsoft Equation Editor and handwriting. In this study, we used no handwriting recognition; our aim was to determine what the raw usability of each modality would be in this task without the interference of individual system nuances. The study showed that students who entered math equations via handwriting input were three times faster, were less prone to errors in input, and enjoyed their experience more than those who typed.[5] In the classroom, this can translate to increased depth or breadth of coverage by virtue of the extra time afforded, and to improved student motivation by virtue of their increased engagement.

Next, we applied handwriting-based input to a learning situation.[6] In this study, we compared a simple type-in interface with a handwriting input space. Both the typing and the handwriting interfaces were simple, unstructured, and unconstrained input spaces. We provided no special-purpose math menus or widgets in either case. The 48 participants each came to our lab for a two-hour session. Results from this study showed that students using handwriting finished the learning session in half the time of their typing counterparts, yet we found no difference in their learning. In a classroom situation, this method would allow teachers to give students more practice or move on to more advanced material in the curriculum more quickly.

The speed benefits of handwriting only increased as the problems got more complex, as in problems with fractions, for example. In their own words, students commented that handwriting "made it easier" and "took a shorter time"—statements that lend support to the hypothesis that handwriting involves less cognitive load. Handwriting also emerged as the winning modality in terms of user preference. All students were exposed to both modalities in a second phase of the study. As the pie chart in Figure 2 shows, students showed a strong preference for handwriting. Out of 46 total students, over 80 percent preferred handwriting.

Figure 2. Pie chart distribution of students' preferences for each input modality tested in the study. The majority of students chose one of the two conditions in which handwriting was offered.

Conclusions and future work

Our work in adapting, deploying, and testing handwriting-based interfaces in ITS for algebra equation solving is ongoing, and so far has provided positive evidence in favor of handwriting interfaces for learning applications. (Some further technical details can be found elsewhere.[15]) Students can learn the same amount in half the time using handwriting input versus typing-based interfaces. In addition, the decrease in extraneous cognitive load allows students to focus more directly on the mathematics and finish their work with a deeper understanding of the material.

We will continue to evaluate new versions of our prototype in classroom studies. Our next steps involve exploring other strategies for enhancing recognition accuracy. Once these enhancements are in place, we will compare the handwriting-based tutor to the standard Cognitive Tutor classroom practice, focusing on differences in learning and cognitive load. This work should help shed light on how to design instructional paradigms that can take advantage of the benefits of handwriting input for learning.

In addition to studying handwriting input, we are considering using speech input for error correction. Systems perform more effectively when the modality of repair is different from the modality of entry, in part because users tend to over-enunciate (in speech) or trace heavily (in writing).[16] Speech seems to be a logical option because it can be highly accurate when the symbol vocabulary is minimal, and because it doesn't require that students return to the keyboard. Furthermore, it could be pedagogically effective to allow students to self-explain their problem-solving process in speech as they write. We are also considering using multimedia output for our ITS. Our system can present to students animated diagrams helping to explain the problem-solving process when the student needs a hint. Mayer's work exploring the principles of multimedia learning serves as design guidelines for including multimedia output.[3]

As we mentioned, while this project focuses on high school algebra, it's likely to generalize to other types of math and to other levels of students. Our future efforts in this line of work will explore other domains, such as calculus, geometry, and physics, which rely even more heavily on spatial information in annotations. MM

Acknowledgments

National Science Foundation award #SBE-03554420 and the Pittsburgh Science of Learning Center support this work. We thank the many people at Carnegie Mellon for their input and support.

References

1. C. Wood, "Education," *PC Magazine*, 2002; http://www.pcmag.com/article2/0,4149,15154,00.asp.
2. A.T. Corbett, "Cognitive Computer Tutors: Solving the Two-Sigma Problem," *Proc. User Modeling*, A. Kobsa, B.P. Woolf, D.N. Chin, and A.D. Esteban eds., LNCS 2109, Springer Berlin / Heidelberg, 2001, pp. 137-147.
3. R.E. Mayer, *Multimedia Learning*, Cambridge Univ. Press, 2001.
4. Nat'l Center for Education Statistics, *Int'l Outcomes of Learning in Mathematics Literacy and Problem Solving*, 2005; http://nces.ed.gov/pubs2005/2005003.pdf.
5. L. Anthony, J. Yang, and K.R. Koedinger, "Evaluation of Multimodal Input for Entering Mathematical Equations on the Computer," *Proc. Conf. Human Factors in Computing Systems* (CHI), ACM Press / New York, 2005, pp. 1184-1187.
6. L. Anthony, J. Yang, and K.R. Koedinger, "Benefits of Handwritten Input for Students Learning Algebra Equation Solving," *Proc. Artificial Intelligence in Education*, IOS Press / Amsterdam, 2007, pp. 521-523.
7. J. Sweller, "Cognitive Load During Problem Solving: Effects on Learning," *Cognitive Science*, vol. 12, no. 2, 1988, pp. 257-285.
8. K.R. Koedinger, and V. Aleven, "Exploring the Assistance Dilemma in Experiments with Cognitive Tutors," *Educational Psychology Rev.*, vol. 19, no. 3, 2007, pp. 239-264.
9. B.Y. White and J.R. Frederiksen, "Inquiry, Modeling, and Metacognition: Making Science Accessible to all Students," *Cognition and Instruction*, vol. 16, no. 1, 1998, pp. 3-118.
10. J.R. Anderson et al., "Cognitive Tutors: Lessons Learned," J. Learning Sciences, vol. 4, no. 2, 1995, pp. 167-207.
11. S. Smithies, K. Novins, and J. Arvo, "Equation Entry and Editing via Handwriting and Gesture Recognition," *Behaviour & Information Technology*, vol. 20, no. 1, 2001, pp. 53-67.
12. E. Tapia and R. Rojas, "Recognition of On-Line Handwritten Mathematical Expressions Using a Minimum Spanning Tree Construction and Symbol Dominance," *Graphics Recognition: Recent Advances & Perspectives*, J. Lladós and Y.-B Kwon eds., LNCS 3088, Springer Berlin / Heidelberg 2004, pp. 329-340.
13. R. Zanibbi, D. Blostein, and J.R. Cordy, "Recognizing Mathematical Expressions Using Tree Transformation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, 2002, pp. 1455-1467.
14. V.I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," *Soviet Physics Doklady*, vol. 10, no. 8, 1966, pp. 707-710.
15. L. Anthony, J. Yang, and K.R. Koedinger, "Adapting Handwriting Recognition for Applications in Algebra Learning," *Proc. ACM Workshop Educational Multimedia and Multimedia Education*, ACM Press / New York, 2007, pp. 47-56.
16. S. Oviatt, "Taming Recognition Errors with a Multimodal Interface," *Comm. ACM*, vol. 43, no. 9, 2000, pp. 45-51.

Lisa Anthony *is finishing her PhD at the Human–Computer Interaction Institute at Carnegie Mellon University. Her research interests include multimodal interfaces, such as pen, speech, and gestures, and focus on applying these input modalities to new domains with a user-centered design perspective. Anthony has an MS in computer science from Drexel University and an MS in human–computer interaction from Carnegie Mellon University. Contact her at lanthony@cs.cmu.edu.*

Jie Yang *is a senior systems scientist at the Human–Computer Interaction Institute at Carnegie Mellon University. His research interests include multimodal interfaces, computer vision, and pattern recognition. Yang has a PhD in electrical engineering from the University of Akron. Contact him at jie.yang@cs.cmu.edu.*

Kenneth R. Koedinger *is a professor of human–computer interaction and psychology at Carnegie Mellon University and the Carnegie Mellon Director of the Pittsburgh Science of Learning Center. His research interests include creating educational technologies that increase student achievement and developing computer simulations of student thinking. Koedinger has a PhD in psychology from Carnegie Mellon University. Contact him at koedinger@cmu.edu.*