NumberLand: An Arithmetic Tutoring System

Final Project Report

Nathaniel Raley

Dept. of Educational Psychology

University of Texas at Austin

Introduction

My experience working as a one-to-one tutor in different subjects and with students of various ages has given unique breadth to my understanding of tutoring as an important tool for education. This subjective claim has firm support in the research literature; in an often-quoted study, respected educational psychologist Benjamin Bloom reports that the average student tutored one-to-one performed two standard deviations better than students who were taught in a conventional classroom setting, a finding which held across grade-levels and subject matter (Bloom, 1984). But the benefits of this instruction would seem to be the province of the wealthy few, those who could afford to hire full-time tutors for their children. Thankfully, technology is moving in a direction that promises to change all of that. Having revolutionized science, commerce, and communication in a matter of decades, computers are poised to do the same in education. And with almost 2.5 billion people across the globe connected by the internet, now the world's largest repository of knowledge is for most only a mouse-click away; technology is moving education down from ivory towers and unfettering it from constraints of space and time.

Technology has certainly made its way into the classroom: in 2009, 97 percent of US teachers had one or more computers located in the classroom every day (NCES, 2010), and the ratio of students to computers in the classroom was 5.3 to 1 (and this was five years ago)! The flip-side of this technological surge is that we have created a "knowledge economy" that requires its workers to keep pace with rapidly changing demands; accordingly, job skills in many fields (particularly information/technology) will need to be updated continuously. These sorts of issues, the dream of "a teacher for every student," and support for efficient lifelong learning maybe effectively resolved through the development and implementation of computer-based tutoring systems.

Currently, the quantitative capabilities of American students are severely lacking relative

to students in other developed nations, and changes in mathematics education are sorely needed. In 1997, the National Assessment of Education Progress (NAEP) showed that the average 17-year-old high school student's mathematical performance fell in the lower end of the "basic" range. To place this on a global scale, American students' scores on important international proficiencies tests, such as the Program for the International Student Assessment (PISA), are lower than those of more than 30 other developed countries, with 12 countries having at least twice the percentage of high-ranking students (Hanushek & Rivkin, 2012). I believe that adaptive tutoring systems have the potential to keep students from falling behind their international peers in mathematics and to help give students the quantitative skills they will need to succeed in a modern workplace. Indeed, the US Department of Education has evaluated the effectiveness of such a program that helps high school students learn algebra and geometry (Koedinger et al., 1997).

The topic of educational technology typically brings to mind images of souped-up whiteboards, kids wasting time on iPads, watching movies or playing games with flashy graphics and over-the-top multimedia; given these many missteps and wrongheaded applications of technology to education, it is easy to see where this reputation came from. However, I don't see that happening for these adaptive tutoring systems; I believe that, when used in traditional classrooms with caring teachers, they can provide all the advantages of individualized instruction at a fraction of the cost. I see them saving time by speeding through topics that a student already understands, focusing in depth on topics that are troublesome, and never losing hope or patience with students. I believe that, among other aims, this technology has the potential to accommodate individual differences and respond effectively to them, to track student collaboration and identify student contribution during groupwork, and to motivate bored, tuned-out students; in short, I think that its addition to instructional design would go a long way toward optimization of the

learning process.

Problem Description and Rationale

My project goal was to build a basic tutoring system to teach arithmetic skills to elementary school children, with a focus on number sense techniques instead of rote memorization. I used to work at a Math tutoring franchise, and I really liked the techniques they used to get young kids to stop counting on their fingers. I was trained to use these techniques and to work with students individually using a dry-erase board; I would explain the technique, show an example, and then give the student a set of problems to practice with. I would then monitor the student's progress and make notes in the student's binder about what I had observed. For example, I recorded how long it took the student to answer the questions, how many they got correct, and whether or not I thought they had mastered the technique. The next tutor would read my notes about the student's progress and would pick up where I had left off; the student spend a little time each session working with us individually on the dry-erase board until they had mastered the techniques and could do all basic arithmetic operations quickly in their head. These approaches were extremely effective in getting students to stop counting on their fingers and to start thinking in depth about mathematical topics such as the number line, thinking in groups, and even basic algebra.

In 2010, the Common Core standards for mathematics were released and have been adopted by most states and have broken with traditional curricula in their emphasis on numerical fluency over rote computation. These are grade-level specific standards to be met by each student at the end of the academic year; different states use different examinations to assess whether these standards have been met. I was pleasantly surprised how much the numbersense methods my coworkers and I were using at Mathnasium resembled the Common Core guidelines for second and third grade (see Appendix A).

I have never written a “game” type program before; this type of program relies on a constant stream of input from the user and runs continuously until terminated. The number and type of function calls, the amount of I/O, and the duration of game-play are all variable, and this makes optimization and parallelization of my code difficult. Because my knowledge of game design was very limited before I began this project, I failed to realize that I would be limited in this way.

Method

The program is written in the c++ programming language, which I decided to use given the amount of input and output the program would need to handle. The general structure of the main program (project.cpp) consists of two nested do-while loops. Before entering the first loop, the user is prompted to enter their user name. If no data file is found in the users/ directory corresponding to that username, a new user data file is created. If a user data file is found, that file is opened and the user data is read into the appropriate arrays and used to update user parameters. In addition to the date and time of gameplay, there are 8 user parameters that are continuously read and written to the user's file. This is the data that is recorded for each level: (1) total time spent answering questions, (2) total number of questions answered, (3) total number of questions answered correctly, (4) total number of questions missed, (5) total number correct in a row, (6) average amount of time spent per question, (7) number of hints requested, and (8) number of hints requested per question answered. These values are used to make decisions about whether the student has mastered a given topic and what types of problems the student needs to work on (described below).

After reading the student's file, the program enters the first do-while loop, which outputs the main menu and waits for the user to input their menu selection. When the user inputs a valid choice, the program enters the second do-while loop. If the student has unlocked the level and already played the level (based on total problems data), then a math problem of the correct type

is randomly generated and presented to the student; if the student has not unlocked the level, they are told that they must complete the previous level first; if it is the student's first time playing the level, the teach() function is called, which outputs instructional material for the student to read. Several levels have interactive lessons, but some just output a paragraph or two of text explaining the technique and giving an example of the technique being used. When the student has finished reading the material, they are told that they are about to be presented with questions that they should answer by typing the correct number and pressing enter.

The randomly generated problem is then presented to the student, the time() function begins counting the number of seconds it takes the student to input their answer, and the program waits for input. When the student gives an answer, it is evaluated for correctness; if it is correct, the student is told that they are correct, they are given a new randomly generated question of the same type, and their user data is updated accordingly (for the current level, “total questions” and “number of questions correct” are incremented by 1; “average time per question” variable is updated; “total time” incremented, etc.). If the student is incorrect, they are told to try again (“total questions” and “total questions missed” variables for the current level are incremented by 1, as is the timing data); the student answers until they are correct. When this happens, the loop cycles and a new random question from the same domain is generated, and the process repeats.

At any time during the course of solving problems, the student may input 'q' to return to the main menu, or 'h' to get a hint. When a hint is requested, the hint() function is called, which outputs text that explains the technique in a different way and gives a different completed example. The “number of hints” variable is incremented. As soon as the mastery criterion is reached, students are informed that they have unlocked the next level. The criterion can be easily changed, but I set it up to require 95% accuracy AND an average time per question of less than 7 seconds OR 20 correct in a row. The levels are based on the Math Facts procedure I helped to develop while I was working at Mathnasium; these techniques are not proprietary in any way.

Indeed, they were developed by my coworkers and myself to address what we perceived to be gaps in the proprietary curriculum. Each level centers on one component of learning mental arithmetic, and there are 7 levels total: Doubles (e.g., 4+4), Doubles plus/minus one (e.g., 6+7=6+6+1), Combinations that make ten (e.g., 7+3, 6+4), Working with ten (place value, e.g., 10+2, 10+7), How far apart (e.g., "how far apart are 2 and 7?"), Break down (e.g., 6+__=9), Crossing over ten (takes the most teaching; e.g., 5+9, 15-8). The way that it is taught, the last level allows students to bring all the earlier skills to bear on a single problem.

When students have achieved mastery on this final level, they unlock the quiz, which can then be selected from the main menu (option "8"). The quiz generates problems at random from all of the previous categories; to pass the quiz, the student must get 2 of each problem type correct without missing one of that type in between, but this criterion is also completely adjustable and probably ought to reflect the age and grade level of the student. When students pass the quiz, advanced game play is unlocked! They restart the entire game with all of the same questions except now, the random numbers that are generated come from rand()%100 instead of rand()%10; thus instead of "Doubles plus/minus 1" questions like 7+8, the student will see double-digit questions like 41+42. For "Working with ten", students now get questions like 68+10 instead of 8+10. This allows students to continue to use the techniques they have acquired to solve more difficult problems mentally, instead of having recourse to a calculator or an algorithm like "line up the decimals and add the columns". Choosing option "9" on the main menu, (EXIT PROGRAM), returns '0' to main and terminates the program. Choosing option "0" on the main menu (View My Stats) pipes the give student's user data to gnuplot and generates a histogram so the student can see their progress and how close they are to unlocking the next level.

Results

Since I have described the functionality of the program above in some detail, in this section I will provide several images showing the system in use (Figures 1-4). Captions will provide the context for each.

Figure 1. Screenshot demonstrating a new student on the first level; the top is the first lesson, and below this the student has requested a hint by pressing 'h'.

```
Breakdown:
2 + ___ = 4
2

********** You are CORRECT, Excellent work!

Avg. time = 1.68132 seconds | % correct =95.6044
(Press 'q' to return to main menu...)


Breakdown:
2 + ___ = 7
5

********** You are CORRECT, Great job!


****************************************
**LEVEL CLEARED! You unlocked Level 7!**
****************************************
Avg. time = 1.68478 seconds | % correct =95.6522
(Press 'q' to return to main menu...)
```

Figure 2. Screenshot showing a student advancing from level 6 to level 7; information about average time per question and percent correct is displayed (optional)

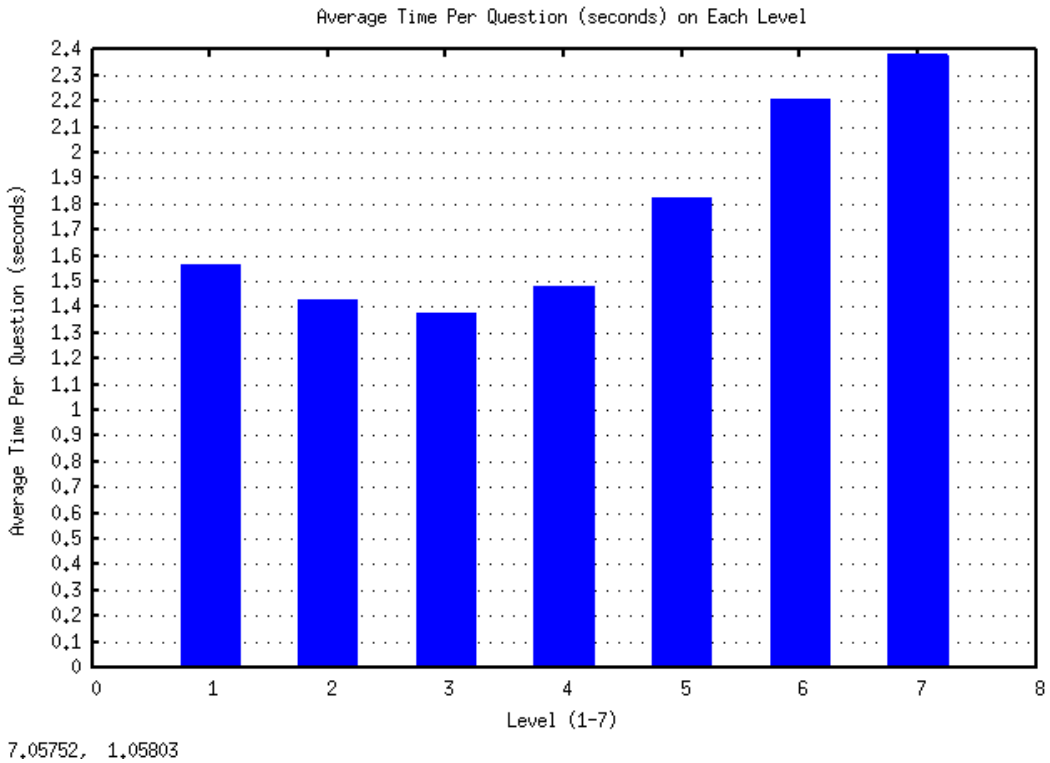

Figure 3. "View my Stats" gnuplot output for average time per question on each level

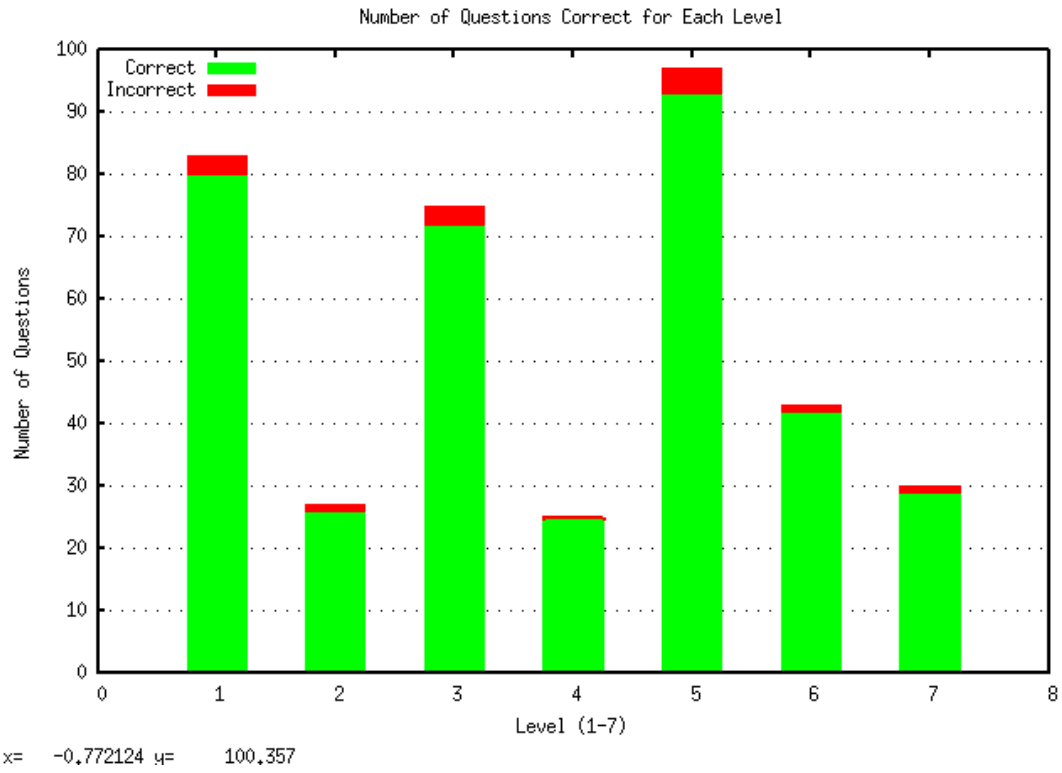

Figure 4. "View my Stats" gnuplot output for number of questions correct vs. incorrect

Limitations and Future Directions

In the future, I would like to conduct a pilot study to assess the efficacy of this program with young students who are struggling in mathematics, or perhaps persons of any age who wish to improve their mental arithmetic. There are many more techniques for becoming proficient at mental arithmetic with larger numbers, and it would not be very difficult to implement these in the present program—I would simply have to write more instructional pieces and more hints. I would also like to develop a graphical user interface for the game so that it is more visually appealing to younger children than a blinking cursor in a computer terminal. The constant input-output sequence of the game is good and holding students attention as they respond to stimuli and feedback, but a more visually appealing interface would be nice. I implemented a very basic

widget with 8 buttons using GTKMM and hooked up the connections to the main menu, but I could not figure out how to generate multiple windows using these widgets. I also adapted a widget for student sign in, that has a text field for logging into the program; I spent many hours trying to get the main menu widget to pop up after user-name entry, but I was not able to do so. I still have the code for all the things I tried, and I hope to get this working in the future. I need friends with small children who can beta test it for me! Also, ultimately I would like to have a better student model that makes use of Bayes nets to update mastery probabilities instead of relying on strict criteria for determining whether or not a student advances.

Appendix A:

Common Core Standards for Math: Number & Operations in Base 10 (Grades 1 & 2)

CCSS.Math.Content.1.NBT.A.1
Count to 120, starting at any number less than 120. In this range, read and write numerals and represent a number of objects with a written numeral.

Understand place value.

CCSS.Math.Content.1.NBT.B.2
Understand that the two digits of a two-digit number represent amounts of tens and ones. Understand the following as special cases:
CCSS.Math.Content.1.NBT.B.2.a
10 can be thought of as a bundle of ten ones — called a "ten."
CCSS.Math.Content.1.NBT.B.2.b
The numbers from 11 to 19 are composed of a ten and one, two, three, four, five, six, seven, eight, or nine ones.
CCSS.Math.Content.1.NBT.B.2.c
The numbers 10, 20, 30, 40, 50, 60, 70, 80, 90 refer to one, two, three, four, five, six, seven, eight, or nine tens (and 0 ones).
CCSS.Math.Content.1.NBT.B.3
Compare two two-digit numbers based on meanings of the tens and ones digits, recording the results of comparisons with the symbols >, =, and <.

Use place value understanding and properties of operations to add and subtract.

CCSS.Math.Content.1.NBT.C.4
Add within 100, including adding a two-digit number and a one-digit number, and adding a two-digit number and a multiple of 10, using concrete models or drawings and strategies based on place value, properties of operations, and/or the relationship between addition and subtraction; relate the strategy to a written method and explain the reasoning used. Understand that in adding two-digit numbers, one adds tens and tens, ones and ones; and sometimes it is necessary to compose a ten.
CCSS.Math.Content.1.NBT.C.5
Given a two-digit number, mentally find 10 more or 10 less than the number, without having to count; explain the reasoning used.
CCSS.Math.Content.1.NBT.C.6
Subtract multiples of 10 in the range 10-90 from multiples of 10 in the range 10-90 (positive or zero differences), using concrete models or drawings and strategies based on place value, properties of operations, and/or the relationship between addition and subtraction; relate the strategy to a written method and explain the reasoning used.

References

Bloom, B. S. (1984). The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. *Educational Researcher*, *13*(6), 4–16.

Hanushek, E. A., & Rivkin, S. G. (2012). The Distribution of Teacher Quality and Implications for Policy. *Annual Review of Economics*, *4*(1), 131–157.

Koedinger, K. R., Anderson, J. R., Hadley, W. H., & Mark, M. A. (1997). Intelligent Tutoring Goes To School in the Big City. *International Journal of Artificial Intelligence in Education (IJAIED)*, *8*, 30–43.