Workshop W6 on Empirical Methods for Tutorial Dialogue Systems

Held during ITS 2002, the Sixth International Conference on Intelligent Tutoring Systems

San Sebastian, June 4, 2002
Preface

Tutorial Dialogue Systems have been an area of great emphasis in the research field of Intelligent Tutoring Systems, particularly in the past 5 years. As a sign that the area of Tutorial Dialogue Systems is maturing, a great deal of empirical work in this area is being presented at tutoring conferences, focusing both on evaluation of implemented systems (both formative and summative) and transcript annotation and analysis. The goal of this workshop is to bring together researchers working on aspects of these problems in order to take inventory of what we have already learned from these endeavors and to build a group consensus for what are the most important questions to pursue in our continued work in this area.

We received a wide range of very high quality submissions, which we believe will provide a springboard for a very productive workshop discussion. The papers we received fall neatly into three general categories (1) transcript annotation and analysis (2) progressive refinement evaluations (3) full system evaluations.

Work in tutorial dialogue analysis makes it possible for us to learn from what works and doesn't work in human-human tutorial dialogue. Work in this area has brought to the foreground open questions related to the coding schemes being developed for tutorial dialogue, the kinds of features of tutorial dialogue that should be captured in the coding schemes, and the level of analysis that is most useful in order for example to help designers of Tutorial Dialogue Systems. We received three papers focusing on these issues. In particular, Jordan and Siler as well as Core et al. focus on different aspects of the general topic of student initiative. Litman presents a proposal for a project that involves using tutorial dialogue analysis in its development stages. In this segment of the workshop, in addition to discussion of these specific papers, Mark Core will give an invited talk discussing previous work on the topic of coding initiative in dialogues, and there will be a coding exercize to give workshop participants the opportunity to gain hands-on experience with this type of empirical work.

During the development stages of a system, pilot testing and progressive refinement evaluations make it possible to identify strengths and weaknesses in various approaches, both on the level of methodology and implementation. Mostow et al. and Marineau et al. describe their methodology for analyzing the logfiles from student interactions with their systems.
Fiedler and Gabsdil describe a testbed for experimenting with different dialogue strategies. Rose et al. and Hockey et al. both focus on progressive evaluations focusing on specific problems. In particular, Rose et al. focus on the problem of selecting interventions for high performance students, while Hockey et al. focus on the problem of providing constructive feedback in the case where the system fails to understand what the student says.

The ultimate goal of our work in tutorial dialogue systems is to build and evaluate effective tutorial dialogue systems that yield significant learning gains. Nevertheless, many open questions are left to be answered about how best to evaluate these systems. For example, what are key indicators of dialogue effectiveness and how do they relate to learning? We received four papers describing whole system evaluations, in particular Heffernan, Siler et al., Aist et al., and Aleven et al. In addition to discussion of these specific papers, this session will feature an invited talk by Ken Koedinger discussing the current state of the art in evaluating tutorial dialogue systems.

We are grateful to our very helpful organizing committee, who helped us in a wide range of ways including offering advice on the organization of this workshop, reviewing papers, and being willing to take an active role in leading the coding exercises and giving invited talks. This group of individuals includes Jeff Rickel, Johanna Moore, Art Graesser, Pamela Jordan, Diane Litman, Barbara Di Eugenio, Jack Mostow, Mark Core, Claus Zinn, Greg Aist, Ken Koedinger, and Reva Freedman. We couldn't have done it without you guys!

We look forward to an exciting workshop in San Sebastian. See you there!

Sincerely,

Carolyn P. Rose (University of Pittsburgh)
Vincent Aleven (Carnegie Mellon University)
Organizing Committee

Carolyn Penstein Rose (co-chair), University of Pittsburgh
Vincent Aleven (co-chair), Carnegie Mellon University
Gregory Aist, NASA Ames Research Center
Mark Core, University of Edinburgh
Barbara DiEugenio, University of Illinois at Chicago
Reva Freedman, Northern Illinois University
Art Graesser, University of Memphis
Pamela Jordan, University of Pittsburgh
Kenneth Koedinger, Carnegie Mellon University
Diane Litman, University of Pittsburgh
Johanna Moore, University of Edinburgh
Jack Mostow, Carnegie Mellon University
Jeff Rickel, University of Southern California
Claus Zin, University of Edinburgh
Workshop Schedule

9:00 – 9:10 Opening Remarks

Overall System Evaluations
9:10 – 9:55 Ken Koedinger (invited talk)
9:55 – 10:20 Heffernan
10:20 – 10:45 Siler et al.
10:45 – 11:10 Aist et al.
11:10 – 11:30 Coffee Break
11:30 – 11:55 Aleven et al.
11:55 – 12:15 Discussion

Annotation
12:15 – 1:00 Mark Core (invited talk)
1:00 – 3:00 Lunch
3:00 – 3:20 coding exercise discussion
3:20 – 3:45 Jordan and Siler
3:45 – 4:10 Core et al.
4:10 – 4:30 Litman

Formative Evaluations
4:30 – 4:50 Fiedler and Gabsdil
4:50 – 5:10 Hockey et al.
5:10 – 5:25 Coffee Break
5:25 – 5:50 Mostow et al.
5:50 – 6:16 Marineau et al.
6:15 – 6:35 Rose et al.
6:35 – 6:55 Discussion
6:55 – 7:00 Closing Remarks

Evening Dinner in Biarritz (details TBA)
Table of Contents

iii  Preface

v   Organizing Committee

vii  Workshop Schedule

1  Web-Based Evaluation Showing both Motivational and Cognitive Benefits of the Ms. Lindquist Tutor
   Neil T. Heffernan

9  Evaluating Knowledge Construction Dialogs (KCDs) versus minilessons within Andes2 and alone
   Stephanie Siler, Carolyn Penstein Rosé, Tim Frost, Kurt VanLehn, and Peter Koehler

16 Experimentally Augmenting an Intelligent Tutoring System with Human-Supplied Capabilities:
    Adding Human-Provided Emotional Scaffolding to an Automated Reading Tutor that Listens
    Gregory Aist, Barry Kort, Rob Reilly, Jack Mostow, and Rosalind Picard

29 More on pilot-testing a tutorial dialog system that supports self-explanation; Good feedback on
    explanations helps; how we can generate more of it?
    Vincent Aleven, Octav Popescu, and Kenneth R. Koedinger

39 Student Initiative and Questioning Strategies in Computer-Mediated Human Tutoring Dialogues
   Pamela W. Jordan and Stephanie Siler

46 Initiative in Tutorial Dialogue
   Mark G. Core, Johanna D. Moore, and Claus Zinn

56 Adding Spoken Dialogue to a Text-Based Tutorial Dialogue System
   Diane J. Litman

62 Supporting Progressive Refinement of Wizard-of-Oz Experiments
   Armin Fiedler and Malte Gabsdil

70 Targeted Help: Embedded training and methods for evaluation
   Beth Ann Hockey, Gregory Aist, Jim Hieronymus, Oliver Lemon, John Dowding

75 Viewing and Analyzing Multimodal Human-computer Tutorial Dialogue: A Database Approach
   Jack Mostow, Joseph Beck, Raghu Chalasani, Andrew Cuneo, and Peng Jia

85 AutoTutor's Log Files and Categories of Language and Discourse
   Johanna Marineau, Andrew Olney, Max Louwerse, Natalie Person, Brent Olde, Suresh Susarla,
   Patrick Chipman, Art Graesser, and the Tutoring Research Group

93 Can we help students with a high initial competency?
   Carolyn P. Rose, Kurt VanLehn, and Pamela Jordan
Web-Based Evaluation Showing both Motivational and Cognitive Benefits of the Ms. Lindquist Tutor

Neil T. Heffernan  
neil@cs.cmu.edu  
School of Computer Science  
Carnegie Mellon University

Abstract
In the main conference, Heffernan and Koedinger (2002) report upon the Ms. Lindquist tutoring system. The evaluation results reported were preliminary and limited in that we showed that students learn more when they get to engage in a dialog, but we did not control for time on task so were not sure if it was worth the additional time a dialog takes. This paper reports the latest web-based experimental results, which show learning gains as well as motivational benefits of the system. Experiment 3 found that students learned more, in the same period of time, using Ms. Lindquist, even though the students did about half the number of problems than those in the control condition.

Key words: Intelligent tutoring systems, teaching strategies, experiment, student learning, algebra.

Introduction
Several groups of researchers are working on incorporating dialog into tutoring systems: for instance CIRCSIM-tutor (CIRCSIM-Tutor, 2002), Autotutor (Graesser et al., 2001) and Atlas-Andes (Rose, et al., 2001). The value of dialog in learning is still controversial, and this paper reports on the empirical results of a system that uses dialog.

In the main Intelligent Tutoring Systems conference (that this workshop paper appears as a part of), Heffernan and Koedinger (2002) report upon the Ms. Lindquist tutoring system that uses dialog. The Ms. Lindquist tutoring system helps students become proficient in writing expressions for algebra word problems. This system is of the "coached practice" variety that does not offer explicit instruction (i.e., long web pages or lectures), but instead is meant to scaffold "learning by doing" while students practice their problem solving skills. An assumption in the development of this system was that students would learn more if they could have an intelligent dialog rather then simply being told the answer. In earlier work (Heffernan, 2001) showed that students learned more when they engaged in a dialog with the Ms. Lindquist tutoring system, but did not investigate if it was worth the extra time spent. This paper reports the latest results of experiments that were conducted by students using the tutor via the internet (at www.AlgebraTutor.org ). Each student who visits the web site is informed that they are taking part in an experiment. Then they are randomly assigned to a condition in the experiment. There are several experimental conditions in these experiments, each of them representing different types of "intelligent" feedback that involve a tutorial dialog. These tutorial strategies are based upon our observation of an experienced tutor (Heffernan, 2001). For instance, the "Concrete Articulation" strategy conducts a dialog based upon the idea of using a concrete instance. The following dialog is an example of the Concrete Articulation Strategy and was collected as part of Experiment 3:

Tutor: In the marching band there are 7 fewer flute players than drummers. If the number of drummers is "x", write an expression for the total number of flute players and drummers.
Student [21]: 7-x+x \{\text{Correct answer is } x+(x-7)\}

Tutor: No. We will come back to the total number of flute players and drummers. Let me try to break this down for you. Please compute the number of flute players if the number of drummers was 10 drummers?

Student: 10-7

Tutor: Correct. Now use "x" instead of 10 to again express the number of flute players.

Student [9]: x-7

Tutor: Correct. Now, lets try this question again: Please write an expression for the total number of flute players and drummers.

Student [13]: x-7+x

Tutor: Correct. Let us go onto the next problem.

Another tutorial strategy is the "Explain in English First" Strategy, which asks students to use pull-down menus that use English phrases, before being asked to give an agent in algebraic symbols. What follows is an example of this strategy:

Tutor: A farmer collects "x" eggs from his chickens, and wants to put them into cartons to sell to the supermarket. A carton contains 12 eggs. Write an expression for how many cartons are needed to store the eggs.

Student [42]: 12/x

Tutor: No. We will come back to the amount of cartons needed. Let me try to break this down for you. Can you explain how you would find the amount of cartons needed?

Student [211]: the amount of cartons needed is equal to \{1st menu\} "the number of eggs" \{2nd menu\} "divided by" \{3rd menu\} "the size of a carton."

Tutor: Correct. Now, follow that up by writing an expression for the amount of cartons needed.

Student [13]: x/12

Tutor: Correct. Let us go onto the next problem.

A third strategy is the "Worked Example" strategy that tells the student the answer and then asks them to explain parts of the answer (See Heffernan & Koedinger (2002) for an example). The control strategy is simply to tell the student the correct answer if they make a mistake (i.e., "No. A correct answer is 5m-100. Please type that.") If a student does not make an error on a problem, and therefore receives no corrective feedback of any sort, then the student has not participated in either the control condition or the experimental condition for that problem. In both Experiment 1 and in Heffernan (2001), progression though the curriculum is controlled by mastery learning whereby a student is given problems until they get some number (usually 2 or 3 problems) of problems correct in a row. Students then took a posttest. Experiments 2 and 3 differ in that there is also a pretest, as well as the fact that we controlled for time on task.

The Ms. Lindquist's curriculum is composed of 5 sections, starting with relatively easy one-operator problems (i.e., "5x"), and progressing up to problems that need 4 or more mathematical operations to correctly symbolize (i.e., "5x+7*(30-x)"). At the beginning of each curriculum section, a tutorial feedback strategy is selected that will be used throughout the section whenever the student needs assistance. Because of this set-up each student can participate in essentially 5 separate experiments, one for each curriculum section the student completes. We would like to learn which tutorial strategy is most effective for each curriculum section.

Since its inception in September 2000, over 10,000 individuals have logged into the tutoring system via the website, and hundreds of individuals have stuck around long enough (e.g., 30 minutes) to provide potentially useful data. About one in every 6 individuals log back in at some later point. The systems architecture is such that a user downloads a web page with a Java applet on it, which communicates to a server located at Carnegie
Mellon University. Student's responses are logged in files for later analysis. Individuals are asked to self-identify themselves as either a student, teacher, parent or researcher. We collect no identifying information from minors. Students are asked to make up a login name that is used to identify them if they return at a later time. Students are asked to specify how much math background they have. We anticipate that some teachers will log in and pretend to be a student, which will add additional variance to the data we collect thereby making it harder to figure out what strategies are most effective, therefore we also ask at the end of each curriculum section if we should use their data (i.e., did they get help from a teacher, or are they really not a student). Such individuals are removed from any analyses. It is recognized that there will probably be more noise in web based experiments due to the fact that individuals will vary far more than would normally occur in an individual classroom experiments (Ms. Lindquist is used by many college students trying to brush up on their algebra, as well as student just starting algebra), nevertheless, we believe that there is still the potential for conducting experiments studying student learning. Even though the variation between individuals will be higher, thus introducing more noise into the data, we will be able to compensate for this by generalizing over a larger number of students than is possible in traditional laboratory studies.

In all of the experiments described below, the items within a curriculum section were randomly chosen from a set of problems for that section (usually 20-40 such problems per section). The posttest items were fixed, (i.e., all students received the same 2 items post test for the first section, as well as the same 3 item post test for the second section, etc.) We will now present the experiments we performed.

Experiment 1

The first experiment was designed to replicate the results of Heffernan (2001), which showed that if the number of problems was controlled for, rather than time on task, students learned more from a dialog with Ms. Lindquist, than if they were simply told the answer. After collecting data for several months, we analyzed 3800 individual log files. About 2000 of them did not get beyond the 10-15 minute tutorial, and therefore never began the curriculum. Another 500 more did not do more than a single problem in the curriculum. These groups were dropped from the analysis. Hundreds of students were thrown out of the analysis if they got the first two problems correct, and therefore did not receive any tutoring. We were left with 623 student files for analysis. Our goal was to find out which of the tutorial strategies let to the greatest learning. We used a mastery-learning algorithm that for the first curriculum section pushed them onto the next section after getting 2 problems correct in a row. The results we report on relate to the first curriculum section. Once a student reached the mastery criterion of two problems correct in a row, the student was given a two-item posttest.

There were actually three different experimental conditions in this experiment, with each condition being represented by one of the tutorial strategy mentioned in the introduction. The control condition was the one described in the introduction that told students the answer and then went on to do more problems.

Results for Experiment 1

While doing the analysis for this experiment, we encountered a problem that we should have anticipated. Students that were placed into the control condition used the system for a shorter period of time than those in the control. This "drop-out" rate was significantly higher in the control condition than in any of the experimental conditions. Of the 623 individuals analyzed, 47% of the 225 that received the control condition dropped out, while only 28% of the other 398 dropped out. This difference was statistically significant at the p<.01 level. There was no statistically significant difference between the drop-out rates of the three experimental conditions.

Because of this massive selection effect, we do not bother to report any detailed analysis of the learning results. We will note that the "Explain in English First" tutorial strategy seemed to be the most effective for the first curriculum section, while the "Concrete Articulation" strategy appeared to be the most effective for the second curriculum section. These were merely "suggestions" and not to be taken too seriously, due the potentially serious threat to the validity of this experiment due the selection effect related to dropouts. We did

---

1 Many individuals skip the demonstration, and then realize that this tutor does not address the skills they are interested in, such as symbolic manipulation. Many students are disappointed that the web site does not allow them to submit their own problems, such as their homework problems.

2 Because we did not have a pretest, we could not determine if it was weaker or stronger students that were responsible for the increased drop-out rates. In part, to help to deal with this issue, in the version used in Experiments 2 & 3 we included a pretest.
however use these as guesses in picking which of the tutorial strategies to use in our more refined experiments (Experiments 2 & 3).

We conclude that, as far as from a motivational point of view, the intelligent feedback was superior at getting students to persist in tutoring. We now move on to report Experiment 2 and 3.

**Experiment 2**

After Experiment 1, we made several changes to the system including coming up with a way to deal with the drop-out problem, by focusing our analysis only to those students that were doing the tutor as part of a class assignment. A student entering the Ms Lindquist tutoring site was asked if they were students as well as if they were being required to do Ms. Lindquist by their teacher. If they were being required, they were asked for their teacher's name. Over a period of a few months, we collected over a hundred such files, most of them from 5 different teachers. The teacher that sent the largest number of students to the site, whom we will call "Mr. X", sent about 76 students. We decided to analyze just these students.

We know little about Mr. X, but we can infer some things. From the time-stamps on these files, it appears the students used the system during two classes (One on a Monday, and the other a Friday), and did not return to the system for homework (which is possible since its running over the internet). Every student clicked on the button indicating they were "In 7th or 8th grade". Furthermore, it appears that Mr. X students were from three different classes. We can only guess that Mr. X is a math teacher. This person took three groups of students to a computer lab (as indicated by time stamps), and supervised them while they worked through the tutor. There is a mechanism for students to request the system to send a progress reports to their teacher, but this facility was only used by a few students, so it seems likely that Mr. X did not grade his students according to the measures the systems provides. We also have no idea if this teacher was walking around the lab helping students, or even if the teacher was present. Regardless of whether students were being given assistance by their teacher, we have no reasoning to believe that we would be helping students in the control condition any differently than those students in the experimental condition, so we therefore believe these results to be worth considering. As an experimenter used to conducting studies in classrooms, this sort of information is often important to understand why the experiment turned out the way it did, and of course, it would be nice to have that sort of information for this experiment.

These results were also collected using a slightly different version of the software in which we added a pretest that matched the posttest, thereby allow us to look at learning gains. Another change was the fact that this version controlled for time on task by giving the posttest after a set period of time (that varied according the curriculum section but was somewhere between 6 minutes to 15 minutes). After the posttest, students were moved onto the pretest if they had already reached the mastery criterion, or were given more practice if they had not yet reached the mastery criterion.

We report results from the experienced that was run on the first curriculum section as Experiment 2 and will report the results from the second curriculum section as Experiment 3. There were not enough students who finished the third curriculum section to analyze. The experimental condition in Experiment 2 received the "Explain in English" tutorial strategy, while in Experiment 3 the experimental condition received the "Concrete Articulation" strategy.

<table>
<thead>
<tr>
<th>N</th>
<th>Gain (Pre-Post in # Probs. (p=.54)</th>
<th># Problems Done. (p=.0003)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std.Dev.</td>
</tr>
<tr>
<td>Control</td>
<td>33 .121</td>
<td>.545</td>
</tr>
<tr>
<td>Experiment</td>
<td>29 .034</td>
<td>.566</td>
</tr>
<tr>
<td>Total</td>
<td>62 .081</td>
<td>.552</td>
</tr>
</tbody>
</table>

**Results for Experiment 2**

Mr. X's had 76 students to being with. We excluded 14 of them because they got every practice problem correct, and therefore received no corrective feedback of either the experimental type (i.e., dialog) or of the control type (i.e., being told the answer to type in). The results are summarized in Table 1.

---

3 Student might have been helping each other, but the fact that the problems the students saw were randomly ordered helps to mitigate against cheating by making it harder for a student to just copy answers from each other.
Not surprisingly, since engaging in a dialog takes more time than simply being told the answer, students in the control condition solved a statistically significant larger number of (8.3 problems versus 4.6 problems, p<.0003) in the same period of time.

Unlike Experiment 1, (where there was a confound caused by more drop-outs in the control group) all of Mr. X’s students completed the first section. We refer the difference between the pretest score and the posttest score as the learning gain. Between pretest and posttest, a period of time lasting 6.5 minutes, students learning gain was an average of .081 problems (which is a 4% gain). This difference was not statistically significant for any of the individual conditions (i.e., meaning the hypothesis that the mean was significantly different than zero was not supported), nor overall. The reason students did not appear to learn in this section is probably due to the fact that students came in already knowing this skill rather well (pretest scores=1.58, or 79%, with 40 of 62 students getting both pretest problems correct, evenly split between conditions). Given that there is no evidence of learning, it is not surprising that there was no statistically significant effect of condition upon learning gain (p=.54). We now turn to the results of the second curriculum section where we will see that there was no problem of students entering with too much knowledge.

Experiment 3

After completing the first section, Mr. X’s students were either moved onto section 2 or given more practice on Section 1, if they had yet to demonstrate mastery, by getting two problems correct in a row. Two students did not even get to the second section, due to this requirement.

The time between pretest and posttest was 10 minutes. The students went on to the second curriculum section that involved writing expressions that had two-operators (e.g., 800-40*m). This is what we report as Experiment 3. The control condition was the same as in Experiments 1 and 2. Students in the experimental condition received the Concrete Articulation Strategy for feedback.

Results for Experiment 3

The problems that students solved during this experiment were harder than those of Experiment 2, as measured by the fact that of the 74 students who completed the posttest, their average score on the three items was 1.068 correct (or 36% correct). Therefore there was much less chance of a potential "ceiling effect" than in Experiment 2. During the tutoring session, students got 39% of the problems correct on the first try and therefore received tutoring on the remaining 61% of problems (those in the control condition were again simply told the answer).

61 of Mr. X’s students went on to complete section 2. Three of them never made any errors, so where dropped from the analysis since they received neither the control nor the experimental feedback. Unlike in Experiment 1, there was no reliable differently drop-out rate due to condition (8 in the control condition did not finish, while 7 in the experimental condition did not finish). This lack of an interaction between conditions and drop-out rate suggests that the method of looking at students who were required by their teacher to work on Ms. Lindquist appears to be a nice way to avoid the confound of differential drop-out rates between conditions.

<table>
<thead>
<tr>
<th>Table 2: Students learned more even while doing fewer problems.</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Control</td>
</tr>
<tr>
<td>Experiment</td>
</tr>
<tr>
<td>Total</td>
</tr>
</tbody>
</table>

Given that time was controlled for, it is not surprising that the average number of problems done differed significantly (p<.001) by condition (Control=6.9 problems, Experiment=3.5) (See Table 2).

Averaged over both conditions, the average learning gain of 0.31 problems (or 10% for the 3 problem pre-post test) was statistically significant (p<.007 when compared with the null hypotheses of the learning gain being equal to zero). Interestingly, the learning gain in the control condition was 0.138 problems, while in the experimental condition it was 0.483 problems. This difference in learning gain between conditions approached

4 One student did 34 one-operator problems in a row, never getting two correct in a row. This probably suggests a student who was not reading the problems, and was simply typing in the answers provided by the computer. The student did happen to be in the control condition, where this is possible.
statistical significance ($F(1,56=2.5), p=.12$). The effect size\(^5\) was a respectable .5 standard deviations. Figure 1 shows that even though students in the experimental condition solved about half as many problems, they learned more while doing so.

Figure 1: Students did almost half as many problems in the experimental condition (left), but had higher learning gains (right) between pretest and posttest of close to _ a problem out of a 3-item test, for a gain of about 16%.

Table 3: Student’s learning gain (or loss) broken down by condition.

<table>
<thead>
<tr>
<th>Learning Gain</th>
<th>Experiment</th>
<th>Control</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>2</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>14</td>
<td>26</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>6</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>29</strong></td>
<td><strong>29</strong></td>
<td><strong>58</strong></td>
</tr>
</tbody>
</table>

Table 3 shows how the average learning gain of 0.31 problems, reported above, is broken down by condition. We see that the students in the experimental group tested to learn more on average. There was one student that had a learning gain of 3 problems and this person was in the control group. Upon inspection of this student's file, we found that the student did not complete two of the pretest items (probably just hit the return key instead of answering them). Furthermore, this student did only two practice problems before getting to the posttest. On the second practice problem this student got the wrong answer and then was told the answer, however he/she refused to type that answer in and instead typed "garbage" answers for 30 consecutive turns. It seems reasonable to consider how sensitive the results reported above are to the presence of this one student that appears to be an "outlier" student who is over three standard deviations from mean for all students. It turns out that if this student is excluded, then the average learning gain in the control become a very small .04 problems. Our tests of statistical significance tell us that this small number is not statistically significantly different than zero (p=.55) leading us to reject the hypotheses that student's pretests and posttests results differ significantly. Furthermore, the interaction between condition and learning gain switches from marginal significance to become statistically significant (from p=.13 to p=.03). The effect size goes from .35 to .56 standard deviations. This further supports the hypothesis that students really did learn more in the experimental condition, even though they did fewer problems.

\(^5\) Effect size is defined as the difference between the two groups divided by the standard deviation of the control group.
Hypothesis: Dialog encourages learning because it is viewed as a penalty

Looking for instances that seemed to suggest where a learning event might have happened, we read over all the student transcripts that showed a learning gain in the experimental condition in Experiment 3. We failed to find examples of what appeared to be clear examples of what looked like great tutoring. (The students themselves seemed to show lots of "sloppiness"). Because of this, we wondered why did the experimental condition show higher learning gains than the control condition? One alternative hypothesis to explain these results is that students in the experimental condition were more motivated to get an answer correct because they perceived the ensuing dialog as a penalty. In the control condition students can take a guess at a problem, and if they are wrong they are simply told the answer, but in the experimental condition, they will get asked new questions which they might view negatively.

To guard against this hypothesis explaining our results, we looked to see if students in the experimental condition of Experiment 3, spent more time composing their response on the posttest than those students in the control. It turned out that both groups took the same amount of time (The experimental group took 58 seconds while the control group took 60 seconds, a difference that was not statistically significant (p=.8)). It is also true that both groups took the statistically significant same amount of time to compose their initial response for each new problem during the practice period. (The experimental group took 71 seconds while the control group took 70 seconds, a difference that was not statistically significant (p=.6)). Therefore, the hypothesis that students might learn more from dialog because they view the dialog as a penalty, and consequently concentrate more, seems not to be supported by the data.

Web Based Experiments

Web based experimentation holds the promise of making it possible to run experiments more efficiently by eliminating the need to supervise these experiment directly (see Birnbaum (2000) for a survey of this growing field). In theory, web experiments can drastically increase the rate of knowledge accumulation.

Experiment 1 showed one of the potential threats to internal validity of any between-subjects experiment on the web. Steps like asking students when they finish a section if their data should be used is one of the useful steps that helps to wed out data that should not be used. Experiments 2 and 3 showed that it is possible for to run a valid between-subjects experiment if we take some steps to ensure that the two conditions will have similar completion rates. Nevertheless, of course, there are still threats to the experiments validity, which can better be controlled for in a laboratory experiment (i.e., Heffernan 2000). For instance, it might have been the case that Mr. X helped the students in the experimental condition more, because they got these more unusual dialogs, and that it is this help that explains that groups higher learning gains.

My recommendation to others is that web-based experiments can be a good way to cheaply test systems, but should also be used in conjunction with classroom and laboratory experiments.

Conclusion

There appears to be motivational benefits as well as learning benefits to student that engage in a dialog with Ms. Lindquist. Experiment 1 showed students would do more problems if given one of the dialog conditions. Experiment 3 showed that students learned more with a dialog in the same period of time, even while doing fewer problems.

Future Work

For future work, we want to replicate the results in Experiment 3 in a more laboratory like setting, hoping to get a more statistically significant effect and thereby build more confidence in the conclusion in this paper. We will continue to add new tutorial strategies and evaluate them. It might be that students learn more if the tutor starts out with the "worked example" tutorial strategy, and then moves to one of the other strategies. We would also like to investigate the effectiveness of reflective follow-up questions. In particular, we would like to give these reflective follow-up questions at points where it appears the student likely guessed the answer. We would also like to investigate if there is a benefit to having multiple strategies. Our long-term goal is to have built an effective tutoring system that is based upon scientific evidence that documents both learning effectiveness as well as motivation gains. In doing so, we will establish knowledge about student cognition as well as creating an explicit theory of tutoring.

[^6]: Kim (2002) is presently engaged in an effort that is similar to the effort that went into building Ms. Lindquist.
Acknowledgement

This research was funded in part by the Spencer Foundation as well as an NSF grant to the CIRCLE center in Pittsburgh.

References


CIRCSIM-Tutor (2002) (See http://www.csam.iit.edu/~circsim/)


Heffernan, N. T., & Koedinger, K. R. (2002) An intelligent tutoring system incorporating a model of an experienced human tutor. To be published in the Sixth International Conference on Intelligent Tutoring Systems


Evaluating Knowledge Construction Dialogs (KCDs) versus minilessons within Andes2 and alone

Stephanie Siler, Carolyn Penstein Rosé, Tim Frost, Kurt Vanlehn
Learning, Research and Development Center
University of Pittsburgh, Pittsburgh, PA 15260
siler@pitt.edu

Peter Koehler
Department of Physics, University of Pittsburgh, Pittsburgh, PA 15260

Abstract
In this paper, we will compare student learning of basic physics concepts when they engage in natural language dialog, specifically, in Knowledge Construction Dialogs (KCDs) with student learning when they simply read about the physics concepts (minilessons). Because KCDs require increased student activity through incremental question-answering, we expected students to learn more from the KCDs than the minilessons. We conducted a study in which we compared student learning from problem-solving in Andes2 with either KCDs or expository minilessons. No learning differences were found. We then conducted a follow-up study in which students either worked through 30 KCDs or read 30 corresponding minilessons, and found higher learning gains for students in the minilesson condition. However, because these results may have been influenced by technical difficulties experienced by students in the KCD condition or student fatigue in the KCD condition, we conducted another study in which we reduced the number of KCDs and minilessons in the intervention phase and fixed the problems leading to server crashes. In this study, there was no difference between conditions in student learning; however, more subjects need to be run before we can make any conclusions regarding the relative effect of student learning from KCDs and minilessons. Finally, we discuss possible explanations for the null results and plans for a future study.

keywords: interactive human-computer dialog

Introduction
When designing intelligent tutoring systems, it is important to apply our understanding of how students learn to its design (e.g., Anderson, Corbett, Koedinger & Pelletier, 1995). One factor that correlates with student learning is student activity. For example, generating rather than reading words has been shown to promote later recall of those words (e.g., Slamecka & Graf, 1978). Promoting students’ self-explanations, which includes relating new information to students’ prior knowledge, has been shown to promote student learning (e.g., Chi, DeLeeuw, Chiu & LaVancher, 1994; Renkl, 1997). Furthermore, promoting student activity via scaffolding by human tutors, which included incrementally questioning students on a given conceptual topic, was associated with student learning in Chi, Siler, Jeong, Yamauchi and Hausmann (2001). These empirical findings suggest that promoting student activity in a similar manner in intelligent tutoring systems may also increase student learning.

With these principles of student learning in mind, we developed Knowledge Construction Dialogs (KCDs) (Freedman et al., 2000; Rosé et al., 2001). KCDs are interactive directed lines of reasoning that are each designed to lead students to learn as independently as possible one or a small number of concepts. When a question is presented to a student, the student types a response in a text box in natural language. The student may also simply click on Continue, and thus neglect to answer the question. If the student enters a wrong or empty response, the system will engage the student in a remediation sub-dialogue designed to lead the student to the right answer to the corresponding question (refer to the italicized text in the example KCD, below). The system selects a subdialogue based on the content of the student’s response, so that incorrect responses that provide evidence of an underlying misconception can be handled differently than responses that simply show ignorance of correct concepts. Once the remediation is complete, the KCD returns to the next question in the directed line of reasoning.
Here is an example KCD with remediation subdialogues italicized:

**Tutor:** Let's think about the difference between speed and velocity. A closely related distinction is that of the difference between distance traveled and displacement from the origin. Take as an example a bumblebee flying from point A to point B by means of a curvy path. If you draw a vector from point A to point B, you will have drawn the bee's displacement vector. What does the displacement vector represent?

**Student:** The distance traveled by the bee from its starting point to its ending point.

**Tutor:** The displacement vector only represents the net change in position accomplished by the bee's motion. So the displacement vector does not tell you how far the bee traveled. It only tells you the net change in the bee's position. If you want to know the actual distance traveled by the bee, what entity do you have to measure the length of?

**Student:** The bee's path.

**Tutor:** Right. The bee's displacement vector is obviously a vector. Is the actual distance traveled by the bee a vector or a scalar?

**Student:** Scalar

**Tutor:** Great. So distance traveled is the scalar counterpart to displacement, which is a vector quantity. Now, can you tell me what is the difference between speed and velocity?

**Student:** Velocity is a vector composed of speed and direction.

**Tutor:** Speed is a scalar. So when you measure the bee's speed, do you want to use the distance traveled, which is a scalar, or the displacement, which is a vector?

**Student:** Distance traveled.

**Tutor:** Good. So the equation for speed is the length of the path traveled by the body divided by duration of the motion, regardless of how curvy the path is, even if the path leads back to where the body started from.

Prior empirical work (Rosé, Jordan, Ringenberg, Siler, VanLehn & Weinstein, 2001) demonstrated that students learned more from the Andes quantitative physics tutoring system (Gertner & VanLehn, 2000) when it was enhanced with KCD capabilities than when it did not include dialog capabilities. However, the KCDs in this experiment had different content than the expository minilessons used in the control condition. In particular, the minilessons only summarized the main point of the corresponding KCD, they did not present the entire line of reasoning employed by the KCD to teach the concept. Thus, the positive effect could have been due to differences in the content rather than the dialogue vs. monologue difference.

To address this possibility, in this paper, we will compare student learning of basic physics concepts from Knowledge Construction Dialogs (KCDs) to student learning from minilessons, non-interactive lessons that contain all of the conceptual content of the corresponding KCDs. Thus, in both conditions, students were exposed to the same lines of reasoning to teach each concept. For each KCD, the corresponding minilesson was systematically derived from the main directed line of reasoning of the KCD, but did not include any of the content from potential remediation subdialogues. The minilessons and the directed lines of reasoning of the corresponding KCDs have about the same average number of words (259 and 264, computed over 15 different minilessons/KCD pairs, a non-significant difference). The content of the KCD directed lines of reasoning and their corresponding minilessons are intended to be virtually identical.

Here is an example minilesson:

Here are a few things to keep in mind when calculating acceleration for a body at rest. Acceleration is change in velocity over time. If velocity is not changing, then there is zero acceleration. Now, if for a finite time interval the velocity remains zero, then it is true that it is not changing. Therefore when a body is at rest during a time interval, its acceleration within that time interval is also zero.

Here is the KCD corresponding to the minilesson:

**Tutor:** Here are a few little things to keep in mind when computing the acceleration vector for a body at rest. Acceleration is change in what over time?

**Student:** velocity.

**Tutor:** Right.

If the velocity is not changing, what is the magnitude of the acceleration?

**Student:** zero.

**Tutor:** Sounds good.
If velocity remains zero within a time interval, how much is it changing?

**Student:** zero.

**Tutor:** Super.

So if a body is at rest during a time interval, what is the magnitude of the acceleration within that interval?

**Student:** 0.

**Tutor:** Correct.

In this paper, we will discuss student learning from KCDs and minilessons both within a computer tutoring system, Andes2, and on their own (outside of any tutoring system). Additionally, we will consider some qualities of the KCDs that are associated with learning gains. Because they are more consistent with the principles of active learning, we expected students to learn more from interacting with KCDs than from reading minilessons.

**Fall 2001 Evaluation**

To determine whether the natural language *interactions* rather than the conceptual content of the KCDs was responsible for student learning gains in the Rosé et al. (2001) study, we ran a study in the Fall of 2001. Subjects for this study were paid volunteers who either were taking or had taken an introductory physics course. Students completed a pretest that consisted of 30 multiple-choice questions designed to assess students’ knowledge of the target concepts taught by the KCDs or minilessons that could be elicited during problem-solving and 6 quantitative multi-step problems. The maximum possible score on the multiple-choice section of the pretest was 41. Students were given step-by-step instructions on using Andes2 to solve two warm-up problems, and then were to solve 8 problems in Andes2 on their own. In the intervention phase of this study, we compared two versions of Andes2, Andes2-KCD and Andes2-minilesson. The only difference between these two versions of Andes2 was that a student could receive a KCD in Andes2-KCD following signs the student did not understand a relevant concept, whereas in Andes2-minilesson version the student would instead receive the corresponding minilesson. After the intervention phase, students completed the posttest, which was identical to the pretest.

Of the subjects who took both the pretest and posttest (12 Andes2-minilesson and 14 Andes2-KCDs subjects), the mean pretest score for multiple-choice questions of Andes2-minilesson subjects was higher (but not significantly higher) than for the Andes2-KCD subjects (refer to Table 1, below). The mean posttest score for the multiple-choice questions was marginally significantly higher for Andes2-minilesson subjects (refer to Table 1). No subjects reached the maximum possible score on either the pretest or posttest; thus, there was no ceiling effect. Mean times per problem solved were 0.91 and .84 hours, respectively, a non-significant difference. There was no interaction between condition and either pretest scores or total time on training. There was no difference between conditions after covarying out time on training and pretest scores.

<table>
<thead>
<tr>
<th></th>
<th>Andes2-KCD</th>
<th>Andes2-Minilesson</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pretest</td>
<td>24.3 (6.6)</td>
<td>28.8 (7.6)</td>
</tr>
<tr>
<td>Posttest</td>
<td>26.6 (6.9)</td>
<td>30.3 (7.1)</td>
</tr>
</tbody>
</table>

To see if a more fine-grained analysis would uncover any differences between conditions, we did the following. For pretest questions that students did not correctly answer, we looked at whether students correctly answered that question on the posttest. Instances in which students received a KCD or minilesson on a topic they did not understand in the pretest were considered learning opportunities. We expected learning opportunities to result in learning more frequently if students had received KCDs or minilessons than if they did not receive such remediation. For the students who completed the posttest, there was a total of 82 learning opportunities for the 14 students in the Andes2-KCD condition (on average, 5.9 per subject) but only 22 learning opportunities for the 9 students in the Andes2-minilesson condition (on average, 2.4 per subject; due to technical difficulties, we lost data of three subjects in this condition). Unfortunately, there weren’t enough learning opportunities to compare students’ gain scores on individual KCD/minilesson topics (which would control for concept difficulty), so all learning opportunities were aggregated. Students in the Andes2-KCD condition had a 41% likelihood of gaining on a concept given that they received the corresponding KCD,
whereas students in the Andes2-minilesson condition only had a 23% likelihood of gaining, given that they received the corresponding minilesson. This difference in gain, however, was not significant.

This non-significant result may have been due to students in the Andes2-KCD condition not answering all questions in the KCD. Thus, we looked at the log files of the KCD dialogs to see whether subjects who could gain on the KCD topics were actually answering the questions. We excluded the learning events for which students did not answer all questions in the relevant KCD (i.e., there was at least one “I don’t know” or nonsensical response in the log files. “I don’t know” responses in log files indicate that either the subject typed “I don’t know” or simply clicked “Continue”, without entering a response, which is recorded in the log files as an “I don’t know” response. A nonsensical response includes non-word responses such as “ll”). We found that some of these students sometimes did not respond to the questions, and, based on the times between turns logged in the KCD dialogs, they did not appear to be reading the questions either. One student often canceled KCDs immediately when they were elicited by typing “cancel”. However, it is possible that instances of “I don’t know” or nonsensical responses had different causes, including lack of student effort, neglecting to respond because the answer was known, and actually not knowing the answer. Instances of actually not knowing the answer may lead to cognitive conflict, which is associated with learning (e.g., Dreyfus, Jungwirth, & Eliovitch, 1990; Bearison, Magzamen, & Filardo, 1986; Tudge, 1989). An analysis of the time between turns may help us uncover which of these possibilities is more likely in different cases. Thirty-four learning opportunities were excluded because the student did not attempt to respond to each question in the KCD, leaving 48 learning opportunities for students in the Andes2-KCD condition.

Using only learning opportunities in which students in the KCD condition answered all questions in the KCDs and reached the end of the KCDs, gains were compared between conditions. Reducing our data in this manner may unfairly favor the KCD condition because we have no similar way to reduce the data in the minilesson condition. However, we performed this analysis to determine if lack of student responses may have contributed to the overall null finding. Given that a student in the KCD condition did not know a concept in the pretest and had worked through the entire KCD once in the learning intervention, there was a 41% likelihood of gaining (20 out of 48 learning opportunities). Given that students in the minilesson condition did not know a concept in the pretest but received the minilesson addressing that concept in the learning intervention, they were only 23% likely to gain (5 out of 22 opportunities). Though in the expected direction, again this difference was not significant. The possibility exists that students in the minilesson condition did not always read the minilessons. To address this possibility, we looked at the times minilessons were displayed, which were recorded in the log files. We found a bimodal distribution of times clustered at 4.5 seconds and at 56 seconds. We eliminated the 16 minilesson learning opportunities whose times were clustered around 4.5 seconds. Of the remaining 6 minilesson learning opportunities, students gained on 2 (33%).

It is also possible that students were receiving inappropriate feedback for their responses during KCDs. To test for this possibility, we analyzed roughly one-third of the KCD transcripts and found a 100% precision (the percentage of students’ answers coded as correct that actually were correct) and a 96% recall (the percentage of correct answers identified as correct). Thus, students did receive negative feedback for correct responses 4% of the time. Though infrequent, the impact that receiving negative feedback had on students’ effort and motivation is currently being investigated. Specifically, we are looking at instances in which students received negative feedback on correct responses and seeing whether this was associated with a subsequent increase in students’ “I don’t know” or nonsensical responses. A second type of inappropriate feedback is receiving positive feedback for wrong or off-target responses, which may also confuse students. This is also being investigated.

Additionally, because KCD/minilesson topics were aggregated to achieve statistical power, we looked at the possibility that the concepts KCD students gained on were generally more difficult than the concepts that students in the minilesson condition gained on. Difficulty of a concept was operationalized as the average probability that the 26 students who completed the posttest demonstrated a lack of understanding of those concepts in the pretest. The average learning opportunity concept was more difficult in the Andes2-minilesson condition than in the Andes2-KCD condition (58% versus 45% chance of answering incorrectly in the pretest, respectively); thus, concept difficulty was not responsible for KCD subjects failing to out-gain minilesson subjects.

Because prior knowledge highly correlates with student learning, a second important factor to consider is differences between conditions in students’ pretest scores. However, the average pretest scores associated with a learning event were similar across Andes2-KCD and Andes2-minilesson conditions (22.4 and 24, respectively).
**Spring 2002 Follow-up study**

To increase the number of learning opportunities by exposing students to all KCDs and minilessons and to increase the likelihood that students’ knowledge gains are prompted by the KCDs and minilessons themselves rather than Andes, we ran a follow-up experiment. We advertised our study on a website that listed experiments available to students taking an introductory psychology class, who were required to participate in experiments for course credit. We also posted flyers around campus. Subjects who were recruited from the psychology subject pool received course credit for their participation; subjects recruited from campus flyers were paid $7/hour. All subjects had taken or were currently taking an introductory physics course. Subjects completed a physics pretest that was similar to the tests taken by subjects in the Fall 2001 evaluation but included additional multiple-choice questions designed to assess students’ knowledge of concepts addressed by additional KCDs or minilessons that were not relevant in the Fall 2001 evaluation. Because students were not solving physics problems in this study and thus, we did not expect any learning gain at the problem level, the pretest in this follow-up study also did not include any quantitative problems. After completing the pretest, subjects read through a set of 30 minilessons or worked through the corresponding set of 30 KCDs. Immediately after completing the set of minilessons or KCDs, all students completed the physics posttest, identical to the pretest. Both tests consisted of 44 multiple-choice and short-answer questions designed to tap students’ knowledge of the concepts addressed by all of the KCDs and minilessons. The maximum possible score on both the pretest and posttest was 47 (three questions were broken down into 2 sub-parts).

The mean pretest scores of the 11 students in the KCD and the 10 students in the minilesson condition who read through all 30 KCDs or minilessons were similar (refer to Table 2, below). Students in the minilesson condition scored marginally higher on the posttest (refer to Table 2). After covarying students’ pretest scores (there was no interaction between students’ pretest scores and condition), students in the minilesson condition still out-performed students in the KCD condition, $F(1,18) = 3.8$, $p < .10$.

<table>
<thead>
<tr>
<th>Table 2. Mean scores (and standard deviations)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KCD condition</strong></td>
</tr>
<tr>
<td>Pretest 25.6 (8.7)</td>
</tr>
<tr>
<td>t(19)=0.64, $p = .53$</td>
</tr>
<tr>
<td>Posttest 28.5 (7.6)</td>
</tr>
<tr>
<td>t(19)=1.84, $p &lt; .10$</td>
</tr>
</tbody>
</table>

Again, to get a more precise measure of the relative learning benefits specifically from working through KCDs versus reading minilessons, we performed the same fine-grained learning opportunity analysis as in the Fall 2001 Andes2-KCD evaluation. In the KCD condition, students gained on 61 of the 223 opportunities (only 27%), whereas in the minilesson condition, students gained on 84 of the 191 opportunities (or 44%). Consistent with the overall difference between conditions, this difference in gain was significant, $\chi^2 = 12.5$, $p < .001$. KCD students gained on 32 of the 118 learning events, or 27%, identical to the rate including all learning opportunities. Once again, students’ “I don’t know” responses did not appear to be associated with decreased learning.

One explanation for this unexpected result was that the high frequency of server that occurred while students worked through the KCDs interfered with their learning. Another possible factor was that students in the KCD condition, who took longer to finish the 30 KCDs than minilesson subjects took to read, simply became more tired.

**Follow-up study to Spring 2002 study**

To test these two possibilities, in a second follow-up study, we reduced the number of KCDs and minilessons to 15 and fixed the problems leading to server crashes. Twelve students who were currently taking or had recently taken introductory physics class were recruited from flyers posted around campus. They were paid $7/hour. Twelve subjects were randomly assigned to each condition (6 in each condition). As in the initial follow-up study, students completed the multiple-choice pretest (identical to the pretest given to subjects in the initial Spring 2002 study), worked through the 15 KCDs or read the 15 corresponding minilessons, then immediately completed the multiple-choice posttest (identical to the posttest given to subjects in the initial Spring 2002 study). The maximum possible score on both the pretest and posttest was 47. The mean pretest score of KCD subjects was significantly higher than that of Minilesson subjects (refer to Table 3, below). The mean posttest score of subjects in the KCD was marginally significantly higher than that of the Minilesson subjects (refer to Table 3). The maximum posttest score in the KCD condition was 43; thus, no ceiling effect existed. After covarying pretest scores (again there was no interaction between students’ pretest scores and condition), there
was no difference between conditions in posttest scores. Thus, mental fatigue and/or server problems may have contributed to some extent for the initial follow-up results. However, a null finding may simply be due to low number of subjects run in this study.

Table 3. Mean scores (and standard deviations)

<table>
<thead>
<tr>
<th></th>
<th>KCD condition</th>
<th>Minilesson condition</th>
<th>( t^1(10) ) = 2.63, p &lt; .05</th>
<th>( t^1(10) = 2.11, \ p &lt; .10 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pretest</td>
<td>34.7 (8.8)</td>
<td>24.3 (3.8)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Posttest</td>
<td>36.8 (8.2)</td>
<td>28.5 (5.2)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( ^1 \)two-tailed t-tests.

We again performed the same fine-grained analysis as in the prior two studies, in which we looked at subjects’ likelihood of gaining on a concept, given they had seen the relevant KCD or minilesson. Because the pretest and posttest both included questions addressing concepts not addressed by the KCDs or minilessons, this analysis was even more sensitive to the effects of minilessons versus KCDs on student learning relative to differences in total test scores. Subjects in the KCD condition gained on 24 of the 51 learning opportunities (or 47%), whereas subjects in the minilesson condition gained on 33 of the 83 learning opportunities (or 40%). Consistent with the total test score results, this difference was not significant. However, before we can conclude no learning differences, we need to run more subjects.

Qualities of KCDs

Because generating responses should lead to more effective learning (e.g., Slamecka & Graf, 1978), we looked more closely at how concepts were elicited within the KCDs. For some concepts, students were asked to generate a response for a specific concept. For example, in the KCD that discusses Newton’s Third Law, the student is asked to state the relationship between the directions of a force and its reaction force. For other concepts, the student is not questioned on the concept but is told the concept. For example, the centripetal acceleration relationship \( a = \frac{v^2}{r} \) is stated near the end of a KCD. For the 15 KCDs given to subjects in the last study, we determined whether the corresponding test questions targeted concepts that students were questioned on or stated in the KCDs. For KCD subjects in both follow-up studies, gains were more likely on questioned than stated concepts (40 out of 82, or 49% of questioned concepts were gained on versus only 13 out of 56, or 23% of stated concepts were gained on, a significant difference, \( \chi^2 = 9.2, \ p = .002 \)). However, questioned concepts seemed to be less difficult than stated concepts: on average, students in both Spring evaluations were 44% likely to not have known a questioned concept on the pretest but 77% likely not to have known a stated concept.

Current directions

In summary, we have no evidence that engaging students in dialog enhanced student learning over having students simply read the directed line of reasoning of the dialog, though it may just be that we didn’t run enough subjects to see a difference. However, these results suggest the possibility that the directed lines of reasoning rather than the dialogue led to student learning in the study reported in Rosé, Jordan, Ringenberg, Siler, VanLehn & Weinstein (2001), in which students learned more from the KCD-enhanced Andes than when Andes did not include KCDs. Perhaps responding to questions within the KCDs led students to attend better to the individual steps of the dialog but not to the target concept(s) of the dialogs, which were assessed during testing.

To assess this possibility, we plan to run a study this Fall with the following four conditions: (1) students will read minilessons, (2) students will read minilessons and summarize each one, (3) subjects will work through KCDs, and (4) subjects will work through KCDs and summarize each. We will assess student learning of both target concepts and concepts presented in the individual steps of the KCDs. Our hope is that asking students to summarize after working through each KCD will lead to increases in learning of both types of concepts.

Acknowledgements

This research is supported by the Office of Naval Research, Cognitive and Neural Sciences Division MURI Grant N00014-00-1-0600 and NSF Grant 9720359 to CIRCLE, a center for research on intelligent tutoring.
References


Experimentally Augmenting an Intelligent Tutoring System with Human-Supplied Capabilities: Adding Human-Provided Emotional Scaffolding to an Automated Reading Tutor that Listens

Gregory Aist1, Barry Kort2, Rob Reilly2, Jack Mostow3, and Rosalind Picard2

1 With Project LISTEN 1996-2001; now Visiting Scientist, MIT Media Lab; and Research Scientist, Research Institute for Advanced Computer Science, Mail Stop T27-A, NASA Ames Research Center, Moffett Field CA 94035-1000 USA
GregoryAist@yahoo.com

2 MIT Media Lab, Building E15, 77 Massachusetts Ave., Cambridge MA 02139 USA
http://affect.media.mit.edu/AC_research/lc/

3 Project LISTEN, RI-NSH 4213, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh PA 15213
http://www.cs.cmu.edu/~listen

Abstract. This paper presents the first statistically reliable empirical evidence from a controlled study for the effect of human-provided emotional scaffolding on student persistence in an intelligent tutoring system. We describe an experiment that added human-provided emotional scaffolding to an automated Reading Tutor that listens, and discuss the methodology we developed to conduct this experiment. Each student participated in one (experimental) session with emotional scaffolding, and in one (control) session without emotional scaffolding, counterbalanced by order of session. Each session was divided into several portions. After each portion of the session was completed, the Reading Tutor gave the student a choice: continue, or quit. We measured persistence as the number of portions the student completed. Human-provided emotional scaffolding added to the automated Reading Tutor resulted in increased student persistence, compared to the Reading Tutor alone. Increased persistence means increased time on task, which ought lead to improved learning. If these results for reading turn out to hold for other domains too, the implication for intelligent tutoring systems is that they should respond with not just cognitive support – but emotional scaffolding as well. Furthermore, the general technique of adding human-supplied capabilities to an existing intelligent tutoring system should prove useful for studying other ITSs too. Keywords: emotional scaffolding, affective computing, guided oral reading, intelligent tutoring systems.
1 Introduction

Educators have long been aware of the importance of motivational factors in learning. Beyond such intuitive awareness, Coles (1999) notes that “Some connections between learning and emotions have been well documented: for example, poor learning can produce negative emotions; negative emotions can impair learning; and positive emotions can contribute to learning achievement and vice versa” (Coles 1999). Neuroscience and cognitive science have made important strides in recent years unraveling the mystery of the role of emotions in cognition, decisions, and learning (e.g., Damasio 1994, Isen 2000).

Today, researchers are mapping out the emotional factors that arise in the course of learning, in an effort to craft more effective educational resources which stimulate and maintain the motivation to learn (Lepper et al. 1993; Kort et al. 2001). Lepper et al. (1993) analyzed how expert tutors use motivational strategies; del Soldato and Boulay (1995) proposed incorporating motivational strategies into tutoring systems. In this paper, we look at adding human-supplied emotional scaffolding to an existing intelligent tutoring system – as a means of exploring the potential benefits of enabling intelligent tutoring systems to be emotionally appropriate.

But what does it mean for an intelligent system to be emotionally appropriate? We can consider first the capability of sensing affective information, and separately the capability of responding in emotionally appropriate ways. A system that can sense affective information (either passively, e.g. by computer vision, or actively, e.g. by asking) may or may not reveal its awareness much of the time. A system that can respond in emotionally directed ways makes decisions based on perceptions and prior knowledge about how best to respond or to not respond, taking into account the student’s affective state. This problem of deciding how to respond is a qualitatively different research problem from the sensing problem. Thus we can place emotionally appropriate computational systems on the chart shown in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>Does not sense/perceive affective information</th>
<th>Senses/perceives affective information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does not supply emotionally directed responses</td>
<td>The norm for computer software.</td>
<td>Example: System silently records examples of user frustration, perhaps for later review by quality assurance team.</td>
</tr>
<tr>
<td>Supplies emotionally directed responses</td>
<td>Example: System supplies spoken apology triggered by network delay</td>
<td>Example: System detects student frustration; presents a slightly easier task.</td>
</tr>
</tbody>
</table>

Table 1. Degrees of kindness: affective computing in sensing and acting.

Ultimately, affective computing systems should both sense and respond to emotions. Research into automated sensing of emotions is underway as described elsewhere (Picard 1997). In the meantime, we would like to further the research agenda of automated responding without having to wait for automated human-level sensing of emotions – and to take advantage of whatever opportunities for improving intelligent tutoring systems may present themselves along the way. A promising way to start is to identify specific aspects of state-of-the-art intelligent tutoring systems where emotional scaffolding (provided at least initially by human wizards) can substantially improve over current practice. In this paper, we present one such study exploring emotional awareness in ITSs, carried out as a cooperative effort between Carnegie Mellon’s Project LISTEN (Mostow & Aist FF 2001) and MIT’s The Affective Learning Companion (Picard 1997; Kort et al. 2001).

The remainder of the paper is as follows. First, we describe a series of quick, informal pilot user studies that we conducted in order to verify underlying assumptions, evolve our research questions, and refine the experimental protocols. Second, we describe the main study we carried out, and its principal results. We then discuss further implications for intelligent tutoring research.
2 Computer Tutor plus Human Wizard

Our fundamental research question for this paper was “In what ways can human-level emotional scaffolding improve on the state of the art in intelligent tutoring systems?” In order to address this question, we used techniques drawn from ablation studies and Wizard of Oz studies. Ablation studies, a common method for studying intelligent tutoring systems, take a working system and deconstruct it, systematically disabling various components and examining the effectiveness of the remaining system. The Wizard of Oz method, a familiar technique for designing interactive systems, replaces the "brains" of a yet-to-be-built computer program with a human wizard, pulling the strings from behind the scenes. In some variations, the wizard automates him- or herself out of the loop (wholly or partially).

Our experiments were as follows. First, we inverted the ablation study. We focused on adding components and testing for increased effectiveness, rather than removing components and testing for decreased effectiveness. Second, we did not want to embed one particular hypothesis in stone (that is, C++ code). Therefore, we imbued Project LISTEN’s Reading Tutor with a human wizard filling in the emotional gaps. We thus created a hybrid with a human in the loop: Reading Tutor Plus.

(We note that this methodology owes a debt of inspiration in part to some sections of Alex Rudnicky’s proposal to develop automated techniques to allow a computer to gradually learn how to do computer technical support (Rudnicky, internal presentation to CMU Sphinx group, fall 2001).)

In this study in particular, we began with Project LISTEN’s Reading Tutor, which listens to children read aloud, and helps them learn to read (Mostow & Aist FF 2001; Figure 1). The Reading Tutor helps a student read a story as follows. The Reading Tutor displays a story one sentence at a time, and listens to the student read all or part of the sentence aloud. The student can click Back to move back to a previous sentence, and Go to move on. The Reading Tutor responds with help modeled in part after expert reading tutors, but adapted to the strengths and weaknesses of computer technology. The Reading Tutor provides praise for good or improved reading.

In our experiments, the affective component consists of the wizard injecting emotionally directed remarks over the same audio circuit by which the student hears the Reading Tutor – without, so far, controlling the Reading Tutor’s behavior directly as is normally done in Wizard of Oz studies. In the initial study design, both the Reading Tutor’s own praise and the human-supplied emotional scaffolding were active. For reasons we describe later, in the main study design, the Reading Tutor tutored the same as it ordinarily would – but without the praise components previously introduced into the system. But what should the human wizard say? If the human wizard can say anything at all, the only reasonable conclusion from statistically reliable differences between trials with human support and those without (we argue) is that the difference is due to the (active) presence of the human wizard. While probably true, such a distinction is not terribly interesting since it does not shed any light on exactly what the wizard did that was helpful; nor is that distinction terribly useful since it does not shed any light on exactly how to automate the wizard's scaffolding.

We therefore set about to identify concrete ways in which to limit what the wizard said, while still yielding a protocol that we expected to have significant and substantial impact on the tutoring process.
3 Pilot Studies

In order to acquire a set of age-appropriate, relatively similar texts, we decided to adapt a set of limericks written in the 19th century by Edward Lear for research use (see Aist (2000) for previous use of these limericks.) Each limerick contained a single difficult word. The first question we decided to investigate, in late August 2001, was whether an encounter with a new and difficult word would trigger a visible emotional response. We used a within-subject design consisting of repeated trials with words in different conditions for different subjects. This methodology, familiar in medical research as a repeated trials design, has been adapted to intelligent tutoring research in the form of experiments embedded into the course of normal student-Reading Tutor interaction (Mostow and Aist FF 2001; Mostow et al. NAACL 2001). Students read eight children's limericks, four in each of two conditions, as follows:

(experimental) the limerick contained the original difficult word, such as irascible, or (control) the limerick contained a substituted easier word or phrase, such as grouchy.

The conditions were designed so that the experimental condition would elicit a visible emotional response (such as frustration), but the control condition would not. We collected videotaped data from five students. (The first two students' data is more or less unusable due to bad or missing audio.) We wanted to see whether any experiment seeking to manipulate difficulty in order to trigger visible student responses (at the level of individual words) might succeed. Thus for this first pilot study we simply watched the videotapes and formed informal impressions. (Our follow-on study, as reported later in this paper, used more systematic experimental techniques.) Our impressions of the pilot data were as follows: Students did indeed occasionally respond visibly to difficult words, but infrequently; furthermore, whether a response was forthcoming did not seem to be uniformly predictable whether for a given student or a given word. Nonetheless, we were relieved that these observations bolstered one of the foundational assumptions of embedding emotional awareness into intelligent tutoring systems: difficult tasks encountered on an intelligent tutoring system can sometimes trigger visible emotional responses.

We then sought to develop a protocol whereby the human wizard could intervene at specific limited points with emotionally appropriate responses, and the outcome measures would include measures of learning.
In late September 2001, we set up a protocol where students read a series of limericks interspersed with definitions of the hard (target) word. For example:

There was an Old Man of Cape Horn,
Who wished he had never been born;
So he sat on a chair,
Till he died of despair,
That dolorous Man of Cape Horn.
We can say a person is dolorous if they are mournful, or feel really bad.

The human wizard could see and hear the student over a dedicated television link, and was initially restricted to providing emotional scaffolding, such as praise or encouragement, between the limerick and the definition. The student could see and hear the Reading Tutor, and could hear but not see the wizard. The control condition lacked such emotional scaffolding. Both conditions included the previously developed Reading Tutor-supplied praise as described in Mostow and Aist (CALICO 1999) and subsequently updated in later versions of the Reading Tutor.

Our experience after working with just a few students reading under this protocol, however, showed clearly (at least to us) that this limited amount of emotional scaffolding was simply not enough to reasonably expect any difference between the emotionally supported (experimental) trials and the neutral (control) trials – no difference in moment-by-moment observable behavior would, we suspect, mean no differences in student behavior over the course of an entire session. Furthermore, the presence of Reading Tutor-supplied praise might very well confound any effects of human-supplied praise.

4 Revised Protocol

Therefore, we designed a revised protocol. We decided to greatly expand the times when emotional scaffolding was permitted, as follows. First, we made the points when the wizard could provide emotional scaffolding include not just between limericks and definitions, but between every pair of sentences. We also allowed the wizard to intervene to alleviate frustration with a difficult sentence by suggesting that the student move on to the next sentence: a “mercy rule” to maintain a good pace of reading. Finally, we suppressed the Reading Tutor’s built-in praise, so that such praise would not interfere with the wizard in the experimental condition – or dull the contrast between Reading Tutor plus wizard vs. Reading Tutor alone. Why not include the unablated Reading Tutor as an alternate control condition? The number of students we were able to run in this study was too small to permit three conditions; we decided that the comparison of most interest was the Reading Tutor with no emotional scaffolding vs. the Reading Tutor with human-supplied emotional scaffolding.

We used two sets of 10 limericks, with each set sorted in approximate order of the difficulty of the target word. In previous work we had found that second and third graders did not remember much from these limericks when tested on a later day (Aist 2000.) This suspicion that second and third graders were too young for these limericks was confirmed when we tried to have one second grade student read them; we agreed that she was in over her head.

Thus we decided to come up with a different text for the second and third graders: We had these students read a list of progressively harder words.

We decided to get a list of words that would progressively stretch children’s decoding skills. Thus we constructed the list(s) of progressively harder words (Form A, and Form B) as follows.

First, we selected those words with a frequency of two per million words, as given by the Brown corpus (Kucera and Francis 1967).

Second, we excluded those words marked by the MRC Psycholinguistic Database as usually capitalized, for example Lionel, Belgium, Unitarianism.
Third, we sorted the remaining words by the number of syllables and the number of letters.

Fourth, we selected by hand several sets of words, preferentially selecting decodable words (those which can be read left-to-right according to common letter-to-sound rules of English.) The words ranged from one-syllable, three letter words like \textit{bib} and \textit{den} to six-syllable, thirteen- and fourteen-letter words like \textit{acceptability} and \textit{transcendentalism}.

Fifth, we paired the words (e.g. \textit{complimentary} paired with \textit{differentiate}; \textit{acceptability} paired with \textit{accessibility}.)

Sixth, we randomly assigned one word of each pair to Form A, and the other word of each pair to Form B. Each word on Form A thus had a twin on Form B.

egin{center}
\begin{tabular}{l}
\textbf{If someone is accomplished they must be...} \\
\hline
lazy  \\
selfish  \\
lively  \\
skilled
\end{tabular}
\end{center}

\textbf{Figure 2. Vocabulary question after each limerick.}

One of our expectations was that emotional scaffolding would lead to increased student attentiveness, which would lead to increased retention of the vocabulary information presented in the limericks and accompanying vocabulary explanations. Another expectation was that human-provided emotional scaffolding would increase students' persistence. Normally, the length of time that a student uses the Reading Tutor is strongly affected by the length of the story, the length of the class period, and other factors essentially out of our control.

In order to provide for an explicit measure of memory of the word that it had just explained, the Reading Tutor asked a vocabulary question after each limerick (Figure 2.) There were no questions during the wordlist task. In order to provide for an explicit measure of persistence, we inserted choice points into the wordlist and limerick tasks. After every ten words (for the wordlists) or after every limerick, including its corresponding definition and vocabulary question, (for the limericks), the Reading Tutor displayed as text and spoke aloud a prompt inviting the student to click Goodbye to stop, or Go to continue reading. It then alternately flashed the Goodbye and Go buttons six times.
We determined who would receive which condition when as follows. In each study, we matched students by classroom (and thus grade), and by gender. Each student read one of the versions on (their own) Day 1, either with or without emotional scaffolding. Approximately one week later, that student read the other version in the other condition. We counterbalanced treatment by version and order.

We measured the following dependent variables.

- **persistence**: the number of times the student clicked Go to continue reading instead of clicking Goodbye to quit. We would expect persistence to be important because increased persistence means more time on task – a key ingredient of success in reading.
- **memory**: the number of words a student got right on limericks quiz.

Figure 3 shows the experimental and control conditions: in the experimental condition, a human wizard observed the student on a television monitor in an adjacent room and provided emotional support over the student's headphones. In the control condition, the wizard helped the student log in and choose the correct story – but after that simply observed, and the Reading Tutor provided help as usual (minus its pre-existing emotional support, which had been suppressed for this experiment.)

The human wizard provided emotional support as follows.

1. Selecting phrases from a web page, to play pre-recorded audio.
2. Speaking live into a microphone to provide help that was (not yet) available in pre-recorded form.

As the experiment progressed, the wizard added phrases used in the previous days’ experiment to the web page – automating himself (partially) out of the loop.

To characterize the interaction between the human wizard, the student, and the Reading Tutor, we transcribe below an extended example of one student using the Reading Tutor without emotional support, and the same student using the Reading Tutor while receiving emotional scaffolding from the human wizard. We transcribed these sessions on a turn-by-turn basis from videotape, noted the begin and end times for the first set of words, and added a brief description of the student’s behavior when faced with the choice to continue or quit.
Student¹ without emotional support
October 31, 2001
Wordlist Form A

(Wizard directs student to choose “Word List Form A”)
RT displays text “Word List Form A”
W: “Can you read that?”
S: “Word”
RT: “Word”
S: “List Form A”
RT displays and speaks: Now you’ll see some words to read.
S clicks Go
RT displays and speaks: Please read each word aloud.
S: “Please read each word aloud.”
S clicks Go.
11:49:14 AM
RT displays: den
S: “den”
S clicks Go
RT displays: eel
RT says “starts like eels”
S: “ee- eel”
RT displays: fry
RT says: rhymes with wry
S: “fry?”
RT displays: gym
RT sounds out gym
S: “gym”
S (yet again): “gym”
RT displays: kin
RT: n here makes the sound /n/
(At this point the RT may have heard itself, and thus went on to the next word right away.)
RT displays: kit
RT says: “rhymes with lit”
S: “kit”
RT displays: pal
RT sounds out: “al”
S: “pale”
RT displays: ram
RT sounds out ram
S: “ram”
RT displays: spa
RT breaks spa down into onset and rime
S: “spa”
RT: displays vow
S: “vow”
RT displays and says: You can click Goodbye to log off, or Go to continue reading.” (Goodbye and Go alternately flash.)
11:50:48 AM S narrows eyes, scratches chin, then clicks Goodbye.

Same student with emotional support
October 17, 2001
Wordlist Form B

(Wizard directs student to choose “Word List Form B”).
RT displays text “Word List Form B”
RT says “Word List Form B”
RT says “Starts like World”
RT says “Word List Form B”
W: “Can you read that?”
W: “Can you read that out loud, Danny?”
RT: sounds out form
S (overlapping): “Word List From B” [sic]
W: “Good.”
RT displays and speaks: Now you’ll see some words to read.
W: “You can read this aloud, or click Go.”
S: “Now you’ll see some words to read.”
W: “Very good.”
S clicks Go.
RT displays and speaks: Please read each word aloud.
S: “Please read each word aloud.”
S clicks Go.
11:53:27 AM
RT displays: bib
S says: “bib”
S clicks Go
RT displays: fin
RT sounds out fin
S says: “fin”
S clicks Go
RT displays: fad
RT says: “Rhymes with fats”
S: “fad”
W: “You’re doing fine.”
RT displays gel
RT sounds out gel
S: “gel”
W: “Good.”
RT displays hop
S: “hop”
S clicks on Go.
RT says: “Sorry, can’t go on right now.”
RT displays: lug
S says “jug”
RT: mmm
RT says “lug”
S: “lug”
W: “Good.”
RT displays: maw
RT says “a w here makes the sound aw”
S: “w – maw”
W: “Good.”
RT says “maw. maw.”
S: “maw”
RT displays: pup
RT says “starts like pun”
RT: mmm
S: “pawp”
RT displays sip
RT breaks sip down into onset and rime
W: “Well done.”
RT displays: wed
RT sounds out wed

¹ Subject label: mDS.
5 Results

We present the results from the outcome variables persistence and memory. Out of 17 students who began the study, 14 students completed it, as shown in Table 2.

Table 2. Persistence and memory results from study comparing Reading Tutor alone to Reading Tutor plus human-supplied emotional scaffolding.

<table>
<thead>
<tr>
<th>Number</th>
<th>Gender</th>
<th>Persistence (Control)</th>
<th>Persistence (Experimental)</th>
<th>Correct (Control)</th>
<th>Correct (Experimental)</th>
<th>Word List / Limericks</th>
<th>Which story first?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>F</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Word List</td>
<td>Control</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>10</td>
<td>10</td>
<td>0.6</td>
<td>0.8</td>
<td>Limericks</td>
<td>Control</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>10</td>
<td>5</td>
<td>0.6</td>
<td>0.44</td>
<td>Limericks</td>
<td>Experimental</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>Word List</td>
<td>Control</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>10</td>
<td>10</td>
<td>0.4</td>
<td>0.6</td>
<td>Limericks</td>
<td>Experimental</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
<td>Word List</td>
<td>Control</td>
</tr>
<tr>
<td>7</td>
<td>F</td>
<td>2</td>
<td>2</td>
<td>0.4</td>
<td>0</td>
<td>Limericks</td>
<td>Control</td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td>4</td>
<td>15</td>
<td></td>
<td></td>
<td>Word List</td>
<td>Control</td>
</tr>
<tr>
<td>9</td>
<td>M</td>
<td>1</td>
<td>0.83</td>
<td></td>
<td></td>
<td>Limericks</td>
<td>Control</td>
</tr>
<tr>
<td>10</td>
<td>M</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>Word List</td>
<td>Control</td>
</tr>
<tr>
<td>11</td>
<td>M</td>
<td>8</td>
<td>0.83</td>
<td>0.5</td>
<td></td>
<td>Limericks</td>
<td>Experimental</td>
</tr>
<tr>
<td>12</td>
<td>M</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>Word List</td>
<td>Experimental</td>
</tr>
<tr>
<td>13</td>
<td>M</td>
<td>3</td>
<td>10</td>
<td>0.33</td>
<td>0.6</td>
<td>Limericks</td>
<td>Experimental</td>
</tr>
<tr>
<td>14</td>
<td>M</td>
<td>10</td>
<td></td>
<td></td>
<td>0.83</td>
<td>Limericks</td>
<td>Control</td>
</tr>
<tr>
<td>15</td>
<td>M</td>
<td>6</td>
<td>10</td>
<td>0.67</td>
<td>0.9</td>
<td>Limericks</td>
<td>Control</td>
</tr>
<tr>
<td>16</td>
<td>M</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>Word List</td>
<td>Control</td>
</tr>
<tr>
<td>17</td>
<td>M</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td>Word List</td>
<td>Experimental</td>
</tr>
</tbody>
</table>

Figure 4 shows persistence in the control condition vs. persistence in the experimental condition – each point represents one student except where noted. Persistence in the control condition averaged 6.4 ± s.d. 5.1 for wordlists and limericks combined; in the experimental condition, 8.6 ± s.d. 4.8 for wordlists and limericks combined, a significant difference (p = 0.041 by 2-tailed paired T test; F = 5.07, p = 0.044 by repeated measures analysis of variance with story kind (wordlist or limericks) as a between-subjects factor; F = 4.46, p = 0.061 by repeated measures analysis of variance with story kind and order as between-subjects factors) with an effect size of 0.44.

Boys persisted on average twice as long with emotional scaffolding as without; however, many of the girls were already at or near maximum persistence:

Boys: N=7, 6.1 ± 3.4 (expt.) vs. 2.7 ± 2.4 (control), p=.007 by paired T-test.
Girls: N=7, 11.0 ± 4.9 (expt.) vs. 10.0 ± 4.5 (control), p=.582 by paired T-test.

No significant differences were found on memory (Figure 5), which was measured only for the limericks.

One might suggest that it wasn't the emotional support that made the difference, but rather that the student knew a real human was present and aware of what the student did. However, in this experiment the wizard was in
fact present during the control condition, but served merely as an observer. The difference between conditions therefore cannot lie in presence alone, but rather somewhere in the realm of supplying emotional scaffolding.

Figure 4. Persistence in the control condition and in the experimental condition.

Figure 5. Memory in the control and experimental conditions (limericks only).

We had initially noted some signs that emotional support might lead to prolonged engagement. As the wizard said later:

At least a few subjects seemed to be responsive to emotional scaffolding. There were a few subjects in the Experimental Condition who seemed to remain engaged longer as a result of judiciously timed encouragement just as they were deciding whether to continue or quit. (Barry Kort, via intragroup email, November 9, 2001).
However, as the experiment progressed, we had reason to suspect that we were going to end up with student frustration swamping any other effects of affect. As the wizard later recalled:

The most frequent source of frustration was when the RT failed to accept a perfectly articulated sentence, phrase, or word. There were selected words that it consistently failed to recognize time and again. Some subjects repeated the fragments with increasing levels of annoyance. Some even cursed at the machine. The other annoyance was the long delays before it advanced to the next fragment. (Barry Kort, via intragroup email, November 9, 2001).

Thus it was gratifying to see that emotional scaffolding did in fact lead to significantly greater persistence (for boys). In fairness to the Reading Tutor, we should point out that mitigating factors include laptop sound card problems, and performance slowdowns caused by memory leaks and full disk. Furthermore, these problems – and their severity – don’t seem representative of the Reading Tutor’s normal performance in the field. We do see false alarms and frustration. However, they’re mitigated by technical factors (the desktop computers used for 84 of the 99 Reading Tutors in daily use at schools for the 2001-2002 school year don’t have many of the problems we found on the laptop used in this experiment, some of which have now been fixed), by context (the students get to take turns picking what to read – not just limericks or word lists), and by the Reading Tutor’s built-in emotional support (the unablated Reading Tutor praises students (Mostow & Aist CALICO 1999)).

One kind of result from this study was simply the types of emotional support produced – fodder for possible later automation. The wizard produced various kinds of support. For example, the wizard sometimes provided praise such as “Good” or “Very good”, or encouragement such as “You’re doing fine.” For example, after the system offered a choice to click Go to continue, or Goodbye to log off – when the student was deciding whether to continue reading or not, the wizard sometimes would supply judiciously timed encouragement such as “You’re doing fine.” This and other kinds of encouragement ought to provide a rich source of potential Reading Tutor interventions.

It was also interesting to note the interactions between normal Reading Tutor behavior and the interventions supplied by the wizard. For example, when words in the wordlist became excessively difficult, the Reading Tutor modeled the pronunciation so that the student didn't even have to read the word. In the word-list exercise, the RT typically spoke every word upon display after the first few sets of words. Many subjects stopped looking at the screen and just repeated what they heard. The Reading Tutor normally provides assistance both preemptively and on demand – but in this case this assistance changed the task sufficiently to make it substantially easier. For another example, there were cases where “Sometimes, I would cue up a praise response, only to have the RT negate it by requiring the subject to repeat the phrase.” (Barry Kort, personal communication, November 9, 2001). In these cases, the Reading Tutor and the wizard were giving mixed messages – presumably leading to a less engaging experience than one where the praise and the task structure matched. Such clashes between the Reading Tutor and the human tutor help reveal the limitations of our current implementation of this “Reading Tutor plus” methodology. Further tweaking of the Reading Tutor installation for these tasks – wordlists and limericks – and a tighter integration of the human-generated responses into the dialog management of the Reading Tutor, ought in principle to alleviate such problems.

6 Future Work

We noted that students experienced frustration at times with the Reading Tutor. From the research viewpoint, this frustration represents both threat and opportunity. It’s a threat to the study to the extent that it dominates affect and swamps the effects we’re trying to study. It’s an opportunity to the extent that generating frustration lets us study emotional scaffolding methods for overcoming it – or reveal areas on which to work towards the goal of modifying the system to be less frustrating.

As a reviewer noted, “Although the human wizard is pretty constrained in when and how he can interact with students, he still has significant freedom in what he chooses to say and when he chooses to intervene, so it isn’t clear that the authors will be able to automate the types of things the wizard is doing.” Further experiments
might be directed at finding out what an emotionally supportive automated tutor should say – in the face of limited sensing, or when enabled with richer senses including vision processing.

We found logistics to be a challenge, due to physical separation between where the researchers were based and where we had a partnership with an elementary school. Later experiments might modify the methodology – perhaps separating the wizard from the student not just by a television connection to the next room, but by a Webcam so that the student and the wizard could be anywhere, or at least anywhere with a fast Internet connection.

Future work towards specifying exactly what emotional scaffolding is and how and when it helps could proceed by transcribing and analysing the videotaped interactions between the Reading Tutor, the student, and the human wizard for content and effectiveness. As one reviewer pointed out, “Clearly the biggest area for future work is understanding the human’s utterances and when they are used so that they can be automated.”

Future work towards improving the Reading Tutor in terms of the affective quality of children’s experience might take several directions. If we can understand when kids get confused, we may be able to modify the design to reduce confusion or add suitable prompts. Improving speech recognition accuracy would help a lot, and we’re working on it, but it won’t happen overnight. If kids get distracted because they have to wait too long for the Reading Tutor to prepare for its next action, or if it takes too long to detect when they reach the end of the sentence, we may need to run on a faster platform, or modify the code. If the limericks in this experiment bore kids enough to be a problem, we may need more interesting materials to use when conducting later experiments – but it may be difficult to give them the nicely controlled experimental properties that the limericks provide.

Future work towards a model of emotions and learning that works for children’s reading would need to take into account the intrinsic reward structure of reading. Learning to read may simply not involve lots of insights. The one big aha! is grasping the alphabetic principle – “cracking the code” in the sense that there exists a code. The rest may simply be lots of practice to acquire lots of specifics – particular letter-sound mappings, and meanings of thousands of specific words. That is perhaps a very different set of intrinsic rewards than a set of problem-solving tasks, each task with its own “aha!”.

7 Conclusion

We summarize this paper’s contributions as follows.

First, we presented a novel finding on the role of emotions in learning: Human-provided emotional scaffolding in the context of an intelligent tutoring system for oral reading leads to increased student persistence. This paper presents the first statistically reliable empirical evidence from a controlled study for the effect of human-supplied emotional scaffolding on student persistence in an intelligent tutoring system: Children’s persistence was higher with emotional scaffolding than without it. As a reviewer of an earlier version of this paper noted, “affective interaction is an essential feature of a much work in animated pedagogical agents. Results from such work have already demonstrated that interventions by animated agents capable of affective intervention yield positive results in terms of persistence and subjective experience. For example, Lester et al. document the persona effect and its role in promoting learning. This study differs in that it attempts to focus specifically on the effects of emotional scaffolding.” Thus we submit for the research community’s consideration that intelligent tutoring systems ought to incorporate emotional awareness (sensing and responding) into their behavior.

Second, we presented examples of emotional scaffolding. We presented some types of specific interventions used by an expert to provide emotional scaffolding shown to increase persistence.

Finally, we presented a case study in studying ITSs by adding human expert capabilities to existing tutoring systems. This experiment tested the effects of adding a human expert capability (in this case emotional scaffolding) to an existing intelligent tutoring system with minimal or no code changes. Perhaps this notion may prove useful not only for exploring emotional scaffolding, but other aspects of intelligent tutoring systems as well.
8 Acknowledgements

This work was supported in part by the National Science Foundation under Grant No. REC-9979894 (Project LISTEN), and under Grant No. REC-0087768 (The Affective Learning Companion). Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation or the official policies, either expressed or implied, of the sponsors or of the United States Government.

For Project LISTEN’s Web site, see http://www.cs.cmu.edu/~listen. For the Affective Learning Companion’s Web site, see http://www.media.mit.edu/affect/AC_research/lc/index.html. Early discussions (c. 1997) on supplementing the Reading Tutor with emotional awareness were had with John Kominek.

References

More on pilot-testing a tutorial dialog system 
that supports self-explanation 
Good feedback on explanations helps; how we can generate more of it? 

Vincent Aleven, Octav Popescu, and Kenneth R. Koedinger 
Human-Computer Interaction Institute 
Carnegie Mellon University 
aleven@cs.cmu.edu, octav@cmu.edu, koedinger@cs.cmu.edu 

Abstract Self-explanation is an effective metacognitive strategy. How can an interactive learning environment 
best help students to state explanations in their own words? In developing the Geometry Explanation Tutor, we 
have made the assumption that self-explanation can best be supported by means of a restricted form of dialog in 
which the system provides detailed feedback. We tested this assumption using data from a pilot study of limited 
duration in which the tutor was used in a school. This study yielded a corpus of 185 dialogs involving 791 
attempts to explain a geometry problem-solving step. A human rater assessed the local appropriateness of the 
system’s feedback following each explanation attempt. We semi-automatically determined the degree of 
progress made by students following each feedback message. We found that higher-quality feedback tends to be 
followed by greater progress on the part of students and less regression. This result lends some support to the 
assumption that detailed feedback is useful. If this is so, then how can the system generate better feedback? 
Low-rated feedback was due to (1) lack of coverage in the system’s NLU component and Explanation Hierarchy 
and (2) lack of ways of dealing with errors of commission by the student. These shortcomings can be and to 
some extent have been addressed without adding major new components to the system’s architecture. 

1. Introduction 
Self-explanation is an effective metacognitive strategy (Chi, 2000), but not all students are equally good at 
producing effective self-explanations (Renkl, 1997). Students often do not produce good self-explanations, even 
when prompted to explain, (Renkl, et al., 1998). Therefore, an important research question is: how can 
interactive learning environments help students learn through self-explanation? 

Aleven and Koedinger (2002) showed that students learn geometry with deeper understanding when 
the system requires that they explain their problem-solving steps “by reference”, that is, by selecting from a 
glossary the name of a geometry principle that justifies each step. Conati and VanLehn (2000) investigated the 
use of a Bayesian student model to point students to the steps in worked-out physics examples that they might 
benefit from explaining. They found that this method is effective for a subset of students. In both systems, 
students explain by means of menus and templates and receive feedback on their explanations. 

It is plausible that students learn with even greater understanding when they can explain their steps in 
their own words, as opposed to using a structured computer interface. For example, explaining in one’s own 
words may have a generation effect. However, the results with systems in which students explain in their own 
words (by means of the keyboard) without receiving feedback on their explanations are mixed at best. 
Hausmann and Chi (in press) found that a simple prompting system that does not provide feedback can help 
increase the number of self-explanations generated by students, as compared to the number of self-explanations 
typed spontaneously into a text editor, but they found also that students produce fewer self explanations when 
they type than when they explain orally. Similarly, Aleven and Koedinger found that without feedback students 
provide few explanations and even fewer good explanations (Aleven & Koedinger, 2000). Renkl (2002) showed 
that students studying probability theory examples learn better when their self-explanations are supplemented 
with “instructional explanations” that the system provides on demand, even when students do not receive 
feedback. However, he did not compare these conditions to ones in which students explain orally or in which 
they receive feedback. Thus, in interactive learning environments, feedback seems needed in order to provide 
students with an incentive to explain. In this sense, computer-based learning environments seem to differ from 
human one-on-one tutoring, where feedback from the tutor is not crucial (Chi, et al., 2001). 

But what kinds of feedback will be most helpful? How effective is generic feedback that says only 
whether an explanation is adequate? Is more detailed feedback better? In developing the Geometry Explanation 
Tutor, a system designed to help students state general explanations of geometry problem-solving steps in their 
own words, we have taken the viewpoint that generic feedback is not likely to be sufficient with the given 
subject matter and student population (high-school students) and that feedback based on a detailed analysis of
students’ explanations is more effective. The system’s architecture reflects this viewpoint. In this paper, we examine this assumption using data collected during a small pilot study carried out in a school in the Pittsburgh area. We also discuss, based on our analysis of these data, how the system can be improved.

2. The Geometry Explanation Tutor

The Geometry Explanation Tutor provides guidance during problem-solving practice. In addition, it requires that students provide general explanations of their problem-solving steps in their own words (Aleven, et al., in press; 2001). It helps students, through a restricted form of dialogue, to improve their explanations and arrive at explanations that are mathematically precise and that identify the critical features of the problem. So far, the system’s dialogues deal with the topics covered in one of the six units that make up the tutor curriculum, the Angles unit, which deals with the geometric properties of angles. The Geometry Explanation Tutor was built on top of an existing Cognitive Tutor (Anderson, et al., 1995) for geometry problem solving, the Geometry Cognitive Tutor™. This tutor was developed by our research group in tandem with a full-year high school geometry curriculum. The combination of these two components has been shown to be better than traditional geometry classroom instruction (Koedinger, et al., 2000). The tutor and the curriculum are being marketed commercially and are in use in about 100 schools in the United States (see http://www.carnegielearning.com).

The system’s architecture and knowledge sources have been designed with two key assumptions in mind: First, as mentioned, the system should provide feedback based on a detailed analysis of the student’s explanation. Second, the system must be able to deal primarily with errors of omission, explanations that focus on the right geometry rule but omit critical information. Our analysis of a corpus of student explanations indicated that these types of errors were more frequent than errors of commission.

Each explanation attempt is processed in three steps: First, the system’s natural language understanding (NLU) component builds a semantic representation. This representation is then classified with respect to a fine-grained set of explanation categories, represented in the system’s Explanation Hierarchy. Finally, the system decides how to respond, based on the classification of the explanation and its determination of what geometry theorem(s) or definition(s) the student should be explaining.

The system’s NLU component is primarily knowledge-based, in contrast to many other tutorial dialogue systems, which rely on statistical approaches (Graesser, et al., 2001), key phrase spotting (Evans et al., 2001), or bypass language understanding altogether (Heffernan & Koedinger, in press). The NLU component uses a left-corner chart parser with a unification-based grammar formalism to parse the input (Rosé & Lavie, 1999). The Loom description logic system (MacGregor, 1991) is used to represent the semantic content of the student’s input. The system’s Explanation Hierarchy is also represented in Loom. It contains 149 categories which represent common ways in which students express or attempt to express geometry rules in their own words. The hierarchy was developed based on our analysis of a corpus of student explanations. For each relevant geometry rule, the Explanation Hierarchy contains one or more categories representing correct and complete ways of stating the rule. For example, category COMPLEMENTARY-ANGLES-SUM-90 represents all correct and complete statements of the definition of complementary angles, including “the sum of the measures of complementary angles is 90 degrees.” For each relevant geometry rule, the Explanation Hierarchy also contains numerous categories that represent commonly occurring incomplete ways of stating the rule. For example, category COMPLEMENTARY-ANGLES-90 represents sentences meaning “complementary angles are 90 degrees,” which falls just short of being a complete and correct explanation. The Explanation Hierarchy contains a small number of categories representing incorrect ways of stating rules, such as a category for statements like “complementary angles sum to 180.” The Explanation Hierarchy is instrumental in producing detailed feedback to student explanations. Each explanation category in the Explanation Hierarchy has an associated sequence of (canned) feedback messages, which are used by the system as detailed below.

Once the student’s explanation has been classified, the system decides how to respond according to its dialogue management strategy shown in Table 1. The algorithm is implemented by means of production rules. Classifying an explanation generally results in a set of explanation categories, rather than a single explanation category. Some of these categories may represent errors of omission, some may represent errors of commission. By and large, the system deals with errors of omission before it deals with errors of commission. This policy was adopted because we assumed that when a student explanation is classified under multiple explanation categories, these categories are likely to stem from different (non-overlapping) parts of the explanation, for example, different clauses of a multi-clause sentence. It seemed most productive in such situations to give feedback on the part with the error of omission, since that part at least focuses on the right geometry rule.

An error of omission occurs when the student’s explanation is classified under a category that is more general (i.e., subsumes) than a category representing complete statements of the geometry rule being explained. When such an error occurs, the system first determines which of the categories under which the explanation...
Table 1: Dialogue management strategy for providing feedback on a student’s explanation, based on (a) the list of the geometry rule(s) that can justify the current step and (b) the set of explanation categories under which the explanation was classified is closest (in terms of distance in the Explanation Hierarchy) to the correct category. It then displays the first of the sequence of feedback messages associated with the selected category (lines 3, 4, 6, 7 in Table 1). However, when an explanation attempt is classified under the same set of categories as the previous attempt, the system instead selects the next message in the sequence selected previously (line 5). This way, it is able to provide more specific advice when the student does not succeed in improving an explanation. (During the pilot study reported in this paper, a simpler approach was used, as detailed below.) The system’s feedback messages initially give away as little as possible, while still providing some useful information. The first message in a feedback sequence usually does no more than hint that what the student said is an overgeneralization. Subsequent messages provide more specific information, as is illustrated in the following dialogue fragment, in which three consecutive explanation attempts all classify under the same explanation category, namely, ANGLES-90:

Student: angles are 90
Tutor: You are going in the right direction. But you probably did not mean to say that all angles are equal to 90 degrees?
Student: angles are 90 degrees
Tutor: Exactly what kinds of angles are 90 degrees?
Student: these angles are 90 degrees
Tutor: For starters, you need to make it clear that your rule is about complementary angles.

Errors of commission occur when the student focuses on the wrong geometry rule or introduces errors in the statement of a rule. When this happens, the explanation is classified under a category that does not subsume (in the Explanation Hierarchy) a category representing complete statements of a correct rule, but subsumes only categories representing incorrect rules. The system has two ways of dealing with errors of commission. First, it is able to produce specific feedback in response to a small number of common errors of commission, such as focusing on the supplementary angles rule when explaining steps that involve the linear angles rule. During the pilot study, this particular error occurred frequently. The tutor responded, quite appropriately, as follows:

Student: supplementary angles are 180
Tutor: You are trying to explain what it means for angles to be supplementary. And doing a good job. But you need to explain WHY they are supplementary.

The system produces this kind of feedback messages in much the same way that it deals with incomplete explanations that focus on a correct rule (see lines 3-7 in Table 1), with one difference: the system has the ability to select a message associated with the combination of the explanation category and the rule to be
explained (in those few cases where a message for such a combination has been provided by the system developer). When an explanation classifies under category like for example SUPPLEMENTARY-ANGLES-180, the system is able to present a different message depending on whether the rule to be explained is linear pair or supplementary angles. The second way in which the system responds to errors of commission is by presenting a generic feedback message (“You are focusing on the wrong rule”), as can be seen in step 14 and 15 of Table 1.

3. A pilot study in a school
At the end of the 2000-2001 school year, we conducted a pilot study in order to get a sense of how well the system was working. During this study, the Geometry Explanation Tutor was used briefly in a suburban junior high school in the Pittsburgh area, as part of a 9th-grade Integrated Mathematics II course. This course covered a number of topics in geometry and algebra. Approximately 30 students of ages 14 and 15 participated in the pilot study. The students were “honors students,” which means that within their school and grade level they were among the best in terms of academic ability and diligence. During two 40-minute class periods, the students worked in pairs on the Geometry Explanation Tutor, running the tutor on the school’s wireless PC laptops. Earlier during the semester, they had learned about angles and their interrelationships (the topics covered in the tutor’s Angles unit, on which we focused in the study) but they did not have any computer tutoring related to these topics. The geometry instruction earlier during the semester was based on a different curriculum than that of which the Geometry Explanation Tutor is part. The study constituted the first time that the tutor was used in a school. The study was somewhat limited in scope and duration: the students did not work through the complete Angles unit or encounter all theorems that are addressed in this unit. Further, the students who used the tutor probably were of somewhat higher ability and were somewhat better prepared than the average student in the target population.

4. Data analysis
The logs of the student-tutor interactions contained information about 185 dialogues comprising 791 attempts to explain a geometry theorem or definition. This amounts to 12.3 ± 4.6 dialogues per pair of students. Students arrived at a complete explanation in 75% of the 185 dialogues. In a companion paper (Aleven, et al., in press) we report some results from the analysis of these data. Summarized briefly:

• There was evidence that students learned to explain the various geometry theorems and definitions involved in the Angles unit, as they worked with the Geometry Explanation Tutor: the number of attempts needed to arrive at a complete and correct explanation of a rule was lower on later opportunities (2.5 attempts) to explain that rule than it was on first opportunities (4.7 attempts).

• The tutor’s NLU component classifies student explanations in a reasonably accurate manner: Three human raters classified the student explanations collected in the pilot study. We then computed the inter-rater agreement for each pair of human raters and also the inter-rater agreement between the system and each of the human raters. The NLU component agreed with human raters somewhat less often than human raters agreed among themselves (the averages of the kappa coefficients were .65 and .81, respectively), but the gap was not too big.

In the current analysis, we are interested first in testing the hypothesis that detailed feedback on explanations is useful and helps students to improve their explanations. We look whether there is a relation between (1) the quality of the feedback provided by the system—at this stage of the development of the system, not all feedback is (equally) good (2) the amount of student progress immediately following a feedback message. Second, in order to produce information that will be useful during the further development of the system, we look in detail at points in the dialogue where the system’s feedback seems to leave room for improvement. The current analysis of feedback quality is complementary to the analysis of the classification performance of the system’s NLU component (Aleven, et al., in press). In that analysis, we asked whether the system could classify student input accurately with respect to its current set of explanation categories. The current analysis focuses on the question whether the right feedback can be generated with this set of explanation categories. Thus, inherent in the current analysis are the questions of whether we have identified all useful explanation categories and of whether the system’s strategy for deciding what feedback to give (shown in Table 1) is appropriate.

4.1. Local quality of system feedback
In order to assess the quality of the system’s feedback, the first author graded the system’s responses to the 791 student explanation attempts on a 5-point scale—the better the feedback, the higher the grade. The grades were based on the grader’s overall sense of how appropriate a given feedback message was, taking into consideration
only the explanation attempt to which the feedback was given but no dialog history. In order to make sure that the grader could not look at the larger dialog context in which a response was given, the set of explanation-feedback pairs was scrambled into a random order before the grader went to work. This analysis therefore focuses on the *local* appropriateness and clarity of the feedback. When feedback received a low rating, this could be due to imperfections in any or all of the processing stages of the system.

In advance, we did not formulate specific criteria for each grade level. After the fact, we can discern the following trends: Five points were assigned when the system response seemed to be right on target or when the system correctly recognized that an explanation was complete and correct (of the system responses that received grade 5, about half received the grade for this reason). Four points were given when the feedback was good, but could be improved in one way or another. Three points were allowed for feedback that was OK but either did not focus very clearly on the explanation’s shortcomings or could be improved in more than one way. Two points were given when it was hard to see how the feedback could be helpful or when the feedback was confusing or awkward, such as feedback saying that the student should explain why angles are supplementary when the students were actually attempting to explain just that. Finally, one point was given when the feedback made no sense or when it pointed the student in the wrong direction, for example by saying that the student was focusing on the wrong rule where the student was actually focusing on the correct rule. Also, 1 point was given when the tutor missed a correct and complete explanation. Items where the NLU component could not produce an analysis (see step 17 in Table 1). Such responses are to be avoided—by hand—by the first author. The procedure used for hand-labeling the 791 explanations is described in more detail in Aleven, et al. (in press).

Examples of some of these categories are shown in Aleven, et al. (in press). Using these definitions, we computed the degree of progress made in each explanation attempt, based on the explanation categories that were assigned “by hand” by the first author. The procedure used for hand-labeling the 791 explanations is described in more detail in Aleven, et al. (in press).

---

1 Under this definition, adding something redundant to an explanation is considered to be lateral movement, as seems appropriate. However, adding something to an explanation that is wrong in the current context is also considered to be lateral movement. Instead, it should probably be regarded as regression. Thus, the Lateral Movement category is somewhat heterogeneous. However, it is not clear how to define a better criterion based on the explanation categories.
Table 2: Degree of progress made as a function of the rating given to the feedback immediately preceding the explanation attempt. The percentages represent the frequency of the given progress category, given a particular feedback rating (i.e., they are the percentage of the row total).

<table>
<thead>
<tr>
<th>Feedback Rating</th>
<th>Progress (N=157)</th>
<th>Lateral Move (N=157)</th>
<th>Same Classification (N=157)</th>
<th>Regression (N=157)</th>
<th>Total (N=367)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>26% (N=20)</td>
<td>17% (N=13)</td>
<td>39% (N=30)</td>
<td>17% (N=13)</td>
<td>100% (N=76)</td>
</tr>
<tr>
<td>2</td>
<td>30% (N=31)</td>
<td>20% (N=20)</td>
<td>29% (N=30)</td>
<td>21% (N=21)</td>
<td>100% (N=102)</td>
</tr>
<tr>
<td>3</td>
<td>39% (N=20)</td>
<td>12% (N=6)</td>
<td>31% (N=16)</td>
<td>18% (N=9)</td>
<td>100% (N=51)</td>
</tr>
<tr>
<td>4</td>
<td>59% (N=26)</td>
<td>20% (N=9)</td>
<td>11% (N=5)</td>
<td>9% (N=4)</td>
<td>100% (N=44)</td>
</tr>
<tr>
<td>5</td>
<td>64% (N=60)</td>
<td>14% (N=13)</td>
<td>15% (N=14)</td>
<td>7% (N=7)</td>
<td>100% (N=94)</td>
</tr>
<tr>
<td>Total</td>
<td>43% (N=157)</td>
<td>17% (N=61)</td>
<td>26% (N=95)</td>
<td>15% (N=54)</td>
<td>100% (N=367)</td>
</tr>
</tbody>
</table>

In analyzing the degree of progress made in students’ attempts at explaining geometry rules, we exclude a number of items that are not relevant. First, we obviously need not be concerned with progress following the last explanation attempt in each dialogue, so we omit the last explanation attempt in each of the 185 dialogues. Further, we disregard the SameInput category, that is, explanation attempts that were exactly the same as the previous attempt. This kind of repetition occurred rather frequently (218 out of 791 explanation attempts). We suspect however that these repetitions were due mostly to imperfections in the tutor’s user interface that led students to submit explanations to the tutor inadvertently. (The user interface has been improved meanwhile.) Since the current analysis focuses on dialogue management, not the user interface, it seems better to disregard these attempts as well. However, below we report results of statistical tests both with and without these items. Third, we disregard the items for which the feedback was not rated, as detailed above. The results reported here pertain to the remaining 367 explanation attempts.

As shown in Table 2 (bottom row), students made progress on 43% of the explanation attempts. They regressed on 15% of the steps, and some amount of stagnation occurred on the remaining 43% of steps (Lateral Movement or Same Classification). The progress rate following good feedback (i.e., feedback that received grade 5) is about what we would want it to be in a tutoring context, namely, 64%. This number might seem a bit low, given that we were dealing with good and well-prepared students. However, a much higher progress rate would probably indicate that the students are getting too much help, which may be detrimental to learning. We see also that students were sometimes able to make progress even after bad feedback (i.e., feedback that received grade 1), but not very often (26%). Interestingly, the probability of Progress increased as the feedback rating increased. Further, the frequency of regression and input classified under the same set of categories as the previous attempt decreased as the feedback rating increased. Lateral movement seemed to occur at a fairly constant rate, regardless of the feedback rating, perhaps reflecting the fact that lateral movement is a fairly heterogeneous category. A Chi-square test revealed a significant association between the feedback rating and the progress category, regardless of whether we exclude explanations that were the same as the previous (χ²(12)=55.0, p<.0001) or include them (χ²(16)=62.8, p<.0001).

The observed relation between local feedback quality and progress rate provides some support for one of the main assumptions underlying the design of the Geometry Explanation Tutor, namely, that the content of the feedback matters and that the function of feedback is more than just to provide an incentive for students not to leave the explanation boxes in the tutor’s user interface blank: Detailed feedback seems to help students to improve explanations. On the other hand, we cannot entirely rule out an alternative interpretation of these data, namely, that the inadequate explanations upon which it is difficult for students to improve tend the be the ones for which it is also difficult for the tutor to produce good feedback. In other words, it is still possible that the probability of improvement depends less on the feedback quality and more on the complexity of the explanation. At this point, we do not know how one might distinguish between the two interpretations.

6. Analysis of system feedback

If good feedback seems to lead to student progress, how can the system produce more of it? To answer this question, we look in more detail at the feedback items that seem to offer most room for improvement.

6.1. When good feedback does not lead to progress …

We first looked at items where the feedback received a high rating (4 or 5) and yet no progress was made by the student on the subsequent explanation attempt. In advance, these items seemed interesting because they might
tell us that we had been making wrong assumptions and that the feedback message was not as good or as clear as we thought. By and large, however, this analysis did not reveal significant flaws in the system’s methods or knowledge base, although it did help us identify some feedback messages that could be phrased better.

When good feedback does not lead to progress, the system should notice the lack of progress and provide more specific feedback. The system version that was used in the pilot study was not capable of doing so. Since then, the system has been augmented so that it does provide more specific feedback when an explanation attempt is classified under the same set of categories as the previous attempt on the same step (see Table 1, steps 3-7). The current data suggest that it will be useful to extend this mechanism so that it handles all instances where a student does not make progress, including regression and lateral movement.

6.2. When feedback receives a low rating …

We also looked at items that received a low local feedback rating, in order to identify the most frequent causes of such low ratings. We wanted this analysis as much as possible to reflect the current capabilities of the system, as opposed to those at the time of the pilot study, since a number of improvements have been made since then. Therefore, instead of looking at the feedback messages in the data set collected during the pilot study, we had the system re-generate the feedback for each of the student inputs, using its current method, shown in Table 1. This method is slightly different from the method used during the pilot study, in the following way: When an explanation is classified under multiple categories, the system used during the pilot study randomly selected a category on which to base feedback. The current system picks the category that is closest to the correct category (see step 7 in Table 1). Second, in the current analysis, we wanted to focus on dialogue management, not NLU, since we had already used the data from this study to evaluate the NLU component (Aleven, et al., in press). In other words, we sought to reduce the influence of NLU errors on the analysis. Therefore, the re-generated feedback was based on the classification categories that had been “hand-assigned” by a human rater, rather than those generated by the system.

The same grader graded the re-generated set of feedback messages in the same way as before. As it turned out, for 115 explanation attempts the re-generated feedback was the same as the original feedback, offering an opportunity to compute the intra-rater reliability of the ratings. The intra-rater reliability score for the 115 items was 60% ($\kappa = .44$) when we count as agreement only those instances where the exact same rating was assigned twice. The intra-rater reliability score is 93% ($\kappa = .85$) when we consider as being in agreement a pair of ratings that differ by no more than one point. Thus, the grader did not reliably assign the same grade to the same item, but was very reliable in assigning a grade that is off by no more than 1.

The re-generated feedback messages received an average rating of 3.93 ± 1.33, as shown in Table 3. We inspected the 153 items that received a low feedback rating (either 1 or 2) in order to identify ways in which the system might be improved.

6.2.1. Lack of coverage

A significant portion of the low quality feedback turned out to be due to lack of coverage, sometimes in the Explanation Hierarchy only, sometimes in the NLU unit more broadly. These problems can be (and to some degree have been) addressed by adding knowledge to the system. Here we briefly discuss some examples of these coverage problems: First, students often referred to linear angles as “angles that form a line” or “angles on a line”. These statements represent exactly the kind of free-form expression that we want to encourage with the Geometry Explanation Tutor. Much headway has been made in augmenting the tutor’s NLU component so that it can deal with expressions referring to linear angles as angles that form a line. Second, the students frequently referred to the “whole and parts postulate.” This phrase was not recognized by the tutor, as the given postulate is not part of the geometry curriculum of which the tutor is part. This is an example where testing a system in a slightly different context than the one for which it is designed leads to suboptimal performance. Third, some parts of the Explanation Hierarchy were sparsely populated, such as those dealing with angle addition and angle bisection. Therefore, the tutor was not in a good position to provide detailed feedback on incomplete

---

2 This random selection strategy seemed a good idea at the time but it did not work well. It was based on the assumption that feedback based on each of those categories would be equally helpful and that it would be good to switch between these categories as the given set of classification categories would repeat during a given dialogue.

3 In fact, the re-generated feedback does not fully reflect the current system’s capabilities. The method used to generate feedback was slightly different from the current method, shown in Table 1. It did not produce increasingly specific feedback messages when an explanation attempt was classified under the same category as the previous attempt. In other words, the method used omitted lines 4, 6, and 7. This is appropriate when focusing on the local quality of feedback.

4 The Cognitive Tutor Tutor Geometry™ curriculum includes a number of theorems that together cover the same ground as the whole and parts postulate.
Table 3: Rating of feedback following various types of student explanations. The type of an explanation was determined based on the explanation categories assigned to that explanation by a human grader. Errors of omission and errors of commission were defined as described in section 2.

<table>
<thead>
<tr>
<th>Student Explanation Type</th>
<th>Nr. Of Items</th>
<th>Perc. of Items</th>
<th>Feedback Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct</td>
<td>238</td>
<td>30%</td>
<td>4.84</td>
</tr>
<tr>
<td>Omission</td>
<td>344</td>
<td>43%</td>
<td>3.66</td>
</tr>
<tr>
<td>Error of Commission</td>
<td>43</td>
<td>5%</td>
<td>3.16</td>
</tr>
<tr>
<td>Omission &amp; Error of Commission</td>
<td>78</td>
<td>10%</td>
<td>2.81</td>
</tr>
<tr>
<td>Irrelevant</td>
<td>88</td>
<td>11%</td>
<td>3.45</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>791</strong></td>
<td><strong>100%</strong></td>
<td><strong>3.93</strong></td>
</tr>
</tbody>
</table>

explanations of these theorems. Finally, the system does not have a category for explanations that merely state in words or symbols the arithmetic involved in finding the numeric answer. Therefore, it is not able to respond by saying that the students should focus on explaining why those arithmetic steps were the right ones to take.

6.2.2. Errors of commission

A second cause of low-rated feedback was the system’s limited ability to handle errors of commission. As mentioned, we had designed the system’s dialogue management strategy so that it deals with errors of omission first, regardless of whether there are errors of commission also5. However, the main assumption underlying this policy, namely, that when an explanation classifies under multiple categories, these categories relate to different parts of the explanation, was not always met. Some of the low rated feedback was due to the fact that “errors of omission first” strategy is not appropriate when there are multiple explanation categories related to the same part of the explanation. In particular, the system was not very successful, when an explanation focuses on the right rule but contains both errors of omission and commission, as happened in about 10% of explanation attempts. As Table 3 indicates, the feedback following such explanations tended to receive a fairly low rating. For example, on a step that involved the isosceles triangle theorem, we saw the following exchange:

**Student:** two corresponding angles of an isosceles triangles are congruent

**Tutor:** Correct, except for one thing: is any pair of angles in an isosceles triangle congruent?

The student’s explanation contains an error of omission (only the base angles of an isosceles triangle are congruent) as well as an error of commission (the use of the term corresponding angles is inappropriate in this context). The tutor feedback fails to point out the error of commission and is confusing. (It received grade 2.) A better response would have been to focus on the error of commission and to make it clear to the student that it is inappropriate to mention corresponding angles in this context. The policy of focusing on errors of omission first is to blame. The tutor's NLU component had (correctly) classified this explanation under two explanation categories, (1) CONGRUENT-ANGLES-OF-ISOSCELES-TRIANGLE, which represents an incomplete statement of a correct rule, or an error of omission and (2) CONGRUENT-CORRESPONDING-ANGLES, which is not related to a correct rule (i.e., represents an an error of commission). The example strongly suggests that the tutor’s policy should be changed so that errors of commissions are handled first. This alternative policy would help also with other examples, such as the errors of commission are italicized):

- “if two angles are supplementary they equal 90 degrees” (in a step dealing with complementary angles)
- “perpendicular bisectors equal 90 degrees” (in a step dealing with adjacent complementary angles but not angle bisection)
- “consecutive angles add to a larger angle” (in a step dealing with angle addition, use of the term consecutive where adjacent is needed)
- “complementary angles linear pair equals 90 degrees angle addition” (student takes a blanket approach in step dealing with complementary angles, as a result of consecutive unsuccessful attempts at improving the explanation; it would probably be best to tell the student simply to focus on the complementary angles and to get rid of the rest)

The same change in policy would be helpful in dealing with explanations that contain redundant terms but are not strictly wrong, such as in the following examples (the redundant words are shown in italics):

---

5 The exception are correct and complete statements of a wrong geometry rule (lines 8 and 9 in Table 1), but in practice they occur very infrequently. During the pilot study, only 14 out of 791 explanation attempts were of this type.
• “two adjacent complementary angles equal 90 degrees”
• “isosceles triangles have two consecutive congruent angles”
• “adjacent linear pair sum to 180 degrees”

It will not be difficult to extend the system to handle these errors of commission (step 3 in Table 1). The existing mechanism may need to be extended. There are a number of open questions, such as how the messages should be phrased, whether the content of these messages should depend on what the correct rule is, to what categories these messages should be attached, exactly when they should be given, and whether it is helpful to take into account whether the explanation categories relate to different parts of the explanation.

Some low quality feedback following errors of commission was simply due to lack of coverage in the Explanation Hierarchy. On one particular problem, students often tried to explain their answer in terms of the similar triangles theorem. The correct explanation however involved the isosceles triangle theorem. (The problem did in fact involve a pair of congruent triangles, but this fact was irrelevant.) The tutor was not able to recognize this error of commission, since it has no representation of the similar triangles rule. This rule is dealt with in a different curriculum unit, which normally comes after the Angles unit. However, the students in the pilot study had learned about the similar triangles theorem already before working on the Angles unit, illustrating again how slightly out-of-context use of a dialogue system may lead to unanticipated problems.

7. Discussion and Conclusion

We report on an analysis of 185 dialogues between students and the Geometry Explanation Tutor. A human rater assessed the quality of the system’s responses to student explanations in these dialogs. (In order to reduce the influence of NLU error, we had the system re-generate the feedback messages, based on the classification of student input done by a human.) About 20% of the feedback messages received a low grade. Rating the quality of the system’s feedback was very helpful in focusing the search for ways of improving the system. The most frequent cause for low-rated feedback was lack of coverage in the system’s NLU component and/or Explanation Hierarchy. The second most frequent cause was the system’s limited ability to handle students’ errors of commission.

These shortcomings can be addressed mostly within the system’s current dialogue management framework. In order to address the lack of coverage, more knowledge needs to be added. We have already made significant headway in this regard. Further, in order to deal better with errors of commission the system’s tutorial strategy shown in Table 1 must be changed, so that errors of commission are handled first, before errors of omission. This is a good strategy from a pedagogical viewpoint, as errors of commission are likely to be detrimental to learning. Further, it is likely that the system’s feedback messages in response to errors of omission will work better when errors of commission have already been addressed. Many of the system’s feedback messages have been written based on the assumption that there are no errors of commission. It looks like these problems can be addressed without going too far outside the current dialogue management framework. Finally, the system’s multi-level feedback mechanism needs to be extended so that it recognizes all situations where students do not make progress, not just the ones where the classification of the student’s explanation is the same as it was in the previous attempt. That is, the system should also respond appropriately to lateral movement and regression by the student. We foresee that the proposed changes will considerably improve the overall quality of the dialogue.

The data from the pilot study provide evidence for a relation between the local appropriateness of the system’s feedback messages and the degree of progress made by the student immediately following the feedback message: Better feedback tends to be followed by greater progress and less regression on the next attempt. This is an interesting result that lends some support to a fundamental assumption underlying the design of the Geometry Explanation Tutor, namely, that tutorial dialogue with detailed feedback is a good way of supporting self-explanation. However, we cannot entirely rule out an alternative assumption underpinning the result, namely, that the explanations for which the system does not provide good feedback tend to be the ones that are most difficult to improve for students. In order to determine more conclusively that detailed feedback is helpful, it seems necessary to do a controlled experiment comparing a system version that provides detailed feedback (e.g., the current version) against one that provides generic feedback only.

The current study has a number of limitations. First, the quality of the system feedback was assessed by the first author who also happens to be the author of the system’s feedback messages. We do not know if an experienced tutor would have assigned the same grades. Second, the study focused on the local appropriateness of feedback. When grading the system feedback, the grader took into account only the explanation to which the message was a response, no previous dialog history. It is possible that the grades would have been different if the grader had looked at a larger context. This remains as an interesting question for future work. A third shortcoming of the current study is that we did not validate whether the computational notion of progress we
used corresponds to more intuitive notions. None of these limitations seems terribly important for our current purpose (formative evaluation), but should be addressed when an analysis of this type would be done as part of a more summative evaluation.

It is an interesting open question how the local progress measure used in the current study, which is essentially a performance measure, relates to learning. It seems reasonable to assume that little learning could occur if students were not making progress on their explanations—conversely, a high progress rate would seem to be beneficial for learning, at least if it was not achieved solely by means of overly helpful feedback. This is plausible especially given that self-explanation has been shown to be related to learning in a number of cognitive science studies (Chi, 2000). Therefore, if we assume that the current system does not err on the side of being too helpful—a reasonable assumption in light of the fact that the system’s feedback messages were designed to provide no more help than needed—the result presented here can be seen as good news. Nonetheless, the real question remains how well students learn when they explain their steps in their own words, as compared to simpler forms of supporting self-explanation, such as that described in Aleven and Koedinger (2002). We are currently involved in a larger study that addresses just that question.

8. References


Student Initiative and Questioning Strategies in Computer-Mediated Human Tutoring Dialogues

Pamela W. Jordan and Stephanie Siler
Learning Research and Development Center
University of Pittsburgh
Pittsburgh PA 15260
[pjordan,siler]@pitt.edu

Abstract

In this paper we explore student initiative in human tutoring dialogues and how it potentially influences tutor questioning strategies. We describe our annotation scheme for tutor questions and student responses in which student initiative is simply defined as any unsolicited response from the student. Although we were successful at reliably annotating student initiative, we examine what is categorized as student initiative and propose some further distinctions that may be useful for improving mixed-initiative interaction in tutoring systems.

Keywords: annotation, human dialogue, initiative, dialogue strategies

1 Introduction

In human tutoring dialogues, a tutor who ignores signs of student confusion in order to complete his own plan runs the risk of preventing learning (Chi, 1996). We also know that active learning, which can involve a student making inferences, elaborating, justifying, integrating, predicting, etc., can be beneficial for students (Chi et al., 2001). Ideally then one would expect that maximizing the amount of confusion addressed and maximizing active learning behaviors in students to be worthy tutorial goals. Although human tutors may find it difficult to refrain from taking on the active learning role themselves (e.g. giving long explanations) and may at times fail to recognize or choose to ignore signs of confusion or unprompted active learning on the part of their students, they are undoubtedly better at recognizing it and responding to it than our current intelligent tutoring systems are. Our hope is that we can find enough instances of this sort of behavior in human tutorial dialogues that we can learn to mimic the successes in our intelligent tutoring systems.

For the purpose of this paper we will call these unexpected student behaviors that we wish to recognize and to which we wish to react and to encourage, student initiative. These behaviors are initiatives because they have introduced or started something new that was not part of the tutor’s current plan; they are an interruption to the forward progress of the tutor’s current plan relative to his last plan step and how he expects the student to contribute to this plan. When a tutor reacts to a student initiative, it has the effect of changing the direction of the tutoring session and helps to customize it to meet the needs of the individual student.

For example, if the tutor is asking a series of focused, short-answer questions, then the tutor is not planning to elicit an elaboration or some other form of active learning from the student and is expecting to continue on with his line of reasoning with some slight adjustments given the kinds of errors he expects to see. So if the student decides to elaborate after answering one of these questions, it was the student’s own initiative to do so and the tutor can either ignore this unexpected additional response and continue forward with his plan or he can react to the elaboration by either terminating or interrupting his current plan and addressing the student’s unexpected contribution. If the unexpected student

\footnote{This work was funded by NSF grant 9720359 and ONR grant N00014-00-1-0600}
contribution shows promise, we might expect the tutor to switch away from a focused line of questioning to a more open-ended one in which he tries to encourage further active learning behavior. However, if the contribution reveals some misconceptions or errors the tutor may want to pursue this further by eliciting more active learning behavior (e.g. Why do you think this is true?) or initiating a different focused line of questioning.

In human-computer interactions, our goal is to move away from the one-sided interactions that are typical of our current software applications toward mixed-initiative interactions that better reflect the type of collaborative interactions we are accustomed to when interacting with each other. With one-sided initiative, either the human initiates every action or the computer does. When the computer is in control of the interaction, typically it is not looking for the human to initiate anything without invitation and when the human is in control, he rarely sees the computer initiate an unwarranted action since it is so poor at recognizing when it is helpful and appropriate to interrupt the software user. With mixed-initiative interactions, either participant can initiate actions without invitation so a software agent needs to be able to recognize when it will be helpful to interrupt and when to allow itself to be interrupted.

There are many proposals and definitions as to what qualifies as initiative because what is of greatest importance varies by the type of interaction involved as well as what we are able to computationally model (see (Chu-Carroll and Brown, 1998) for a survey of these proposals). The work in (Whittaker and Stenton, 1988; Walker and Whittaker, 1990) is the basis of one often used definition of dialogue initiative. It is simple to implement and has been successful at modelling some aspects of advisory and directive dialogues. But the emphasis of this work was to model topic shifts and to recognize topic boundaries. It looked at when shifts can occur and how the shifts are negotiated using high-level, easily recognizable speech acts. Utterances are classified into 4 types: assertions, commands, questions and prompts and correspondences between topic boundaries and these speech acts were analyzed to arrive at rules for predicting when a topic shift occurs. The rules indicate that a speaker has control when his last utterance is 1) an assertion unless it is a response to a question, 2) a command or 3) a question unless it is in response to a question or command. With a prompt or an explicit repetition of what has already been stated so far, the speaker is signalling that he is relinquishing control. But if the speaker has not relinquished control, the hearer can initiate taking over control by providing new, unsolicited information regardless of whether or not he first responds to the speaker’s previous contribution. These rules are actually better for predicting topic initiative than topic control since it overlooks the fact that topic initiatives can fail (i.e. the hearer can reject the topic).

But as pointed out by further examinations of collaborative dialogues (e.g. (Jordan and Di Eugenio, 1997; Chu-Carroll and Brown, 1998)), topic initiative alone is not enough to adequately inform mixed-initiative interactions and misses a sizeable number of initiatives (e.g. 27% according to (Chu-Carroll and Brown, 1998) for one type of advisory dialogue). To help correct these shortcomings, (Chu-Carroll and Brown, 1998) suggests that initiative be defined as actions that direct how the agents’ task should be accomplished and actions that initiate the establishment of mutual beliefs. But we claim that this particular definition of initiative, while more comprehensive, needs some adjustment to be useful for modelling mixed-initiative in tutorial dialogue. We propose to give more emphasis to the initiation of active learning behaviors. Explanations and justifications are understandably excluded as initiatives by the above definitions (i.e. they support the establishment of mutual beliefs but do not initiate them) but as we pointed out earlier they are examples of important learning actions on the part of the student.

So as not to exclude any type of initiative a priori, we chose to call any unsolicited action an initiative in our initial study of tutorial dialogue. Although we are able to reliably annotate student initiative using this definition, we will need to refine it in order to be able to automatically recognize it within a tutoring system. But before we do so, we first need to understand what kinds of initiative are going to be most useful in tutorial dialogue by learning about which kinds of initiative are most influential in dialogue strategies.

In this paper, we will describe how we characterized unsolicited student actions and our annotation results. We will begin to explore this data by estimating which of these initiatives are topic initiatives and which are not. Our estimate shows that 78% of all student initiatives are non-topic initiatives. Finally we will make some proposals for how to better understand the non-topic initiatives and how to identify strategies useful for mixed-initiative interactions. While topic initiative within tutorial dialogue
has been explored (Shah et al.; Graesser and Person, 1994), other types of initiative have not yet been
as fully appreciated.

2 Corpus Overview

Our corpus is a collection of computer-mediated human tutoring dialogues in which a tutor presents a
student with a qualitative physics problem from a set of 30 such problems. We currently have collected
199 dialogues in which 5 tutors and 35 students interact. We have analyzed 15 of these dialogues for
student initiative by annotating whether the student is providing new unsolicited information. These
15 dialogues represent interactions between 1 tutor and 4 students working on a set of 11 problems.

For this particular corpus we did not measure student learning gains but it is still worthwhile to study it
under the assumption that any tutoring is expected to be effective. We are currently collecting a similar
corpus for 10 training problems where we are measuring learning gains. For both corpora, all the
students have recently taken a high-school or college-level introductory physics course and experienced
physics professors are providing a majority of the tutoring. Although some of the tutors in our first
corpus are physics graduate students, our analysis in this paper is of an experienced physics professor.

For each tutoring session the tutor and student are in separate rooms. All of the dialogues about
qualitative physics are conducted via the computer interface. The student and the tutor both explicitly
give up their dialogue turn by pressing a submit button and during a turn no interruptions are allowed.
While restricted turn-taking alters the nature of the dialogues (Oviatt and Cohen, 1991; Whittaker,
1995; O’Conaill et al., 1993; Jordan, 2000), it is still a valid (Clark, 1996) and successful form of
human communication (Jordan, 2000). Restricted turn-taking has the advantages of simplifying both
the dialogue analysis and the implementation of a computer tutor by eliminating overlapping language
and the need to determine the intended sequencing.

Both the student and the tutor’s computer interfaces comprise an area in which the physics question
is displayed, an essay entry window and a dialogue window. The tutor also has the ability to select
which question to present to the student and can enable and disable the student’s essay and dialogue
windows in order to encourage the student to separate the task of essay writing from that of engaging
in a dialogue.

After the tutor presents a qualitative physics question, the student enters his answer and explanation in
the essay window, as in (1), and then the student and tutor engage in a dialogue to correct and improve
that response, as in (2)\(^1\). The essay becomes part of the student and tutor’s common ground for the
dialogue so that the dialogue and essay are not independent. The tutor’s highest level dialogue topics
clearly depend upon the essay but once a topic is introduced the dependence upon the essay lessens.

(1) Q: When a car without headrests on the seats is struck from behind, the passengers often suffer neck
injuries. Why?
R: Because of the force of the car from behind hitting them, their heads are knocked back. This is
because, their heads want to travel at the same speed that their car is moving and when the car hits
them from behind the force causes this motion to be stopped.

(2) T0: What is the direction of the force experienced by the car when it is hit from behind? [CL]
S0: the direction is towards the car that hit it. [NI]
T1: Do mean that the car which is hit from behind will experience a force directed backwards? [CL]
S1: i guess it would be experience a force forwards [I]
T2: What will this force cause the car to do? [CL]
S2: accelerate [NI]
T3: Now analyze what will happen to the head, the neck if the head rest was not there? [OE]
S3: ok if the headrest was not there, when the car from behind hit the car it would give it acceleration,
the persons body would move at the speed of the car during the acceleration, but the head would
not. [NI]
T4: Why should the neck get hurt? [OE]

\(^1\)The notations at the end of each turn will be used and explained in later sections.
3 Annotating Student Initiative

We define any utterance that provides new unsolicited information as a show of student initiative. Since our definition of student initiative depends upon the tutor’s expectations, we first annotated tutor questions as to whether the expectation is for a small, finite set of subject relevant responses with no larger discourse relation other than a response to a question being involved, or whether there is an expectation for an informational relation (such as justification) in which the content that is to be provided by the student cannot be characterized as a small finite set. We called the first category of tutor questions closed questions and the second open-ended questions.

Student initiative then was annotated based on whether the student did more than simply respond in the expected way relative to the category assigned to the tutor question. As part of our annotation guidelines, we decided that we should never look ahead in the dialogue when determining the categories for tutor questions. We did not want the annotation of the tutor question to be influenced by how the student actually responded since it was to represent what we thought the tutor expected from the student. In the dialogue shown in (2), we illustrate our annotation categories using the following notation at the end of each turn:

[NI]: student response with no show of initiative
[I]: student response with initiative
[OE]: open-ended question from the tutor
[CL]: closed question from the tutor

To assess the intercoder reliability of our categorizations, we used the Kappa coefficient of agreement (Krippendorff, 1980; Carletta, 1996). The Kappa values for agreement on annotating tutor questions and student responses is shown in the first row of Table 1. A Kappa value is constrained to the interval [0,1] where K=0 means the agreement is no different than chance and K=1 means there is perfect agreement. Using Krippendorff’s scale to assess the Kappa values, where K < .67 is discounted, K > .8 is conclusive and anything in between is tentative, we had good agreement on student responses but not on tutor questions.

Since the categorization of student responses is dependent upon the category of the tutor question, one may wonder how the intercoder reliability for student responses can be good while that for tutor questions is not. Consider that a student can fail to respond in the expected way to either category of question. If one annotator characterizes a question as open-ended, the student response is almost always going to be classified as a non-initiative. However, the student may still fail to respond fully to this question with, for example, the justification the annotator may have thought the tutor was expecting and may just give an answer with no justification. If another annotator classifies the same question as a closed question, the student response will still be annotated as a non-initiative since the student did not give an unsolicited justification along with the response. This sort of annotation disagreement seems reasonable to expect in this case since the student may also have missed that the tutor was implicitly requesting a justification. The second row of Table 1 shows the Kappa values when we remove the cases in which we claim the tutor’s intent is potentially ambiguous to the student. But for this ambiguity argument to become convincing, clearly we need to subcategorize the non-initiative cases and indicate whether the perceived tutor expectation was met or not.

Although we created a taxonomy of question types to characterize open-ended and closed questions as part of our annotation guidelines, we did not annotate the question subcategories and check the
intercoder reliability for them. However, it is still informative to look at these subcategories since it better defines how we chose to classify tutor questions.

For closed questions the subcategories are:

- Tutor asks student for body which force acts on (limited by problem statement)
- Tutor asks student for force (limited number of forces, in part because limited number of bodies that can cause forces in problem statement)
- dichotomous questions (e.g. yes/no, either/or)
- relational questions (e.g. is the body1’s velocity greater than or equal to body2’s)
- directional questions (e.g. in which direction is this acceleration)

For open-ended questions the subcategories are:

- why or how types of questions (e.g. “ok, but why do you say so?”)
- definitional questions (e.g. “what is force?”)
- "what happens when..?” questions (e.g. “what will happen to the head and neck if the head rest was not there?”)
- Tutor requests terminology (e.g. “what will you say in scientific terms about the motion of the block?”)
- Tutor asks student for complex calculation or reasoning (e.g. “if the rock falls through this distance, what will be its final velocity?”)

From this subcategorization, one can see that our closed questions are similar to the scaffolding questions and our open-ended questions to the prompting questions described in (Chi et al., 2001).

In developing these subcategories for our annotation guidelines, we did not resort to classifying tutor questions simply on the basis of syntactic form. For example, we could not declare that all syntactic yes/no questions were closed questions. This is because many of the syntactic yes/no questions were implicit why questions in the context of the dialogue as in (3).

(3) Are you suggesting that acceleration, height of fall and initial velocity can be different for the two balls and yet they will take the same time to hit the ground?

4 Proposals Regarding Student Initiative and Closed vs. Open-ended Questions

Looking again at the question subcategories in the previous section, in the case of the closed question subcategories, we see that the tutor is requesting that the student either make a low effort inference or clarify and disambiguate a statement that the student made earlier whereas with the open-ended questions the tutor is requesting that the student engage in what (Chi et al., 2001) calls deeper forms of construction that lead to active learning. For example, why or how questions request that the student make inferences and integrate materials by asking for justifications, “what happens when..” questions request that the student make predictions, and asking for the value of a quantity requests that the student do some further problem solving to arrive at a value. Furthermore, our question categories bear some resemblance to the student question categories in (Graesser and Person, 1994) and it may be fruitful to use these in helping us better characterize the purpose of the student initiatives. So we propose that we should try to categorize unsolicited student contributions by which question subcategory they could hypothetically be responding to in addition to whether student turns adequately meet the tutor’s expectations. We expect that this categorization would help us to better distinguish which kinds of initiative will be most valuable to recognize in tutorial dialogue.
<table>
<thead>
<tr>
<th>15 dialogue sessions - 1 annotator</th>
<th>Closed</th>
<th>Open-Ended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response only</td>
<td>103</td>
<td>104</td>
</tr>
<tr>
<td>Response + explanation or expansion</td>
<td>29</td>
<td>6</td>
</tr>
<tr>
<td>Response + question</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Change previous response</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Totals</td>
<td>138</td>
<td>114</td>
</tr>
</tbody>
</table>

Table 2: Preliminary Subcategorization of Student Responses Relative to Tutor Question Types

So why should we be as interested in recognizing active learning initiative as we have been in topic initiative (e.g. responding to student questions)? In the case where the student is showing active learning initiative, we need to recognize it and encourage the student to continue with it by giving feedback. If a tutoring system ignores it, it may discourage the student from taking on the role of an active learner when they are ready for it and the system simply mis-predicted when it will be most successful at prompting active learning behaviors from the student. Furthermore, in those cases where the tutor is prompting for active learning behaviors, we can examine when it is that the tutor is going to be most successful at getting the student to engage in active learning behavior.

We propose that closed and open-ended questioning are two high-level strategies that a tutoring system may want to choose between and that student initiative could be one important factor to consider in that decision process. The open-ended questioning prompts for active learning whereas the closed questioning gives more guidance and prepares the student for active learning behaviors. In the 15 dialogues that we’ve annotated for open-ended questions and closed questions, the tutor has an almost equal distribution of both types of questions across the dialogues (55% closed vs. 45% open-ended) so it seems that these may indeed be two important strategies that are involved successful tutoring.

We’ve begun a subcategorization of the student responses relative to tutor question types but have not yet checked the intercoder reliability for the response subcategories. Table 2, shows the distributions of this subcategorization relative to closed and open-ended questions according to one annotator. From this we can estimate topic initiative by grouping together student turns that are responses followed immediately by questions and student turns that are changes to previous responses. We will declare all other initiatives to be non-topic or possibly active learning initiatives (e.g. responses followed by unsolicited explanations or expansions). We see overall that there are more non-topic than topic initiatives (78% vs. 22%). It is also the case that there is much more student initiative with the closed questions than with the open-ended questions (25% vs. 9% \(\chi^2 < .001\)) and that 82% of the initiatives following closed questions are non-topic initiatives. This further suggests that non-topic initiatives may be a signal for the tutor to discontinue a closed questioning strategy.

What we haven’t yet begun to address in our analysis is how the tutor reacted to the student initiatives and the interactions between the closed and open-ended questioning strategies. Some potential hypotheses of interest for us to explore in human tutorial dialogue are:

Hypothesis-1: the less disruptive an initiative is the more likely it is that the tutor responds to it.

Hypothesis-2: when the current closed or open questioning strategy fails switch to the other.

Hypothesis-3: when the student demonstrates active learning behavior switch to or continue a strategy that encourages it.

References


Initiative in Tutorial Dialogue*

Mark G. Core and Johanna D. Moore and Claus Zinn

Division of Informatics,
University of Edinburgh, 2 Buccleuch Place
Edinburgh EH8 9LW, UK
[markc|jmoore|zinn@cogsci.ed.ac.uk

Abstract

One-on-one human tutoring has been shown to be a very effective method of instruction. Many researchers have argued that good human tutors encourage knowledge construction by getting students to do as much of the work as possible and to maintain a feeling of control, while providing students with enough guidance to keep them from becoming too frustrated or confused. So-called “Socratic tutoring” has been proposed as the best way to accomplish these tasks. To implement Socratic tutoring in a dialogue-based ITS we must understand it in terms of the features that characterize dialogue systems such as: dialogue strategies, initiative management, dialogue acts, and turn taking. In this work we investigate initiative in tutorial dialogue by comparing tutoring sessions with a Socratic style to sessions with a didactic tutoring style. Results show that the Socratic dialogues were more interactive than the didactic dialogues; on average students spoke more in the Socratic condition, and tutors asked more questions and made fewer statements. Students learned more in the Socratic condition but due to small student numbers this can only be regarded as a trend. Surprisingly, students took more initiative in the didactic dialogues. Unlike we expected, student initiative was not the key element that makes Socratic tutoring effective.

keywords: initiative annotation, initiative analysis, tutorial dialogue

1 Introduction

Studies show that one-on-one human tutoring is more effective than other modes of instruction. The average student who received one-on-one tutoring with an expert tutor scored 2 standard deviations above the average student who received standard classroom instruction (Bloom, 1984). Current intelligent tutorial systems relying on graphical user interfaces reliably produce effect sizes of 1 standard deviation above students only receiving traditional instruction (Anderson, Corbett, Koedinger, & Pelletier, 1995).

What is it about human tutoring that better facilitates learning? Many researchers argue that it is the collaborative dialogue between student and tutor that promotes the learning (Merrill, Reiser, & Landes, 1992a; Fox, 1993; Graesser, Person, & Magliano, 1995). Through collaborative dialogue, tutors can intervene to ensure that errors are detected and repaired and that students can work around impasses (Merrill, Reiser, Ranney, & Trafton, 1992b). Previous research has also shown that students must be allowed to construct knowledge themselves to learn most effectively (Chi, Bassok, Lewis, Reimann, & Glaser, 1989; Chi, de Leeuw, Chiu, & Lavancher, 1994; VanLehn, Siler, Murray, & Baggett, 1998). The consensus from these studies is that experienced human tutors maintain a delicate balance allowing students to do as much of the work as possible and to

*The research presented in this paper is supported by Grant #N00014-91-J-1694 from the Office of Naval Research, Cognitive and Neural Sciences Division. Thanks to Jean Carletta and our reviewers for their comments on this work.
maintain a feeling of control, while providing students with enough guidance to keep them from becoming too frustrated or confused.

So-called "Socratic tutoring" has been proposed as the best way to accomplish these tasks. But what are the features that characterize Socratic tutoring dialogues? To implement Socratic tutoring in a dialogue-based ITS we must understand it in terms of the features that characterize dialogue systems such as: dialogue strategies, initiative management, dialogue acts, and turn-taking. In this paper, we explore the role of initiative in Socratic dialogue and study the relationship between initiative and learning.

Our experimental paradigm consists of collecting dialogues between students and a Socratic tutor (a tutor who asks questions instead of giving explanations) and contrasting them with dialogues between a student and a didactic tutor (a tutor who gives explanations followed by questions to test student understanding). Our hypotheses were that the Socratic sessions would be more successful and the reason for their success would be that students were encouraged to take more initiative.

2 Previous Work

2.1 Defining Initiative

Sinclair and Coulthard (1975) developed a dialogue grammar for classroom discussions. Their minimal unit of dialogue is the exchange which is composed of an initiating move, an optional responding move, and an optional feedback move. Whoever makes the initiating move is said to have initiative for the exchange. Moves consist of an uninterrupted sequence of words by a speaker. Moves have a structure of their own but we will discuss only the "head" element (primary function) of moves. The head elements of initiating moves can be questions, directives, or informative statements; informative statements are distinguished from replies in that their primary purpose is to give information, not to respond. By primary purpose, we mean the function of the move as mutually agreed upon by the dialogue participants.

Sinclair and Coulthard describe cases of "re-initiation" where the student replies incorrectly and the original initiating question is reasked. When the tutor accepts the student answer then the exchange is over. However, re-initiation does not capture all the ways an exchange can be disrupted. Linell et al. (1988) discuss how a responder can ask for clarification, challenge the speaker, and change topics as well as respond directly to an initiating move. Linell et al. do not assign initiative directly to speakers but instead rank speaker moves based on how much "they can be regarded as governing or steering the ensuing dialogue and as being governed or commanded by the preceding dialogue" (p. 419). For example, an utterance which is not a response in any way but requires a response from the listener is ranked highest with a value of six. Minimal responses are at the other end of the scale (with a rank of two); they invite no response and give no more information than required.

Shah (1997) defines student initiative more directly as "any contribution by the student that attempts to change the course of the [tutoring] session" (p. 13). Shah's goal was to study cases where student behavior deviated from the tutor's expectations as encoded in the tutor's discourse plan as defined in (Freedman, 1996). In terms of annotating student initiative, Shah takes a practical approach tailored to her corpus of remediation dialogues. In these dialogues, tutors would quiz students about the answers they gave during problem solving (rather than having students self-explain or actually problem solve). In this corpus, student initiatives are student utterances that are not answers to questions. Shah assumes that these initiatives are dealt with exclusively by the next tutor speech act and that the tutor then takes back initiative. Thus, the tutor always has initiative except during student initiatives and the tutor's responses to those initiatives.

Chu-Carroll and Brown (1998) state that it is important to differentiate initiative (they call it dialogue initiative) from task initiative. They define dialogue initiative by stating that it "tracks
the lead in determining the current discourse focus” (p. 6)\(^1\) and that task initiative “tracks the lead in the development of the agents’ plan” (p. 6). Presumably, determining the discourse focus means something like setting the discourse segment purpose in Grosz and Sidner’s (1986) theory of discourse. What it means to take the lead in developing the agents’ plan depends on the plan representation but informally can refer to adding or taking away actions from the plan, rearranging actions, or setting parameters.

Although Chu-Carroll and Brown claim that dialogue and task initiative can be annotated directly from these definitions, it is worth noting that several research projects (Strayer & Heeman, 2001; Jordan & Di Eugenio, 1997; Doran, Aberdeen, Damianos, & Hirschman, 2001; Walker & Whittaker, 1990) have adopted the dialogue-initiative annotation guidelines proposed in (Whittaker & Stenton, 1988). Whittaker and Stenton refer to dialogue initiative as control of the dialogue and define rules for determining who has control (e.g., commands mean the speaker has control, questions mean the speaker has control unless the question follows a previous question).

2.2 Studies of initiative in human-human corpora

Work in computational linguistics (Walker & Whittaker, 1990) has successfully used initiative as a tool for dialogue analysis. Walker and Whittaker studied two dialogue genres, advisory dialogues (ADs) and task oriented dialogues (TODs). In the ADs they examined, novices were asking experts for help with either their finance or software problems. In the TODs, an expert was guiding a novice through the construction of a water pump over the telephone or via keyboard (chat). Walker and Whittaker labeled these dialogues for initiative based on the guidelines in (Whittaker & Stenton, 1988).

In the TODs, most of the time the expert has initiative; the ADs have closer to an equal sharing of initiative. In TODs, the expert had initiative 91% of the time while only having initiative 60% of the time in finance ADs and 51% of the time in software ADs. This distribution reflects the fact that in ADs, the user must provide the expert with details of the problem to be solved. The expert would let the novice describe the problem in their own words rather than continually prompting them for the information.

Another issue Walker and Whittaker investigated was how initiative was passed from one speaker to another. Of particular interest are abdications, initiative changes marked by prompts - utterances with no propositional content. They directly signal the listener to take initiative. Abdications were more prevalent in the TODs; novices would interrupt experts to report problems and then use abdications to signal the end of the interruption. The ADs were more collaborative and replied less on abdications.

Shah (1997) investigated initiative in CIRCSIM dialogues, typed human-human tutoring dialogues dealing with the circulatory system. Her corpus consisted of students’ initial tutoring session and a subsequent session with each of the same students. She labeled any student utterance that was not an answer to a question an initiative; these were then categorized based on communicative goal (e.g., challenge, support, repair, request information). Shah found that the initial sessions have twice the number of student initiatives as the set of subsequent sessions. The nature of the student initiatives changes as well. The proportion of student initiatives associated with confusion (long pauses and self repairs) decreases in subsequent sessions and the proportion of challenges increases. Shah also looked at tutor reactions to student initiatives; she found that tutors sometimes rejected student initiatives, but she did not investigate what triggered such actions.

(Graesser & Person, 1994) labeled student questions (a subset of the initiatives studied by Shah) in a corpus of tutoring sessions for a research methods course. Graesser and Person developed a taxonomy of different question types. Of specific interest are deep-reasoning and knowledge deficit questions. Deep-reasoning questions involve causal reasoning and hypothetical situations. Knowledge deficit questions are triggered when a student realizes an inconsistency or gap in his understanding or gets stuck on a problem. Graesser and Person found that in the first half of the

---

\(^1\)The page numbers come from the digital version: http://citeseer.nj.nec.com/244268.html
course there was a negative correlation between overall number of student questions and exam scores. In the second half of the course, there were positive correlations between exam scores and the proportion of student questions that were deep-reasoning questions and the proportion of student questions that were knowledge deficit questions.

Our study focused solely on initiative and did not address the difficult problem of categorizing question semantics. Initiative is a noisy measure of student participation. Shallow questions such as “What do I do next?” are treated the same as insightful questions such as “Is a load basically the opposite of a source?”. Despite this interference, we hoped that high levels of initiative would characterize students who took control of their learning and as a result scored well in the post experiment test.

3 Study of Socratic v. Didactic Tutoring

In our study we created a corpus of Socratic and didactic tutoring dialogues. The first hypothesis to be tested was that the experimental manipulation was successful - that Socratic dialogues were more interactive than the didactic dialogues. The second hypothesis was that the Socratic dialogues were more successful in promoting learning. The third hypothesis was that more student initiative is a critical part of what makes Socratic tutoring successful; although if the student takes too much initiative we would expect performance to drop off based on studies indicating the problems with unstructured exploratory learning (Pea & Kurland, 1984; Piroli & Anderson, 1985).

3.1 Method

The setting for this study is a course on basic electricity and electronics developed with the VIVIDS authoring tool (Munro, 1994). Students read textbook-style lessons and then perform labs using a circuit simulator with a graphical interface. (Rosé, Moore, VanLehn, & Allbritton, 2000) describes an experiment where students went through these lessons and labs with the guidance of a human tutor. Before the lessons students were given pretests to gauge their initial knowledge. After two tutoring sessions, students took the same tests again. We refer to the difference in their scores as learning gain. There were three sets of tutoring sessions: 1) the trial sessions where the tutor was not given any instructions on how to tutor, 2) the Socratic sessions where the tutor was instructed not to give explanations and instead ask questions, and 3) the didactic sessions where the tutor was encouraged to give explanations and then probe student understanding with questions. During these sessions, the student and tutor communicated through a chat interface. We will refer to the logs of this chat interface as the BEE dialogues. They are publicly available at http://www.cogsci.ed.ac.uk/~jmoore/tutoring/BEES_corpus.html.

In section 2.1, we reviewed various definitions of initiative. From these definitions, we chose Chu-Carroll and Brown’s (1998) definition of (dialogue) initiative. The alternative would be using Linell et al.’s definition of initiative (1988). It is not clear whether Chu-Carroll and Brown’s description of initiative as setting the discourse segment purpose is superior to Linell et al.’s dialogue move ranking. In the end, we chose Chu-Carroll and Brown’s definition because of the availability of easy-to-use annotation guidelines (Whittaker & Stenton, 1988). Note, we do not attempt to annotate task initiative here; in the discussion section we mention future work on annotating scaffolding and question and answer types. This work should allow us to measure task initiative as well.

In the rest of this section we give details of Whittaker and Stenton’s annotation guidelines. Whittaker and Stenton define a set of rules for assigning initiative to every turn (uninterrupted sequences of words by one speaker) in a dialogue. The turn must be classified into one of the following types based on its main purpose:

- **assertions** — declarative turns used to state facts.

---

2 By session, we mean all the dialogue between the tutor and one particular student.
• **commands** — turns intended to instigate action.

• **questions** — turns intended to elicit information.

• **prompts** — turns not expressing propositional content (e.g., “yeah”, “okay”).

We use the rules below to assign initiative. These are the same as the rules given by Whittaker and Stenton except that we make the assumption that a statement following a question responds to that question. See our corpus web page for more details.

```python
if turn = command then speaker has initiative
if turn = question then
    if last_turn = question or command then listener has initiative
    else speaker has initiative
if turn = statement then
    if last_turn = question then listener has initiative
    else speaker has initiative
if turn = prompt then listener has initiative
```

A benefit of this annotation scheme is that in our corpus the majority of turns can be automatically labeled: **questions** often ended in question marks; **commands** often started with verbs; a list of common **prompts** (“okay”, “yeah”) allowed most of these to be labeled, and **statement** could be used to label everything else.

We needed human annotators to correct the automatic labeling. One of the authors of the paper and another annotator (not a project member) corrected the utterance type annotations (the author only corrected 3 out of the 23 dialogues used in this study).

The annotators had a reference manual and trained on trial sessions of the dialogues. To test interannotator reliability, the author and external annotator labeled the same 757 utterances of non-training data; the resulting interannotator reliability as measured with the kappa statistic was 0.92. Generally, kappa values above 0.8 are considered acceptable and values between 0.8 and 0.67 marginal. To download the annotation manual or get more details of the methodology consult the web page, [http://www.cogsci.ed.ac.uk/~jmoore/tutoring/BEE_corpus.html](http://www.cogsci.ed.ac.uk/~jmoore/tutoring/BEE_corpus.html).

### 3.2 Analyses and Results

#### 3.2.1 Were the Socratic dialogues really Socratic?

Following the methodology of (Chi, Siler, Jeong, Yamauchi, & Hausmann, 2001), we verified that the experimental manipulation was successful - that the Socratic dialogues were more interactive. Chi et al. looked at the pure quantity of language from student and tutor as well as the number of moves such as self-explaining, giving feedback, asking a question, and scaffolding. We also started by looking purely at the quantity of language output by the tutor and student. Our test data consists of 23,451 words, 2853 utterances and 1547 turns of Socratic dialogue and 26,195 words, 2993 utterances, and 1378 turns of didactic dialogue.

We first focused on the number of words spoken, in particular, the percentage of words spoken by the student. The average per session for the didactic dialogues was 26%; the average for the Socratic dialogues was greater, 33% (t=2.26, df=18, p < 0.05). Another relevant statistic is the average number of words per tutor utterance. The average per session for the didactic dialogues was 9.42 words/utt and the average for the Socratic dialogues was less, 8.4 words/utt (t = 2.33, df=18, p < 0.05).

Following Chi et al., we also look at the average number of utterances per turn; if the Socratic dialogues were more interactive, then we should see fewer utterances per turn. More utterances per turn would suggest long explanations. Indeed, the Socratic dialogues had a lower average utts/turn, 2.38 than the didactic dialogues, 3.02 (t = 4.9, df=18, p < 0.05).

---

3These guidelines are based on comments by Krippendorff (1980) (as summarised in Carletta, 1996). Krippendorff considered the case of two annotated variables. He said that comparisons were reliable when the kappas for these variables were above 0.8. If the kappas were between 0.6 and 0.8, then tentative conclusions can be drawn.
<table>
<thead>
<tr>
<th>Didactic</th>
<th>Statement</th>
<th>Command</th>
<th>Prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td>29%</td>
<td>47%</td>
<td>9%</td>
<td>13%</td>
</tr>
<tr>
<td>Socratic</td>
<td>42%</td>
<td>34%</td>
<td>6%</td>
</tr>
</tbody>
</table>

Table 1: Breakdown of Tutor Moves

Although we do not have the detailed annotations of (Chi et al., 2001), we did annotate for questions, statements, commands, and prompts. In Table 1, we see that tutors did ask significantly more questions in the Socratic condition (t = 5.85, df=18, p < 0.001) and issue significantly fewer statements (t = 4.16, df=18, p < 0.001). The differences in commands and prompts are not significant (t = 1.99, df=18, NS and t=1.48, df=18, NS).

Table 2 shows a breakdown of student moves: student commands are not included because only two occurred in the whole corpus. The difference between student questions in the two conditions is not significant (t = 1.56, df=18, NS) nor is the difference between student prompts (t = 1.83, df=18, NS). There is a significant difference between student statements in the two conditions (t=2.27, df=18, p < .05) reflecting the earlier result that students spoke more in the Socratic condition. Perhaps with more data the differences in questions and prompts would be greater. Currently we cannot draw any conclusions from the breakdown of student moves.

Although this analysis is course grained, it will tend to blur the differences between Socratic and didactic dialogues rather than create a false distinction. The fine grained analysis used in (Chi et al., 2001) uses categories such as scaffolding and explanation. Scaffolding includes tutor statements that hint rather than explain. In our analysis all statements are regarded equally. Despite this fact, we do see a difference in the percentage of statements and questions made by the tutor in the two conditions. This evidence along with the results on quantity of language shows that the Socratic dialogues were more interactive.

### 3.2.2 Are Socratic dialogues more effective?

This analysis is reported on in (Rosé et al., 2000) and summarized below. In terms of raw learning gain data, the Socratic dialogues were more effective: the effect size ([mean Socratic learning gain - mean didactic learning gain]/standard deviation of didactic gain score) was 1 standard deviation in favor of Socratic tutoring. Small student size means this result can only be taken as a trend as verified using an ANCOVA with dialogue type as the independent variable, pre-test score as the covariate, and post-test score as the dependent variable F(1,18) = 3.13; p < .1.

(Chi et al., 2001) also investigated the differences between Socratic and didactic tutoring. They found that students performed equally well in the two conditions. However, a possible confound in favor of didactic tutoring was that tutors when in the didactic condition sometimes gave away post-test question answers in their explanations. Students tutored with the Socratic method had to do more work to score equally well on the post-test.

Thus, the evidence favoring Socratic tutoring is weak and work still needs to be done to verify its effectiveness. In the meantime, we can still analyze didactic and Socratic dialogues in terms of the features that characterize dialogue systems, in particular, initiative management. Eventually we will need to implement one of these two methods in our tutorial dialogue system and will need to know how to manage initiative.

### 3.2.3 Initiative analysis

Our first analysis was to measure the average percentage of turns for which students had initiative in the Socratic and didactic dialogues. Surprisingly, students had initiative for fewer turns on
Figure 1: Learning Gain Comparisons

average (10%) in the Socratic dialogues than in the didactic dialogues (21%). Informal analysis of the dialogues showed that students did not take advantage of the fact that Socratic dialogues were more interactive; they did not often take over the conversation. In the didactic dialogues, we noticed more student questions (initiatives) following explanations than in the Socratic dialogues. Perhaps the long explanations confused students.

We next tested the relationship between initiative and learning gain. Since initiative factored more heavily in didactic dialogues we hypothesized that the degree to which students took initiative might correlate negatively with learning gain. Since Socratic and didactic dialogues also differ in interactivity, we tested the relationship between learning gain and the interactivity measures of average percent of words and utterances spoken by the student and average percent of tutor utterances that were questions. Figure 1 shows this data: the left graph shows that initiative varies erratically as learning gain increases; there is no relationship (Pearson’s r=-.0689, n=23, NS) between these variables. The left graph also shows average percentage of words spoken by the student: it does have a relationship with learning gain (Pearson’s r = 0.6, n = 23, p < 0.005). The right graph shows the relationship between percentage of utterances spoken by the student and learning gain (Pearson’s r = 0.56, n = 23, p < 0.005), and the relationship between average percentage of tutor utterances that were questions and learning gain (Pearson’s r = 0.46, n = 23, p < 0.05). Since percentage of tutor utterances that were statements did not have a relationship with learning gain (Pearson’s r = -0.3143, n=23, NS), we did not plot these results. We focused on tutor questions and statements because in section 3.2.1 we found significant differences between the Socratic and didactic dialogues in these utterance types.

In section 2.2, we discussed the work of Walker and Whittaker on investigating initiative in the genres of advisory dialogues (ADs) and task oriented dialogues (TODs). Walker and Whittaker also investigated the difference between TODs in a spoken (telephone) modality and a typed (computer chat) modality. The results of their study are shown in columns 3-6 of Table 3 and the corresponding measures from our study are in columns 1 and 2. The Socratic dialogues have almost the same average expert initiative as TODs. In the TODs, the expert would issue a series of commands. In the Socratic dialogues the tutor was issuing a series of questions.

The second row of the table shows average percentage of initiative changes that were abdications. Abdications are the use of prompts to give away initiative; these often occur after interruptions to signal the original speaker to continue. Walker and Whittaker noted that TODs had more interruptions and thus more abdications in the spoken modality. However, the typed TODs had fewer abdications than either the spoken TODs or the ADs. Modality has an impact on how initiative is managed.

In the didactic and Socratic dialogues (both of which are typed) shown in columns 1 and 2,
<table>
<thead>
<tr>
<th></th>
<th>Didactic</th>
<th>Socratic</th>
<th>AD Finance</th>
<th>AD Software</th>
<th>TOD Phone</th>
<th>TOD Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert-Initiative</td>
<td>79%</td>
<td>90%</td>
<td>60%</td>
<td>51%</td>
<td>91%</td>
<td>91%</td>
</tr>
<tr>
<td>Abdication</td>
<td>2.32%</td>
<td>0.43%</td>
<td>38%</td>
<td>38%</td>
<td>45%</td>
<td>28%</td>
</tr>
</tbody>
</table>

Expert-Initiative - % of total turns with expert initiative
Abdication - % of initiative shifts that are abdications

Table 3: Initiative Measures for six Corpora

we see that abdications are rarely used. A number of reasons are possible. In the typed TODs, communication consisted of two simultaneously updated channels. In the tutoring dialogues, participants would send each other short messages. This modality, typed text and restricted turn taking might have reduced the number of abdications. Another possible factor is that students in this study were relatively passive; the tutor could not rely on them to take initiative if she uttered a prompt. The tutor’s initiative management also played a role. In our dialogues, after the student took initiative, the tutor would address the student’s turn and then often take back initiative not giving the student a chance to utter a prompt.

4 Discussion

Although we make some simplifying assumptions in annotating initiative, it is unlikely that a more complex annotation scheme will change these results. Our two major assumptions are: (1) questions following questions are assumed to be asking for clarification and not considered as having initiative, and (2) all statements following questions are assumed to answer these questions and are not considered as having initiative.

Although these assumptions are not always correct, defining a more precise annotation procedure is problematic. For some questions (e.g., “what time is it?”) annotators can agree what statements are answers and what statements give more information than requested. For other questions (e.g., “what causes current to flow?”) it is difficult to define a limited set of answers. Questions following questions are similarly difficult to deal with. Factors such as the content of the question, how many previous interruptions have occurred, and the response of the original speaker impact whether the question is perceived as taking initiative. Each factor is vague (e.g., does the second interruption automatically take initiative?) and the interactions between them not well defined.

A second point is the strength of the results. The statistics presented here verify that the average amount of student initiative was higher in the didactic condition. If we refine our definition of initiative, it must change the data sufficiently such that the hypothesis that the didactic dialogues have more initiative is no longer valid, and the change in data will need to be drastic enough to verify the hypothesis that the Socratic dialogues have more initiative. Changing a few examples where our annotation rules failed will not be sufficient.

In section 2.2 we described initiative as a noisy measure; all student questions are treated the same no matter how deep or shallow. Thus, it seems likely that we simply need a finer measure of active learning. Such a measure will also allow us to revisit the question - “Are the Socratic dialogues really Socratic?”. We showed that the Socratic dialogues were more interactive but if the tutor was asking mostly shallow questions then the dialogues should not be called Socratic.

The question taxonomy in (Graesser & Person, 1994) is one way to identify deep tutor and student questions. (Jordan & Siler, 2002) suggests going further and classifying student answers. Although a tutor may ask a shallow question the student may give more information than requested acting as if a deep question had been asked.

We also plan to study scaffolding based on the results in (Chi et al., 2001). Chi et al. performed a corpus collection similar to ours coming up with a set of didactic and Socratic dialogues. Learning gain results for the two groups were indistinguishable but Chi et al. found that tutors when in the didactic condition sometimes gave away post-test question answers in their explanations. Chi et al. cite a greater amount of scaffolding episodes and a greater amount of student reading as the reasons that the Socratic students were able to construct the answers that subjects in the didactic
sessions were simply told.

Although initiative did not turn out to be related directly to learning gain, we saw that on average students had initiative for 10% of the time in Socratic dialogues and 21% of the time in didactic dialogues. The question is whether successful tutorial dialogue is necessarily mixed initiative. Currently most tutorial dialogue systems either never take initiative or never allow the student to have initiative. Our hypothesis is that mixed-initiative systems would be more successful. A dialogue system evaluation in the database retrieval domain (Chu-Carroll & Nickerson, 2000) found that a mixed-initiative dialogue manager resulted in higher user satisfaction and better task efficiency than a system-initiative dialogue manager.

This study does not say anything about user-initiative systems nor about the dialogue genre of tutoring. Thus, we plan to run experiments with our machine tutor comparing both user and system initiative dialogue management to mixed-initiative dialogue management. If our hypothesis is correct and mixed-initiative tutors perform better, then further work must be done to determine when tutors should take and give away initiative and what types of linguistic signals are used when giving or taking initiative. The latter phenomenon can be complicated as participants may enter into a negotiation subdialogue to settle who will take initiative.

References


Adding Spoken Dialogue to a Text-Based Tutorial Dialogue System

Diane J. Litman
Learning Research and Development Center
and
Computer Science Department
University of Pittsburgh
Pittsburgh, PA 15260 USA
litman@cs.pitt.edu

Abstract
Spoken dialogue is a natural and highly desirable form of student-computer interaction, which provides both opportunities and challenges to the area of dialogue-based tutoring systems. This paper describes a new research project wedding research in spoken language technology and instructional technology, with the goal of promoting learning gains by enhancing communication richness.

Keywords: Spoken Dialogue, Prosody, Annotation, Evaluation

1 Introduction
As dialogue systems for tutorial applications become more common (Rose and Freedman, 2000), it will become crucial to increase their communicative effectiveness by taking advantage of advances in human language technology. For example, as evidenced by other research projects such as Project LISTEN (Mostow and Aist, 2001) and DC-TRAIN (Fry et al., 2001), the time is ripe for exploring whether spoken language interactions can and/or should replace the text-based interactions found in most dialogue-based intelligent tutoring systems. This paper describes a new research project which will wed research in spoken language technology with instructional technology, with the goal of promoting learning gains by enhancing communication richness.

Moving towards spoken dialogue systems for tutorial applications has many potential benefits. First, spoken language interaction is generally considered to be the most natural and easy to use form of natural language interaction. Second, a user’s speech contains prosodic information not present in text-based interaction, which could be used to monitor both the dialogue and pedagogical effectiveness of student-computer interactions. Third, going to a fully speech-based dialogue system is a natural complement to existing talking head tutors that already focus on spoken language output, e.g., (Graesser et al., 2001; Rickel and Johnson, 2000). The hands-free aspect of spoken language will also more easily support the addition of other interface modalities, for example pointing and clicking. Finally, because tutoring is a challenging and underexplored application area for the computational linguistics community, a spoken dialogue tutoring system is likely to not only advance the state of the art in tutoring, but also in spoken dialogue systems.

2 Research Questions
My new research project will entail building a spoken language interface to an existing text-based tutoring dialogue system, and conducting experiments with users in order to address the following set of research issues:

• What are the advantages – as well as disadvantages – of using a speech-based rather than text-based tutorial dialogue system?
• Can prosody be used to infer pedagogically significant information?
• Can the tutoring system make use of such inferences?
• Can existing robustness mechanisms deal with the noise from Automatic Speech Recognition?

3 Technical Approach

The research will enhance a text-based dialogue tutoring system now being developed at the University of Pittsburgh, by first providing and then exploiting spoken language capabilities. The research will be conducted in three phases (baseline system implementation and evaluation, data analysis, and enhanced system implementation and evaluation), and will need to address the workshop focus areas (evaluation, annotation, and creation/use of public resources). The research will build on the Why2 project, a collaboration between the University of Pittsburgh and the University of Memphis in the area of dialogue-based explanation tutoring (Graesser et al., 2002). In Why2, a student first provides a natural language answer to a qualitative physics problem. The tutor then engages the student in a natural language dialogue to provide feedback and correct misconceptions, and to elicit more complete explanations.

In Phase 1, a baseline spoken dialogue version of Why2 will be developed. This system will basically replace the text-based input and output modalities of the current version of Why2 with spoken language input and output. Because the use of automatic speech recognition will increase the noise at the input level, the system will also provide a new test for Why2's current robustness mechanisms. This spoken language baseline will then be empirically evaluated, providing a new experimental condition for examining dialogue and tutoring effectiveness. This will complement the existing evaluation efforts for Why2, which will compare the Pittsburgh text-based natural language dialogue system, a human tutoring condition, a text-based reading (i.e., non-dialogue) tutoring system, and the Memphis natural language tutoring system (which among other things includes a talking head). The evaluation will also generate a corpus of human-computer dialogues, which will be used to drive the next phases of research.

In Phase 2, a corpus analysis will be performed, which will be used to guide the development of an enhanced spoken dialogue version of Why2 in Phase 3. The enhanced version of the system will attempt to increase pedagogical effectiveness, by taking advantage of information that is only available in speech. We will manually annotate user utterances for potential pedagogical classifications of interest (e.g., annoyance, frustration, confusion, boredom, certainty, etc., as discussed below),¹ and automatically label prosodic and other features that can be computed at runtime. We will then use machine learning to predict the classes from the automatically obtainable features.

In phase 3 of the research, a new version of the system based on the corpus analysis will be implemented. This enhanced system will use the learned classifier to guide dynamic adaptation of the dialogue and pedagogical behaviors of the tutoring system. We will conduct empirical evaluations to compare the original and enhanced spoken language versions of Why2, with respect to student learning gains and dialogue system performance.

4 Previous Research and Implications for Tutorial Dialogue

My focus on discovering pedagogically-driven techniques for deciding when and how to intervene during intelligent tutoring will build heavily on my previous work in using prosodic and other automatically computable features to identify and recover from communication problems. Over the last few years, I have shown that features automatically computable at system run-time can be used to predict problems that can successfully benefit from some sort of system intervention. The hypothesis of my new work is that a similar approach will prove useful for predicting and adapting to problematic pedagogical situations.

¹Determining what can be annotated with high reliability will likely be a significant research effort in its own right. Until our human-computer spoken dialogue data becomes available, during Phase 1 we plan to conduct preliminary annotation experiments and refine our coding manuals using existing (text-based) Why2 dialogue data. We might also conduct a Wizard of Oz study to collect human-human spoken language data.
4.1 Non-Tutorial Dialogue

My previous work in non-tutorial dialogue focused on system detection of and adaptation to user problems at both the dialogue and utterance levels. At the dialogue level, our results were based on an examination of two different corpora, and used two different definitions of problematic dialogue situations. In a corpus of human-computer dialogues obtained from experiments with the TOOT train-timetable dialogue system, we used a machine learning methodology to identify problematic dialogue situations defined as dialogues where the percentage of correctly understand utterances was below some threshold. Using a set of 23 features representing five types of knowledge sources (acoustic, dialogue efficiency, dialogue quality, experimental parameters, and lexical), we were able to predict whether a dialogue was problematic with respect to speech recognition performance with 77% accuracy (Litman et al., 1999). In human-computer data from a deployed call routing system, we used a similar approach to predict problematic dialogues defined in terms of unsuccessful task completion (Walker et al., 2000a).

Using a set of automatic, task independent features representing three knowledge sources (acoustic, semantic, and dialogue), problematic dialogues with respect to task success could be identified with 87% accuracy, and predicted from just the first exchange in the dialogue with 72% accuracy.\(^2\)

We also showed the utility of predicting problematic dialogue situations for triggering system adaptation. Using a ruleset learned from training data, we modified the TOOT system to incrementally predict whether a user was having speech recognition problems as a dialogue progressed, and to adapt to a more conservative set of dialogue strategies whenever the user’s dialogue was predicted to be problematic. Our main result was that for a population of novice users, adding adaptation to TOOT significantly increased the task success rate from 23% to 65% (Litman and Pan, 2002).

While our initial work focused on predicting and adapting to problems at the (sub)dialogue-level, our more recent research with the TOOT corpus focused on modeling user problems at the utterance level, and investigating the types of system dialogue strategy changes that might be warranted after a single problematic utterance. In particular, we focused on predicting several different turn categories potentially useful for error handling: misrecognitions, corrections, and aware sites. We also greatly expanded our feature set for representing our dialogue data, to include prosodic features (pitch, loudness, turn duration, silence between turns, silence within turns, and speaking rate), more features derived from the speech recognition component and its inputs and outputs, as well as a set of history features for the dialogue as a whole and for more local dialogue contexts. Our machine learning results showed that the use of prosodic features - in conjunction with other features that were known or automatically detectable by a spoken dialogue system - yielded the best prediction performance. For example, our combined feature set accurately predicted speaker turns that were misrecognized – with 93% accuracy for predicting the presence of transcription errors, and 90% accuracy for predicting the presence of semantic errors (Litman et al., 2000).\(^3\)

We also found that our features could be used to predict user corrections of system errors with 84% accuracy (Hirschberg et al., 2001), and aware sites – turns where the user first becomes aware that the system has made an error, with 89% accuracy (Litman et al., 2001). All of these accuracy rates were significant improvements over baseline figures.

Other research has shown the utility of adapting to user problems at the utterance level. A spoken dialogue system that automatically adapted initiative based on participant roles, features of the current utterance and dialogue history (Chu-Carroll, 2000) outperformed a non-adaptive version in terms of usability and efficiency (Chu-Carroll and Nickerson, 2000). Dynamically deciding whether to confirm each user utterance during a task-oriented dialogue was also shown to improve system performance (Smith, 1998).

4.2 Tutorial Dialogue

For my current work I plan to take a similar approach, but now predict and adapt to problematic pedagogical situations in tutorial dialogues. Perhaps the most direct application of our previous work would be to attempt to detect and correct “tutorial-level misrecognitions.” Even if there is perfect speech recognition, other components of the system can still fail. In the text-based version of Why2,

\(^2\)This work has also been extended to predicting understanding errors at the utterance level (Walker et al., 2000b).

\(^3\)The utility of acoustic-prosodic features has also been shown by (Levow, 1998).
for example, there are cases where the student answers the system correctly, but the system believes that the student’s response was in fact incorrect (e.g., because the natural language component couldn’t handle the particular wording used by the student).

I also plan to pursue detecting and responding to certain affective states, motivated by recent work in both the tutoring and prosodic literatures. Evens has proposed detecting problematic affective student states (e.g., confusion, boredom, anger, and frustration) in the context of the Circsim-Tutor text-based dialogue system, with the goal of having the system then take steps to rectify the problematic situation (Evens, 2002). We hope to also pursue the detection and exploitation of such states. Initially we will focus on using the prosodic features examined in our earlier work for this type of prediction, building on current work exploring the use of prosody, sometimes in conjunction with other types of information, to detect emotional speaker states from speech (e.g., happy, afraid, angry, sad, or neutral (Polzin and Waibel, 1998); annoyance/frustration versus other (Shriberg et al., 2001); and emotional utterances such as anger and irritation that can signal “trouble in communication” and trigger a system action to prevent further breakdown (Batliner et al., 2001)). We will also look at related tasks such as detecting off-talk (Siepmann et al., 2001) and subjective language (e.g., expressing emotions, opinions, etc.) (Wiebe et al., 1999), which might also be useful for triggering system intervention.

Our longer term goal is to identify and rectify other types of potentially problematic pedagogical situations in tutorial dialogue (e.g., particular levels of student initiative (Jordan and Siler, 2002; Moore, 2002; Roque, 2002)), using our previous prosodic and other predictors.

5 Expected Contributions

Spoken dialogue is a natural and highly desirable form of student-computer interaction, which provides both opportunities and challenges to the area of dialogue-based tutoring systems. The opportunity arises from the presence of speech features such as prosody that serve as data for comprehending the nature of the student-computer interaction. The challenge is first in interpreting the speech itself and then in making productive use of those features.

The research and resulting technology from this proposal will hopefully lead to more natural and effective dialogue-based systems for tutoring applications. The major contributions will be 1) the development of a spoken dialogue tutoring system that can be empirically compared with text-based dialogue systems, and 2) an understanding of whether and how information from the speech signal can be used to predict pedagogically significant information that can be usefully exploited for learning gains.

Acknowledgements

Many thanks to Pam Jordan and to the reviewers of this paper for extremely helpful comments, and to Kurt VanLehn for his encouragement and help in undertaking this project. This research is supported by the National Science Foundation under Grant No. 9720359 to the Center for Interdisciplinary Research on Constructive Learning Environments (CIRCLE) at the University of Pittsburgh and Carnegie-Mellon University.

References


Supporting Progressive Refinement of Wizard-of-Oz Experiments

Armin Fiedler
Department of Computer Science
Saarland University
P.O. Box 15 11 50
D-66041 Saarbrücken, Germany
afiedler@cs.uni-sb.de

Malte Gabsdil
Department of Computational Linguistics
Saarland University
P.O. Box 15 11 50
D-66041 Saarbrücken, Germany
gabsdil@coli.uni-sb.de

Abstract. We present the architecture of DiaWoZ, a system supporting the design and execution of Wizard-of-Oz experiments to collect data from tutorial dialogues. The architecture is highly modular and allows for the progressive refinement of the experiments by both designing increasingly sophisticated dialogues and successively replacing simulated components of the tutoring system by actual implementations. The dialogues to be examined are specified in a language that combines finite-state approaches with information states, thereby allowing for great flexibility. Therefore, the architecture should also be appropriate for examining dialogues in general.

Keywords: Wizard-of-Oz experiments, dialogue systems, tutoring systems

1 Introduction

State-of-the-art tutorial systems use dialogues about the subject of the tutorial session to make the student develop the tutorial goal, but only few systems are able to perform a natural language style interaction [Person et al., 2000; Alevin and Koedinger, 2000; Heffernan and Koedinger, 2000]. In general, these systems have no means of entering flexible, natural language input. Instead, menu-based input or exact wording of the input is required. Moreover, the content of the tutorial sessions is quite limited. Moore [2000] empirically proved the necessity of natural language dialogue capabilities for the success of tutorial sessions. Wizard-of-Oz (WOz) experiments proved to be an appropriate approach to collect data about dialogues in complex domains [Bernsen et al., 1998].

Our newly started project DIALOG [Pinkal et al., 2001] aims at a mathematical tutoring system that features a full-fledged natural language dialogue management component. The tutoring system draws on the mathematical assistant system ΩMEGA [Benzmüller et al., 1997], a powerful proof development system. A major task in this project is to collect data about tutorial dialogues between a human student and the mathematical tutoring system. To do so, we plan a series of WOz experiments, in which the dialogue specifications become increasingly sophisticated and the meanwhile implemented components of the tutoring system are evaluated. To support the design and execution of the planned WOz experiments we are currently developing a tool, called DiaWoZ, that allows for rapid design and realization of experiments.

In Section 2, we discuss the planned WOz experiments. Section 3 is devoted to the architecture of DiaWoZ and Section 4 discusses the dialogue specification for a short example dialogue. We conclude the paper with a discussion of the current state and future developments.

2 Wizard-of-Oz Experiments

Wizard-of-Oz (WOz) experiments are a common means to collect human-machine interactions in the development cycle of dialogue systems (cf. [Bernsen et al., 1998]). In a WOz experiment subjects interact with a
system that is feigned by a human, the so-called wizard. Thus, WOz experiments generally allow one to capture the idiosyncrasies of human-machine as opposed to human-human dialogues [Fraser and Gilbert, 1991; Dahlbäck et al., 1993]. Although these experiments are rather costly in terms of preparation, execution, and evaluation, we consider them to be the method of choice to approach and explore more complex domains such as tutorial dialogues.

2.1 Progressive Refinement of Wizard-of-Oz Experiments

In our approach, we plan to progressively refine consecutive WOz experiments as depicted in Figure 1. Firstly, we want to collect some initial data and ideas about what human tutors do when they tutor students in the subject under consideration. In these “classical” WOz experiments the tutor is free to enter utterances without much restriction. Refinement at this stage merely means to define subdialogues or topics the wizard has to address during the dialogue, but without committing him to any predefined sequence of actions.

Secondly, we aim at setting up experiments where the dialogue specifications are spelled out in increasingly more detail, thereby limiting the choices of the wizard. These experiments will enable us to formulate increasingly finer-grained hypotheses about the tutoring dialogue, and to test these hypotheses in the next series of experiments.

Thirdly, we want to evaluate already existing components of the dialogue system before other components have been implemented. For example, if the dialogue manager and the natural language generation component are functional, but natural language analysis is not, the wizard has to take care of natural language understanding. In such a hybrid system, the wizard can still control the generated system utterances, allowing him to overrule choices by deciding for an alternative utterance defined by the dialogue specification. Since we expect that the inclusion of system components will have an effect on the dialogues that can be performed, the dialogue specification ought to be refined again whenever a new system component is added.

2.2 Tools Supporting Wizard-of-Oz Experiments

To support the wizard in his task of simulating a complex dialogue system, we are currently developing a tool called DiaWoZ. DiaWoZ is inspired by different existing dialogue building and WOz systems. MDWoz [Munteanu and Boldea, 2000] and SUEDE [Klemmer et al., 2000] are two examples of finite-state based systems for designing and conducting WOz experiments. MDWoz features a distributed client-server architecture and includes modules for database access as well as visual graph drawing and inspection. SUEDE provides a sophisticated GUI for drawing finite-state diagrams, a browser-like environment for running experiments, and an “analysis mode” in which the experimenter can easily access and review the collected data. The drawback of these systems, however, is that they only allow finite-state dialogue modeling, which is restricted in its expressiveness. Development environments like the CSLU toolkit [Sut-
ton and Cole, 1998], on the other hand, offer more powerful dialogue specifications (e.g., by attaching program code to states or transitions), but do not support the WOz technique.

The architecture of DiaWoZ and its dialogue specification language are designed to support the progressive refinement of experiments as discussed in Section 2.1. The dialogue specification enhances finite-state dialogue modeling with the possibility of storing variables that are global to the whole dialogue or local to some subdialogue. The variables are collected in a so-called information state, which allows us, for example, to parameterize transitions. Moreover, transitions have preconditions and effects. The preconditions restrict the applicability of transitions to certain configurations of the information state; the effects change the information state. Adding an information state to a finite state automaton renders the dialogue specification very flexible and allows us to define a wide range of dialogue models: from models that are purely finite-state by leaving the information state empty to the information-state–based dialogue modeling approach proposed in the TRINDI project (cf., e.g., [Traum et al., 1999]) by defining a degenerated automaton that consists of only one state and arbitrarily many recursive transitions. It is also possible to define a single state with one recursive transition without any preconditions and effects that allows arbitrary input by the wizard. This last setting is used for conducting “classical” WOz experiments. The combination of finite state automata with information states gives us the advantages of both approaches without committing to their respective drawbacks. This allows us to devise non-trivial dialogues that can still be handled appropriately by the wizard.

The modular architecture of DiaWoZ, which is discussed in more detail in the next section, allows us to refine the dialogue specification between experiments, and successively replace wizard controlled components by implemented modules of the tutoring system.

3 The Architecture of DiaWoZ

We assume that the task of authoring a dialogue to be examined in a WOz experiment is usually performed distinct in time and place from the task of performing the corresponding WOz experiment. To reflect this distinction, we decided to divide DiaWoZ into two autonomous subcomponents, which can be run independently: the Dialogue Authoring and the Dialogue Execution components. Figure 2 depicts the architecture of DiaWoZ.

The Dialogue Authoring component is a tool to specify the dialogues that are to be examined in a WOz experiment. Using the Graphical Dialogue Specification module, which allows for drag-and-drop construction of the dialogue specification, the experimenter can assemble a finite state automaton augmented with information states as the specification of a dialogue. A Validator ensures that the dialogue specification

![Figure 2. The architecture of DiaWoZ.](image-url)
meets certain criteria (e.g., every state is reachable from the start state, and the end state can be reached from every state). The complete dialog specification is passed to the Dialogue Execution component.

The Dialogue Execution component first parses the dialogue specification and constructs an internal representation of it. This internal representation is then used by the Executor to execute the automaton. The Executor determines which state is the currently active one and which transitions are applicable. Depending on the dialogue turn these transitions are passed to a chooser. The Generation Chooser receives the possible transitions that, in their effect, generate the tutor’s next utterance. The Analysis Chooser receives possible transitions that analyze the subject’s utterances. Both choosers may delegate the task of choosing a transition to specialized modules, such as an intelligent tutoring system to determine the next help message or a semantic analysis component that analyzes the subject’s utterance. Moreover, both choosers may also inform the wizard of the available options via the Wizard Interface and thus allow the wizard to pick a transition.

The tutoring system’s resp. wizard’s utterances are presented to the subject via the Subject Interface, which also allows the subject to enter text. The Logging Module structures and stores relevant information of the dialogue for subsequent examination by the experimenter.

DiaWoZ is devised as a distributed system, such that the Dialogue Authoring and the Dialogue Execution components, the Wizard and Subject Interfaces, and the Logging Module each can be run on different machines. The components are being implemented in Java and the communication is via sockets using an XML interface language. In the remainder of this section, we discuss the main components of DiaWoZ in some more detail.

3.1 The Dialogue Specification

As mentioned earlier, a dialogue specification in DiaWoZ consists of a finite state machine combined with an information state.

The finite state automaton is defined by a set of states and a set of transitions from states to states. Furthermore, the dialogue specification language allows for the definition of global variables, which are accessible from all states of the automaton. In addition, local variables can be defined for each state. The information state is conceived as the set of global and local variables that are accessible from the current state.

In contrast to the usual custom, the transitions have preconditions and effects. The preconditions are defined in terms of variables in the information state and restrict the set of applicable transitions for the current state dependent on the information state. The effects can both change the information state by setting its variables to different values and result in a function call triggering an observable event such as an utterance. In particular, transitions can be parameterized in terms of the variables of the information state and the values, to which they are changed in the transitions’ effects.

The dialogue specification language is defined in XML, which makes it rather clumsy to read, but easy to validate.

As an example consider the following task from algebra: An algebraic structure \((S, \circ)\), where \(S\) is a set and \(\circ\) an operator on \(S\), should be classified. \((S, \circ)\) is a group if (i) there is a neutral element in \(S\) with respect to \(\circ\), (ii) each element in \(S\) has an inverse element with respect to \(\circ\), and (iii) \(\circ\) is associative. In a tutorial dialogue, the tutor must ensure, that the student addresses all three subtasks to conclude that a structure is a group. An appropriate dialogue specification is given in Figure 3. The initial information state is displayed on the left side, while the finite-state automaton is shown on the right side.

State 1 is the start state. In State 2, there are three transitions \(t_1\), \(t_2\), and \(t_3\), which lead to parts of the automaton that represent subdialogues about the neutral element (States 3 and 6), the inverse elements (States 4 and 7), and associativity (States 5 and 8), respectively. The information state consists of three
global variables NEUTRAL, INVERSE, and ASSOCIATIVE capturing whether their corresponding have been solved. The preconditions of the transitions are the following:

\[ t_1: \text{NEUTRAL} = \text{open} \]
\[ t_2: \text{INVERSE} = \text{open} \]
\[ t_3: \text{ASSOCIATIVE} = \text{open} \]

The remaining transitions are always applicable.

The effects of the transitions \( t_1' \), \( t_2' \), and \( t_3' \) consist in changing the value of NEUTRAL, INVERSE, and ASSOCIATIVE, respectively, to done. Moreover, each transition produces an utterance in the dialogue. We will give more detail about the utterances in Section 4.

3.2 The Executor

The Executor is responsible for traversing the finite-state part of the dialogue specification. In particular, the Executor keeps track of the current state of the finite-state automaton and of the current information state. It calculates the set of applicable transitions in the current state based on the transitions’ preconditions and the information state. For example, if State 2 is the current state and the value of \( \text{NEUTRAL} \) is done, \( t_1 \) is not applicable. When a transition has been chosen, the Executor applies the transition, that is, it calculates the new state of the finite-state automaton and updates the information state as defined by the effects of the chosen transition. For example, when leaving State 3 by applying \( t_1' \), NEUTRAL is set to done.

3.3 The Choosers

The architecture depicted in Figure 2 includes two transition choosers. The Analysis Chooser manages the transitions that are responsible for analyzing the subject’s utterances. The task of the Generation Chooser is to choose the next action that should be performed by the system. We decided to make a clear-cut distinction between the two choosers in our architecture for two reasons. First, it prevents us from intermingling the transitions that encode the turns of the tutoring system with those that encode the subject’s turns, and thus supports a clearly modular design. Second, it allows us to add newly implemented subcomponents of the tutoring system that can provide the chooser with enough information to automatically choose a transition. For example, it should be possible to add to the Generation Chooser a software component that generates help messages without affecting other subcomponents of the Generation Chooser or the Analysis Chooser. Thus, the choosers allow for the progressive refinement of consecutive experiments. In general, the transition picked by the chooser can be presented to the wizard to confirm or overrule this choice.

3.4 The Wizard and Subject Interfaces

The Wizard Interface (cf. Figure 4) includes a frame that displays the dialogue as it has unfolded so far as well a split window, which displays both the options provided by the Generation Chooser (indicated
by Tutor Choices) and the options provided by the Analysis Chooser (indicated by Subject Choices) in two distinct frames. At each point in time only one of the choosers is enabled depending on the dialogue turn. Figure 4 shows the situation, where the subject entered an utterance, which the wizard analyzes by choosing entries in pull-down menus. Note that the Generation Chooser is disabled, since we are in the subject’s dialogue turn. The disabled chooser still shows the options from the previous dialogue turn. To allow for parameterized transitions we use pull-down menus that facilitate the wizard’s task substantially. Other options may be displayed as simple buttons or edit fields.

![Figure 4. The Wizard Interface window.](image)

The subject interface is rather simple. Although a multi-modal interface is desirable, DiaWoZ currently allows only for text output and input.

4 An Example Dialogue

To show how DiaWoZ works, let us come back the example dialogue specification given in Figure 3. It covers the following example dialogue (where $Z$ denotes the set of integers):

(U1) **Tutor:** To show that $(Z, +)$ is a group, we have to show that it has a neutral element, that each element in $Z$ has an inverse, and that $+$ is associative in $Z$.

(U2) **Tutor:** What is the neutral element of $Z$ with respect to $+$?

(U3) **Student:** $0$ is the neutral element, and for each $n$ in $Z$, $-n$ is the corresponding inverse.

(U4) **Tutor:** That leaves us to show associativity.

Let us now examine in detail how this dialogue unfolds. Starting in State 1, there is only one transition that can be picked, namely $t_0$. It leads to State 2 and outputs utterance (U1). In State 2, all three transitions $t_1$, $t_2$, and $t_3$ can be picked, because their preconditions are fulfilled. The wizard chooses $t_1$, which leads to State 3 and produces the tutor’s utterance (U2). Now, the student enters utterance (U3). Note that the student not only answers the tutor’s question, but also gives the solution for the second subtask about the inverse elements. Since there is no natural language understanding component included in the system in
our example setting, the wizard has to analyze the student’s utterance. To allow for that, DiaWoZ presents the window depicted in Figure 4 to the wizard, where the field titled “Repeat” stands for transition $t_4$, while the field titled “Correct Answer” denotes transition $t'_4$. The wizard instantiates the parameters of $t'_4$ by choosing the value done for the variables NEUTRAL and INVERSE of the information state to be set by the effect of $t'_4$. Note that this choice reflects that the student overanswered the tutor’s question. Moreover, note that due to the overanswering the tutor should not choose the subtask about the inverse elements in the next dialogue turn, but instead proceed with the remaining problem about associativity. With clicking OK in the “Correct Answer” field, transition $t'_4$ is selected. Thus, the Executor updates the information state by setting the values of NEUTRAL and INVERSE to done and brings us back to State 2. This time, only transition $t_3$ is applicable, which produces utterance (U4).

5 Conclusion

We presented the architecture of DiaWoZ, a Wizard-of-Oz tool, which can be used to simulate human-machine interactions in complex domains. In the design, we put great emphasis on modularity and clear interface specifications. To define dialogues, we augmented finite-state approaches of dialogue modeling with information states. This gives us great flexibility in designing dialogue specifications. Hence, the architecture is not restricted to tutorial dialogues, but also usable for examining dialogues in general. One of the main features of the architecture is that it allows for the progressive refinement of consecutive WOz experiments: It is possible both to refine the dialogue specification between experiments and to successively add and evaluate already implemented modules of the tutoring system.

DiaWoZ is a distributed system that is being implemented in Java using XML for the communication between modules via sockets. Since XML parsers are available for almost every programming language, new modules can be programmed in arbitrary programming languages and added to the system. So far, we have implemented the distributed infrastructure of DiaWoZ including the communication via XML messages. We have also implemented part of the Executor and prototypes of the Wizard Interface, the Subject Interface, and the graphical dialogue specification and inspection tool.

We are currently investigating how we can further extend the dialogue specification language to allow for the definition of sub-automata. This would enable us to compose dialogues out of smaller building blocks that can be easily substituted or shared between consecutive experiments. Moreover, we are currently including a powerful representation of the discourse structure in DiaWoZ. The discourse structure will both represent the rhetorical relations between discourse segments and serve as a dialogue history to support an envisioned dialogue planning module.

Acknowledgments

We thank Christian Korthals and Elsa Pecourt for not only implementing DiaWoZ, but also discussing its design. Moreover, we are especially thankful to the anonymous reviewers for many useful comments.

References

Targeted Help: Embedded training and methods for evaluation

Beth Ann Hockey¹, Gregory Aist¹, Jim Hieronymus¹, Oliver Lemon², John Dowding¹

¹Research Institute for Advanced Computer Science
RIACS, MS T27A-2, NASA Ames Research Center, Moffett Field CA 94085
{bahockey, aist, jimh, jdowding}@riacs.edu

²Center for the Study of Language and Information
Stanford University, Stanford CA 94305-4115 USA
lemon@csli.stanford.edu

Abstract
We describe the notion of Targeted Help: embedding training on which utterances to use with a spoken dialogue system into the system’s interactions with users. Targeted Help has three components: what the system heard, what the problem was with the utterance, and what the user might want to say instead. We describe each part in turn and offer a preliminary version of how one might want to evaluate each component and the overall system as well. This paper explores possibilities for doing useful evaluation with a corpus on the domain not specifically collected as a user test for Targeted Help. Evaluation with existing corpora is of interest because it would allow reuse of corpora and facilitate evaluation earlier in the development of a system.

Keywords: spoken dialogue, embedded training

1. Introduction

As part of a spoken dialogue system, grammar based recognizers tuned to a domain perform very well, in fact better than comparable Statistical Language Models (SLMs) for in-coverage utterances (Knight et al. 2001). However, users, particularly non-expert users of dialogue systems, will produce utterances that are out of coverage. The goal of our Targeted Help approach is to make use of these out of coverage utterances to provide the user with more specific information in cases in which the system is not able to understand their speech. Currently most systems respond to recognition failures with, at best, some version of “Sorry I didn’t understand you, try again”. This type of response provides the user with no information about why their utterance failed.

We want instead to work toward a system that creates more informative prompts for the user. For instance, if the user is unfamiliar with a push-to-talk device, the Targeted Help component could work out that the user was already speaking when the push-to-talk button was pressed, and respond with Please be sure that the button is pressed before you begin speaking. Similarly if the user produces a word that is not in the system’s vocabulary, the Targeted Help could report that the system doesn’t know the word. The idea is to have embedded tutoring about the dialogue system. Systems that use grammar-based language models will obviously benefit from teaching the user the bounds of the system quickly, and enabling them to use within-grammar utterances.

There are a number of strategies for what type of Targeted Help system response might be produced. The previous example illustrates a diagnostic response. Another possibility is to produce an in-coverage example sentence similar to what the user said. In our system we have in fact done both. The utterances produced by our implementation of Targeted Help are composed of one or more of the following pieces:

- **What the system heard:** A report of the backup SLM recognition hypothesis,
- **What the problem was:** Description of the problem with the user's utterance (e.g. the system doesn't know the word), and
- **What you might say instead:** A similar in-coverage example.
Some obvious questions that arise in implementing the Targeted Help strategy are: 1) what can be known about the user's utterance if the system is not able to understand it but only has the imperfect backup recognizer output? and 2) in what way can example utterances generated by the Targeted Help agent be made similar to the user's utterance? One aspect of similarity that we suspect is important is matching the user's utterance type (e.g., yn-question, wh-question, command etc.) because otherwise the user is likely to be misled into thinking that a particular utterance type is impossible in the system. It also is likely to help naive users learn the coverage of the system if the examples give them valid uses of in-coverage words they produced in their utterance because it gives them both confirmation that those words are in coverage and an in-coverage pattern to imitate.

Even when the user produces an out of coverage utterance they are likely to produce some in-coverage words. Our Targeted Help system runs a fall-back speech recognizer that is being driven by a category-based statistical language model. When the grammar-based recognizer fails, the Targeted Help agent looks for within-domain words in the recognition hypothesis from the fall-back recognizer. This gives us a set of target words from which to match the example to the sentence type of the user's utterance: wh-question, yn-question, answer, or command.

Furthermore, for commands, which are a large percentage of the utterances, we also attempt to produce an in-coverage example that is interpretable by the system. These examples are intended to demonstrate how in-vocabulary words from the backup recognizer hypothesis could be successfully used in communicating with the system. For example, if the user says something like Go and measure the pressure at the, and the fall-back recognizer detected the words measure and pressure, the Targeted Help agent could provide an example grammatical utterance like measure the pressure at the commander's seat.

In this submission we will follow one of the suggestions in the call for papers and describe planned evaluations for this system. We wish to address in particular the question of what useful evaluation can be done prior to a full blown user test based on previously collected corpora. While reasonable designs for user tests for this system are fairly straightforward, the questions of how to design the corpus investigation and what one can reasonably expect to learn from it, are fairly uncharted and we think would benefit from discussion by the workshop participants.

2. System Description

For a system in which to implement Targeted Help, we are working with a simulated Personal Satellite Assistant (Rayner, Hockey, and James 2000), which controls a simulated PSA robot in a simulation of the Space Shuttle. (We are also working with the WITAS system for controlling a simulated helicopter in urban search and rescue (Lemon et al. 2002).) The system features an asynchronous dialogue manager (Boye, Hockey, and Rayner 2000), combined with the Nuance Communications Speech Recognizer, the Gemini language understanding system (Dowding et al. 1993), and the Festival Speech synthesis package. The components are integrated using SRI's Open Agent Architecture (OAA) (Martin et al. 1998). The Targeted Help capabilities are implemented through a secondary SLM speech recognizer and a separate Targeted Help agent.

3. Component-wise Evaluation

There are two major components of the Targeted Help system that can be evaluated, the backup SLM speech recognizer and the Targeted Help agent that uses the recognizer output to formulate responses to the user. Within the Targeted Help agent there are subcomponents that can be evaluated separately in addition to doing evaluation of the Targeted Help agent as a whole. What we are ultimately interested in is whether the Targeted Help prompts assist users in performing more effectively when interacting with the system, whether users learn about the system faster enabling them to produce fewer out of coverage utterances and whether this saves users time and effort. Clearly full-blown user tests could be designed to investigate these issues, however user test are time consuming and expensive. At a relatively early stage in a research project it would be useful to be able to evaluate the system and its components in some less demanding way to determine whether the approach is productive, and gain insights into how to improve it before running a user study. This approach would allow the research questions to be clarified, making better use of a later user study. In this paper we will investigate what can be learned from a previously collected corpus that was not designed for testing Targeted Help. In fact, this corpus of interactions from 25 users was collected on a version of the RIALIST Simulated Personal Satellite Assistant System (PSA) before the idea of targeted help had even been formulated. We have tried out some of the ideas for evaluation that we describe later in this paper by creating a batch mode for the Targeted Help agent and running it on the corpus. In the following sections we will briefly discuss the SLM, and more thoroughly
examine two aspects of the Targeted Help agent: problem diagnosis and in-coverage example generation. In each section, in addition to reporting measurements on the PSA corpus, we will discuss possible measures that could be calculated, what information those measures could give you about the performance of the system or component and what the assets and drawbacks are for each measure.

The corpus we are using consists of 6264 utterances. This data was collected from 25 subjects not previously involved in the development of the system; each subject was first given a uniform short introductory session, and then asked to solve a number of tasks which involved using the speech interface to the simulated robot. We have begun some of the evaluation described below focussing on the out of coverage utterances. Targeted Help is triggered when the main recognizer fails and the secondary SLM recognizer succeeds so the utterances that are out of coverage for the language model of the main recognizer are certain to trigger Targeted Help. There could also be cases of failed recognition for in coverage utterances but there is also the possibility that the back up recognizer would recognize them correctly and they would resume normal processing rather than producing a Targeted Help message.

3.1 What the system heard: SLM recognizer

The most common way to measure recognizer performance is to calculate word error rate (WER) (Zue & Cole 1997). WER compares the recognizer output to a reference string, generally the transcribed input utterance and gives the portion of words in the recognizer output that differ from the refernece string. The WER calculation takes into account substitutions, deletions and insertions of words. To evaluate the performance of the SLM feeding the Targeted Help agent we used out of coverage utterances from our corpus and found the WER for the current instantiation of the recognizer to be 50.6%. WER is relatively straightforward to calculate and gives a reasonable idea of how close the process being evaluated is to the reference string. In the Targeted Help case we get an estimate of how likely the recognizer output is to have something useful in it. However, this evaluation alone does not provide information about ultimate functionality.

Another common measure is sentence error rate (SER). SER measures how many sentences have one or more word errors in them. SER is not a very useful measure in the case of the Targeted Help SLM because we expect just about every sentence to have at least one error. Fortunately,. for Targeted Help we only really need the back up SLM recognizer to get enough of what the user said to construct a useful system response. For novel applications like the PSA there is typically not enough training data for statistical recognizer performance to be very good. Our current SLM was trained on approximately 4500 utterances from 3 different domains. It has category back off (that is trigrams are based on categories generalized from the data rather than individual words) as a way to compensate for having only a small amount of data. While we will continue to work on improving the performance it is also completely clear that we should continue to work on developing Targeted Help to be successful with less than stellar recognizer output, because in many spoken dialogue systems there will be only a small amount of training data available. Given only small amounts of training data there will be severe limitations on how much the performance of the SLM can be improved. The Targeted Help strategy is a way to make use of both the potential for good accuracy of grammar based recognizers and the broader but much less accurate performance of SLMs in conditions of limited training data.

3.2 What the problem was: Problem Diagnosis

Another major function of the Targeted Help agent is to make a diagnosis of what was problematic about the user’s utterance. The agent has several possibilities – such as unknown word, unknown verb subcategorization and early endpointing (when the recognizer incorrectly identifies the end of an utterance before it is actually finished) – that it evaluates. More than one can be relevant simultaneously. The values of these individual judgements are not as important as the pattern as whole. Consequently, the obvious thing to look at is a vector of these values and compare that to a vector of values for the same features generated by a human judge or judges. Preliminary attempts indicate that this is an easier judgement task than creating target example sentences. We still need to address the problem of what constitutes a meaningful measure of similarity for these vectors. For evaluating the applicability and coverage of the problem diagnosis we plan to also look at what percentage of the time the Target Help agent gets a diagnosis versus producing the default message for the diagnosis portion of the system response.

3.3 What you might want to say instead: Example sentences
One part of what Targeted Help produces is an in coverage example sentence. What can we learn from the corpus data? The standard approach to evaluation of recognizers provides inspiration for one way to measure the quality of the examples produced. In an ideal world we could use WER to compare the example produced by Targeted Help to “perfection” where:

\[
\text{Perfection} = \text{the utterance that would cause the user to produce the successful/right thing on their next utterance & for ever after.}
\]

In our less than ideal world the obvious problem is where to find an approximation to perfection. There are several potential sources of this gold standard each with its own advantages and disadvantages:

The first possibility is: \(\text{Gold\_standard}_1 = \text{next effective,in-coverage user utterance}\). This scheme has the advantage of comparing Targeted Help examples to actual user utterances and avoiding judgement error. The logs of the interaction will show when the user’s utterance resulted in the system performing an action, consequently the identification of the next effective, in-coverage user utterance can be quite objective. \(\text{Gold\_standard}_1\) will produce a reference string for each item triggering Targeted Help unless the problematic utterance is the end of the interaction. Also, \(\text{Gold\_standard}_1\) is guaranteed to be in-coverage. In addition there are possibilities for semi-automatic identification. Important drawbacks to \(\text{Gold\_standard}_1\) are the assumptions it requires such as:

- User maintains same goal, rather than abandoning it and doing something completely different
- User keeps rephrasing at the same level of detail e.g. “measure temperature at flight deck” versus “Go to flight deck. Measure temperature”; and
- User produces minimal changes to utterance in attempting to find something successful.

If these assumptions don’t hold the Targeted Help message will be evaluated as having an undeservedly higher WER. So even though we can get a \(\text{Gold\_standard}_1\) sentence for each failure many of them may not be useful.

A second variation would be: \(\text{Gold\_standard}_2 = \text{next effective, in-coverage user utterance for which human judge determines that assumptions hold}\). The advantage over \(\text{Gold\_standard}_1\) is that the cases with failed assumptions that are not a reasonable point of comparison will be eliminated (to the limit of the judges’ accuracy). The main drawback is that this non-automated subjective step of doing the judgements is introduced.

Another possibility is: \(\text{Gold\_standard}_3= \text{user provided utterance in response to system asking what they meant}\). Building evaluation into the dialogue system would mean that comparisons would be with actual user utterances and the process would be semi-automated. Potential difficulties are that having this type of evaluation/data collection embedded in the task is likely to make interacting with the system very annoying to users. Variability in users’ responses is likely to be very high leading to the need for very large numbers of subjects.

Another possibility is: \(\text{Gold\_standard}_4 = \text{What human judge(s) say the "perfect" example is}\). The advantage to this approach is that you avoid the assumptions of \(\text{Gold\_standard}_1\). The main drawbacks are that \(\text{Gold\_standard}_4\) sentences will not be actual user utterances and the judges will not be completely accurate. In particular even judges that understand the grammar coverage quite well may produce out of coverage utterances. However we expect the percentage of out of coverage utterances for \(\text{Gold\_standard}_4\) to be less than the number of \(\text{Gold\_standard}_1\) items that violate crucial assumptions.

A variation of \(\text{Gold\_standard}_4\) is to have a judge (who is not familiar with the coverage) perform a generate-and-test evaluation by thinking of potential examples and then running each possibility through the parser for the language model to see if it parses. We tested this approach on 200 utterances with the results shown in table 1. RPrompt is the Targeted Help prompt produced by using the SLM recognizer hypothesis as input. TPrompt uses the transcription of the user utterance as input to the Targeted Help agent thereby showing pure Targeted Help agent results. As we can see from the table WER is very high. In looking through the Targeted Help output however, there are quite good examples which are technically errorful. The trouble with WER doesn’t take into account paraphrasing, or intent. Small words that might not matter much in interpretation count as errors. For example, if \(\text{Gold\_standard}_4 = \text{“Go to the mid deck and measure the pressure”}\) and the example produced by Targeted Help was “Go to mid deck and measure pressure” we get 2 errors for an example that is basically ideal. So WER are probably falsely high.
The paraphrasing problem might be improved by considering some type of bag-of-words approach as has been discussed in the generation and translation literature. Getting at intent is more difficult since without some directly measurable indicator, identification of intent will be entirely subjective.

4. Overall Evaluation

In thinking about overall evaluation it is clear that corpus data of this type will provide information about user performance. However, we can answer questions like “If targeted help worked perfectly, how often would it apply and how much effort would it save?” To determine how often Targeted Help would apply we need to add together the number of out of coverage utterances, which is easy to determine (in our corpus, 533) and some in-coverage utterances that failed to be recognized correctly by both recognizers (slightly harder to count), about 9% for our corpus. This number also allows us to calculate an upper bound on how many user utterances could be avoided if, for each possible type of system command or dialogue move, the Targeted Help message allowed the user to get back on track immediately and avoid all subsequent mistakes. The upper bound is the number of times Targeted Help triggers minus one. We also plan to look at how often Targeted Help produced the default message vs. the non-default message to evaluate the effectiveness of our range of diagnosis types.

References


Viewing and Analyzing Multimodal Human-computer Tutorial Dialogue: A Database Approach

Jack Mostow, Joseph Beck, Raghu Chalasani, Andrew Cuneo, and Peng Jia

Project LISTEN\(^1\), 4213 RI-NSH, 5000 Forbes Ave, Carnegie Mellon University, Pittsburgh, USA
Telephone: 412-268-1330 (voice) / 268-6436 (FAX); email: Mostow@cs.cmu.edu

Abstract. It is easier to record logs of multimodal human-computer tutorial dialogue than to make sense of them. In the 2000-2001 school year, we logged the interactions of approximately 400 students who used the Reading Tutor and who read aloud over 2.4 million words. This paper discusses some of the problems in extracting useful information from such logs and the difficulties we encountered in converting the logs into a more easily understandable database. Once log files are parsed into a database, it is possible to write SQL queries to answer research questions faster than analyzing complex log files each time. The database permits us to construct a viewer to examine individual Reading Tutor-student interactions. This combination of queries and viewable data has turned out to be very powerful. We provide examples of questions that can be answered by each technique as well as how to use them together.

Keywords: viewing and analyzing tutorial dialogue, session logs, evaluation tools, Project LISTEN, Reading Tutor, multimodal dialogue, speech I/O, session browser, database queries, perl, MySQL

1 Introduction

It is easier to record logs of multimodal human-computer tutorial dialogue than to make sense of them. We discuss this problem in the context of Project LISTEN’s Reading Tutor, which listens to children read, and helps them. The resulting multimodal dialogue includes mouse clicks and speech as student input, and spoken and graphical assistance as tutor output. In the 2000-2001 school year, hundreds of students used the Reading Tutor daily at three elementary schools, reading over 2.4 million words. How can we analyze so much data? The Reading Tutors logged thousands of sessions, but the logs are too detailed to see the forest for the trees.

An alternate way to capture detailed tutorial interactions in human-viewable form is to videotape them, as we have done in a number of studies [7]. Video has obvious advantages, but many drawbacks. Video is laborious to record well at schools, invades privacy, can distort student behavior, captures only one level of externally observable detail, omits internal events and tutorial decision processes, and is tedious to analyze or search. To avoid these drawbacks, we describe a database approach we developed to view and analyze logs of children’s interactions with Project LISTEN’s Reading Tutor.

1.1 Project LISTEN’s Reading Tutor

The aspects of the Reading Tutor relevant to this paper are the channels of communication between Reading Tutor and student, and the forms in which the Reading Tutor logs them. For publications on successive versions and evaluations of the Reading Tutor [4], please see www.cs.cmu.edu/~listen.

\(^1\) This work was supported by the National Science Foundation under Grant Number REC-9979894. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation or the official policies, either expressed or implied, of the sponsors or of the United States Government. We thank other members of Project LISTEN who contributed to this work; MySQL’s developers; the students and educators at the schools where Reading Tutors recorded data; and the anonymous reviewers for making this paper better than we intended.
A Reading Tutor session consists of logging in followed by a series of stories, which the student and Reading Tutor take turns picking. Each story is a sequence of a few types of steps: assisted reading, writing, listening, and picking. The Reading Tutor may insert a preview activity beforehand, and/or a review afterward, also built out of the same types of steps.

The Reading Tutor’s output is textual, graphical, and audio. It displays text on the computer screen, sentence by sentence, for the child to read aloud. It gives various sorts of help on a word or sentence by playing human or synthesized speech, with graphical cues such as highlighting words as it speaks them.

The Reading Tutor inputs the student’s speech, mouse clicks, and keyboard presses. It uses a less than perfectly accurate speech recognizer to produce a time-aligned transcript of each student utterance.

1.2 Data recorded by the Reading Tutor

The Reading Tutor logs information in various forms. Although other tutorial dialogue systems may be organized very differently, we suspect that they may encounter some of the same problems we identify, such as proliferation of log entry types, the need to identify a few key control points, and the need to identify the appropriate level(s) of detail and aggregation at which to represent and analyze tutorial interactions.

The .wav file for each utterance contains the digitized speech. Its “hypothesis” file contains the speech recognizer’s time-aligned transcript, one word per line, showing its start and end time as offsets in centiseconds from the beginning of the utterance:

```
BY 227 249
11 250 278
```

Utterance files are a simple representation, both human-readable and conducive to automated analysis, especially since this representation has remained stable for years. However, they are incomplete, capturing only what students said and what the recognizer “heard,” not what else the student or Reading Tutor did. The “sentence” file shows the alignment of this hypothesis against the sentence text, using –1 to indicate that a sentence word had no hypothesis word aligned against it:

```
121
divided
by
11
is
1
```

The Reading Tutor records tutorial interactions in detailed logs, starting a new log each time it is launched. Log files are tens of megabytes in size and many thousands of lines long. Each line is generated by a “logprint” utility function that records a sequential event number, a severity level, a timestamp, the number of digitized speech samples recorded so far, the name of the function that logged this event, and an event description whose form and content depend on the type of event logged and is often much longer than in the examples, which were chosen for their brevity. For example, this pair of lines in the log indicates that an utterance has ended and been captured:

```
16466, Notice, "Tue Apr 10 12:30:20.387 2001", 10763200,
"CListener::Finalize Utterance", "End Utterance"
```

```
16467, Notice, "Tue Apr 10 12:30:20.417 2001", 10763200,
"CCapture::WriteWaveFile(int)", "Wrote File:
d:\listen\cd\Tue-Sep-19-23-44-58.093-2000\Capture\fAT6-6-1994-08-01\dec-fAT6-6-1994-08-01-Apr10-01-12-30-14-902.wav"
```

Over years of development, the Reading Tutor code has accumulated over a thousand calls to logprint for various purposes, including debugging and performance tuning. They record not only externally observable events, but also internal decisions at various levels of control. For example, suppose the student clicks on the word “cat.” The Reading Tutor first computes the types of help it can give on this word, such as speaking the word, sounding
it out phoneme by phoneme, and so forth. From this set it chooses to give, say, a rhyming hint. From its table of rhymes, it randomly chooses the word “bat,” and verifies that it has a recording of “bat.” Then it queues a sequence of audio and graphical actions to say “rhymes with,” display the word “bat” beneath the word “cat,” and say “bat.” The log file records this sequence of actions as separate events, but does not explicitly link them together as a single abstract event of the form “give help of type h on word w from time t1 to t2.” One reason for this is the difficulty in logging intervals rather than single events. Obviously a log message cannot be generated at time t1 that also provides the endtime. Logging the event at time t2 would destroy the chronological ordering of the log files. Since the log files were designed to be human readable, this ordering is an important property.

Although a detailed record can be useful for debugging, it is impractical to write scripts to parse and analyze a thousand different types of log entries. Over time we have reorganized the code to instrument key control points. For example, all graphical and audio output goes through a single “player” object, and key tutorial events are logged by a single function named CUserHistory::RecordEvent.

1.3 Previous ways to view and analyze Reading Tutor data

The Reading Tutor’s logs are hard for humans to read and for programs to analyze. We first tried to solve the readability problem by making the Reading Tutor generate summaries, whose level and purpose varied.

Reading Tutors connected to the Internet via each school’s local area network with an automated script every night to ftp summary data back to Carnegie Mellon to be mirrored and archived. These summaries provided a human-readable view of what the student was doing in the session – mostly for us researchers, since teachers have little time to peruse such detail.

Between student sessions, each Reading Tutor displayed a summary to let us monitor its overall usage and reliability objectively, both for a given day and historically, rather than rely on reported teacher impressions. For example, the following summary was displayed on a Reading Tutor in a school lab:

Today : 1 readers, median session 24 minutes, total 26 minutes, 0 program restarts

Average (in 154 days used): 2 readers, median session 18 minutes, total 39.3 minutes, 2.9 program restarts.

Last... start: 4/05/2001 5:03 AM; login: 4/05/2001 12:24 PM; story: 4/05/2001 12:24 PM; utterance: 4/05/2001 12:34 PM; logout: 4/05/2001 1:00 PM

The Reading Tutor displayed a student roster to monitor individual usage, facilitate scheduling, and indicate student performance. The roster showed how long each student had read that day, the student’s name (altered in this paper to protect anonymity), his or her current level, number of distinct stories and words seen to date, and usage, e.g.:

24 min. Sarah Reader (level B) has read 145 stories and 1460 words, in 88 17.1 minute sessions from 10/23/2000 to 4/05/2001

Clicking on the student’s name brought up the “Reading Journal” summary generated for that day’s session, listing the sequence of activity steps in the session. For example, the April 5 session comprised 6 activities, totalling 12 steps, one of them described as follows:

Reading Tutor chose “How To Make Cookies By Emily Mostow.” (level B). Sarah has not finished this activity before.

April 5, 2001 At 12:24:28 PM: Started reading "First, Get The Batter."

New words: Put Ingredients Batter Oven

April 5, 2001 At 12:26:52 PM: Finished this step after 2 minutes, with 5.8 words accepted per minute.
Summaries can help expose bugs concealed by more detailed descriptions. For example, browsing the first (1999) version of the Reading Journal quickly exposed a logic error that sometimes caused the Reading Tutor to pick stories at too high a level—a bug that appeared only after prolonged use, since it occurred only when the student had finished all the stories at a given level.

The summaries were generated as plaintext or HTML, and aimed at human viewing. For automated analysis, we wrote special-purpose perl scripts specific to particular research questions. For example, by analyzing hypothesis and sentence files we were able to measure total usage and the amount of rereading [6]. Other questions required scripts to analyze log files. These scripts were complex enough for us to doubt their correctness. We allayed some of these doubts (and found bugs to fix) by hand-analyzing randomly selected examples, but still did not entirely trust the scripts.

The Reading Tutor bases some of its decisions on the student’s history. For example, it needs to keep track of which words and stories the student has seen before. For this purpose it maintains a database of previous events. Some of our previous analyses used this type of database. The database for a single Reading Tutor was considerably more selective than the detailed log files, but over the course of a year often grew to over 100 megabytes. Nevertheless, it lacked some information needed to view or analyze tutorial interactions in detail.

2 A database approach

A July 2000 talk by Dr. Alex Rudnicky on his session browser for the Communicator system [1], coupled with our desire to understand how students were using the Reading Tutor at schools, inspired the vision of a log viewer to display Reading Tutor interactions at multiple levels of detail. We report on work in progress toward this vision—bugs and all.

Our approach parses utterance and log files into a database, enabling us to answer research questions by writing concise database queries instead of complex perl scripts. In contrast to the summaries recorded by the Reading Tutor, the log viewer uses stored queries to generate views at different levels dynamically, allowing us to modify the view form and content. We now explain how we represent, populate, query, and view the database.

2.1 Database schema for Reading Tutor data

Figure 1 summarizes the entity relationship schema we developed to model data from the 33 Reading Tutors used during the 2000-2001 school year by hundreds of students at three elementary schools. Database schemas are important to get right, and this one took weeks to finalize. To edit the schema, we used Microsoft Visio™, which keeps track of fields and dependencies while providing a graphical view that lists each table’s fields (not shown here) but suppresses other details such as the data type of each field.

The entities modeled in the schema range in grain size from schools down to individual word encounters. Arrows encode many-to-one relations. For example, each Reading Tutor was launched once a day, more if it crashed and was restarted; a school had multiple students; a launch had some number of student sessions.

We faced a challenge in figuring out how to model the elements of a session. One choice was to model a session as a sequence of story encounters, a story as a sequence of sentence encounters, and a sentence encounter as a sequence of utterances. This model matched a useful simplified view of a session as consisting of picking a series of stories to read, and it used appropriately different fields to describe stories, sentences, and utterances. For example, story tables had a field for the difficulty level of the story, sentence encounter tables had a field for when the student began reading this sentence, and utterance tables had a field for name of the wav file that contained the student’s speech. However, this structure did not match the reality that stories were multi-step activities, that the Reading Tutor often inserted a teaching activity before a story, and/or a review afterwards, and that the Reading Tutor architecture treated each session as a uniform tree structure of such steps. Such a step tree would be easy to represent recursively in the schema, but would omit bona fide distinctions among stories, sentences, and utterances. How to reconcile these two models? We decided to encode both models in parallel, relating them by linking each sentence encounter to the step during which it occurred.
We used separate tables to represent mouse clicks, graphical actions, and Reading Tutor audio output, because they have different parameters. However, it is not very useful to view them separately, so we wound up generating merged tables that combined them into a unified sequential view.

We did not attempt to model the Reading Tutor’s decision processes, only its interactions with students. We did include “Abstract Response” to model feedback as occurring either before the student began to read the sentence, after the Reading Tutor detected a mistake, as backchanneling (active listening such as “uh huh”) or as praise. But just because the schema could represent abstract responses did not mean that the parser could always extract or infer them from the logs, or figure out which words they involved, so this table was incompletely populated. It was not always feasible to determine what word received this help, because help actions were not logged as abstract “molecules” such as “sound out CAT,” but instead as individual “atomic” graphical actions and audio segments, such as “highlight the letter C” and “play phoneme_K.wav.”

2.2 Populating the database

To populate the database, a perl script parsed through individual log files in a single forward pass. Each line was either disregarded as unimportant, used to maintain state that may be relevant for future records (e.g., user name, session start time), or was used to generate a particular record. The parser scanned for 33 different types of lines in a log – 13 different kinds of CUserHistory::RecordEvent lines, plus 20 other types of lines.

When the parser generated a record, it called a filer function to handle the communication with the database and create the necessary records in dependent tables, such as Sentence for a sentence encounter record. The filer returned an internal id from the database so that future related records, such as audio output within a sentence encounter, could refer to it.

For example, this line marked the start of a sentence encounter with the text “Dividing By 11”:

15023, Notice, "Tue Apr 10 12:27:13.648 2001", 7776000, "CParagraph::SetSentence", "Dividing By 11"

Accordingly, the script created a Sentence Encounter record with 12:27:13.648 as its start time, “Dividing By 11” as its text, and (based on state maintained from parsing previous lines) the internal (database) ids of the launch, session, story encounter, and step. 740 lines later, the script found a “coach_goes_forward” event, which enabled it to fill in the end time of the Sentence Encounter as 12:28:35.386.
Similarly, the parser started an Utterance record when it encountered a “user_begins_turn” event, and completed it when it encountered a pair of lines in the log showing that an utterance ended and was captured. The parser extracted from these lines the start time, end time, event number, and wav file for the utterance. From the state it maintained, it filled in the sentence encounter ID and its number of utterances so far. For each word in the sentence file, the parser created a Word Encounter record to specify which word, if any, was aligned against the sentence word, and computed its start and end times from the time offsets.

2.3 Implementation of the multi-level log viewer

Given the database, the viewer was straightforward to implement. We used MySQL Database-Server (see www.mysql.com) to serve the database, perl DBI packages to interface to the database, perl and CGI scripts to generate the views, and Apache Web-Server [8] to serve the views. All of these packages are free to download. To save time, we adopted a uniform tabular style rather than craft more specialized, aesthetic views, as we would if they were intended for teachers and not just us.

A link to a view is encoded as a call to the script that generates that view. For example, clicking the link http://logviewer.cmu.edu:9876/cgi-bin/storyList.pl?session_id=8562 calls the script storyList.pl to list the story encounters for session number 8562. The script executes a database query to retrieve them:

```
select story.story_id, title, level, file_path, story_encounter_id, start_time, end_time, type_desc, student_level, byte_offset, event_number, initiative, sms, ems
from type_description,story, story_encounter
where story_encounter.story_id = story.story_id and
      story_encounter.session_id = $sessionid and
      type_description.type_id = story_encounter.exit_through
order by start_time, sms
```

The script outputs an HTML table with a row for each record returned, embedding any links to other views.

2.4 Views: what the browser shows

In general, the log viewer generates a view as a list of records in an HTML table, with one row per value, and column headers at the top. One or more fields in each row may contain clickable links to more detailed views. To protect student anonymity, we restrict access and omit or alter names in our examples.

The highest-level view lists the (three in this database) values of School, with columns for school name, location, number of Reading Tutors, and number of students. Clicking on a school name brings up a table of Reading Tutors at that school, with columns for machine name, description, and number of launches. Clicking on a machine name brings up a chronologically ordered table with one row for each launch of the Reading Tutor on that machine. Each row includes the date and time of the launch, which version of the Reading Tutor was launched, the number of sessions during that launch, and a link to the log file for the launch. Clicking on a session count links to the list of sessions, and so on.

Clicking on a session’s story count, step count, or student name brings up a table of stories read, session steps, or student information, respectively. For example, Figure 2 lists stories read in one session. Each row shows the story encounter’s start time, end time (omitted here to save space), duration, number of sentences read, total number of sentences in the story, title, reader level, how the encounter ended, story level, and who picked the story.

This example revealed two instructive problems. A bug in the populating script inflated the number of sentences in the story to include previews and reviews. Second, the “select_response” value for how the last story encounter ended indicates that the log was missing expected information. From the log, it appears that the user timed out, but no Pause event was recorded. As these examples illustrate, a database populated from log files not designed for that purpose can be informative but buggy or incomplete.
Figure 2: Table of activities for a session

Figure 3 shows the table of sentence encounters for the story “How to Make Cookies by Emily Mostow” (written by the first author’s then-8-year-old daughter). The first two records come from preview activities that introduced the words “oven” and “batter,” which the student was encountering for the first time in the Reading Tutor. The fifth record shows that the sentence “Then put it in the oven” took 33 seconds, with 2 utterances and 3 other actions, totaling 5 – few enough to list here, which is why we chose this example.

Figure 3: Table of sentence encounters for the story “How to Make Cookies by Emily Mostow”

Besides starting at the list of schools and browsing downward to more detailed views, we wanted to view specific entities found by queries. We therefore provided a more direct form of access by inputting the entity’s database id. The same integer represents different ids in different tables, so the user must also select the type of table – of schools, Reading Tutors, launches, sessions, story encounters, sentence encounters, or utterances.
Figure 4: Table of actions and utterances for the sentence encounter “Then put it in the oven”

3 Using the database to answer research questions

We are using the database both to replicate previous studies [3] rapidly and to answer new research questions [5], thanks to the (relative) ease of constructing queries and validating their correctness. For statistical analysis we use SPSS’s ability to import data from the database, and our SQL client (urSQL)’s ability to export results into Excel.

3.1 Constructing queries

To answer a research question, we formulate it as an SQL query. As a simple example, were students likelier to back out of a story if the Reading Tutor chose it than if they did? This query counts how often students backed out of stories, disaggregated by who chose:

```sql
SELECT se.initiative, COUNT(*)
FROM student_click sc, story_encounter se, sentence_encounter sen
WHERE
  sc.start_time = se.end_time
  AND sc.type_id = 8
  AND sc.sentence_encounter_id = sen.sentence_encounter_id
  AND sen.story_encounter_id = se.story_encounter_id
GROUP BY se.initiative
```

The basic logic (in the `WHERE` condition) is to find story encounters that ended when the student clicked `Back` (click type 8) out of a sentence and this click occurs at the same time a story ended. One problem could arise if a student clicked back at the same time someone else finished a story on a different computer. This case should not count as backing out of a story. To avoid this miscount, the `WHERE` clause also specifies that the click must occur in a particular sentence in a story that ends at the same time as the click. The first line specifies what data to collect – the initiative (who chose the story) and how many items met the condition.

<table>
<thead>
<tr>
<th>Item</th>
<th>Initiative</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(null)</td>
<td>140</td>
</tr>
<tr>
<td>2</td>
<td>student_initiative</td>
<td>703</td>
</tr>
<tr>
<td>3</td>
<td>tutor_initiative</td>
<td>2457</td>
</tr>
</tbody>
</table>

Table 1: Results of query to determine when students backed out of stories

The query results in Table 1 (based on about 400 students) support a Yes answer to the research question: the Reading Tutor chose the story in 2457 cases where the student backed out, the student chose in 703, and who chose was not specified in 140 cases. Null values may be deliberate for introductory tutorial “stories” presented automatically to newly enrolled students, or may be caused by bugs in generating or parsing the logs. They seem inevitable in a large database, but if rare enough do not preclude informative analysis, as this example illustrates.
3.2 Using the viewer to debug and improve queries

Queries are excellent tools for generating summary results, but are less powerful for examining individual cases. For example, to discover that not all the null initiatives were due to introductory tutorials, we used a query to find instances of stories with null initiatives. Then we used the viewer to examine their text and tutorial interactions. This combination of techniques was useful on several occasions.

For example, to analyze how much time students spent waiting for the Reading Tutor to respond, we developed a query to compute the delay from the last word a student says in a sentence to the first word s/he says in the next sentence. This query is 75 lines long because it requires several steps: finding the last word of a sentence and the time it was uttered, finding the time the student uttered the first word of the next sentence, ensuring that both sentences are in the same story, etc. After debugging each step, we ran the full query. By sorting the resulting delays, we found some too long to be believable. The table of results identified the story encounters where they occurred. By using the viewer to browse these story encounters, we found the bug: between some sentence encounters were activities where students were supposed to write, which the query erroneously included as part of the delay. It is important to note that it is not necessary to create a new view for each research question. For example, the view shown in Figure 3 was used to find the flaws in the query to compute how long students were waiting. However, this view was designed before we knew that we would be conducting this analysis, and could be used to verify other queries. The same tactic of viewing outliers traced supposedly hours-long sessions to a bug in filling in missing end times.

An additional benefit of the log viewer is that by presenting student-computer interactions in a more understandable form, it makes it easier for people with incomplete knowledge of the project to take part in data analysis. Our project had a near perfect split between those people who understood how the Reading Tutor worked and those people who could write SQL queries. The viewer allowed those unfamiliar with the tutor to perform sanity checks on their queries (as in the case of student writing activities, mentioned above). People who were less familiar with SQL used the viewer to examine unlikely query results and to find glitches in the database. This dichotomy of project members’ knowledge is not unique to Project LISTEN; finding some means to work around this gap is very helpful.

3.3 Benefits of using a database

One benefit of using the database is the ease of getting summary information from the database. We have had project meetings where questions were raised and immediately addressed by a quickly written query (e.g. “How many times did the Reading Tutor provide each type of help?”). Although is is slower to get detailed information about each student, rather than a summary, from the database, it is comparable to or better than the prior technique of using perl scripts. Using queries, a complex analysis of how long students delayed before beginning to pronounce a word took 3 weeks to go from an idea to writing up results. This is comparable to the amount of time it took to write the perl scripts. However, the perl scripts were developed by someone who had been working on Project LISTEN for significantly longer (2 years) than the person who was developing the queries (6 months). Also, the work using queries analyzed how these delays related to students’ performance on paper tests [2], so the investigation was somewhat more complex than the original analysis [3].

The comparison of SQL vs. perl is not quite a fair one, as the difference in ease of use has less to do with the languages than with the data each of them processes. SQL queries manipulate a structured database that we took time to set up, while the perl scripts had to work with low-level log files. To create the database, we had to debug a set of perl scripts. This task was time intensive, but only had to be done once. Its constant cost is amortized over all of the analyses performed. So for investigating a small set of research questions a database might not be worth the cost of setting up, but for more open-ended investigations it is.

4 Conclusions

We detailed a database approach to view and analyze multimodal tutorial dialogue, and how we applied it to Reading Tutor data. We now summarize its caveats, then its benefits.
It was hard to develop a good schema and useful views, especially for pre-existing logs that lack some desired information, at least in easy-to-extract form. We plan to make the Reading Tutor generate the database records in real time, or at least more parseable logs.

The database must be robust to tutor crashes and bugs. For example, when a crash ends a log prematurely, the end time of events in progress must be filled in. Parsing the logs exposed some Reading Tutor bugs, such as assigning the same filename to two utterances, which must not be allowed to corrupt the database.

Populating the database took weeks for our data, with 2.4 million word encounters. In the 2000-01 school year, 33 Reading Tutors each recorded hundreds of logs, typically tens of megabytes long, with thousands of utterance files. To avoid duplicating data, a long populating process must be robust to stops and restarts.

Although the database takes long to design and construct, it pays off in queries much shorter than perl scripts, because they are expressed more declaratively. Database technology absorbs much of the complexity of searching and assembling data. When necessary, we speed up queries by adding appropriate indices, but that type of optimization is easier and less bug-prone than rewriting conventional procedures to speed them up.

Views package certain types of queries in a reusable, understandable form, easier to use than querying the database directly – especially for views that integrate multiple tables, and for users more fluent at clicking on links than at formulating SQL queries. It is hard to design views both concise and detailed enough to be useful. Some of our views list hundreds of records – too many to fit on one screen or peruse easily. Views should summarize lower-level details in informative aggregate form, for example, durations and counts of utterances and actions in a sentence encounter. Queries make such aggregation easier, less bug-prone, and more flexible than in procedural tutor code.

We use queries both to answer statistical questions by aggregating over lots of data, and to find examples of particular phenomena, such as outlier values. We use the viewer to inspect such examples in detail, finding bugs or unexpected cases that refine the question. Finally, simply browsing our data at multiple levels often exposes interesting phenomena.

**References (also see www.cs.cmu.edu/~listen)**

AutoTutor's Log Files and Categories of Language and Discourse

Johanna Marineau1, Andrew Olney1, Max Louwerse1,
Natalie Person2, Brent Olde1, Suresh Susarla1, Patrick Chipman1, Art Graesser1, and the
Tutoring Research Group

1Department of Psychology, 202 Psychology Building,
The University of Memphis, Memphis, TN 38152 USA
jmarinea@memphis.edu

2Department of Psychology, 2000 North Parkway,
Rhodes College, Memphis, TN 38112 USA

Abstract: AutoTutor is a computer tutor that interacts with students in a multi-turn tutorial dialog with natural
language. AutoTutor has been developed for computer literacy and conceptual physics, showing learning gains of .6
standard deviation units. This paper documents AutoTutor's log files. The log files record pertinent information
about each turn that the student contributes, including syntactic parsing, speech act segmentation and classification,
coverage of the expected content, the quality of the student contribution, and other dimensions of language and
discourse. The log file also records relevant information about the selection of tutor’s dialogue moves through the
Dialogue Advance Network (DAN).

Keywords: log files, intelligent tutoring systems, annotation, mark-up initiatives, dialog, AutoTutor

Introduction

AutoTutor is a computer tutor that holds conversations with students in natural language (Graesser, Person, Harter, &
TRG, in press; Graesser, VanLehn, Rose, Jordan, & Harter, 2001; Graesser, P. Wiemer-Hastings, K. Wiemer-
Hastings, Kreuz, & TRG, 1999). In each tutoring session, AutoTutor presents a number of questions (or problems)
that require deep reasoning, explanations, and approximately a paragraph of information in the answer (or solution).
A multi-turn exchange between the tutor and student is needed for a full answer to evolve. AutoTutor facilitates the
evolution of an answer by generating a number of different categories of dialog acts, such as pumps (“Tell me
more.”), prompts for specific information, hints, assertions, summaries, corrections, and feedback about the student’s
contributions (positive, neutral, negative). These dialog acts are sensitive to the quality and nature of student
contributions. There also is an animated conversational agent (i.e., talking head) that presents the tutor’s
contributions.

The argument has been made that that there is something about natural language and discourse that appears to be
effective in promoting learning gains in human tutoring (Corbett, Anderson, Graesser, Koedinger, and VanLehn,
1999; Graesser, Person, & Magliano, 1995). This has motivated a number of researchers to incorporate natural
language dialog facilities in their tutoring systems (Aleven, Popescu, & Koedinger, 2001; Freeman, 2000; Moore,
1995; Rose, Jordan, Ringenberg, Siler, VanLehn, & Weinstein, 2001). Indeed, we have found that AutoTutor
produces a learning gain of .6 standard deviation unit compared to a reread control in our evaluations of AutoTutor
in the domain of introductory computer literacy.

It is beyond the scope of this paper to describe the mechanisms that process student contributions, generate dialog
moves, and manage the dialog. AutoTutor uses Latent Semantic Analysis (Landauer and Dumais, 1997; Landauer,
Foltz, and Laham, 1998) to represent world knowledge and evaluate the conceptual quality of student contributions
in comparison with expected answers to deep-reasoning questions (Graesser, K. Wiemer-Hastings, P. Wiemer-
Hastings, Harter, Person, and the TRG, 2000; Franschetti, Karnavat, Marineau, Callie, Olde, Terry, and Graesser,
2001). A Dialogue Advance Network (DAN) serves as a dialog management facility to handle mixed-initiative
dialogue (Person, Graesser, Kreuz, Pomeroy, and the TRG, 2001). There is curriculum script that stores concepts, the
sets of deep-reasoning questions, expected answers to questions, families of hints and problems for each expected
answer, sets of misconceptions, corrections of errors, and other content about the subject matter. The information in
the student’s turn is compared to the expected answers and misconceptions by using LSA in the match processes.
The tutor’s dialog moves are formulated by the DAN in conjunction with LSA and the curriculum script.

When each student’s turn is processed, mechanisms are needed to parse and segment the natural language input, to
assign speech acts and clauses to discourse categories, to evaluate the quality of the student’s contributions, and to
formulate the dialog moves of the tutor. Researchers have proposed a number of category systems for classifying
and marking up the speech acts in the tutorial dialog (Core, Moore, & Zinn, 2000; Person et al., 2001; Shah, Evens,
Michael, & Rovick, 2002). Some of these analytical systems require expert human judges to segment, mark up, and
classify the constituents, whereas other systems have most or all of these components automated. In the case of
AutoTutor, all of the components are automated. Manual coding of over 100 hours of human to human tutoring
sessions was done to determine the tutoring strategies and discourse patterns of untrained human tutors (Graesser &
Person, 1994; Person, Graesser, Magliano, & Kreuz, 1994). A five-step dialogue frame and the dialogue moves
identified in this analysis were used in the development of AutoTutor. The log files reflect the components of the
tutorial dialogue that were identified as important for supporting the dialogue management and in evaluating of
system's effectiveness. In the future, the log files will be analyzed to determine the relationship between discourse
patterns and learning gains. Just as important, there will be a comparison between the discourse categories
automatically assigned in the log file and categories assigned by human experts on the basis of discourse theories.
Discrepancies between the log files and human experts will guide future developments of AutoTutor.

Transcripts and log files of the tutoring sessions record any aspect of the tutoring session that will be of further help
in evaluating the quality of the tutor-student interactions, and the computational architecture of the system. The log
files in AutoTutor record the student-tutor dialogue, syntactic parses of the student’s turns, the categories of the
speech act classifier, states of the DAN, and LSA evaluations of the student’s input. The remainder of this paper
discussed the contents of the log files as they are currently implemented in AutoTutor.

Contents of the Log Files

We are currently conducting a large-scale experiment on the tutoring of conceptual physics at three universities
(University of Pittsburgh, Rhodes College and the University of Memphis). All of the tutoring systems present
conceptual physics problems such as the one below.

When a car without headrests on the seats is struck from behind, the passengers often suffer neck
injuries. Why do passengers get neck injuries in this situation? Explain.

It should be noted that a good answer would require about a paragraph of information because the student is expected
to explain his/her reasoning. In this section we will examine a section of one of the intelligent tutoring systems,
called Why/AutoTutor. Why/AutoTutor’s log file was collected during one of these physics tutoring sessions. As
we discuss different aspects of the log files, we will show an excerpt from the log file. The Appendix contains the
complete section of the subtopic being discussed.

Each session begins by recording the subtopic being tutored, the level of difficulty (easy, moderate, hard), and the
time and date the session began. In the section of the log file we discuss here, the topic is “neck-injury” and the
session has a low level of difficulty.

An important component of a dialogue management system is the classification of the input to determine the user's
intent (Allen, Byron, Dzikovska, Ferguson, Galescu & Stent, 2001). Dialogue acts can have multiple levels of
meaning. One can ask “Can you repeat the question?”, assert “I would like you to repeat the question.”, or issue a command “Repeat the question.” The speaker’s intent is the same, but they are different speech acts with different surface forms. Various mark-up languages have been developed to classify the user's input into categories that are useful to dialogue management systems in particular domains. DAMSL (Dialogue Act Mark up in Several Layers) is a classification scheme developed by the analysis of advice-giving and task-oriented dialogues (Allen & Core, 1997). This classification scheme includes categories for accepting and rejecting proposals that are specific to the types of dialogues analyzed. Shah et al (2002) have proposed another classification scheme based upon an analysis of computer mediated human to human tutorial dialogue. Student initiatives, defined as any student contribution that is not an answer to a tutor's question, are categorized along four dimensions: communicative goal, surface form, focus, and hedging. In tutorial dialogue it is necessary to appropriately classify the student’s input. If the system is not responsive to the student's directives and questions, then they quickly become frustrated and limit how they interact with the system.

In order to capture the user’s intent (at least to some extent) and to encourage mixed-initiative dialog, AutoTutor uses a speech act classifier that categorizes speech acts in a student’s turn into one of 24 possible categories, including question categories, and metacommunicative and metacognitive frozen expressions (see Louwerse, Olney, Mathews, Marineau & Mitchell, submitted; Olney, Louwerse, Mathews, Marineau, Mitchell, submitted). The questions are classified into 16 categories based on a taxonomy developed by Graesser, Person, and Huber (1992). The classifier uses both a syntactic and a semantic analysis for this purpose, and correctly classifies questions with 86% accuracy (Olney et al, submitted). The speech act classifier for AutoTutor was developed to provide speech act classification at a more general level. As we develop AutoTutor to become more responsive to student's questions, we needed a mechanism that would accurately identify the type of question the student was asking without being restricted to a particular domain. The log files contain the parser output as well as the speech act category (“contribution” in the example below).

In the log file below AutoTutor first uses one of a large variety of canned expressions classified according to their use in dialog (prompt-marker, elaboration-marker, hint-marker, etc.). It then introduces a context and asks a question. The time it takes AutoTutor to respond and the time of the student to respond is presented next, followed by the student input. The category in which the student’s answer is classified is given next, in addition to a parse of the student answer that was partially used in determining the speech act category.

TUTOR-DM-147> Let's talk about something else.
TUTOR-DM-148> When a car without headrests on the seats is struck from behind, the passengers often suffer neck injuries. Why do passengers get neck injuries in this situation? Explain.
AT-TRT-UTTERANCE-TIME-54> 21.6320000000019
AT-SRT-UTTERANCE-TIME-54> 144.946999999997
STUDENT-55> The passengers get neck injuries because the car behind them is applying a force in one direction and there is nothing to absorb the force of the head snapping back
TUTOR-SAC-53> CONTRIBUTION
PARSE-53> (S (NPL (DT The) (NNS passengers)) (VP (VBZ is) (VP (VBG applying) (NPL (DT a) (NN force)) (PP (IN in) (NPL (CD one) (NN direction)))))) (CC and) (VP (ADVP (RB there)) (VBZ is) (NP (NN nothing) (TOINF (VP (TO to) (VP (VB absorb) (NP (NPL (DT the) (NN force)) (PP (IN of) (NPL (DT the) (NN head) (VBG snapping) (NN back)))))))))

In addition to each student and tutor turn, the log files provide a continuous dialogue history of the student's contributions used in calculating the subtopic coverage score. The current contribution is the student's current input to be analyzed. Since this is the first contribution of the student, the current contribution is the same as the dialog history. The dialog history can get quite long of course.

DIALOG-HISTORY-49> the passengers get neck injuries because the car behind them is applying a force in one direction and there is nothing to absorb the force of the head snapping back body
Using the dialogue history and the student's current contribution, the log file records the extent to which this information covers expectations and misconceptions that are anticipated in the curriculum script. These coverage scores may correspond to global and local spans of information. LSA cosine metrics (which vary from 0 to 1) assess the extent to which the entire dialog history and the immediate local contribution cover particular bad answers (called “badPoints”), and expected good answers (called “goodPoints”). There are other coverage metrics, which need not be addressed, in the present context. There is a completeness measure that assesses the proportion of expectations that are covered (i.e., the cosine metric exceeds some threshold). The verbosity measure is an index of how wordy the student has been throughout the previous session.

TUTOR-ASR-95> misconceptionAnswer: 0.59989697
TUTOR-ASR-95> subTopicFirstMove: 1.0
TUTOR-ASR-95> topicCoverage: 0.38809592
TUTOR-ASR-95> scriptCoverage: 1.0
TUTOR-ASR-95> completeness: 0.25
TUTOR-ASR-95> badPoints(4): 0.36137545
TUTOR-ASR-95> badPoints(3): 0.23158917
TUTOR-ASR-95> badPoints(2): 0.7409022
TUTOR-ASR-95> badPoints(1): 0.6127475
TUTOR-ASR-95> subTopicCoverage: 0.25
TUTOR-ASR-95> badAnswer: 0.7409022
TUTOR-ASR-95> verbosity: 1.0
TUTOR-ASR-95> goodPoints(4): 0.55672544
TUTOR-ASR-95> goodPoints(3): 0.6629203
TUTOR-ASR-95> goodPoints(2): 0.22583447
TUTOR-ASR-95> goodPoints(1): 0.49844313
TUTOR-ASR-95> topicAbility: 0.4932672
TUTOR-ASR-95> scriptAbility: 0.6085928
TUTOR-ASR-95> goodAnswer: 0.7571305

TUTOR-ASR-95> misconceptionPoints(1): 0.59989697
TUTOR-ASR-95> subTopicAbility: 0.4932672

LOCALASSESSMENTS-48>
TUTOR-ASR-96> misconceptionAnswer: 0.59989697
TUTOR-ASR-96> subTopicFirstMove: 0.0
TUTOR-ASR-96> topicCoverage: 0.2966336
TUTOR-ASR-96> scriptCoverage: 1.0
TUTOR-ASR-96> completeness: 0.25
TUTOR-ASR-96> badPoints(4): 0.36137545
TUTOR-ASR-96> badPoints(3): 0.23158917
TUTOR-ASR-96> badPoints(2): 0.7409022
TUTOR-ASR-96> badPoints(1): 0.6127475
TUTOR-ASR-96> subTopicCoverage: 0.25
TUTOR-ASR-96> badAnswer: 0.7409022
TUTOR-ASR-96> verbosity: 1.0
TUTOR-ASR-96> goodPoints(4): 0.55672544
TUTOR-ASR-96> goodPoints(3): 0.6629203
TUTOR-ASR-96> goodPoints(2): 0.22583447
TUTOR-ASR-96> goodPoints(1): 0.49844313
TUTOR-ASR-96> topicAbility: 0.49326718
TUTOR-ASR-96> scriptAbility: 0.60769707
TUTOR-ASR-96> goodAnswer: 0.7571305
Based on the good and bad answer scores calculated for the student's current contribution, the program provides mean and z-scores for the good and bad aspects. Using these scores the student is provided with positive-positive, positive-neutral, neutral-neutral, neutral-negative or negative-negative feedback. The tutor uses these categories of feedback for generating the appropriate discourse markers (e.g. excellent, good, okay, I see, not really, nonsense).

Each expectation (aspect) of the ideal answer has hints, prompts, and elaborations (assertions) that are written in the curriculum scripts (Graesser, et al., 1999). After selecting the appropriate type of dialogue move for AutoTutor to use next, the log files display a selection of the particular dialogue move chosen, and the LSA cosine score that indicates which of the moves would best increase the aspect coverage if the student answers correctly. The system then records the aspect and dialogue move the tutor uses next.

Once a student has provided the correct answer to one expectation (aspect) of the ideal answer, the log files show the expectation rankings by which the system chooses the next expectation to address. LSA coverage scores are shown for the expectations that have not yet been fully answered (sub-threshold scores). The coherence score is an LSA score ranking each expectation by how well it would contribute to the overall coherence of the dialogue, i.e., overlap with the previous expectation that was just covered. The redundancy score indicates how much the coverage of one particular expectation would cover other expectations in the ideal answer.

Subthreshold Ranks

Aspect (4) In an attempt to produce the required large force, the neck gets stretched and may get injured damaging its muscles and ligaments. 0.55672544
Aspect (1) When a car is struck from behind the force of impact will cause a large forward acceleration of the car. 0.49844313

Coherence Ranks

Aspect (4) In an attempt to produce the required large force, the neck gets stretched and may get injured damaging its muscles and ligaments. 0.55672544
Aspect (1) When a car is struck from behind the force of impact will cause a large forward acceleration of the car. 0.49844313

Redundancy Ranks

Aspect (4) In an attempt to produce the required large force, the neck gets stretched and may get injured damaging its muscles and ligaments. 0.55672544
Aspect (1)  When a car is struck from behind the force of impact will cause a large forward acceleration of the car.

The student provided two correct expectations in the ideal answer. Aspect (4) was chosen as the answer aspect that would assist the student in the completion of the ideal answer. After each aspect change, the tutor re-evaluates and generates the dialogue move most likely to lead the student toward the ideal answer.

Discussion

Log files are used in AutoTutor for keeping records of tutoring sessions, with the hope of incrementally improving the system’s performance. Analysis of AutoTutor's log files from sessions in computer literacy provided useful information that was used to improve the quality of the dialogue and the functioning of LSA in the most recent version of AutoTutor. In the initial analysis it was noted that the limited number of meta-cognitive and meta-communicative statements that the tutor offered made his conversational style redundant and less natural. In later versions of AutoTutor, we have expanded the number and variety of meta-cognitive and meta-communicative statements offered by the tutor. Another deficit discovered in this analysis was that the users were unsure of when it was their turn to respond to the tutor. If the tutor's final dialogue move was an elaboration (assertion), the student was unclear how he/she was expected to respond (Person, Graesser, Harter, Mathews, and the TRG, 2000). The tutor's final dialogue move was changed so he would offer a hint, a prompt, or a summary. This change alleviated some of the turn-taking confusion.

Analysis of the LSA scores documented in the log files indicated that there were adjustments that could be made to the curriculum scripts and the LSA space to improve AutoTutor's performance. The LSA values of the students' input were analyzed to assess LSA's ability to accurately evaluate the student's knowledge. A student's answer that used content words contained within the curriculum scripts would artificially inflate the LSA score. The use of important concept words would result in LSA giving the answer a high rating even though it contained a misconception. The curriculum scripts were rewritten to contain generic forms of important content words. These generic concepts were then associated with the more specific terms in the concept list. This prevented high LSA scores for answers that contained the right words but the wrong context. The evaluation of LSA led to the question of what information should be included when training LSA. Would a larger training corpus lead to better performance? What would happen to performance if the corpus contained historical information, or examples of the misconceptions a student might have? Five different physics corpora were analyzed to determine what type of corpora would provide the best LSA performance (Olde, Franceschetti, Karnavat, Graesser, & the TRG, in press). The corpora differed in size and the amount of historical information and misconceptions they contained. The study showed that the size of the corpus had little impact on LSA performance, and a small corpus of relevant material could provide acceptable performance. There was also no benefit in removing irrelevant information, such as historical facts and examples of misconceptions and errors.

A number of markup initiatives have been introduced in the past (Allen & Core, 1997; Carletta, Isard, Isard, Kowtko, Newlands, Doherty-Snoddon, & Anderson, 1997; Di Eugenio, Jordan & Pylkkänen, 1997). The purpose of these markup initiatives is to provide a corpus with a detailed tag set that would facilitate subsequent discourse analyses. These initiatives are extremely useful in many areas of language and dialog analysis. Furthermore, they provide a level of detail that can currently not be achieved by any automatic tagging system. On the down side, these initiatives are extremely labor intensive and time consuming. Automatic tagging is less detailed, but adequate for the purposes of many research projects. It remains to be seen whether Why/AutoTutor's current log file will go the distance in tracking and improving the tutorial dialog, but the log files from the earlier versions of AutoTutor have provided us with valuable information used to improve the conversational quality and pedagogical effectiveness of the current AutoTutor.

Acknowledgements

This research was supported by grants from DoD Multidisciplinary University Research Initiative (MURI)
program administered by the Office of Naval Research (N00014-00-1-0600) and from the National Science Foundation (SBR 9720314 and REC 0106965). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DoD, ONR or NSF.

References


Can we help students with a high initial competency?

Carolyn P. Rosé and Kurt VanLehn and Pamela Jordan

LRDC, University of Pittsburgh
Pittsburgh, PA 15260 USA
rosecp@pitt.edu

Abstract

It is a well known phenomenon that students with high initial pretest scores demonstrate lower learning gains since there is less room for improvement. In this paper we explore a related issue, that of selecting appropriate interventions for students who start out with a high initial competency. We present a mathematical model that makes it possible to project success rate at selecting appropriate interventions based on accuracy at analyzing student performance at a task. This model demonstrates that for high initial competency students, selecting interventions based on any sort of isolated analysis of student performance is doomed to failure even with very high accuracy analysis. We explore an alternative approach to selecting interventions within the context of the Why2 conceptual physics tutoring system (VanLehn et al., 2002) based on prior probabilities of student need. Our analysis demonstrates that within this domain, individual differences between high competency students makes this type of approach equally unsuccessful. We conclude by discussing some alternative approaches to solving this problem, which we are still investigating.

Keywords: knowledge construction dialogues, selecting interventions

1 Introduction

It is a well known phenomenon observed in all types of learning studies that students with high initial pretest scores demonstrate lower learning gains since there is less room for improvement. In this paper we explore a related issue, that of selecting appropriate interventions for students who start out with a high initial competency. We use as our example the Why2 conceptual physics tutoring system (VanLehn et al., 2002). The goal of the Why2 system is to coach students as they explain physics mental models in natural language in response to short essay questions such as, “Suppose you are running in a straight line at constant speed. You throw a pumpkin straight up. Where will it land? Explain.” The Why2 system has at its disposal a library of knowledge construction dialogues (KCDs), i.e., interactive directed lines of reasoning, each of which is designed either to elicit a specific idea (i.e., an elicitation KCD) or to remediate a specific misconception (i.e., a remediation KCD).

When students interact with Why2, they are first presented with an essay question. After reading the essay question, the student types in an initial essay. The system then analyzes the student’s essay in order to detect the presence of misconceptions and required concepts. The system then uses KCDs both for coaching students to insert missing required concepts (elicitation KCDs) and to remove the expression of misconceptions (remediation KCDs). When a student’s essay is analysed, the system constructs a list of missing required points and misconceptions. For each of these items, the student will receive an elicitation KCD for every point analysed as missing and a remediation KCD for every misconception analysed as present.

As the system interacts with the student, it keeps track of which interventions have already been given, and thus can avoid repeatedly offering the same intervention. This contextual information about which
interventions have been offered can be thought of, then, as a sort of primitive user model that aids in future decisions about which of the available interventions it should offer. However, for the initial essay that students type in, the system makes its decision about what intervention to offer purely based on its analysis of the student’s essay. Intuitively, helping students with a low initial competency is easier than for those with a high initial competency since almost any intervention selected is likely to be appropriate since so many are needed. Equally intuitive is the idea that high analysis accuracy is required for selecting interventions for high initial competency students since they need only a very few interventions. Thus, the likelihood is high that a haphazardly selected intervention will not be appropriate, and conversely that the few interventions that are required will not be selected. What is less intuitive is how dire the situation actually is once this phenomenon is quantified, and the conclusion that selecting interventions based solely on an isolated analysis of student performance is doomed to failure with high initial competency students. One possible solution is to use prior probabilities of student need to influence the selection of interventions. Our preliminary investigations of this approach within the context of Why2 have demonstrated that the high degree of individual differences in our domain between high competency students make some sort of student specific student modeling necessary to make this type of approach successful.

We begin by presenting our model for projecting success at selecting appropriate interventions based on accuracy of analysis of student performance. We then describe our preliminary investigations into using prior probability of student need to influence the selection of interventions in Why2. We conclude with some current directions.

2 Selecting Appropriate KCDs

Figure 1: This Figure summarizes our model for predicting KCD precision, recall, and false alarm rate from analysis precision, recall, and false alarm rate. Note that this model only applies to the selection of elicitation KCDs.

<table>
<thead>
<tr>
<th>Point Identified</th>
<th>Point Not in Essay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point Identified</td>
<td>$A$</td>
</tr>
<tr>
<td>Point Not Identified</td>
<td>$C$</td>
</tr>
</tbody>
</table>

Analysis Precision $= A/(A + B)$
Analysis Recall $= A/(A + C)$
Analysis False Alarm Rate $= B/(B + D)$
KCD precision $= D/(C + D)$
KCD Recall $= D/(B + D)$
KCD False Alarm Rate $= C/(A + C)$

In order to build an effective system, it is important both to give students the interventions that they do need, and to avoid giving them extraneous interventions that they do not need. Neglecting to give a student an intervention that is needed means losing an opportunity to teach that student something that student needs to know. Giving an intervention that a student does not need means wasting a student’s time, possibly distracting that student from what that student really needs to learn, and likely annoying or even confusing that student. Thus, we would like to build a system with a high interventions selection recall and low selection false alarm rate, where we define selection recall as the percentage of interventions that a student needs that the system gives. And selection false alarm rate as the percentage of interventions that the student does not need that the system gives.

Nevertheless, analyzing student performance, which in Why2 is an essay, but in another system may be an equation or a proof, etc., is most naturally evaluated separately from that of selecting interventions. For example, in the case of Why2, analysis of student essays is a computational linguistics problem, and performance on this task is most naturally conceptualized as a text classification problem and measured.
in terms of analysis precision, recall, and false alarm rate over a corpus of student essays. Analysis precision is the percentage of required points and misconceptions identified in the student essays that were actually present in those essays. Note that this is undefined in the case that no required points are identified. Related to this notion is analysis false alarm rate, which is the percentage of required points not present in the essay that were incorrectly identified by the system. Analysis recall is the percentage of misconceptions and required points present in student essays that were actually identified by the system. Note that this is undefined whenever there are no required points present in a student essay. Naturally, a system that is good at accurately identifying required points and misconceptions in student essays will also be good at selecting appropriate KCDs to engage students in. However, our mathematical model demonstrates that the relationship between analysis precision, recall, and false alarm rate and KCD precision, recall, and false alarm rate varies widely depending upon the quality of student essays.

Figure 1 presents some equations that describe our mathematical model. Note that this model only applies to the selection of elicitation KCDs. In our domain, missing information from essays is a much more prevalent problem than incorrect information indicating a misconception that we are prepared to handle with an available remediation KCD. As the equations in Figure 1 demonstrate, we define Recall for analysis as the number of required points that WHY2 correctly identifies as present in a student essay divided by the total number of required points actually present in the essay. Precision is the number of required points correctly identified divided by the total number of required points that WHY2 identified, correctly or incorrectly. False alarm rate is computed by dividing the number of required points identified but missing in the essay by the total number of required points missing from the essay. Note that analysis precision, recall, and false alarm rate are a byproduct of the analysis approach we have selected. Different NLU techniques achieve different levels of precision, recall, and false alarm rate. And we are continuing to experiment with different approaches. However, once an analysis approach is selected, we treat analysis precision, recall, and false alarm rate, which measure our accuracy at analysing student essays, as given. On the other hand, you will notice that KCD precision, recall, and false alarm, which measure our accuracy at selecting elicitation KCDs, is a function of analysis accuracy and essay quality. It is also possible to show that, holding KCD accuracy constant, that analysis accuracy varies with essay quality. But in practice, it makes more sense to consider analysis accuracy as inherent in the NLU approach, and KCD accuracy is derived from that. Note that we define essay quality here as the percentage of required points that are included in the student’s essay. It does not take into consideration any expressions of misconceptions or wrong information also present.

Now let’s consider how KCD precision, recall, and false alarm rate are related to analysis precision, recall, and false alarm rate as well as essay quality. As you see from the equations in Figure 1, KCD precision is the number of KCDs correctly given divided by the total number of KCDs given. KCDs are given whenever a required point is not identified. Thus, when analysis recall is low, a lot of points that are present in the student’s essay will not be identified. Thus, the corresponding elicitation KCDs will be incorrectly given. A needed KCD is not given whenever a point is incorrectly identified in a

Figure 2: This Table illustrates how KCD precision and recall vary with essay quality, keeping 0.90 analysis precision and 0.90 analysis recall.

<table>
<thead>
<tr>
<th>Essay Quality</th>
<th>KCD Precision</th>
<th>KCD Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>0.20</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>0.30</td>
<td>0.96</td>
<td>0.96</td>
</tr>
<tr>
<td>0.40</td>
<td>0.93</td>
<td>0.93</td>
</tr>
<tr>
<td>0.50</td>
<td>0.90</td>
<td>0.90</td>
</tr>
<tr>
<td>0.60</td>
<td>0.85</td>
<td>0.85</td>
</tr>
<tr>
<td>0.70</td>
<td>0.77</td>
<td>0.77</td>
</tr>
<tr>
<td>0.80</td>
<td>0.60</td>
<td>0.60</td>
</tr>
<tr>
<td>0.90</td>
<td>0.10</td>
<td>0.10</td>
</tr>
</tbody>
</table>
student’s essay. Thus, when analysis precision is low, many of the points that are identified are not actually present in the student’s essay. Thus, many of the KCDs that the student needs will not be given. KCD false alarm rate is the percentage of KCDs that should not be given but are. Thus, KCD false alarm rate is 1 - analysis recall.

KCD precision and recall are influenced both by analysis precision and recall as well as essay quality. To illustrate this point, let is consider the equations in Figure 1. Notice that A represents the percentage of time that a point is in the essay and identified as in the essay. Notice further that B represents the percentage of time that a point is not in the essay but identified as in the essay anyway. C represents the percentage of time that a case is in the essay and not identified as such. Finally, D represents the percentage of time that a point is missing from the essay and not identified as being in the essay.

When essay quality is high, D will be low. If essay quality is 70%, D can be no greater than 30%. Remember that KCD precision is $D / (C + D)$. If analysis recall is perfect, then C will be 0%, so KCD precision will be 100%. But if analysis recall is less than perfect, even a little bit less, then KCD precision goes down very fast. Since D is small, even if C is equally small, KCD precision is down to 50% already. KCD recall is determined by the relationship between B, the points not in the essay that were identified, and D, the points not in the essay that were not identified, in particular, $D / (B + D)$. Thus, if analysis precision is perfect, then B will be 0%, so KCD recall will be 100%. But if it is not perfect, then it will bring the KCD recall down fast, again because D is necessarily small.

The opposite is the case when essay quality is very low, lets say 20%. In this case, A and C must both be low. Thus, even if analysis recall is 0%, C will be no more than 20%. Let’s say that analysis recall and precision are around 86%. Whenever essay quality is low and analysis precision is reasonably high, B and C must both be small in comparison with D. Analysis precision and recall of 86% would have been disastrous for a high quality essay as we saw above. However, in this case, it would mean that A is 17%, B is 3%, C is 3%, and D is 77%. Thus, KCD precision and recall are both 96%. Therefore, the situation is quite different when essay quality is low. If analysis accuracy is reasonably high, KCD assignment accuracy will also be high.

Therefore, from this mathematical model we can see that as essay quality increases, it becomes much more difficult to do a good job at selecting appropriate KCDs for students. In fact, selecting appropriate KCDs for students with essay qualities of 70% or higher may well be completely out of our reach. In particular, even if analysis precision and recall are almost perfect, specifically at 90%, KCD precision, recall, and false alarm rate become unsatisfyingly low once essay quality is 70% (see Figure 2). Figure 3 shows the projection of KCD precision and recall from our current best analysis performance.

Thus, when we depend upon our analysis of student’s essays to tell us how to select interventions for students, we may be at a loss with respect to helping high end students who produce high quality essays. One might wonder whether it is necessary to worry about the high end students since it is often said that good students will learn even with a bad teacher. Nevertheless, no matter how good the students are, if we fail to present them with instruction on the topics that they lack, they will not have the opportunity.

Figure 3: This Table illustrates how KCD precision and recall vary with essay quality, keeping 0.79 analysis precision and 0.80 analysis recall, which is our current best performance with Why2.

<table>
<thead>
<tr>
<th>Essay Quality</th>
<th>KCD Precision</th>
<th>KCD Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>0.20</td>
<td>0.95</td>
<td>0.95</td>
</tr>
<tr>
<td>0.30</td>
<td>0.91</td>
<td>0.91</td>
</tr>
<tr>
<td>0.40</td>
<td>0.87</td>
<td>0.86</td>
</tr>
<tr>
<td>0.50</td>
<td>0.80</td>
<td>0.79</td>
</tr>
<tr>
<td>0.60</td>
<td>0.69</td>
<td>0.68</td>
</tr>
<tr>
<td>0.70</td>
<td>0.52</td>
<td>0.50</td>
</tr>
<tr>
<td>0.80</td>
<td>0.16</td>
<td>0.15</td>
</tr>
<tr>
<td>0.90</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
to learn those topics. Additionally, we have evidence in the form of expressions of frustration in our log files that students are unhappy when they receive help on many topics that they do not need help on. However, we do not have a definitive answer on what is the minimum acceptable level of KCD precision, recall, and false alarm rate. This is still under investigation.

Interestingly, a similar model has been proposed in the medical field. In that field, it has been determined based on this similar model that the Positive Predictive Value of a diagnostic test depends both on what is called the Specificity as well as the Prevalence of the disease in the population. What is called Sensitivity in that field is analogous to our analysis recall. What they call Specificity is analogous to our KCD precision. What they call Positive Predictive Value is analogous to our analysis precision. And their Negative predictive value is analogous to our KCD precision. Prevalence of disease could be thought of as analogous to our essay quality, or percentage of required points that are present in an essay. It is possible to verify that these pairs represent analogous quantities since both are defined in terms of true positives, true negatives, false positives, and false negatives. In their model, Positive predictive value equals (Sensitivity times Prevalence) divided by ((Sensitivity times Prevalence) + (1 - Specificity)(1 - Prevalence)). Although they are solving for a different quantity than what we are primarily interested in, their model is interesting in that the relationship between the four quantities in their model is the same as that in our analogous model. Thus, the problem of high prevalence (or analogously, high essay quality) is a common problem, and we may learn something from their solutions to their analogous problem. In the medical field, the problem is addressed by using multiple diagnostic tests. Thus, perhaps an analogous approach would prove beneficial here as well.

3 Using Prior Probabilities of Student Need

An evaluation of the WHY2 system is currently underway with undergraduate students having already taken one college level Physics course. We examined 95 student essays collected from 22 students interacting up until now with either one of the two different versions of the WHY2 system (WHY-Atlas and WHY-AutoTutor) to determine how often students type high quality essays. Specifically we looked at essays for the pumpkin problem introduced above. We determined that 21% of student essays (i.e., 20 out of 95 essays) are in the high competency range (i.e., missing 2 or fewer of the 6 required points for the pumpkin problem), however, only 10% of these essays (2 essays) are initial essays. Thus, for the majority of high quality student essays, both versions of WHY2 have access both the analysis of the essay as well as the dialogue history (i.e., record of which interventions have already been given). Thus, the problem is not as dire in practice as it would be if the system did not have access to the dialogue history. However, it is instructive to note that of the 20 high quality essays, 75% of them are at the 67% essay quality level (i.e., missing 2 out of the 6 required points). Thus, very few students are getting past high-mediocre performance at constructing quality explanations. Further analysis is required to determine with high confidence whether this is primarily due to lack of success at selecting appropriate interventions or to some other problem.

To address the problem of selecting an appropriate intervention for a high quality essay, we explored the possibility of selecting based on prior probability of student need rather than on an analysis of student essay quality. Our preliminary analysis demonstrates, however, that this is unlikely to be a solution to the problem. We first checked the probability of an essay including each of the required points over the whole corpus of 95 student essays. We then compared these probabilities with those of in the 20 high quality essays. For ease of reference we will refer to the points as A, B, C, E, J, and K respectively. Point K was the most likely to be included in a student essay. It appeared in 79% of the essays overall and 100% of the high quality essays. Point K occurred in 44% of essays overall and 85% of high quality essays. Point A occurred in 39% occurred in 27% of essays overall and 50% of high quality essays. Point J occurred in 28% of essays overall and 75% of high quality essays. Finally, point E occurred in 18% of essays overall and 30% of high quality essays. Note that the ranking by probability of occurrence for points is not the same for high quality essays as it is for essays overall. Thus, the first caveat we encountered in our analysis was that prior probabilities should be computed over just high quality essays, and not over the entire set of essays. This means that in order to get reliable statistics,

---

1 See www.ipathology.com/Disease%20Prevalence%20Notes.htm and www.epidemiolog.net/studymat/PredictiveValue.xls.
since high quality essays occur only 20% of the time, we need a tremendous amount of data in order to
collect a large enough sample of high quality essays to base the statistics on.

Even if we assume that the current set of 20 high quality essays is sufficient to obtain a probability
distribution that is good enough for our purposes, the numbers we get do not seem to help us do a
better job at selecting appropriate interventions, even for that same set of essays, because of the high
degree of individual differences in student performance on high quality essays. The three points least
likely to occur in high quality essays are E, B, and J. However, if the corresponding elicitation KCDs for
all three of these were given to students who wrote those high quality essays, roughly 2/3 of the KCDs
that would be given to students would be inappropriate. This is because the two points on average
that are missing from these high quality essays varies widely over the set of high quality essays. If only
the top 2 rather than the top 3 interventions were given instead, about half of the KCDs given would
be inappropriate, and students would also be missing about half of the KCDs that they should get.
Considering that these high quality essays are primarily at the 67% quality level, this prior probability
based approach puts us in the same bad position with respect to selecting appropriate interventions
than even our current best analysis approach (see Figure 3). Some combined approach, taking both the
analysis and prior probabilities might prove to be the most effective approach. A more likely approach
to be successful would be to compute some student specific model of prior probability of needing a
specific intervention, possibly based on the student’s pretest performance. Again, further analysis is
required to determine whether this is the case.

Another possibility would be to take a different approach altogether for high quality essays. Perhaps we
could take the production of a high quality essay as an indication that the student might learn better
with a slightly harder problem. If the answer to the slightly harder problem required much of the same
reasoning on a deep level as the easier problem, then the harder problem could be used to teach the
same concepts that the student showed evidence of needing instruction on with the easier problem. If
we take a very conservative approach to essay analysis, such as using LSA with a high threshold, we
can achieve a high precision with an accompanying low recall. If we took such an approach, we
would rarely miss opportunities to teach a student something that was needed, but we would teach on
lots of things that were not needed. However, if we teach these same concepts in the context of a harder
problem, it is less likely that the instruction will seem superfluous. Thus, such an analysis approach
could work well.

As an illustration, let us consider a harder version of the pumpkin problem. Remember that the original
pumpkin problem is as follows: “Suppose you are running in a straight line at constant speed. You
throw a pumpkin straight up. Where will it land? Explain.” In the original pumpkin problem, the
man throws the pumpkin straight up. But consider what happens when the man exerts both a vertical
and a horizontal force on the pumpkin during the toss. In the original problem, the man’s horizontal
velocity remains equal to the pumpkin’s horizontal velocity. Thus, their horizontal displacements from
the point of release will always be equal. And when the pumpkin lands, it will land in the man’s hands.
But in the harder case, the student must reason that during the toss, the pumpkin will accelerate in the
horizontal direction. Thus, although both the man and the pumpkin will move at a constant horizontal
velocity after the toss, the pumpkin’s velocity will be greater. Thus, it’s horizontal displacement from
the origin will always be greater than that of the man. And it will land in front of the man.

If the student produces a high quality essay in response to the easy version of the problem, the system
could reanalyze the student’s essay with a very stringent, conservative approach. Then, we could tutor
on all of the required points analyzed as missing, but in the context of the harder problem. For example,
the system could say, “Since your essay looks very good, let me give you a harder scenario...” Then, for
example, if the analysis of the essay for the easy problem indicated that the student did not articulate
the fact that the horizontal displacement of the man and pumpkin would always be the same, the
system could ask, “In this new scenario, what do you think will be the relationship between the man’s
horizontal displacement and that of the pumpkin at all times after the release?” If the student can
answer this question correctly, the system can assume that the student did not need to be tutored on
the corresponding topic in the easier scenario since the same reasoning is required to obtain a correct
answer here. But if the student gets the answer wrong, the system could begin by tutoring on the
topic in the easier scenario as follows: “Let’s think of how this relates to the previous question then.
To begin with, in the case where the man exerted only a vertical force on the pumpkin, what was the

98
relationship between their horizontal displacements at all times after the release?" This way, we can verify that the student needs tutoring on a topic without implying that the student neglected to include the corresponding information in the student’s essay. In the case where the student did include this information in the essay, the question can be interpreted as simply drawing the student’s attention to something that was articulated earlier. Thus, this answer offers a potential solution both to the problem of neglecting to tutor on topics that are needed and to avoiding frustrating students by tutoring them on topics that they have already demonstrated competence at.

4 Conclusions and Current Directions

In this paper we have explored the issue of selecting appropriate interventions for students who start out with a high initial competency. We present a mathematical model that makes it possible to project success rate at selecting appropriate interventions based on accuracy at analyzing student performance at a task. This model demonstrates that for high initial competency students, selecting interventions based on any sort of isolated analysis of student performance is doomed to failure even with very high accuracy analysis. We explore an alternative approach to selecting interventions within the context of the Wity2 conceptual physics tutoring system (VanLehn et al., 2002) based on prior probabilities of student need. Our analysis demonstrates that within this domain, individual differences between high competency students makes this type of approach by itself equally unsuccessful. We are continuing to collect transcripts of students interacting with the Wity2 system and plan to explore the possibility of combining predictions based on prior probability of student need with predictions based on analysis as well as computing student specific prior probabilities based on performance on the pretest.

5 Acknowledgments

The authors would like to thank the rest of the Natural Language Tutoring group for their collaboration. This research is supported by the Office of Naval Research, Cognitive and Neural Sciences Division MURI Grant N00014-00-1-0600 and NSF Grant 9720359 to CIRCLE, a center for research on intelligent tutoring.

References
