Simple but effective feedback generation to tutor abstract problem solving

Xin Lu, Barbara Di Eugenio, Stellan Ohlsson and Davide Fossati
University of Illinois at Chicago
Chicago, IL 60607, USA
xinlu@northsideinc.com
bdieugen,stellan,dfossa1@uic.edu

Abstract

To generate natural language feedback for an intelligent tutoring system, we developed a simple planning model with a distinguishing feature: its plan operators are derived automatically, on the basis of the association rules mined from our tutorial dialog corpus. Automatically mined rules are also used for realization. We evaluated 5 different versions of a system that tutors on an abstract sequence learning task. The version that uses our planning framework is significantly more effective than the other four versions. We compared this version to the human tutors we employed in our tutorial dialogs, with intriguing results.

1 Introduction

Intelligent Tutoring Systems (ITSs) are software systems that provide individualized instruction, like human tutors do in one-on-one tutoring sessions. Whereas ITSs have been shown to be effective in engendering learning gains, they still are not equivalent to human tutors. Hence, many researchers are exploring Natural Language (NL) as the key to bridging the gap between human tutors and current ITSs. A few results are now available, that show that ITS with relatively sophisticated language interfaces are more effective than some other competitive condition (Graesser et al., 2004; Litman et al., 2006; Evens and Michael, 2006; Kumar et al., 2006; VanLehn et al., 2007; Di Eugenio et al., 2008). Ascertaining which specific features of the NL interaction are responsible for learning still remains an open research question.

In our experiments, we contrasted the richness with which human tutors respond to student actions with poorer forms of providing feedback, e.g. only graphical. Our study starts exploring the role that *positive* feedback plays in tutoring and in ITSs. While it has long been observed that most tutors tend to avoid direct negative feedback, e.g. (Fox, 1993; Moore et al., 2004), ITSs mostly provide negative feedback, as they react to student errors.

In this paper, we will first briefly describe our tutorial dialog collection. We will then present the planning architecture that underlies our feedback generator. Even if our ITS does not currently allow for student input, our generation architecture is inspired by state-of-the art tutorial dialog management (Freedman, 2000; Jordan et al., 2001; Zinn et al., 2002). One limitation of these approaches is that plan operators are difficult to maintain and extend, partly because they are manually defined and tuned. Crucially, our plan operators are automatically derived via the association rules mined from our corpus. Finally, we will devote a substantial amount of space to evaluation. Our work is among the first to show not only that a more sophisticated language interface results in more learning, but that it favorably compares with human tutors. Full details on our work can be found in (Lu, 2007).

2 Task and curriculum

Our domain concerns extrapolating letter patterns, such as inferring EFMGHM, given the pattern ABMCDM and the new initial letter E. This task is used in cognitive science to investigate human information processing (Kotovsky and Simon, 1973; Reed and Johnson, 1994; Nokes and Ohlsson, 2005). The curriculum we designed consists of 13 patterns of increasing length and difficulty; it was used unchanged in both our human data collection and ITS experiments. The curriculum is followed by two

Tutor Moves
Answer student's questions
Evaluate student's actions
Summarize what done so far
Prompt student into activity
Diagnose what student is doing
Instruct
Demonstrate how to solve (portions of) problem
Support – Encourage student
Conversation – Acknowledgments, small talk
Student Moves
Question
Explain what student said or did
Reflect – Evaluate own understanding
Answer tutor's question
Action Response – Perform non-linguistic action (e.g. write letter down)
Complete tutor's utterance
Conversation – Acknowledgments, small talk

Table 1: Tutor and student moves

post-test problems, each 15 letter long: subjects (all from the psychology subject pool) have n trials to extrapolate each pattern, always starting from a different letter ($n = 6$ in the human conditions, and $n = 10$ in the ITS conditions). While the earlier example was kept simple for illustrative purposes, our patterns become very complex. Starting e.g. from the letter L, we invite the reader to extrapolate problem 9 in our curriculum: *BDDFFFCCEEGGGC*, or the second test problem: *ACZDBYYDFXGEWWGI*.[1]

3 Human dialogs

Three tutors – one expert, one novice, and one (the *lecturer*) experienced in teaching, but not in one-on-one tutoring – were videotaped as they interacted with 11 subjects each.[2] A repeated measures ANOVA, followed by post-hoc tests, revealed that students with the expert tutor performed significantly better than the students with the other two tutors on both test problems ($p < 0.05$ in both cases).

36 dialog excerpts were transcribed, taken from 18 different subjects (6 per tutor), for a total of about 2600 tutor utterances and 660 student utterances (transcription guidelines were taken from (MacWhinney, 2000)). For each subject, these two dialog excerpts cover the whole interaction with the tutor on one easy and one difficult problem (# 2 and 9 respectively). 2 groups of 2 coders each, annotated half of the transcripts each, with dialogue moves. Our move inventory comprises 9 tutor moves and 7 student moves, as listed in Table 1.[3] Table 2 presents an annotated fragment from one of the dialogues with the expert tutor. Kappa measures for intercoder agreement had values in the following ranges, according to the scale in (Rietveld and van Hout, 1993): for tutor moves, from moderate (0.4 for *Support*) to excellent (0.82 for *Prompt*); for student moves, from substantial (0.64 for *Explanation*) to excellent (0.82 for *Question*, 0.97 for *ActionResponse*). Whereas some of these Kappa measures are lower than what we had strived for, we decided to nonetheless use the move inventory in its entirety, after the coders reconciled their codings. In fact, our ultimate evaluation measure concerns learning, and indeed the ITS version that uses that entire move inventory engenders the most learning. Please see (Lu et al., 2007) for a detailed analysis of these dialogs and for a discussion of differences among the tutors in terms of tutor and student moves.

The transcripts were further annotated by one coder for tutor attitude (whether the tutor agrees with the student's response – *positive*, *negative*, *neutral*), for correctness of student move and for student confidence (*positive*, *negative*, *neutral*). Student hesitation time (*long*, *medium*, *short*) was estimated by the transcribers. Additionally, we annotated for the problem features under discussion. Of the 8 possible relationships between letters, most relevant to the examples discussed in this paper are *forward, backward, progression* and *marker*. E.g. in ABMCDM, M functions as chunk *marker*, and the sequence moves *forward* by one step, both *within* one chunk and *across* chunks. *Within* and *across* are two out of 4 *relationship scopes*, which encode the coverage of a particular relationship within the sequence.

[1] The solutions are, respectively: *LNNPPPMMOOQQQM*, and *LNZOMYYOQXRPWWRT*.

[2] One goal of ours was to ascertain whether expert tutors are indeed more effective than non-expert tutors, not at all a foregone conclusion since very few studies have contrasted expert and non expert tutors, e.g. (Glass et al., 1999).

[3] There is no explicit tutor *question* move because we focus on the goal of a tutor's question, either *prompt* or *diagnose*.

Line	Utterances	Annotation
38	**Tutor:** how'd you actually get the *n* in the first place?	Diagnose
39	**Student:** from here I count from *c* to *g* and then just from *n* to *r*.	Answer
40	**Tutor:** okay so do the *c* to *g*.	Prompt
41	**Tutor:** do it out loud so I can hear you do it.	Prompt
42	**Student:** *c d e f*.	Explain
43	**Student:** so it's three spaces.	Answer
44	**Tutor:** okay so it's three spaces in between.	Summarize

Table 2: An annotated fragment from a dialogue with the expert tutor

4 Learning rules to provide feedback

Once the corpus was annotated, we mined the expert tutor portion via Classification based on Associations (CBA) (Liu et al., 1998). CBA generates understandable rules and has been effectively applied to various domains. CBA finds all rules that exist in the data, which is especially important for small training sets such as ours.

To modularize what the rules should learn, we decomposed *what the tutor should do* into two components pertaining to content: letter relationship and relationship scope; and two components pertaining to how to deliver that content: tutor move and tutor attitude. Hence, we derived 4 sets of tutorial rules. Features used in the rules are those annotated on the tutoring dialogs, plus the student's Knowledge State (KS) on each type of letter relationship *rel*, computed as follows:

$$KS(rel) = \lfloor \frac{p \times 0.5 + w}{t} \times 5 \rfloor \qquad (1)$$

p is the number of partially correct student inputs, w is the number of wrong student inputs and t is the total number of student inputs ("inputs" here are only those relevant to the relationship rel, as logged by the ITS from the beginning of the session). $KS(rel)$ ranges from 0 to 5. The higher the value, the worse the performance on rel. The scale of 5 was chosen to result in just enough values for KS(rel) to be useful for classification.

We ran experiments with different lengths of dialog history, but using only the last utterance gave us the best results. Three of the four rule sets have accuracies between 88 and 90% (results are based on 6-way cross-validation, and the cardinality of each set of rules is in the low hundreds.). Whereas the tutor move rule set only has 57% accuracy, as for some of the low Kappa values mentioned earlier, our ultimate evaluation measure is that the NL feedback based on these rules does improve learning.

relation-marker = No, relation-forward = Yes, student-move = ActionResponse
→ **relation-forward = Yes**
(Confidence = 100%, Support = 4.396%)

correctness = wrong, scope-within = No, KS(backward) = 0, relation-forward = Yes
→ **tutor-move = Summarize**
(Confidence = 100%, Support = 6.983%)

correctness = wrong, relation-forward = Yes, KS(forward) = 1, hesitation = no
→ **tutor-attitude = negative**
(Confidence = 100%, Support = 1.130%)

Figure 1: Example Tutorial Rules

Figure 1 shows three example rules, for choosing relationship, move and attitude respectively – we'll discuss two of them. The first rule predicts that the ITS will continue focusing on the *forward* relation, if it was focusing on *forward* and not on *marker*, and the student just input something. The second rule chooses the *summarize* move if the student made a mistake, the ITS was focusing on *forward* but not on relationships within chunks, and the student showed perfect knowledge of *backward*.

Two strength measurements are associated with each rule $X \rightarrow y$. A rule holds with confidence *conf* if *conf*% of cases that contain X also contain y; and with support *sup* if *sup*% of cases contain X or y. Rules are ordered, with confidence having precedence over support. Ties over confidence are solved via support; any remaining ties are solved according to the order rules were generated.

For each Tut-Move-Rule $TMR_{i,k}$ whose Left-Hand Side LHS matches IS_i do:

1. **Create and Populate New Plan** $p_{i,k}$:
   (a) *preconditions* = IS_i; *actions* = tutor move from $TMR_{i,k}$; *strength* = confidence and support from $TMR_{i,k}$
   (b) Fill remaining slots in $p_{i,k}$:
      i. *contents* = relationship ∪ scope (from highest ranked rules that match IS_i from relationship and scope rule sets);
      ii. *modifiers* = attitude (from highest ranked rule that matches IS_i from tutor attitude rule set)
2. **Augment Plan:** do the following *n* times :
   (a) make copy of IS_i and name it IS_{i+1};
   (b) change *agent* to "tutor";
   (c) change corresponding elements in IS_{i+1} to move, attitude, letter relationship and scope from $p_{i,k}$;
   (d) from the two rule sets for tutor move and tutor attitude, retrieve highest ranked rules that match IS_{i+1}, $TMR_{i+1,j}$ and $TAR_{i+1,j}$
   (e) add to *actions* tutor move from $TMR_{i+1,j}$; add to *modifiers* tutor attitude from $TAR_{i+1,j}$

Figure 2: Plan generation

5 From rules to plans

For our task of extrapolating abstract sequences, we built a model-tracing ITS by means of the Tutoring Development Kit (TDK) (Koedinger et al., 2003). Model-tracing ITSs codify cognitive skills via production rules. The student's solution is monitored by rules that fire according to the underlying model. When the student steps differ from that model, an error is recognized. A portion of the student interface of the ITS is shown in Figure 4a. It mainly includes two rows, one showing the *Example Pattern*, the other for the student to input the *New Pattern* extrapolated starting with the letter in the first cell.

In model-tracing ITSs, production rules provide the capability to generate simple template-based messages. We developed a more sophisticated NL feedback generator consisting of three major modules: update, planning and feedback realization.

The update module maintains the context, represented by the Information State (IS) (Larsson and Traum, 2000), which captures the overall dialog context and interfaces with external knowledge sources (e.g., curriculum, tutorial rules) and the production rule system. As the student performs a new action, the IS is updated. The planning module generates or revises the system plan and selects the next tutoring move based on the newly updated IS. At last the feedback realization module transforms this move into NL feedback.

The planning module consists of three components, plan generation, plan selection and plan monitoring. A plan includes an ordered collection of tutoring moves meant to help the student correctly fill a single cell. The structure of our plans is shown in Figure 3.

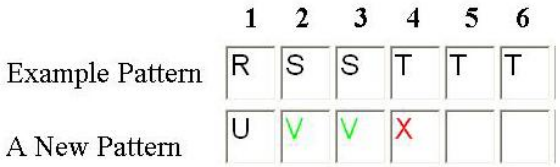
Plan generation generates a plan set which contains one plan for each tutor move rule that matches the current IS_i. Each of these plans is augmented at plan generation time by 'simulating' the next IS_{i+1} that would result if the move is executed but its effects are not achieved. The algorithm is sketched in Figure 2. The planner iterates through the *tutor move* rule set.[4] Recall that our four rule sets are totally ordered. Also, note that each rule set contains a default rule that fires when no rule matches the current IS_i. In Step 1b, at every iteration *only* the rules that have not been checked yet from those three rule sets are considered. In Step 2, n is set to 3, i.e., each plan contains 3 additional moves and corresponding attitudes, which will provide hints when no response from the student occurs. Three hints plus one original move makes 4, which is the average number of moves in one turn of the expert tutor.

An example plan is shown in Figure 3. It is generated in reaction to the mistake in Figure 4a, and by

[4]Since there is no language input, rules which include student moves other than *ActionResponse* in their LHS will never be activated. Additionally, we recast tutor *answers* as *confirm* moves, since students cannot ask questions.

Preconditions	(same as the IS in Figure 4b)
Effects	student's input = *W*
Contents	relationship = *forward* scope = *across*
Actions	*summarize, evaluate, prompt, summarize*
Modifiers	*negative, negative, neutral, neutral*
Strength	conf = *100%*, sup = *6.983%*

Figure 3: An Example Plan



(a) A Student Action in Problem 4

1. **Agent**: *student* (producer of current move);
2. **Relationships**: *forward*, *progress in length*
3. **Scopes**: *across* (for 'forward'), *within* (for 'progress in length');
4. **Agent's move**: *action response*;
5. **Agent's attitude**: *positive* (since student shows no hesitation before inputting letter);
6. **Correctness**: *wrong* (correct letter is *W*);
7. **Student's input**: *X*;
8. **Student's selection**: 4th cell in *New Pattern* row;
9. **Hesitation time**: *no*;
10. **Student's knowledge state (KS)**: *1* (on "forward"), *3* (on "progress in length").

(b) The corresponding IS

Figure 4: A snapshot of an ITS interaction

firing, among others, the rules in Figure 1. The IS in Figure 4b reflects some of the history of this interaction (in the slots *Relationships, Scopes* and *KS*), and as such corresponds to the situation depicted in Figure 4a in a specific context (this plan was generated in one of our user experiments).

The plan selection component retrieves the highest ranked plan in the newly generated plan set, selects a template for each tutoring move in its "Actions" slot and puts each tutoring move onto the dialog move (DM) stack. Earlier we mentioned that rules are totally ordered according to confidence, then support and finally rule generation order. When a plan set contains more than one plan, plans are also totally ordered, since they inherit strength measurements from the rule that engenders the first tutor move in the *Actions* slot.

After the student receives the message which realizes the top tutoring move in the DM stack, plan monitoring checks whether its intended effects have been obtained. If the effects have not been obtained, and the student's input is unchanged, the next move from the DM stack will be executed to provide the student with hint messages until either the student's input changes or the DM stack becomes empty. If the DM stack becomes empty, the next plan is selected from the original plan set and the tutoring moves within that plan are pushed onto the DM stack. Whenever the student's input changes, or after every plan in the plan set has been selected, control returns to plan generation.

The realization module. A tutor move is pushed onto the DM stack by plan selection together with a template to realize it. 50 templates were written manually upon inspection of the expert tutor dialogs. Since several templates can realize each tutor move, we used CBA to learn rules to choose among templates. Features used to learn when to use each template also include the tutor attitude. For the first *Summarize* move in the plan in Figure 3, given the IS in Figure 4b, the rule in Fig. 5 will fire (tutor attitude does not affect this specific rule). As a result, the following feedback message is generated: "From *V* to *X*, you are going forward 2 in the alphabet."

6 Evaluation

To demonstrate the utility of our feedback generator, we developed five different versions of our ITS, named according to how feedback is generated:

1. **No feedback**: The ITS only provides the basic interface, so that subjects can practice solving the 13 problems in the curriculum, but does not provide any kind of feedback.
2. **Color only**: The ITS provides graphic feedback by turning the input green if it is correct or red if it is wrong.
3. **Negative**: In addition to the color feedback, the

scope-within = No, relation-marker = No, relation-forward = Yes, move= Summarize → **template = TPL11**
[where TPL11: From "<reference-pattern>" to "<input>", you are going <input-relation> <input-number> in the alphabet.]

Figure 5: Example Realization Rule

ITS provides feedback messages when the input is wrong.

4. **Positive**: In addition to the color feedback, the ITS provides feedback messages when the input is correct.

5. **Model**: In addition to the color feedback, the ITS provides feedback messages generated by the feedback generator just described.

Feedback is given for each input letter. Positive and negative verbal feedback messages are given out whenever the student's input is correct or incorrect, respectively. Positive feedback messages confirm the correct input and explain the relationships which this input is involved in. Negative feedback messages flag the incorrect input and deliver hints. The feedback messages for the "negative" and "positive" versions were developed earlier in the project, to avoid repetitions and inspired by the expert tutor's language but before we performed any annotation and mining. They are directly generated by TDK production rules.

Although in reality positive and negative feedback are both present in tutoring sessions, one study for the letter pattern task shows that positive/negative feedback, given independently, perform different functions (Corrigan-Halpern and Ohlsson, 2002). In addition, our *negative* condition is meant to embody the "classical" model-tracing ITS, that only reacts to student errors. Hence, in our experiments, we elected to keep these two types of feedback separate, other than in the "model" version of the ITS.

To evaluate the five versions of the ITS, we ran a between-subjects study in which each group of subjects interacted with one version of the system. A group of control subjects took the post-test with no training at all but only read a short description of the domain.[5] Subjects were trained to solve the same 13 problems in the curriculum that were used in the human tutoring condition. They also did the same post-test (2 problems, each pattern 15 letters long). For each post-test problem, each subject had 10 trials, where each trial started with a new letter.

Condition		Score		
		Prob 1	Prob 2	Total
0	Control	36.50	32.84	69.34
1	No feedback	58.21	75.27	133.48
2	Color only	68.32	66.30	134.62
3	Negative	70.33	66.06	141.83
4	Positive	75.06	79.00	154.06
5	Model	91.95	101.76	193.71

Table 3: Average Post-test Scores of the ITS

6.1 Results

Table 3 reports the average post-test scores of the six groups of subjects, corresponding to the five versions of the ITS and the control condition. Performance on each problem is measured by the number of correct letters out of a total of 150 letters (15 letters by 10 trials); hence, cumulative post-test score, is the number of correct letters out of 300 possible.

A note before we proceed. In ITS research it is common to administer the same test before (pre-test) and after treatment (post-test), but we only have the post-test. The pre/post-test paradigm is used for two reasons. First, for evaluation proper, to gauge learning gains. Second, to verify that the groups have the same level of pre-tutoring ability, as shown when the pre-tests of the different groups are statistically indistinguishable, and hence, that they can be rightly compared. Even without a pre-test we can assess this. An ANOVA on "time spent on the first 3 problems" revealed no significant differences across the different groups. Since time spent on the first 3 problems is highly correlated with post-test score (multiple regression, $p < 0.03$), this provides indirect evidence that all subjects before treatment have equivalent ability for this task. Hence, we can trust that our evaluation, in terms of absolute scores, does reveal differences between conditions.

Our main findings are based on one-way ANOVAs, followed by Tukey post-hoc tests:

[5] The number of subjects in each condition varies from 32 to 38. Groups differ in size because of technical problems.

- A **main effect** of ITS ($p \leq 0.05$). Subjects who interacted with any version of the ITS had significantly higher total post-test scores than subjects in the control condition.

- A **main effect** of modeled feedback ($p < 0.05$). Subjects who interacted with the "model" version of the ITS had significantly higher total post-test scores than control subjects, and subjects with any other version of the ITS.

- No other effects. Subjects trained by the three versions "color only", "negative", "positive", did not have significantly higher total post-test scores than subjects with the "no feedback" version; neither did subjects trained by the two versions "negative", "positive", wrt subjects with the "color-only" version.

If we examine individual problems, the same pattern of results hold, other than, interestingly, the *model* and *positive* versions are not significantly different any more. As customary, we also analyze *effect sizes* , i.e., how much *more* subjects learn with the 'model' ITS in comparison to the other conditions. On the Y axis, Figure 6 shows *Cohen's d*, a common measure of effect size. Each point represents the difference between the means of the scores in the 'model' ITS and in one of the other condition, divided by the standard deviation of either condition. According to (Cohen, 1988), the effect sizes shown in Figure 6 are large as concerns the comparison with the "no feedback", "color only" and "negative" conditions, and moderate as concerns the "positive" condition.[6]

ITSs and Human Tutors. After we established that, at least cumulatively, the "model" ITS is more effective than the other ITSs, we wanted to assess how well the "model" ITS fares in comparison to the expert tutor it is modeled on. Since in the human data each post-test problem consists of only 6 trials, the first 6 trials per problem from the ITSs are used to run this comparison, for a maximum total score of 180 (15 letters by 6 trials, by 2 problems). Figure 7 shows the overall post-test performance of all 9 conditions. The error bars in the figure represent the standard deviations.

[6] A very large effect size with respect to control is not shown in Figure 6.

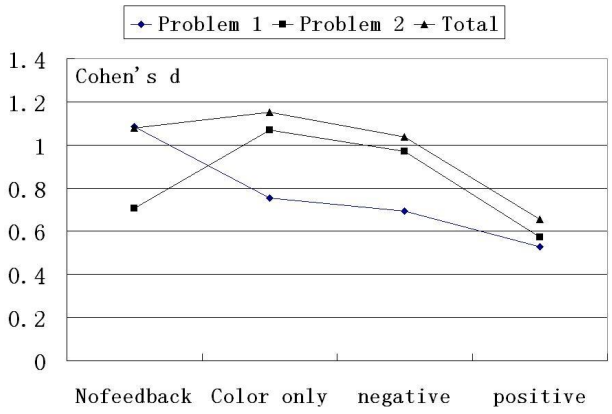


Figure 6: Effect sizes: how much more subjects learn with the "model" ITS

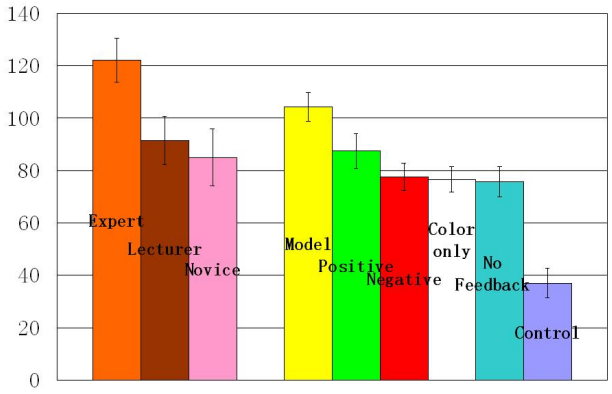


Figure 7: Post-test performance – all conditions

Paired t-tests between the model ITS and each of the human tutors show that:[7]

- on problem 1, the 'model' ITS is indistinguishable from the expert tutor, and is significantly better than the novice and the lecturer ($p = 0.05$ and $p = 0.039$ respectively);

- on problem 2, the model ITS is significantly worse than the expert tutor ($p = 0.020$), and is not different from the other two tutors;

- cumulatively, there are no significant differences between the 'model' ITS and any of the three human tutors.

[7] A 9-way ANOVA among all conditions is not appropriate, since in a sense we have two "super conditions", human and ITS. It is better to compare the 'model' ITS to each of the human tutors via t-tests, as a follow-up to the differences highlighted by the separate analyses on the two "super conditions".

7 Discussion and Conclusions

Our results add to the growing body of evidence that language feedback engenders more learning not only than simple practice, but also, than less sophisticated language feedback. Importantly, our 'model' ITS appears intriguingly close to our expert tutor in effectiveness: on post-test problem 1, it is as effective as the expert tutor himself, and significantly better than the other two tutors, as the expert tutor is. It appears our 'model' ITS does capture at least some features of successful tutoring.

As concerns the specific language the ITS generates, we compared different ways of providing verbal feedback. A subject receives both positive and negative verbal feedback when interacting with the "model" version, while a subject receives only one type of verbal feedback when interacting with the "positive" and "negative" versions (recall that in all these versions including the "model" ITS the red/green graphical feedback is provided on every input). While we cannot draw definite conclusions regarding the functions of positive and negative feedback, since the 'model' version provides other tutorial moves beyond positive / negative feedback, we have suggestive evidence that negative feedback by itself is not as effective. Additionally, positive feedback appears to play an important role. First, the 'model' and the 'positive' versions are statistically equivalent when we analyze performance on individual problems. Further, in the "model" version, the ratio of positive to negative messages turns out to be 9 to 1. In our tutoring dialogs, positive feedback still outnumbers negative feedback, but by a lower margin, 4 to 1. We hypothesize that conveying a positive attitude in an ITS is perhaps even more important than in human tutoring since a human has many more ways of conveying subtle shades of approval and disapproval.

From the NLG point of view, we have presented a simple generation architecture that turns out to be rather effective. Among its clear limitations are the lack of hierarchical planning, and the fact that different components of a plan are generated independently one from the other. Among its strengths are that the plan operators are derived automatically via the rules we mined, both for content planning and, partly, for realization.

It clearly remains to be seen whether our NLG framework can easily be ported to other domains – the issue is not domain dependence, but whether a more complex domain will require some form of hierarchical planning. We are now working in the domain of Computer Science data structures and algorithms, where we continue exploring the role of positive feedback. We collected data with two tutors in that domain, and there again, we found that in the human data positive feedback occurs about 8 times more often than negative feedback. We are now annotating the data to mine it as we did here, and developing the core ITS.

Acknowledgments

This work was supported by awards N00014-00-1-0640 and N00014-07-1-0040 from the Office of Naval Research, by Campus Research Board S02 and S03 awards from the University of Illinois at Chicago, and in part, by awards IIS 0133123 and ALT 0536968 from the National Science Foundation.

References

J. Cohen. 1988. *Statistical power analysis for the behavioral sciences (2nd ed.)*. Hillsdale, NJ: Lawrence Earlbaum Associates.

Andrew Corrigan-Halpern and Stellan Ohlsson. 2002. Feedback effects in the acquisition of a hierarchical skill. In *Proceedings of the 24th Annual Conference of the Cognitive Science Society*.

Barbara Di Eugenio, Davide Fossati, Susan Haller, Dan Yu, and Michael Glass. 2008. Be brief, and they shall learn: Generating concise language feedback for a computer tutor. *International Journal of AI in Education*, 18(4). To appear.

Martha W. Evens and Joel A. Michael. 2006. *One-on-one Tutoring by Humans and Machines*. Mahwah, NJ: Lawrence Erlbaum Associates.

Barbara A. Fox. 1993. *The Human Tutorial Dialogue Project: Issues in the design of instructional systems*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Reva K. Freedman. 2000. Plan-based dialogue management in a physics tutor. In *Proceedings of the Sixth Applied Natural Language Conference*, Seattle, WA, May.

Michael Glass, Jung Hee Kim, Martha W. Evens, Joel A. Michael, and Allen A. Rovick. 1999. Novice vs. expert tutors: A comparison of style. In *MAICS-99, Proceedings of the Tenth Midwest AI and Cognitive Science Conference*, pages 43–49, Bloomington, IN.

Arthur C. Graesser, S. Lu, G.T. Jackson, H. Mitchell, M. Ventura, A. Olney, and M.M. Louwerse. 2004. AutoTutor: A tutor with dialogue in natural language. *Behavioral Research Methods, Instruments, and Computers*, 36:180–193.

Pamela Jordan, Carolyn Penstein Rosé, and Kurt VanLehn. 2001. Tools for authoring tutorial dialogue knowledge. In *Proceedings of AI in Education 2001 Conference*.

Kenneth R. Koedinger, Vincent Aleven, and Neil T. Heffernan. 2003. Toward a rapid development environment for cognitive tutors. In *12th Annual Conference on Behavior Representation in Modeling and Simulation*.

K. Kotovsky and H. Simon. 1973. Empirical tests of a theory of human acquisition of information-processing analysis. *British Journal of Psychology*, 61:243–257.

Rohit Kumar, Carolyn P. Rosé, Vincent Aleven, Ana Iglesias, and Allen Robinson. 2006. Evaluating the Effectiveness of Tutorial Dialogue Instruction in an Exploratory Learning Context. In *Proceedings of the Seventh International Conference on Intelligent Tutoring Systems*, Jhongli, Taiwan, June.

Staffan Larsson and David R. Traum. 2000. Information state and dialogue management in the trindi dialogue move engine toolkit. *Natural Language Engineering*, 6(3-4):323–340.

Diane J. Litman, Carolyn P. Rosé, Kate Forbes-Riley, Kurt VanLehn, Dumisizwe Bhembe, and Scott Silliman. 2006. Spoken versus typed human and computer dialogue tutoring. *International Journal of Artificial Intelligence in Education*, 16:145–170.

Bing Liu, Wynne Hsu, and Yiming Ma. 1998. Integrating classification and association rule mining. In *Knowledge Discovery and Data Mining*, pages 80–86, New York, August.

Xin Lu, Barbara Di Eugenio, Trina Kershaw, Stellan Ohlsson, and Andrew Corrigan-Halpern. 2007. Expert vs. non-expert tutoring: Dialogue moves, interaction patterns and multi-utterance turns. In *CICLING07, Proceedings of the 8th International Conference on Intelligent Text Processing and Computational Linguistics*, pages 456–467. Best Student Paper Award.

Xin Lu. 2007. *Expert tutoring and natural language feedback in intelligent tutoring systems*. Ph.D. thesis, University of Illinois - Chicago.

Brian MacWhinney. 2000. *The CHILDES project. Tools for analyzing talk: Transcription Format and Programs*, volume 1. Lawrence Erlbaum, Mahwah, NJ, third edition.

Johanna D. Moore, Kaska Porayska-Pomsta, Sebastian Varges, and Claus Zinn. 2004. Generating Tutorial Feedback with Affect. In *FLAIRS04, Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*.

Timothy J. Nokes and Stellan Ohlsson. 2005. Comparing multiple paths to mastery: What is learned? *Cognitive Science*, 29:769–796.

Jonathan Reed and Peder Johnson. 1994. Assessing implicit learning with indirect tests: Determining what is learned about sequence structure. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 20(3):585–594.

Toni Rietveld and Roeland van Hout. 1993. *Statistical Techniques for the Study of Language and Language Behaviour*. Mouton de Gruyter, Berlin - New York.

Kurt VanLehn, Arthur C. Graesser, G. Tanner Jackson, Pamela W. Jordan, Andrew Olney, and Carolyn P. Rosé. 2007. When are tutorial dialogues more effective than reading? *Cognitive Science*, 31(1):3–62.

Claus Zinn, Johanna D. Moore, and Mark G. Core. 2002. A 3-tier planning architecture for managing tutorial dialogue. In *ITS 2002, 6th. Intl. Conference on Intelligent Tutoring Systems*, pages 574–584, Biarritz, France.