Proceedings of the Twentieth Annual Conference of the Cognitive Science Society, Madison, WI, 1998.

Using Rule Induction to Assist in Rule Construction for a Natural-Language Based Intelligent Tutoring System

Reva Freedman[†] (freedrk+@pitt.edu)
Learning Research and Development Center
University of Pittsburgh
3939 O'Hara Street #819
Pittsburgh, PA 15260

Yujian Zhou (zhouyuj@charlie.cns.iit.edu)
Michael Glass (glass@charlie.cns.iit.edu)
Jung Hee Kim (janice@steve.csam.iit.edu)
Martha W. Evens (csevens@minna.cns.iit.edu)
Department of CSAM
Illinois Institute of Technology
10 W. 31st Street 236–SB
Chicago, IL 60616

Abstract

We used Quinlan's C4.5 machine learning algorithm to analyze tutorial dialogues as part of the derivation of planning rules for CIRCSIM-Tutor v. 3, a natural-language based intelligent tutoring system. We annotated a corpus of tutoring dialogues with an SGML-based representation of tutorial goals in order to make mechanical processing possible. We looked for rules of the form "under what conditions is goal *x* implemented with plan *y*?". We discovered rules for high-level planning of the tutoring session and dynamic modification of the tutorial agenda. At a lower level of planning, we looked at rules for generating sections of the tutor's utterance. The use of the rule induction algorithm has helped us discover which knowledge available to the planner is significant in making these decisions, as well as producing some decision trees we can actually use in CIRCSIM-Tutor.

Introduction

CIRCSIM-Tutor (v. 3) is a conversational intelligent tutoring system (ITS) which uses natural language for both input and output. The core of CIRCSIM-Tutor is an integrated planning and execution module that uses STRIPS-style planning operators. Here is an example of an operator:

```
(def-operator neural-DLR
  :action ()
  :goal (did-neural-DLR ?vbl)
  :precond ((is-neural ?vbl) ...)
  :recipe  ((did-tutor mechanism ?vbl)
            (did-tutor DR-info ?vbl)
            (did-tutor vbl-value ?vbl))
  :filter ...
  :add    ...
  :del    ...)
```

The goal of the planner is to generate a conversation resulting in the student knowing some desired concepts. Each goal is satisfied by replacing it on the planner's agenda by the prerequisites and recipe steps of an operator which achieves it. When a primitive operator, i.e. an individual speech act request such as *elicit* or *inform*, is reached, it is added to a buffer whose contents will eventually form the tutor's next turn.

When a reply is required from the student, the planner is temporarily suspended and text is generated from the operators in the buffer. After the student replies, the planner resumes operation. The student's reply is one of the factors which is then taken into account in order to generate the following turn.

The question then becomes: where does the operator library come from? In early text generation systems, the operators were obtained from intuition and introspection. More recently, researchers have used a variety of informal means to abstract plan operators from naturalistic corpora (Reiter & Dale, 1997). We have profitably used all of these methods. Recently we have become interested in more formal methods of identifying operators which would provide a good model of our corpus. In this paper we report on some experiments we have done using the decision tree induction program C4.5 (Quinlan, 1993) to identify potential operators in our corpus of human tutorial transcripts.

Rather than obtain rules which could be transferred automatically to an automated system, our goal was to obtain ideas for better modeling our human experts. In addition, we wanted to find out which of the available features were truly useful in explaining tutor behavior.

Our work attempts to discover rules connecting different types of goals in our annotations, rather than discovering relationships between directly observable phenomena such as sentence length. Although it is therefore dependent on solid definitions of the annotation categories, this strategy increases the chance that the rules obtained will be meaningful to humans, because we have chosen features which are meaningful to humans.

[†]This work was performed while Reva Freedman was at the Illinois Institute of Technology.

Background

Roughly speaking, the purpose of the C4.5 program is to provide the best possible decision tree to explain a given set of data. A decision tree is a set of rules for guessing the value of a dependent variable, the result variable, given the values of a set of features.

Each of our experiments attempts to determine a set of rules for making one decision about the text to be generated. As the criteria for making such decisions are encoded as prerequisites for one or more plan operators, appropriate features for experimentation include any predicates whose values would be available during the operation of the planner. These predicates deal with several kinds of knowledge:

- *Tutorial agenda*, all goals which have been chosen but not realized as text yet. Of these, the current goal and its ancestors are most relevant.
- *Tutorial history*, a record of past tutoring goals enabling us to answer questions such as "is this the first time we have tried to explain this concept to the student?". We also maintain a dialogue history, which contains the student's responses
- *Student input.* One of the most important determinants of the tutor's response is what the tutor is responding to, i.e. the student's latest turn. What is generally important is not the content of the student's response but rather its relation to the tutor's question that elicited it. Our categorization of student responses is described in more detail below.
- *Student model.* An important aspect of the student model is the tutor's overall evaluation of the student. In this paper, we use the number of errors in the student's initial solution of the problem as a measure of this value. Note that a dialogue history feature, namely the number of times a question was asked before the student gave the correct answer, could also be used as a measure of student knowledge.
- *Domain knowledge base.* This category includes information about the problem domain as well as information about the task being tutored.

Our human tutors try to maintain a global tutoring plan but at the same time reply to issues raised by students. As a result, turns generated by human tutors tend to have the following structure (Freedman & Evens, 1996), which we have adopted for CIRCSIM-Tutor:

Response to student's previous statement
  Acknowledgment of student's response (1)
  Content-based reply to student (2)
Next step of tutorial plan (3)

The following example illustrates the tripartite structure:

T: Yes, (1)
that's the effect of increased sympathetic stimulation on the myocardium. (2)
But what happens to CC in the DR period? (3)

Each of the three segments is optional. For example, if the content-based reply segment is omitted, the tutor might say:

T: Yes, but what happens to CC in the DR period?

CIRCSIM-Tutor builds each segment separately, then combines them. In this paper we report on four experiments that attempt to determine aspects of the tutor's dialogue. In experiment 1, we look at the high-level structure of the conversation, attempting to find features that determine which tutorial strategy the tutor will choose. In the next two experiments, we look at rules for building the segments of a turn. In experiment 2, we attempt to determine how the tutor decides what to do after receiving the student's response, e.g. continue with the next step of the current plan, add a content-based reply before continuing, or change plans altogether. In experiment 3, we look at features which may determine the existence and type of the acknowledgment. Finally, in experiment 4, we move closer to surface text, choosing one topic which tutors frequently teach and attempting to determine when it will be realized with the text generation primitive ***t-informs*** and when with ***t-elicits***. These experiments cover a spectrum of the types of rules needed for CIRCSIM-Tutor.

The Corpus

Over the last seven years, we have accumulated over 5000 turns of transcripts of human tutoring sessions conducted by typing over a modem. In the tutoring sessions, the students are presented with a hypothetical medical problem which affects blood pressure, such as a hemorrhage or an overly fast pacemaker. The focus of the lesson is on the baro-receptor reflex, the negative feedback loop by which the autonomic nervous system adjusts blood pressure back toward normal. Students are required to predict the qualitative change (increase, decrease or no change) in seven core variables, such as cardiac output and central venous pressure, in each of the three physiological stages which form the negative feedback loop.

In our current protocol, students work the problem one stage at a time, followed by tutoring on their errors. The DR or direct response stage, referred to below, includes the changes that occur after the precipitating event but before the activation of the nervous system. Since three of the seven core variables are largely controlled by the nervous system, they are known as neural variables. In the examples below, the reader will see that whether or not a variable is neural is one of the most important domain-level determinants of various choices.

Tracing the operation of our planner creates a hierarchy of goals. Although nothing in the planner itself requires it, we have attempted where possible to maintain consistency in the type of goal which appears at each level of the hierarchy.

- Physiological stage
- Physiological variable to be taught
- Tutoring strategy
- Topics which implement the strategy
- Text generation primitive (*inform* or *elicit*) for each topic

In the example at the beginning of this paper, *neural-DLR* is a tutoring strategy which consists of the three topics named in the recipe in the order listed.

We annotated one section of our corpus, namely the DR section of every transcript dealing with a broken pacemaker, with the same hierarchical goal structure our planner uses.

Then we added annotations for features available in the planning environment at the time the turn was generated, using each of the sources above. Finally, we used Quinlan's (1993) C4.5 learning algorithm to determine tutoring goals based on subsets of the available features.

The following is an example from the annotated corpus. We used SGML format because a number of utilities are available for processing SGML mechanically. In SGML, as in HTML, the opening delimiter of a piece of text is an identifier in angle brackets, and the closing delimiter is identical except for an initial slash. The identifiers correspond to the names of our plan operators.

The numbers in parentheses are the turn number and sentence number from the original corpus. The ***t-ack*** and ***s-answer*** forms model a cooperating process which responds to student utterances; they do not belong to the goal hierarchy.

```
    <T-does-neural-DLR>
      <T-tutors-mechanism>
        <T-elicits-mechanism>
(29.4) Can you tell me how TPR is controlled?
<S-answer catg=correct>
(30.1) Autonomic nervous system.
</S-answer>
<T-ack type=positive>
(31.1) Yes.
</T-ack>
        </T-elicits-mechanism>
      </T-tutors-mechanism>
      <T-tutors-DR-info>
        <T-informs-DR-info>
(31.2) And the predictions that you are
making are for the period before any neural
changes take place.
        </T-informs-DR-info>
      </T-tutors-DR-info>

      <T-tutors-value>
        <T-elicits-value>
(31.3) So what about TPR?
          ...
<S-ans catg=correct>
(38.4) I would like to change my response re
TPR to zero change.
</S-ans>
<T-ack type=positive>
(39.1) Good.
</T-ack>
        </T-elicits-value>
      </T-tutors-value>
    </T-does-neural-DLR>
```

The example above shows a successful attempt to remediate the neural variable TPR in the DR stage. It shows a tutoring strategy (*neural-DLR*), the three topics which make up the strategy, and the primitives (*inform* or *elicit*) used to realize the topics. A DLR, or directed line of reasoning, is a pre-planned series of questions designed to make a particular point. In this example, ***t-does-neural-dlr*** is realized as a sequence of three topics: ***t-tutors-mechanism*** (to teach the fact that TPR is neural), ***t-tutors-dr-info*** (to teach basic information about the DR stage), and ***t-tutors-value*** (to discuss the corrected value of TPR). Each of these three topics can be realized as an instance of the tutorial primitive ***t-inform*** (conveying the information to the student) or ***t-elicit*** (eliciting the information).

Since the purpose of these experiments was to suggest rules for CIRCSIM-Tutor, we considered only dialogue that will be emulated by dialogue in CIRCSIM-Tutor, eliminating dialogue used for giving instructions, collecting data, and other operations that the graphical user interface will handle.

Choosing a Tutorial Strategy

The first experiment explored how tutoring operators are chosen at the tutoring strategy level. Each case represented one attempt to tutor one variable. The features we considered were:

- The total number of variables predicted incorrectly
- The number of neural variables predicted incorrectly
- Whether the variable is neural or non-neural
- Sequence of the variable within the neural or non-neural group. The tutors usually tutor all the neural variables first, followed by the others. Within each group, the tutors have a standard order but do not include all the variables.
- How many previous attempts had been made to tutor this variable

Our sample included five different tutoring methods:

- ***t-does-neural-dlr***, tutoring using a guided series of questions exploring the behavior of a neural variable
- ***t-shows-contradiction***, tutoring using an inconsistency in the predicted values of the variables
- ***t-tutors-via-determinants***, tutoring using the set of variables which have an effect on the variable which was incorrect
- ***t-moves-forward***, tutoring using causal reasoning from a known-correct variable to the incorrect one
- ***t-tutors-via-deeper-concepts***, tutoring using other concepts and more particular physiological parameters

In this experiment we had 23 cases, each with 5 features. There were 5 possible outcomes. The original tree produced by C4.5 is too specific to be useful. C4.5 produced the following simplified tree:

```
If variable is neural
    if first neural variable, use t-does-neural-dlr
    if second, use t-shows-contradiction
else   {variable is not neural}
    if first non-neural vbl, use t-tutors-via-determinants
    if second, use t-moves-forward
```

This tree misclassifies 3 of the 23 cases, for an error rate of 13%. Of the three which are misclassified, two involve second and subsequent attempts to tutor the same variable. These attempts tend to employ the less common tutoring methods. Although we have coded about half of the transcripts involving these tutoring methods, we have coded only about 10% of the total corpus. We hope that completing the annotation of the corpus will provide criteria for invoking the lesser-used tutoring methods.

What this tree has uncovered is a high-level plan for the entire tutoring session. Previous research has found an algorithm for sequencing the variables to be tutored, and the rule described here tells us which tutoring strategy to use for each variable.

Within each group of variables, neural and non-neural, C4.5 found an interesting rhetorical pattern. The second variable to be tutored in each group uses a method which builds on knowledge taught in the first variable. For example, a common rhetorical pattern for tutoring two non-neural variables v_1 and v_2 is to tutor v_1 via determinants, then move forward to v_2 as follows:

<***t-corrects-variable variable=v***$_1$>
  <***t-tutors-via-determinants***>
    What variables in the prediction table determine v_1?
    (v_1 is tutored, based on its determinants)
</***t-corrects-variable***>
<***t-corrects-variable variable=v***$_2$>
  <***t-moves forward***>
    And what effect would v_1 have on v_2?
    (v_2 is tutored by moving forward from v_1)
</***t-corrects-variable***>

The pattern for neural variables is similar.

Choosing a Response Strategy

In the second experiment, we were interested in categorizing the change (or not) in tutoring goals directly after a student replies to a tutor query. The following are the most frequent categories of student responses in our current markup:

- *Correct.* The student's answer contained the information the tutor was attempting to elicit.
- *Correct but hedged.* The student's answer was correct but contained some indication that the student was unsure of the answer, e.g. saying "maybe heart rate?"
- *Partially correct.* One correct answer, when two or more items were requested.
- *Near miss.* The student's answer was technically correct but was not what the tutor was attempting to elicit, e.g. it was at the wrong level of detail.
- *Don't know.* The student's response was equivalent to "I don't know."
- *Incorrect.*

We tried a number of features, of which the most explanatory turned out to be:

- The category of student response
- Whether the variable is neural or non-neural
- Sequence of the variable within its category
- How many previous attempts had been made to tutor this variable

Possible tutor behaviors were:

- ***Proceed***. The tutor proceeded with the next tutorial goal. This is the normal action when the student gave the desired answer.
- ***Give info and proceed***. The tutor responded with some tutorial information before proceeding with the next tutorial goal.
- ***Give info and re-elicit***. The tutor responded with some tutorial information before asking substantially the same question again.
- ***Give answer and proceed***. The tutor gave the student the answer, then proceeded with the next tutorial goal.
- ***Nested method***. The tutor introduced a new tutoring method to address the current tutorial goal. When the new method terminates, the tutor will return to the next goal of the original method, i.e. the younger sibling of the current goal.
- ***New method***. The tutor abandoned the current method-level tutoring goal and all its descendants, and tried another method to tutor the same variable.

For this experiment we had 57 cases of tutor behavior following a student response. C4.5 produced the following decision tree, which required no simplification:

```
If student answer was correct,
  proceed
If student answer was correct but hedged
  give info and proceed
If student answer was partially correct
  give info and proceed
If student answer was a near miss
  introduce a nested method
If student answer was incorrect
  if variable is neural then
    if this is the first attempt for this variable
      try a new method
    else  {subsequent attempts}
      give info and re-elicit
  else  {non-neural variable}
    if first non-neural variable in dialogue
      give answer and proceed
    else  {subsequent variables}
      give info and re-elicit
```

This tree provides an algorithm for updating the tutoring plan based on the student's response. It correctly classifies 50 of our 57 cases, yielding an error rate of 12%.

The biggest source of error is that the tree underpredicts the use of a new method, predicting only three new methods where seven actually occurred. Trying a new method means, for example, abandoning ***t-tutors-via-determinants*** and switching to ***t-shows-contradiction***. We hope that adding more transcripts to our experiment will shed light on this issue.

Many of the features that we coded were not used by C4.5 in building the final decision tree, most notably the counts of how many predictions were incorrect. This indicates that the tutor's response to a student utterance appears to depend more on the category of the student utterance than on a global assessment of the student's performance.

Choosing an Explicit Acknowledgment

In the next experiment, we explored the style of acknowledgment issued by the tutor. in response to a student answer. In 62 cases of tutor acknowledgments, we categorized the observed acknowledgments as follows:

- *Positive*, e.g. "Correct."
- *Partial*, e.g. "Well that's partly correct, ..."
- *Negative*, e.g. "No."
- *Nil*, no explicit acknowledgment.

The possible features were the same as in the previous experiment. In this experiment C4.5 found significance in the number of times the tutor retried the ***t-elicit*** to which the student was responding. The resulting decision tree is quite messy, with 28 nodes before simplification and 15 nodes after, and the simplified tree misclassifies more than 20% of the cases. However we did obtain a few fairly solid rules:

```
If answer was partially correct
  then issue partial ack
If answer was correct
  then issue positive or nil ack
If the answer was incorrect
  if we are on the first neural or first non-neural vbl
    if we are on the first attempt to tutor that variable
      then nil ack
      else {a subsequent attempt}
      negative ack
```

The distinction between the first attempt (no acknowledgment) and subsequent attempts (negative acknowledgments) appears interesting at first glance. However, further analysis shows that this result is confounded by the results of the first experiment, i.e., first attempts and subsequent attempts generally use consistently different methods. Thus it is possible that a feature of the second method is the underlying cause of the rule above.

There is further evidence that tutorial planning features are not sufficient to explain the use of acknowledgments. For example, our decision trees offer no explanation as to why correct answers receive a positive acknowledgment about three times out of four and no acknowledgment the rest of the time. We suspect that other features affecting dialogue coherence, such as the presence or absence of an initial discourse marker, are related to the decision to use an explicit acknowledgment. Schiffrin's (1987) study of discourse markers lends credence to this theory. For example, according to Schiffrin's analysis, the use of *well* in the following excerpt indicates that the tutor is contradicting the student's wrong answer. This is similar to the purpose a negative acknowledgment would serve.

T: What to you think will happen to SV?
S: No change.
T: *Well*, you predicted that RAP would go down and ...

Also, since DLRs already have internal coherence, it is not surprising to see fewer acknowledgments on student responses to questions inside a DLR.

Choosing a Realization

The final experiment examined the realization of the topic ***t-tutors-dr-info***. This topic is always realized by an instance of the primitive ***t-informs*** or ***t-elicit***. Our question was to determine under what conditions each of these primitives is chosen by the human tutor.

Although ***t-informs*** is usually realized as a declarative sentence and ***t-elicits*** with an interrogative, other alternatives are possible. For example, ***t-elicits*** could be realized as an imperative: "Please tell me ..." (Sinclair & Coulthard, 1975). Thus these rules do not directly determine surface structure, although they do move one level closer to surface structure.

For this experiment we had C4.5 build a decision tree to classify 16 cases where ***t-tutors-dr-info*** is realized as ***t-informs*** (9 cases) or ***t-elicit*** (7 cases). Since this topic occurs when an incorrect prediction for a neural variable is tutored, we chose the following as possible explanatory features:

- Number of variables predicted incorrectly for the stage
- Number of neural variables predicted incorrectly
- Whether or not the tutor has already tutored the topic ***t-tutors-mechanism*** in the course of remediating the incorrect prediction. This topic is used to teach the student that a variable is neural.

The first two features were chosen because they could be used as part of an assessment of the student's performance. Previous work (e.g., Hume et al., 1996) suggests that our tutors change their tutoring style, especially their use of hints, based on their assessment of the student.

The following decision tree was obtained from C4.5:

```
If a t-tutors-mechanism topic has occurred
  then a subsequent t-tutors-dr-info topic will be
    realized as t-inform
  otherwise it will be realized as t-elicit.
```

This rule, which uses only the feature ***t-tutors-mechanism***, correctly classifies 14 of the 16 cases, giving an error rate of 13%. The other two features were ignored.

It is worth noting that none of the readers of the transcripts had previously noticed this relationship.

Having discovered this relationship, we built a 2-by-2 contingency table showing the realization of ***t-tutors-dr-info*** vs. the use of ***t-tutors-mechanism*** (Table 1). We computed $\chi^2 = 6.3$, including the Yates correction for small N, which shows that the relationship is unlikely to be random ($p = 0.02$).

Table 1: Contingency Table for Realizing ***t-tutors-dr-info***

	Has preceding ***t-tutors-mech***?		
	Yes	No	Total
t-inform	9	0	9
t-elicit	2	5	7
Total	11	5	16

Intrigued by the fact that both components of the student assessment were ignored, we decided to see what happened if we eliminated the ***t-tutors-mechanism*** factor instead. An even simpler decision tree was obtained:

```
If all three neural variables were incorrectly predicted
  then use t-elicit
  else use t-inform
```

This tree misclassifies 3 out of the 16 cases, an error rate of 19%. Our tentative understanding of this rule is that when the student makes the same error repeatedly, the tutor switches to a schema where the information involved is specifically probed for.

Related Work

Vander Linden and Di Eugenio (1996a, 1996b) studied rules for distinguishing the occurrences of warnings in written instructions, in particular how to choose among phrases such as *never do X*, *don't do X*, and *be sure not to do X*. They coded instances of such expressions in recipes, instructions for operating equipment, and similar types of written material. The used three semantic features, such as "writer believes that reader knows that doing X is dangerous." They then used a machine learning algorithm to derive decision rules which could be integrated into a text generation application.

Moser and Moore (1995) and Di Eugenio, Moore and Paolucci (1997) studied the use of cue words such as *because* and *also* in tutorial text. They annotated text segments with features describing structural (core/ contributor), intentional, informational and syntactic relations between text segments. They used a decision-tree generating algorithm to test hypotheses about the occurrence and placement of the cue words. Unlike the work of Vander Linden and Di Eugenio, the text segments are coded for a very large number of features, and part of the experiment is to determine which features are significant.

Each of these papers studies relationships between words or syntactic structures in a surface text and semantic or textual features which have been coded. Although our methodology is similar, our work is different because we are relating levels of goal structure to lower-level goals rather than directly to surface text. Thus our work can be used to build rules at several levels of a hierarchical planning process.

Conclusions

We are currently building a dialogue-based intelligent tutoring system which uses a rule-based global planner. To better understand the tutoring strategies used by human tutors, we studied transcripts of expert tutors doing the same task. We annotated this corpus to show the tutorial goals from which utterances could be derived. We used the C4.5 rule induction algorithm to derive rules relating some of our plan operators to the lower-level operators used to instantiate them. Using a small but significant portion of our corpus, we obtained some intuitively satisfying rules at several levels of discourse. Some of these rules are statistically significant, and some point to ideas for future work. In one case C4.5 caught a generalization which our human readers had missed.

We conducted four experiments. The first experiment revealed a high-level plan for a whole stage of the tutoring session. A second related the category of the student answer to a subsequent change in tutorial goals, providing rules for dynamic modification of the tutoring agenda. A third experiment showed that explicit acknowledgments of the student's utterances may not be fully explainable by the tutorial goal structure. A final experiment was just above the level of text realization, finding relationships between student errors and the decision to express certain tutoring goals as ***t-informs*** or ***t-elicits***.

This work has shown that rule induction on a natural-language corpus can produce planning rules which are both useful and interesting. It is a step on the path toward more objective methods of corpus analysis for text generation. We plan to expand our analysis to cover more plan operators and more examples of each in order to produce the most realistic text possible in CIRCSIM-Tutor.

References

Di Eugenio, B., Moore, J. D. and Paolucci, M. (1997). Learning Features that Predict Cue Usage. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, Madrid. `Cmp-lg/9710006`.

Sinclair, J. M. and Coulthard, R. M. (1975). *Towards an Analysis of Discourse: The English Used by Teachers and Pupils.* London: Oxford University Press.

Freedman, R. and Evens, M. W. (1996). Generating and Revising Hierarchical Multi-turn Text Plans in an ITS. *Intelligent Tutoring Systems: Third International Conference (ITS '96)*, Montreal (Springer-Verlag Lecture Notes in Computer Science, 1086). Berlin: Springer.

Hume, G., Michael, J., Rovick, A., and Evens, M. W. (1996). Student Responses and Follow Up Tutorial Tactics in an ITS. *Proceedings of the Ninth Florida Artificial Intelligence Research Symposium (Flairs '96)*, Key West, FL (pp. 168–172).

Moser, M. G. and Moore, J. D. (1995). Investigating Cue Selection and Placement in Tutorial Discourse. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics* (pp. 130–135).

Quinlan, J. R. (1993). C4.5: *Programs for Machine Learning*. Los Altos, CA: Morgan Kaufmann.

Reiter, E. and Dale, R. (1997). Building Applied Natural Language Generation Systems. Journal of Natural Language Engineering, 3, 57–87.

Schiffrin, D. (1987). *Discourse Markers.* Cambridge: Cambridge University Press.

Vander Linden, K. and Di Eugenio, B. (1996a). A corpus study of negative imperatives in Natural Language instructions. *Proceedings of the 17th International Conference on Computational Linguistics (COLING '96)*, Copenhagen. `Cmp-lg/9607014`.

Vander Linden, K. and Di Eugenio, B. (1996b). Learning Micro-Planning Rules for Preventative Expressions. *Eighth International Workshop on Natural Language Generation (INLG '96)*, Sussex, UK. `Cmp-lg/9607015`.

Acknowledgments

This work was supported by the Cognitive Science Program, Office of Naval Research under Grant No. N00014–94–1–0338, to Illinois Institute of Technology. The content does not reflect the position or policy of the government and no official endorsement should be inferred.

In addition to generously sharing with us their knowledge about both pedagogical and domain issues, Professors Joel A. Michael and Allen A. Rovick of Rush Medical College devised the experimental setup for the transcript collection and served as expert tutors in the tutoring sessions.