SGML-Based Markup as a Step toward Improving Knowledge Acquisition for Text Generation

Reva Freedman†, Yujian Zhou, Jung Hee Kim,
Michael Glass and Martha W. Evens

Department of CSAM
Illinois Institute of Technology
10 W. 31st Street 236–SB
Chicago, IL 60616

freedrk+@pitt.edu, zhouyuj@charlie.cns.iit.edu, janice@steve.csam.iit.edu,
glass@charlie.cns.iit.edu, csevens@minna.cns.iit.edu

Abstract

We are investigating computer-assisted methods for identifying plan operators at both the conversational strategy and surface generation levels. We are using standard-conforming SGML markup on our corpus in order to be able to process it mechanically. We are using C4.5 to identify rules of the form "when is goal *x* implemented with plan *y*?". We are currently testing these methods in the knowledge acquisition process for the text generation component of CIRCSIM-Tutor v. 3, a natural-language based intelligent tutoring system.

Introduction

CIRCSIM-Tutor is a conversational intelligent tutoring system (ITS) which uses natural language for both input and output. The text generation component of CIRCSIM-Tutor v. 3, which we are currently implementing, uses a two-phase architecture consonant with the consensus architecture described by Reiter (1994).

A global, top-down *tutorial planner* chooses and instantiates a logic form for the system to say based on available information The tutorial planner starts from a single top-level goal, "generate a conversation resulting in the student knowing <concepts>," and produces a set of primitive goals including typical speech act goals such as *elicit* and *inform*.

The *turn planner* is a paragraph planner which accepts a set of these goals and turns them into a coherent turn of one or more sentences, including the lexical insertion and surface generation phases. Using the two-level model suggested by Robin (1994) and others, the turn planner converts the tutorial planner primitives to one or more semantic forms representing the concepts the tutor wants to convey. These semantic forms are then realized as surface text. (For further details about the CIRCSIM-Tutor planner see Freedman, 1996.)

The basic knowledge representation for the tutorial planner is a sophisticated form of schema which allows static and dynamic preconditions, recursion and full unification. Although neither our planner nor our markup formalism requires it, we have attempted where possible to maintain consistency in the type of action performed at each level of the hierarchy. The most common pattern encompasses the following levels:

- Tutoring strategy
- Speech act
- Linguistic realization of the speech act

If the student answers a question incorrectly, CIRCSIM-Tutor can backtrack at each of these levels: As a result, the transcripts often contain only partial schemata because the tutor changed schemata in mid-stream in response to a student utterance.

The knowledge for both the tutorial planner and the turn planner is stored in operator libraries independent of the planning algorithms employed. In order to create these operator libraries, we must determine how each planner goal is realized at the next level, i.e. under what conditions goal ***t-xxx*** might be realized as ***t-yyy*** or the sequence of ***t-yyy*** followed by ***t-zzz***.

Potential criteria for any of these decisions, i.e. prerequisites for the plan operators, may include information from any of the following sources:

- *Dialogue history*, e.g. is this the first time we have tried to explain this concept. The dialogue history also includes the student's response and our categorization of it.
- *Student model*, e.g. whether we are conversing with a

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.

†Reva Freedman is now at the Learning Research and Development Center of the University of Pittsburgh.

strong or a weak student. We use the student's initial solution of the problem as well as later responses during the conversation to maintain a static and a dynamic evaluation of the student.

- *Domain knowledge base.*

The idea of using machine learning, specifically the use of rule induction, to help identify reasonable rules from transcripts was suggested to us by a series of recent papers connecting text generation goals with textual phenomena (Moser and Moore, 1995; Vander Linden and Di Eugenio 1996a, 1996b; Di Eugenio, Moore and Paolucci, 1997).

Annotating the Corpus

We have accumulated over 5000 turns of transcripts of human tutoring sessions, conducted by typing over a modem. We originally started annotating them in order to manually collect information for text generation. The large size of the corpus and the wide variety of phenomena involved made manual analysis a slow process.

To achieve more accurate and comprehensive results, we decided to formalize our markup. We chose to use fully conformant SGML with the hope that we could use existing software for a large portion of our analysis. With this detailed markup available, many questions about tutorial strategies become more practical to ask.

The following is an example from our corpus. The numbers in parentheses are the turn number and sentence number from the original corpus. For simplicity both in writing and reading the markup, when an argument is relevant to many levels of goals, we only put it on the uppermost level to which it applies. We use a mechanical process to copy the arguments to lower levels. The ***t-ack*** and ***s-answer*** forms are derived from a process which responds to student utterances; they are part of the corpus and the output of CIRCSIM-Tutor, but they are not part of the goal hierarchy of the tutorial planner.

```
    <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>
```

In our tutoring sessions, the students are presented with a hypothetical medical problem which would affect blood pressure, such as a hemorrhage or a pacemaker which runs too fast. The focus of the lesson is on the baroreceptor 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 or decrease, in seven physiological variables, such as cardiac output and central venous pressure, at each of three stages. The DR stage, referred to below, contains the changes that occur directly after the problem has occurred but before the reflex has acted. In our current protocol, the student predicts all seven variables for one stage, the tutor teaches until the mistakes are corrected, and they proceed to the next stage. Among the seven variables, three have some degree of neural (reflex) control. Understanding these is key to understanding how the reflex regulates blood pressure.

When analyzing a tutoring dialogue, each stage is realized as a separate goal. Each stage usually contains goals for remediating each incorrectly predicted variable. The example above shows a successful attempt to remediate the neural variable TPR in the DR stage. We divide each attempt into three levels: *method*, *topic*, and *primitive*. A *method* describes how a tutor teaches something. For example, the method picked in this case was ***t-does-neural-dlr***. 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*** (for the fact that TPR is neural), ***t-tutors-dr-info*** (for basic information about the DR stage), and ***t-tutors-value*** (for the corrected value of TPR). Each of these three topics can be realized as an instance of the primitive ***t-inform*** (conveying the information to the

student) or ***t-elicit*** (eliciting the information).

The transcripts have also been annotated with information about which variables the student answered incorrectly during the initial solution of the problem.

First Experiment

We have used the annotated corpus to perform several experiments using Quinlan's (1993) C4.5 learning algorithm. Here we report on two of them.

The first 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.

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:

- The total number of variables predicted incorrectly[1]
- The 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, which might affect whether the tutor gives information to the student (*inform*) or requests it from the student (*elicit*). In previous work, Hume et al. (1996) have stated that our tutors change their tutoring style, specifically 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 ($p = 0.02$).

[1] All of these numbers refer to the DR stage of the problem.

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

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

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 also. 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%. We have coded approximately half of the transcripts where ***t-tutors-dr-info*** occurs. We intend to run the analysis again when we have coded the rest of the data, but we do not expect any significant changes. 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.

Second Experiment

The second experiment explored how tutoring operators are chosen at the method level. Each case represented one attempt to tutor one variable. The features we picked 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 its category (neural or non-neural). (The tutors usually tutor all the neural variables first, followed by the others.)
- How many previous attempts had been made to tutor this variable

In our sample, five different tutoring methods were attested:

- ***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 variable in category, use ***t-does-neural-dlr***
  if second, use ***t-shows-contradiction***
else /* variable is not neural */
  if first vbl in catg, 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 produce decision trees which cover other phenomena of interest to us, including criteria for invoking the lesser-used tutoring methods, tutoring of third incorrect variables, and multiple attempts to tutor the same incorrect prediction.

Within each group of variables, neural and non-neural, C4.5 noticed 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.

Conclusions

We have used C4.5 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 simple rules which made intuitive sense to us. In one case C4.5 caught a generalization which our human readers had missed. Using SGML to annotate the corpus made this project more feasible by allowing us to automate more of the steps. We plan to expand our analysis to cover more plan operators and more criteria for choosing among them, in order to produce the most realistic text possible in CIRCSIM-Tutor v. 3.

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.

Freedman, R. 1996. Interaction of Discourse Planning, Instructional Planning and Dialogue Management in an Interactive Tutoring System. Ph.D. diss., Dept. of EECS, Northwestern Univ.

Hume, G., Michael, J., Rovick, A., and Evens, M. W. 1996. Student Responses and Follow Up Tutorial Tactics in an ITS. In Proceedings of the Ninth Florida Artificial Intelligence Research Symposium, 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. Morgan Kaufmann.

Reiter, E. 1994. Has a Consensus NL Generation Architecture Appeared, and is it Psycholinguistically Plausible? In Proceedings of the Seventh International Workshop on Natural Language Generation, Kennebunkport, ME, pp. 163–170. Cmp-lg/9411032.

Robin, J. 1994. Revision-Based Generation of Natural Language Summaries Providing Historical Background: Corpus-based Analysis, Design, Implementation and Evaluation. Ph.D. diss., Columbia University.

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. 8th International Workshop on Natural Language Generation, INLG '96, Sussex, UK. Cmp-lg/9607015.

Acknowledgments

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.