IDENTIFYING RELEVANT PRIOR EXPLANATIONS

James A. Rosenblum
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260, USA
Internet: jr@cs.pitt.edu

Abstract

When human tutors engage in dialogue, they freely exploit all aspects of the mutually known context, including the previous discourse. Utterances that do not draw on previous discourse seem awkward, unnatural, or even incoherent. Previous discourse must be taken into account in order to relate new information effectively to recently conveyed material, and to avoid repeating old material that would distract the student from what is new.

Producing a system that displays such behavior involves finding an efficient way to identify which previous explanations (if any) are relevant to the current explanation task. Thus, we are implementing a system that uses a case-based reasoning approach to identify previous situations and explanations that could potentially affect the explanation being constructed. We have identified heuristics for constructing explanations that exploit this information in ways similar to what we have observed in human-human tutorial dialogues.

Introduction and Motivation

We are building an explanation component for an existing intelligent training system, SHERLOCK (Lesgold *et al.*, 1992), which trains avionics technicians to troubleshoot electronic equipment. Using SHERLOCK, trainees solve problems with minimal tutor interaction and then review their troubleshooting in a post-problem *reflective follow-up* (RFU) session where the tutor replays each student action and assesses it as "good" (<+>) or as "could be improved" (<->). After a step is replayed, the student can ask the tutor to justify its assessment.

As an example of the way in which human tutors exploit previous discourse, consider the dialogue in Figure 1, taken from our data. Even though the student has made the same mistake twice, the second explanation looks quite different from the first. Yet the two explanations are related to one another in an important way. In the second explanation the tutor simply reminds the student that she has not determined the status of the main control data signals and that she should do so before testing the secondary control data signals. The tutor expects the student to be able to make use of the previous explanation once he has indicated that it is relevant to the current situation ("for the same reasons given ..." serves this purpose). Accordingly, the tutor does not repeat the detailed explanation of why the main control data signals should be tested first. By generating the second explanation in such a way that it 'meshes' with the first, not only has the tutor corrected the testing mistake of the student, but has forced the student to consider how the two situations are similar. In pointing out this similarity, he has given the student a better understanding of the domain. We call an explanation that is later referred to (explicitly or implicitly) or is integrated into a subsequent explanation the *anchor*.

Clearly it is desirable for a system to produce text that is sensitive to what has been said previously. In order to do this, however, a system must first be able to decide what previous explanation (or part thereof) to use as an anchor. This involves deciding, *in an efficient way*, whether there exist suitable candidates to act as anchor, and if so, which amongst them would be best to use. This paper concentrates on this task.

The Text Planner

For this work, we are extending the text planner built by Moore and Paris (1989). Briefly, it works in the following way. A *communicative goal* (e.g., "achieve the state where the hearer believes that an action could be improved") is formed based upon the student's question. Using its library of plan operators that encode knowledge about tutorial explanations, the system employs a linear planning mechanism to synthesize a response to achieve this goal. The result is a *text plan* for the explanation.

The system then presents the explanation to the user, retaining the plan that produced it in a *dialogue history*. The dialogue history is a record of the conversation that has occurred thus far and includes the user's utterances as well as the text plans that led to the system's responses. In this system, a text plan represents the effect that each part of the text is intended to have on the hearer's mental state, the linguistic strategies that were used to achieve these effects, and how the complete text achieves the overall communicative goal.

TUTOR <-> You tested pin 28 of the A1A3A15. [1]

STUDENT Why is testing pin 28 not a good step? [2]

TUTOR As explained before, the main data inputs to relay B24 are highly suspect at this time. You have tested the low input to B24 but not the high one. Since it is more likely that there may be a problem with the main data signals to relay B24 than with the secondary data signals to relays B25-B30, the main data signal should be tested first. Testing pin 28 is not a bad step, it is just more likely (on the average) that the main data signals are causing the problem. [3]

⋮ *One exchange and one action later.*

TUTOR <->You tested pin 36 of the A1A3A15. [4]

STUDENT Don't I need to test pin 36? [5]

TUTOR You might have to, but for the same reasons given when you tested pin 28, it is generally more efficient to test the main control data signals first, and then test the secondary control data signals if necessary. [6]

Figure 1: Human-Human Advisory Interaction Displaying Contextual Effects

Knowledge Sources for Finding Relevant Prior Explanations

The most straightforward way to find relevant prior explanations is to exhaustively search the system's dialogue history looking for explanations that have certain features. For example, when explaining why a step was assessed as "could be improved," the system could look for previous explanations that justified this type of assessment, and in which the two actions being assessed were similar (i.e., had the same features).

However, this approach is problematic. Explanation plans are large complex structures, and they will accumulate rapidly as the dialogue progresses. Exhaustively searching the discourse history for relevant prior explanations is computationally prohibitive. Thus, we require an indexing strategy that allows the system to find possibly relevant prior explanations in an efficient manner.

To satisfy this requirement, we use case-based reasoning (CBR) to provide a framework in which previous student actions can be efficiently examined to determine which, if any, are relevant when producing an explanation. This approach has the additional advantage of allowing the system to consider what *was* said as well as what *was not* said when planning an explanation. For example, the student may have previously performed an action that displayed some characteristic that the tutor decided not to mention at the time and which would now be appropriate to discuss.

A Case-Based Algorithm

The following aspect of SHERLOCK's reasoning is extremely important to our work. SHERLOCK evaluates each student action by determining which *facets* apply to that action. The facets represent factors that expert avionics tutors use in assessing student's troubleshooting actions (Pokorny and Gott, 1990). To evaluate an action, SHERLOCK finds each facet that applies to it and determines whether that facet should be considered good (g), bad (b), or neutral (n) given the current problem-solving context. For example, the facet "Making a measurement that is off the active circuit path" is considered a b-facet. The representation of a student action includes the list of facets characterizing the action and an assessment (g, b, or n) for each of those facets.

Case-based reasoning generalizes from cases to support indexing and relevance assessment, and can be used to evaluate a case by comparing it to past cases (Ashley, 1992). This seems to describe our task when we treat each student action as a "case". Influenced by the work of Aleven and Ashley (1992), we noted certain similarities between their domain and ours that led us to believe that we could use CBR techniques to identify similar actions as described below.

Our algorithm builds a data structure called a *similarity DAG* (Directed Acyclic Graph) which indicates the previous student actions that are similar to a given action. By similar, we mean similar with respect to a certain class of facets (some combination of g, b, or n). For example, when answering a question about why the current action was assessed as "could be improved," the similarity DAG is built so that it indicates which previous actions were similar to the current action with respect to the b-facets. The root of the DAG represents the current action and the facets of interest (b-facets in our example) that apply to it. Each node in the DAG, including the root, represents a set of student actions that share the same set of interesting facets. The more facets that a node has in common with the current action (in the root), the closer it will be to the root node. Proximity in the DAG corresponds to similarity in facet sets. Basically, the similarity DAG is a partial ordering of the student's actions based on their facet lists.

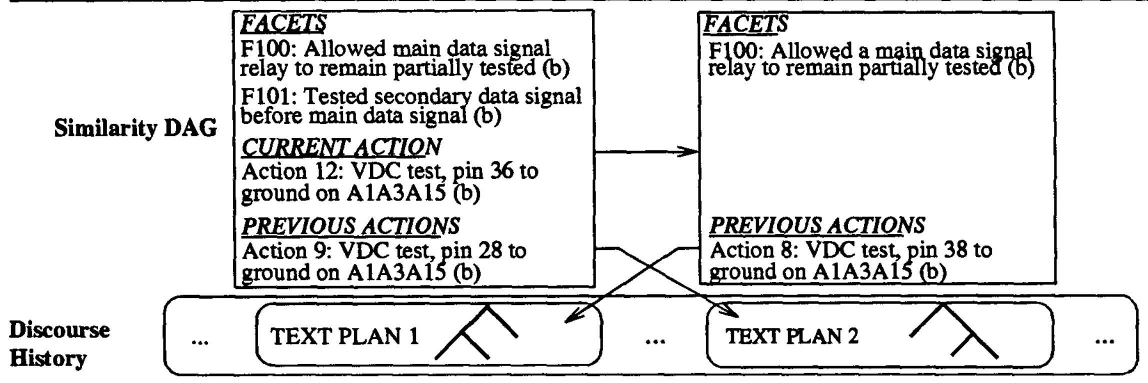
Similarity DAG
FACETS
F100: Allowed main data signal relay to remain partially tested (b)
F101: Tested secondary data signal before main data signal (b)
CURRENT ACTION
Action 12: VDC test, pin 36 to ground on A1A3A15 (b)
PREVIOUS ACTIONS
Action 9: VDC test, pin 28 to ground on A1A3A15 (b)
FACETS
F100: Allowed a main data signal relay to remain partially tested (b)
PREVIOUS ACTIONS
Action 8: VDC test, pin 38 to ground on A1A3A15 (b)
Discourse History
...
TEXT PLAN 1
...
TEXT PLAN 2
...

Figure 2: Data structures when considering how to answer turn 5, Figure 1

Figure 2 shows the similarity DAG that is constructed when the system considers how to answer the question, "Don't I need to test pin 36?" (turn 5 of Figure 1). The facets relevant to the action in question are F100 and F101. The structure indicates that two previous actions – 9 and to a lesser degree 8, are similar to the current situation. Pointers index the dialogue history's record of what was said at those times. At this point, the system has identified candidate situations that are relevant for planning the current explanation. It can now consider these retrieved situations more closely to determine any other facets that they may possess, and can examine the related explanations in the dialogue history to determine what was said about each of the two previous situations. The fact that there are no other nodes in the DAG indicates that there are no other suitable prior situations.

Initial results using this algorithm seem promising. In an analysis of 8 student-tutor protocols involving 154 actions and 22 opportunities for integrating a previous explanation into an answer, the algorithm correctly identified the same previous situations that were used by the human tutor in the actual interactions. In all but 3 cases, when the human tutor did not make a reference to a previous explanation, our algorithm reported no similar prior situation. In the 3 situations where our algorithm identified a similarity not exploited by the tutor, our expert agreed that they would have been useful to incorporate into his explanations.

Lastly, this technique will be useful in answering students' direct questions about the similarities of situations, e.g., "Why is testing 30 good? Isn't it like 36 and 28?" By constructing and consulting a similarity DAG, the system is able to plan responses such as: "Yes, but now you know the main control data signals on pins 33 and 22 are good so you need to test the secondary data signals."

It is important to note that this approach is successful, in part, because the facets are based on a tutor's evaluation of a student's actions, and we are currently addressing only questions that justify these evaluations. We focused on this type of question because 48% of student's queries during RFU are of this type. To answer additional questions in a context-sensitive fashion, we will need to extend our indexing scheme to take the intentions behind an explanation into account as well as the domain content discussed.

Conclusions and Future Work

We have indicated that in order to produce text that is sensitive to the previous discourse, a system must first be able to identify relevant previous explanations and situations. To achieve this first step, a CBR algorithm was introduced that indexes the dialogue history and supplies the explanations with a context in which to be considered. We are devising techniques that use this information to plan subsequent explanations.

References

Aleven, V. and Ashley, K. 1992. Automated generation of examples for a tutorial in case-based argumentation. In *Proc. of the 2nd Int'l Conference on Intelligent Tutoring Systems*, Montreal, Canada.

Ashley, K. 1992. Case-based reasoning and its implications for legal expert systems. *Artificial Intelligence and Law* 2(1).

Lesgold, A.; Lajoie, S.; Bunzo, M.; and Eggan, G. 1992. Sherlock: A coached practice environment for an electronics troubleshooting job. In *Computer Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches*. Lawrence Erlbaum Assoc., NJ.

Moore, J. D. and Paris, C. L. 1989. Planning text for advisory dialogues. In *Proc. of the 27th Annual Meeting of the ACL*, Vancouver, B.C., Canada. 203–211.

Pokorny, R. and Gott, S. 1990. The evaluation of a real-world instructional system: Using technical experts as raters. Technical report, Armstrong Laboratories, Brooks AFB.