Automated Tutoring Dialogues for Training in Shipboard Damage Control

John Fry, Matt Ginzton, Stanley Peters, Brady Clark & Heather Pon-Barry
Stanford University
Center for the Study of Language Information
Stanford CA 94305-4115 USA
{fry,mginzton,peters,bzack,ponbarry}@csli.stanford.edu

Abstract

This paper describes an application of state-of-the-art spoken language technology (OAA/Gemini/Nuance) to a new problem domain: engaging students in automated tutorial dialogues in order to evaluate and improve their performance in a training simulator.

1 Introduction

Shipboard damage control refers to the task of containing the effects of fire, explosions, hull breaches, flooding, and other critical events that can occur aboard Naval vessels. The high-stakes, high-stress nature of this task, together with limited opportunities for real-life training, make damage control an ideal target for AI-enabled educational technologies like training simulators and tutoring systems.

This paper describes the spoken dialogue system we developed for automated critiquing of student performance on a damage control training simulator. The simulator is DC-Train (Bulitko and Wilkins, 1999), an immersive, multimedia training environment for damage control. DC-Train's training scenarios simulate a mixture of physical phenomena (e.g., fire, flooding) and personnel issues (e.g., casualties, communications, standardized procedures). Our current tutoring system is restricted fire damage scenarios only, and in particular to the twelve fire scenarios available in DC-Train version 2.5, but in future versions we plan to support post-session critiques for all of the damage phenomena that will be modeled by DC-Train 4.0: fire, flooding, missile damage, and wall or firemain ruptures.

2 Previous Work

Eliciting self-explanation from a student has been shown to be a highly effective tutoring method (Chi et al., 1994). For this reason, a number of automated tutoring systems currently use NLP techniques to engage students in reflective dialogues. Three notable examples are the medical Circsim tutor (Zhou et al., 1999); the Basic Electricity and Electronics (BE&E) tutor (Rosé et al., 1999); and the computer literacy AutoTutor (Wiemer-Hastings et al., 1999).

Our system shares several features with these three tutoring systems:

A knowledge base Our system encodes all domain knowledge relevant to supporting intelligent tutoring feedback into a structure called an Expert Session Summary (Section 4). These expert summaries encode causal relationships between events on the ship as well as the proper and improper responses to shipboard crises.

Tutoring strategies In our system, as in those above, the flow of dialogue is controlled by (essentially) a finite-state transition network (Fig. 1).

An interpretation component In our system, the student's speech is recognized and parsed into logical forms (Section 3). A dialogue manager inspects the current dialogue information state to determine how best to incorporate each new utterance into the dialogue (Lemon et al., 2001).

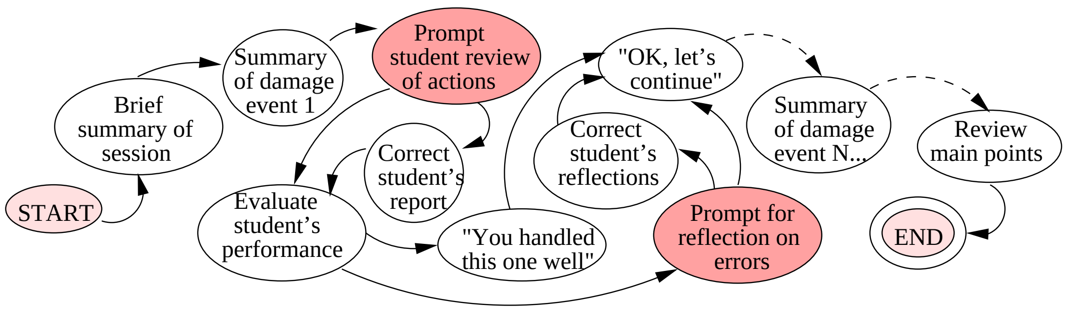

START
Brief summary of session
Summary of damage event 1
Prompt student review of actions
Correct student's report
Evaluate student's performance
"You handled this one well"
Correct student's reflections
"OK, let's continue"
Prompt for reflection on errors
Summary of damage event N...
Review main points
END

Figure 1: Post-session dialogue move graph (simplified)

However, an important difference is that the three systems above are entirely text-based, whereas ours is a *spoken* dialogue system. Our speech interface offers greater naturalness than keyboard-based input. In this respect, our system is similar to COVE (Roberts, 2000), a training simulator for conning Navy ships that uses speech to interact with the student. But whereas COVE uses short conversational exchanges to coach the student *during* the simulation, our system engages in extended tutorial dialogues after the simulation has ended. Besides being more natural, spoken language systems are also better suited to multimodal interactions (viz., one can point and click while talking but not while typing). An additional significant difference between our system and a number of other automated tutoring systems is our use of 'deep' processing techniques. While other systems utilize 'shallow' statistical approaches like Latent Semantic Analysis (e.g. AutoTutor), our system utilizes Gemini, a symbolic grammar. This approach enables us to provide precise and reliable meaning representations.

3 Implementation

To facilitate the implementation of multimodal, mixed-initiative tutoring interactions, we decided to implement our system within the Open Agent Architecture (OAA) (Martin et al., 1999). OAA is a framework for coordinating multiple asynchronous communicating processes. The core of OAA is a 'facilitator' which manages message passing between a number of software agents that specialize in certain tasks (e.g., speech recognition or database queries). Our system uses OAA to coordinate the following five agents:

1. The **Gemini** NLP system (Dowding et al., 1993). Gemini uses a single unification grammar both for *parsing* strings of words into logical forms (LFs) and for *generating* sentences from LF inputs.

2. A **Nuance** speech recognition server, which converts spoken utterances to strings of words. The Nuance server relies on a language model, which is compiled directly from the Gemini grammar, ensuring that every recognized utterance is assigned an LF.

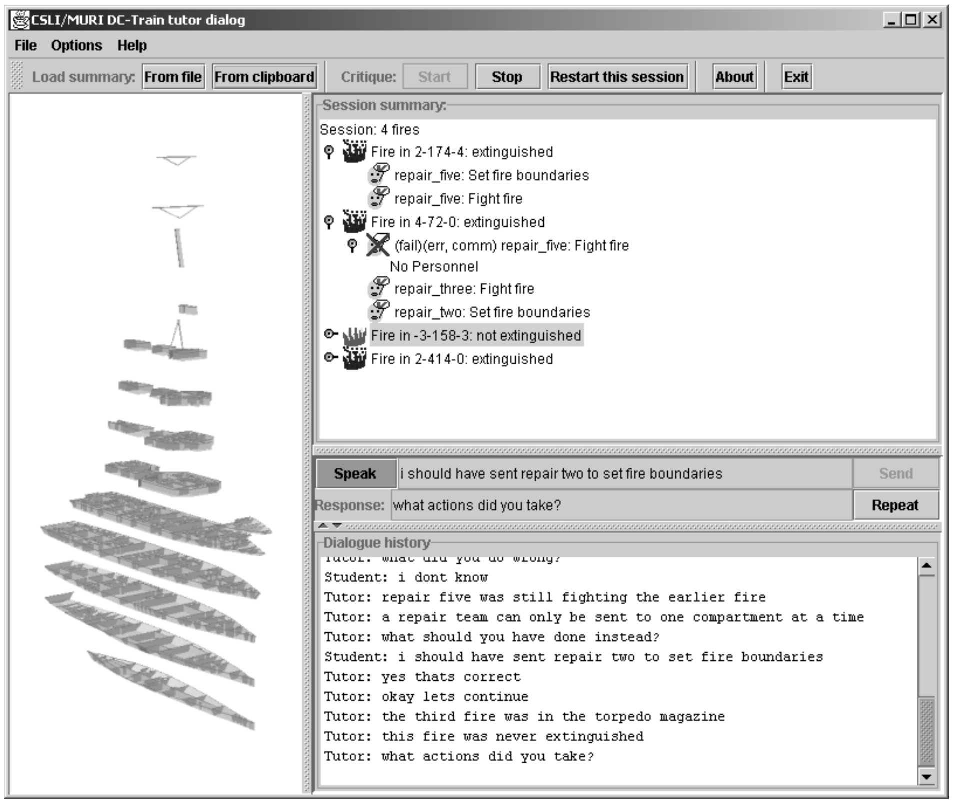
3. The **Festival** text-to-speech system, which 'speaks' word strings generated by Gemini.

4. A **Dialogue Manager** which coordinates inputs from the user, interprets the user's dialogue moves, updates the dialogue context, and delivers speech and graphical outputs to the user.

5. A **Critique Planner**, described below in Section 4.

Agents 1-3 are reusable, 'off-the-shelf' dialogue system components (apart from the Gemini grammar, which must be modified for each application). We implemented agents 4 and 5 in Java specifically for this application.

Variants of this OAA/Gemini/Nuance architecture have been deployed successfully in other dialogue systems, notably SRI's CommandTalk (Stent et al., 1999) and an un-



Session: 4 fires
Fire in 2-174-4: extinguished
repair_five: Set fire boundaries
repair_five: Fight fire
Fire in 4-72-0: extinguished
(fail)(err, comm) repair_five: Fight fire
No Personnel
repair_three: Fight fire
repair_two: Set fire boundaries
Fire in -3-158-3: not extinguished
Fire in 2-414-0: extinguished

i should have sent repair two to set fire boundaries

what actions did you take?

```
Tutor: what did you do wrong?
Student: i dont know
Tutor: repair five was still fighting the earlier fire
Tutor: a repair team can only be sent to one compartment at a time
Tutor: what should you have done instead?
Student: i should have sent repair two to set fire boundaries
Tutor: yes thats correct
Tutor: okay lets continue
Tutor: the third fire was in the torpedo magazine
Tutor: this fire was never extinguished
Tutor: what actions did you take?
```

Figure 2: Screen shot of post-session tutorial dialogue system

manned helicopter interface developed in our laboratory (Lemon et al., 2001).

4 Planning the dialogue

Each student session with DC-Train produces a *session transcript*, i.e. a time-stamped record of every event (both computer- and student-initiated) that occurred during the simulation. These transcripts serve as the input to our post-session **Critique Planner** (CP).

The CP plans a post-session tutorial dialogue in two steps. In the first step, an **Expert Session Summary** (ESS) is created from the session transcript. The ESS is a tree whose parent nodes represent damage events and whose leaves represent actions taken in response to those damage events. Each student-initiated action in the ESS is evaluated as to its timeliness and conformance to damage control doctrine. Actions that the student should have taken but did not are also inserted into the ESS and flagged as such. Each action node in the ESS therefore falls into one of three classes: (i) *correct* actions; (ii) errors of *commission* (e.g., the student sets fire containment boundaries incorrectly); and (iii) errors of *omission* (e.g., the student fails to secure permission from the captain before flooding certain compartments).

Our current tutoring system covers scenarios generated by DC-Train 2.5, which covers fire scenarios only. Future versions will use scenarios generated by DC-Train 4.0, which covers damage control scenarios involving fire, smoke, flooding, pipe and hull ruptures, and equipment deactivation. Our current tutoring system is based on an ESS graph that is generated by an expert model that consists of an ad-hoc set of firefighting rules. Future versions will be based on an ESS graph that is generated by an successor to the Minerva-DCA expert model (Bulitko and Wilkins, 1999), an extended Petri Net envisionment-based reasoning system. The new expert model is designed to produce an ESS graph during the course of problem solving that contains nodes for all successful and unsuccessful plan and goal achievement events, along with an explanation structure for each graph node.

The second step in planning the post-session tutorial dialogue is to produce a **dialogue move graph** (Fig. 1). This is a directed graph that encodes all possible configurations of dialogue structure and content that can be handled by the system.

Generating an appropriate dialogue move graph from an ESS requires pedagogical knowledge, and in particular a *tutoring strategy*. The tutoring strategy we adopted is based on our analysis of videotapes of fifteen actual DC-Train post-session critiques conducted by instructors at the Navy's Surface Warfare Officer's School in Newport, RI. The strategy we observed in these critiques, and implemented in our system, can be outlined as follows:

1. Summarize the results of the simulation (e.g., the final condition of the ship).
2. For each major damage event in the ESS:
   (a) Ask the student to review his actions, correcting his recollections as necessary.
   (b) Evaluate the correctness of each student action.
   (c) If the student committed errors, ask him how these could have been avoided, and evaluate the correctness of his responses.
3. Finally, review each type of error that arose in step (2c).

A screen shot of the tutoring system in action is shown in Fig. 2. As soon as a DC-Train simulation ends, the dialogue system starts up and the dialogue manager begins traversing the dialogue move graph. As the dialogue unfolds, a graphical representation of the ESS is revealed to the student in piecemeal fashion as depicted in the top right frame of Fig. 2.

Acknowledgments

This work is supported by the Department of the Navy under research grant N000140010660, a multidisciplinary university research initiative on natural language interaction with intelligent tutoring systems.

References

V V. Bulitko and D C. Wilkins. 1999. Automated instructor assistant for ship damage control. In *Proceedings of AAAI-99*, Orlando, FL, July.

M. T. H. Chi, N. de Leeuw, M. Chiu, and C. LaVancher. 1994. Eliciting self-explanations improves understanding. *Cognitive Science*, 18(3):439–477.

J. Dowding, J. Gawron, D. Appelt, J. Bear, L. Cherny, R. C. Moore, and D. Moran. 1993. Gemini: A natural language system for spoken-language understanding. In *Proceedings of the ARPA Workshop on Human Language Technology.*

O. Lemon, A. Bracy, A. Gruenstein, and S. Peters. 2001. A multi-modal dialogue system for human-robot conversation. In *Proceedings of NAACL 2001.*

D. Martin, A. Cheyer, and D. Moran. 1999. The Open Agent Architecture: a framework for building distributed software systems. *Applied Artificial Intelligence*, 13(1-2).

B. Roberts. 2000. Coaching driving skills in a shiphandling trainer. In *Proceedings of the AAAI Fall Symposium on Building Dialogue Systems for Tutorial Applications.*

C. P. Rosé, B. Di Eugenio, and J. D. Moore. 1999. A dialogue based tutoring system for basic electricity and electronics. In S. P. Lajoie and M. Vivet, editors, *Artificial Intelligence in Education (Proceedings of AIED'99)*, pages 759–761. IOS Press, Amsterdam.

A. Stent, J. Dowding, J. Gawron, E. O. Bratt, and R. C. Moore. 1999. The CommandTalk spoken dialogue system. In *Proceedings of ACL '99*, pages 183–190, College Park, MD.

P. Wiemer-Hastings, K. Wiemer-Hastings, and A. Graesser. 1999. Improving an intelligent tutor's comprehension of students with latent semantic analysis. In S. P. Lajoie and M. Vivet, editors, *Artificial Intelligence in Education (Proceedings of AIED'99)*, pages 535–542. IOS Press, Amsterdam.

Y. Zhou, R. Freedman, M. Glass, J. A. Michael, A. A. Rovick, and M. W. Evens. 1999. Delivering hints in a dialogue-based intelligent tutoring system. In *Proceedings of AAAI-99*, Orlando, FL, July.