AN ARCHITECTURE FOR SPOKEN DIALOGUE MANAGEMENT

David Duff,
Barbara Gates
Susann LuperFoy

The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22102 USA
{duff|blgates|luperfoy}@mitre.org

ABSTRACT

We propose an architecture for integrating discourse processing and speech recognition (SR) in spoken dialogue systems. It was first developed for computer-mediated bilingual dialogue in voice-to-voice machine translation applications and we apply it here to a distributed battlefield simulation system used for military training. According to this architecture discourse functions previously distributed through the interface code are collected into a centralized discourse capability. The Dialogue Manager (DM) acts as a third-party mediator overseeing the translation of input and output utterances between English and the command language of the backend system. The DM calls the Discourse Processor (DP) to update the context representation each time an utterance is issued or when a salient non-linguistic event occurs in the simulation. The DM is responsible for managing the interaction among components of the interface system and the user. For task-based human-computer dialogue systems it consults three sources of non-linguistic context constraint in addition to the linguistic Discourse State: (1) a User Model, (2) a static Domain Model containing rules for engaging the backend system, with a grammar for the language of well-formed, executable commands, and (3) a dynamic Backend Model (BEM) that maintains updated status for salient aspects of the non-linguistic context. In this paper we describe its four-step recovery algorithm invoked by DM whenever an item is unclear in the current context, or when an interpretation error is, and show how parameter settings on the algorithm can modify the overall behavior of the system from Tutor to Trainer. This is offered to illustrate how limited (inexpensive) dialogue processing functionality, judiciously selected, and designed in conjunction with expectations for human dialogue behavior can compensate for inevitable limitations in SR, NL processor, the backend software application, or even in the user's understanding of the task or the software system.

1. SPOKEN DIALOGUE SYSTEMS

1.1 Integrating Discourse and SR

Waibel et al., (1989) and De Mori et al., (1988) extend stochastic language modeling techniques to the discourse level to improve spoken dialogue systems. The complexity of discourse state descriptions leads to a sparse data problem during training, and idiosyncratic human behavior at run time can defeat even the best probabilistic dialogue model. Symbolic approaches to spoken discourse data identify discourse constraints on language model selection at run time. Our work collects discourse-level processing into a centralized discourse capability as part of a modular user interface dialogue architecture. Its use in a spoken dialogue interface to a distributed battlefield simulation system used for military training is diagrammed in Figure 1.

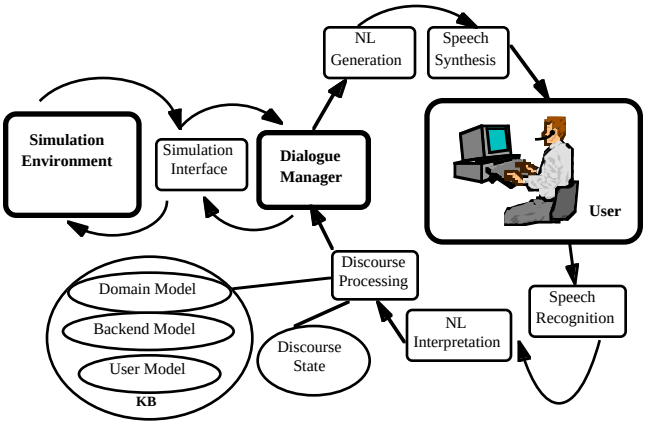

Figure 1: User issues voice commands to a simulation system. Commands are interpreted relative to the context, represented in the discourse state and elements of the knowledge base (KB).

The simulation system is called ModSAF or Modular Semi-Automated Forces. It runs on a network of Sun workstations viewed by instructors and students during mission rehearsal exercises. The user (a commander in training) issues spoken orders in a restricted subset of English called "battle command language." Commands are directed to platoons and companies represented in the simulated world.

The functional interface to ModSAF is through the "Command and Control Simulation Interface Language" or CCSIL (Salisbury, 1995) into which all spoken input utterances must be translated. DM converts the output of a speech recognizer into well-formed CCSIL commands, and presents backend responses to the user. It intercept errors that originate with user behavior or with any component of the input interpretation system. The Discourse Processor (DP) maintains a simple record of the developing linguistic context (Discourse State), and associates new utterances with it and

with stored representations of context in the KB: (1) a User Model, (2) a static Domain Model containing rules for engaging the backend system, with a grammar for the language of well-formed, executable commands, and (3) a dynamic Backend Model (BEM) that maintains updated status for salient aspects of the non-linguistic context.

1.2 HC Dialogue and HCH Dialogue

Real-time spoken dialogue systems are of two types: human-computer dialogue systems or HC (for "Human-Computer") systems, and computer-mediated human-human dialogue systems or HCH (for Human-Computer-Human) systems. In HC systems, the system as a whole is engaged with the user in a dialogue, as in query processing for database retrieval or voice-activated command and control. In HCH systems, the computer only facilitates a dialogue between two humans and is not a party in the dialogue.[1] HCH systems include multi-user dungeons known as MUDs and voice-to-voice machine translation systems (Yamazaki et al., 1994), (Wahlster, 1993). (Waibel, 1995), (Montgomery, 1995).

HCH systems tend to require of the NL (Natural Language) interpreter shallower processing with broader coverage of the input language. In contrast, HC dialogue systems require deep interpretation of a narrower set of input commands. That is, if the user says, "Get one of them and take it to her" in an HC dialogue, the interpreter must fully resolve each reference, "one," "them," "it," and "her" in order to locate the intended wrench and deliver it to the intended human being. The same utterance to an HCH system requires only sufficiently deep interpretation for transmission to the target language correspondents for "one," "them," "it," and "her." In HCH dialogue the system has no power to influence the breadth of language produced by the users, whereas HC dialogue can be designed as a fully system-initiated interaction in which the user responds only to options presented in a restricted vocabulary. In HC dialogue we can rely on human cooperation and adaptability to limit the input vocabulary especially when it corresponds to the limited functionality of the application task.

1.3 Tutoring Versus Training

Differing pedagogical objectives give rise to two modes in instructional HC systems. In "training" mode, the interface is transparent to the user who has the sensation of interacting directly with the backend application. The DM may simply record commands and responses in the training dialogue history, flagging trouble spots for retrospective analysis by an instructor. In contrast, the DM of a "tutoring" system, acts as a third-party mediator whose job includes intercepting user mistakes for correction before passing them to the backend simulation system.

Consider a student's mistaken command to the first platoon to increase its speed by 30 kilometers per hour which would result in an illegal (impossible) velocity. A tutoring system could interrupt the primary command-and-control dialogue between the student and the simulation system to say: "The maximum velocity for a platoon is 60 kph. First platoon is already traveling at 40 kph." In contrast, a training system would send the illegal command to the backend system without modification and leave it to the user to recognize that the wrong platoon was invoked or a problematic velocity value assigned. Our goal is a DM that can be set to operate in either training or tutoring mode by simply adjusting the parameters that control its reaction to student mistakes.

2. DIALOGUE MANAGEMENT

In a deep+narrow HC dialogue, errors can be prevented by designing the NL interface to interpret only those utterances that have meaning to the system and constraining the user's input language to match that narrow sublanguage. It is important to present to the user reliable indicators of the system's restricted language processing ability and avoid an "Eliza-styled" interface that appears superficially to have much greater understanding than it actually does, leading users to say things that cannot be understood in terms of the backend command language. When prevention fails we rely on the recovery mechanism.

The architecture for our work on spoken dialogue interfaces to distributed military simulation systems (Figure 1) was first developed for computer-mediated (HCH) bilingual dialogue in voice-to-voice machine translation applications. The DM is a third-party mediator translating between user and backend languages. Before "translation" from English into the language of the backend system, the DM invokes its recovery algorithm to evaluate the input utterance interpretation against the non-linguistic context representations to do preemptive troubleshooting.

2.2 Dialogue Trouble

Temporary miscommunication in computer-human dialogue (Terveen, 1991) is not an artifact of today's incomplete technology; but an inevitable feature of dialogue interaction. We use Clark and Schaefer's (1987) analysis of human human dialogue as a starting point for the development of a four-step process for dealing with disfluency in HC dialogue. The four steps are Detection, Diagnosis, Repair Plan Selection, and Collaborative Plan Execution. (Prevention is a fifth category of effort that we treat as external to but supportive of the recovery process proper.)

In their study of human-human dialogue, Clark and Schaefer observe a process of grounding by which participants work jointly to ensure listener "acceptance" of each new speaker "presentation" of material for incorporation into the common ground (i.e., the shared understanding) of the discourse. Brennan et al. (1993) and Perez-Quinones (1996) formalize this process for typed database retrieval dialogues so that the human user receives an indication from the system of the internal status of the query interpretation process. Traum and Allen (1992) apply a similar grounding model to the analysis of a corpus of collaborative problem-solving dialogues between humans. In our system, the grounding or acceptance phase serves as the point of attachment for repair dialogues.

[1] Except that HCH systems have their own user interfaces that operate when the need arises for a HC dialogue with one of the users.

We expand the acceptance phase into a 4-step repair-management algorithm invoked for each input utterance.

2.3 Levels of Interpretation Failure

Approximating Clark and Schaefer's eight levels of understanding in human-human dialogue we define eight categories of communication failure in our architecture.

Level 0: The dialogue manager (DM) notices that no speech was received following a prompt to the user.

Level 1: Speech recognition component (SR) reports below-threshold confidence or cannot return any hypothesis at all for the current acoustic input.

Level 2: Sentence interpreter cannot parse the current utterance.

Level 3: Sentence interpreter cannot assign a semantic value to the current utterance in order to translate it to a well-formed CCSIL command. For example, "FIRST PLATOON INCREASE SPEED" is missing an argument for the velocity feature value and in "FIRST PLATOON CHANGE FORMATION TO THREE FIVE KILOS." there is a semantic type conflict.

Level 4: The utterance translates to a well-formed command in the backend language but execution of that command would result in the violation of a constraint in the Domain Model. For example, "FIRST PLATOON INCREASE SPEED TO 400 MILES PER HOUR" is well-formed but it violates the maximum velocity constraint of 60 kph..

Level 5: The command is well-formed and legal but infelicitous in the current discourse context due to a failed precondition in the Backend Model (BEM). For example, "FIRST PLATOON DECREASE SPEED BY TWO ZERO (20) MILES PER HOUR" results in a Level 5 failure if the current velocity of that platoon is only five miles per hour since execution of that command would result in the violation of a range constraint.

Level 6: The backend system rejects the input command for reasons that are beyond the DM's understanding of the backend system as represented in the Domain Model. In other words, our check against the Domain Model and BEM was insufficient to detect a problem.

Level 7: User notices that the backend application's response fails to satisfy the user's objective. Consider the case where the user says "FIRST PLATOON HALT" but the SR component returns "THIRD PLATOON HALT" albeit with a high confidence score. That error may go undetected through Level 6 if first platoon is in motion so that the BEM meets preconditions for halting. In this case, the system issues a command to the simulation to halt the wrong platoon.

2.4 Steps in the Recovery Algorithm

The generic four-step error recovery algorithm we posit for any spoken dialogue system begins when either of the dialogue participants detects an error condition.

1. Detection: In spontaneous dialogue, miscommunications can go undetected by both participants indefinitely. may discover years after the conclusion of a dialogue that I had misunderstood something said to me, or I may never detect the miscommunication. In automated dialogue systems, the SR, parser, or DP can fail to detect its own misinterpretation. If the problem goes undetected by subsequent interpretation processes, the backend application program fails to respond as the user intended and it is the user who must detect the error condition.

2. Diagnosis: Once an error has been detected it is classified into Level (0...7). Detection with no ability to diagnose can an interface that simply requests, "PLEASE REPEAT," regardless of error type, which is neither efficient nor pleasant as an interaction partner. Level 4 errors are detected by the Dialogue Manager with the help of a stored Domain Model which holds the knowledge base representation of constraints that backend system designers wish to have enforced by the user interface. Level 5 errors are detected by the dialogue manager after consulting the Domain Model and BEM. For a detected Domain Model violation, the actual cause could be the user misconception or slip of tongue, SR error, or interpretation failure.

3. Repair Plan Selection: Given a diagnosis, the system has flexibility in selecting a repair plan. In tutoring mode, the dialogue system usually hopes to catch any student misconception, and engage the student in a dialogue to correct the flawed beliefs or reasoning. But depending on the objectives for the immediate tutoring session, it may instead make sense to simply fix this error without notification, to avoid distracting attention away from the primary objective. Given the identical diagnosis, a training system might select a different repair plan to let the student experience the (simulated) consequences of certain errors. A well-formed but disastrous command can be admitted so that the student commander inadvertently destroys a friendly platoon. This experience of negative consequences is a key motivation for the use of simulated training environments.

Finally, when an interpretation module detects an error the design decisions may allow the DM to repair the miscommunication unilaterally, in the process of converting the utterance into a command, without bringing it to the user's attention. A command to the simulation to reduce by 20 mph the velocity of the a platoon traveling 5 mph, causes DM to halt the platoon and inform the user.

4. Execute the plan interactively: The human participant introduces unpredictability during the repair dialogue just as in the main dialogue. This means that execution of the plan constructed in Step 3, must be flexible in the face of a user's failure to conform to the selected plan. This step requires, in the limit, a full-blown reactive planning program. As an approximation we can implement a system that allows for clarification questions where the plan structure predicts a yes/no response to a repair question. We have thus far implemented no plans that grant flexibility to the user so we have little to say about this step.

3. CONCLUDING REMARKS

This paper reports on an in-progress implementation project in which our dialogue manager (DM) supports a spoken dialogue interface to a distributed battlefield simulation system. The DM acts as the mediator in a mixed initiative dialogue, interpreting between one user and the simulation application. For the purpose of this discussion, we have focused on the DM dialogue repair procedure, ignoring (a) the anaphora resolution mechanism, (b) updating routines that incorporate a new input or output utterance and its interpretation into the Discourse State, (c) the plan recognition component, and (d) processing of output utterances from the simulation system to the user. We have yet to address several difficult problems engendered by the multi-user distributed nature of the training system and treat only the dialogue management problem for a single-user interface to the distributed simulation in which other processes (human or machine) may affect the state of the represented world.

It is important to acknowledge that all four steps to our recovery algorithm involve uncertainty, and therefore risk. It may be impossible to determine whether a given unexpected utterance received by the DM is the result of a user misconception, a speech recognition error, or a weakness on the part of the DM itself; i.e., the Domain Model could be missing crucial knowledge about the backend system, DP may have failed during anaphora resolution (interpretation of context-dependent noun phrases), DM may have failed during error detection or diagnosis at some point in the dialogue history leading to a corrupted discourse context representation, etc. Therefore, what is most interesting and potentially fruitful in this enterprise is the discovery or development of reliable heuristics for carrying out the four steps. What we have provided here is an architecture into which those heuristics and rules can be inserted. and coordinated with other centralized discourse functionality.

For future work on the integration of discourse processing and speech recognition, we are looking at uses of Discourse State information to select among multiple alternatives (n-best) hypotheses produced by SR. Secondly, we want to allow the DM to alert SR in advance with a prediction for which speech act is expected next and therefore, which of a set of pre-trained language models to select as the language model for the next input utterance or which certainty threshold to adopt. Finally, we are negotiating a mechanism for SR to preserve and transmit to the discourse processor, prosodic data of value to discourse processing but not currently used by SR or NL sentence analyzer.

4. REFERENCES

1. Brennan, S. E. and E. A. Hulteen. (1993). *Interaction and Feedback in a Spoken Language System*. AAAI Fall Symposium Series Symposium on Human-Computer Collaboration: Reconciling Theory, Synthesizing Practice.
2. Clark, H. and E. Schaefer. (1987). Collaborating on Contributions to Conversations. *Language and Cognitive Processes*, pp. 19-41.
3. Grosz, B. and C. Sidner (1985) *The Structures of Discourse Structure*. Technical Report CSLI-85-39, Center for the Study of Language and Information. SRI International
4. LuperFoy, S. (1992). The Representation of Multimodal User-Interface Dialogues Using Discourse Pegs. Proceedings annual meeting of the Association for Computational Linguistics.
5. Montgomery, C. (1995) *Machine Aided Voice Translation*, presentation to US Army Technology Review Workshop on Spoken Language Processing for Language Sustainment.
6. Oviatt, S. L., P. R. Cohen and A. Podlozny (1990) *Spoken Language in Interpreted Telephone Dialogues.* SRI International Technical Note 496.
7. Perez-Quinones, M. A. and J. L. Sibert (1996) "A Collaborative Model of Feedback in Human-Computer Interaction," In proceedings of ACM SIGCHI Computer-Human Interaction annual meeting.
8. Salisbury, M. R., (1995) "Command and Control Simulation Interface Language (CCSIL): Status Update", Proceedings of the Twelfth Workshop on Standards for the Interoperability of Defense Simulations, 639-649, Orlando, Florida
9. Seneff, S., V. Zue, J. Polifroni, C. Pao, L. Hetherington, D. Goddeau, and J. Glass. (1995). Preliminary Development of a Displayless PEGASUS System. . Proceedings of the ARPA Spoken Language Systems Technology Workshop.
10. Terveen, L. (1991) PhD dissertation, University of Texas at Austin.
11. Traum, D. and James F. Allen, "A Speech Acts Approach to Grounding in Conversation," In Proceedings 2nd International Conference on Spoken Language Processing (ICSLP-92), pages 137-40, October 1992.
12. Wahlster, W. (1994) *Overview of User Modeling for Verbmobil*. keynote presentation to User Modeling biannual workshop.
13. Yamazaki, Y. and T. Morimoto: "ATR Research Activities on Speech Translation", 2nd IEEE Workshop on Interactive Voice Technology for Telecommunications Applications(IVTTA94), pp.61-66, 1994 (Kyoto).