Spoken Variable Initiative Dialog: An Adaptable Natural-Language Interface

Ronnie W. Smith, East Carolina University

Several obstacles have prevented spoken natural-language systems from providing the required performance, including inflexibility, ineffective goal-seeking, and poor speech recognition. However, results from using this system indicate that commercial applications are within reach.

Recent major advances in speech recognition technology have raised expectations about practical spoken natural-language interfaces. Such interfaces give users a flexible way to communicate with the computer, and allow them to keep their hands and eyes on the task at hand — repairing equipment, providing telephone assistance, piloting an airplane, driving a car, operating heavy equipment, and so on. Several obstacles have prevented spoken natural-language systems from providing the required performance, including inflexibility, ineffective goal-seeking, and poor speech recognition. But we can circumvent these problems by embedding the speech recognition technology within a dialog processing mechanism that

- uses problem solving to help the user carry out a task,
- conducts subdialogs (and switches among them as needed) to carry out the subgoals needed to complete the task,
- exploits knowledge about the user to determine what the user knows and what the user must be told,
- exploits context-dependent expectations about user responses to help interpret those responses, and
- engages in dialog in which the user can control the initiative (drives the task), the computer can control the initiative, or something in between.

Such a variable initiative dialog is a major advance that lets natural-language systems communicate effectively with novices (who need a computer-controlled dialog to lead them through the details), experts (who need occasional advice while pursuing their own strategies), and users who fall in between these extremes.

Based on a theory of natural-language dialog that addresses all these issues, my colleagues and I have implemented a system that uses spoken natural language to help users repair electronic circuits. Our integrated dialog-processing model combines a domain problem solver, a general subdialog mechanism, and knowledge about the user to provide timely and coherent assistance to the user. The robust parsing and language-understanding mechanism helps the system to correctly determine the meaning of utterances in spite of misrecognitions. Our results indicate that commercial application of this technology is within reach, and should stimulate thought about how to further improve the quality of spoken natural-language interaction.
Figure 1. The dialog processing architecture.

An integrated model of natural-language dialog processing

Our architecture integrates plan recognition, contextual reference, presuppositions, and user models into a single system (see Figure 1). The dialog controller coordinates the system's processing. The user task to which the dialog pertains is accomplished by completing a set of domain-specific task goals. As a result, the domain processor must recommend a current task goal to the dialog controller. Goal completion is described by a theorem; the dialog controller invokes the theorem prover to determine if a goal has been achieved (by proving a theorem). During theorem-proving, a user model containing axioms about the user is available. If the theorem prover cannot complete the proof because an axiom is missing, the dialog controller uses language to acquire the axiom. It computes a specification of the utterance needed to get the user to verbally supply the missing axiom.

For example, if the controller were trying to learn if there is a wire between connectors 84 and 99, then an appropriate axiom might be fact(wire(84, 99), exist, X) where X is either absent or present depending on whether or not the wire exists. If the knowledge base contains no such axiom, the controller might compute an utterance specification such as question(fact(wire(84, 99), exist, X)) and transmit it to the linguistic interface for verbalization (see the first utterance in Figure 2).

So, theorem-proving is a unifying feature that integrates mechanisms for several independently studied dialog phenomena. It provides the motivation for the dialog to acquire missing axioms. It yields a natural definition of the subdialog — all the language interaction about a recommended task goal. And it provides a natural method for subdialog clarification — simply modify the active theorem to include the new subgoals specifically mentioned.

Figure 3 illustrates this last point for the dialog segment in Figure 2. The dialog has reached main goal 1, where the wire between connectors 84 and 99 is of main interest to the domain processor. The dialog controller tries to carry out the goal of observing this wire, but is stymied by the lack of an axiom. The computer's first question and the user's response are required to acquire the missing axiom about the wire's status.

Once the computer learns that the wire is missing, it initiates a subdialog to add the wire (main goal 2). Utterance 4 initiates a clarification subdialog about this main goal. Consequently, the dialog controller instructs the theorem prover to insert into the active theorem a subgoal for learning how to accomplish the main goal ("learn to do add"). Resumption of theorem-proving leads to an expansion of this subgoal, leading the user through each substep about adding the wire. The first two substeps are to locate connectors 84 and 99. These are satisfied trivially by the theorem prover without the use of language. The knowledge in the user model indicating that the user knows how to locate these connectors was inferred from the user's utterance, "it is not there."

Utterance 7 then introduces the subgoal connect, after which utterance 8 leads to a second clarification subdialog for learning the subgoal's substeps. After completing the first two substeps, utterance 12 indicates the wire has been added. The system searches through the possibilities for missing axioms for the unsatisfied goals introduced at utterances 11, 7, and 3 to determine that utterance 12 supplies the missing axiom for the main goal (adding the wire).

Because each computer utterance addresses a specific missing axiom, the system has specific expectations for the user's response. For example, expected responses to utterance 3 include:
- that the user added the wire,
- that the wire exists,
- that the user does not know how to add the wire, and
- that the user needs help adding the wire.

Similar expectations for user responses at other points in the dialog can be computed from specific domain knowledge about the required action as well as from general dialog knowledge. By searching through
Figure 3. Relationship between theorem-proving and dialog for Figure 2.

the different sets of expected responses, the system can determine which goal is relevant to the user's utterance. In the case of utterance 12, main goal 2 is directly relevant rather than the subgoals introduced at utterances 11 and 7.

After the system determines that main goal 2 is completed, it discontinues proofs of the substeps (it does not even have to start the proof of the subgoal connect (end2, 99)), and then continues the dialog with main goal 3. If the user now realizes that the wire might have been misconnected, he or she might say "Where is connector 99?" in response to utterance 13. This is an interrupt — an unexpected change to another subdialog. Interrupts are detected based on the expectations for user responses in the active subdialog as well as the possible expectations of other subdialogs. A search through these expectations can determine that the user's utterance is relevant not to main goal 3, but to main goal 2, and the proof and corresponding subdialog about main goal 2 can be reopened.

Variable initiative: Giving priority to a conversant's goals

Sometimes a user has sufficient knowledge to accomplish several goals without much computer assistance. At other times, as in the dialog above, the user needs detailed assistance. These varying needs call for a variable initiative dialog: User initiative gives priority to the user's goals of carrying out steps uninterrupted, while computer initiative gives priority to the computer's goals. However, initiative is not an all-or-nothing prospect: either the user or the computer can have the initiative without having complete control of the dialog. We have identified four modes that characterize the computer's level of initiative in the dialog:

- **Directive** — The computer has complete dialog control. It recommends a task goal for completion and uses whatever dialog is necessary to obtain the knowledge to complete the goal. No interruptions to other subdialogs are allowed.
- **Suggestive** — The computer still has dialog control, but not as strongly. It suggests a task goal to perform next, but allows minor interruptions to closely related subdialogs.
- **Declarative** — The user has dialog control. He or she can interrupt to any desired subdialog at any time, but the computer is free to mention relevant, though not required, facts as a response to the user's statements.
- **Passive** — The user has complete dialog control. The computer passively acknowledges user statements, and provides information only as a direct response to a user question.

(This notion of dialog control extends an earlier one based on the utterance type of the speaker: question, assertion, command, or prompt.)

The computer can indicate its level of initiative only through its responses, so the choice of response topic is affected by the dialog mode. In directive and suggestive
mode, the computer has the initiative and should base its response primarily on its own goal. In declarative and passive mode, the computer should select as its response something it believes relevant to the topic something it believes relevant to the perceived user goal.

Suppose that the user states “the light is off,” and that the computer knows that for the light to be lit, the switch must be turned up to activate the power. The computer’s highest priority goal is now to put the switch up. However, depending on the mode, any of the following goals could be selected:

- **Directive** — Goal: make the switch be up. Possible computer response: “Put the switch up.”
- **Suggestive** — Goal: observe the current switch position. Possible response: “What is the switch position when the light is off?”
- **Declarative** — Goal: help the user learn that when the switch is up, the power circuit is activated. Possible response: “The power is on when the switch is up.”
- **Passive** — Goal: help the user learn that the computer understood the user’s last utterance. Possible response: “Okay.”

Figure 4 shows two dialog segments from our experiments with the implemented system (discussed below). In directive mode, the subject performs tasks under the close guidance of the computer, and there is language interaction about each task goal. In declarative mode, the subject independently carries out several necessary task goals without any interaction. By allowing the user to arbitrarily change subdialogs while in declarative mode, the computer can provide the relevant assistance when a potential problem is reported without requiring interaction for task goals already completed.

**Implementation and evaluation**

We implemented our dialog system to evaluate its problem-solving effectiveness in the various modes and to study user behavior with the system. The experimental system helps users repair a Radio Shack 160-in-One Electronic Project Kit. The circuit we focused on causes an LED to alternately display a 1 and a 7. The dialog system can detect errors caused by missing wires as well as a dead battery.

The dialog processor runs on a Sun 4 workstation, with most of the code written in Quintus Prolog, and the parser in C. (We later implemented an improved parser on a Sparc-2 workstation, yielding an average computer response time of 2.2 seconds.) Speech recognition is performed by a Ver-"ex 6000 running on an IBM PC. To improve speech recognition, we restricted the vocabulary to 125 words. A DECTalk DTCC1 text-to-speech converter provides spoken output.

Eight undergraduates used the system in the experiments. The students had completed one computer science class and were taking a second, but none had taken any AI classes nor had they used a natural-language system. Furthermore, none were electrical engineering students, who probably could have fixed the circuit without any computer assistance.

The subjects first recorded their voice patterns for the speech recognizer and practiced using the system on up to four “warm up” problems in which the computer had maximum initiative (directive mode). The subjects then participated in two problem-solving sessions, one in directive mode, and one in declarative mode. Because we have not yet developed a good strategy for automatically changing modes during a dialog, we locked the mode for the duration of a dialog to study the effects of different levels of initiative.

Each subject could work on up to ten problems in each problem-solving session. A total of 141 problems were attempted, of which 118 (84 percent) were successfully completed within our artificial time constraints. (We estimate that 95 percent would have been completed if we had allowed more time.) Of the remaining 23 problems, 22 were terminated early due to excessive time spent on the dialog. Misunderstandings due to misrecognition caused 13 of these failures. Misunderstandings due to inadequate grammar coverage occurred in three failures. In four failures the subject misconnected a wire, in which case the domain processor did not have sufficient knowledge to properly assist the user. In one failure the subject was confused about when the circuit was working. In another there were problems with the system software, and a hardware failure caused termination of the final dialog. Overall, more
than two-thirds of the failures were due to miscommunication, and the vast majority of these were due to misrecognition by the computer. In fact, the speech recognizer correctly recognized only half of the 2,840 utterances word for word.

However, 81.5 percent of the utterances were still correctly interpreted due to our use of an error-correcting parser that transforms grammatical input from the speech recognizer into a grammatical utterance. The parser can insert and delete words, and substitute phonetically similar words such as “eight” and “it.” The “cost” of each such correction depends on its effect on the utterance’s meaning: Inserting or deleting “not” usually has a high cost, while inserting or deleting “a” and “the” usually has a low cost. Each utterance also has an expectation cost based on the likelihood that the utterance would be spoken in the current context. The parser looks for the correction with the lowest overall cost, which is a function of the correction cost and expectation cost.

Despite the parser, 18.5 percent of the utterances were misinterpreted. Due to the potential for confusion, the experimenter was allowed to notify the user whenever a serious misrecognition caused an incorrect interpretation. For example, when one subject said “The circuit is working,” the speech recognizer returned the words “Faster it is working,” which the system interpreted as the phrase “faster.” The experimenter then told the user, “Due to misrecognition, your words came out as ‘faster’.” The experimenter did not tell the user what to do, but merely described what happened. In this way, we tried to restrict the interaction to that between the computer and user as much as is possible given the quality of current commercial, real-time, continuous speech recognition devices, and the domain vocabulary with many phonetically similar words (such as “switch” and “which,” or “eight” and “it”).

The effects of initiative. The two levels of initiative affected the test results as we had expected. When users had the initiative they tended to

- speak longer utterances (37 percent of the utterances in declarative mode were one word versus 60 percent in directive mode).
- speak fewer utterances that had actually been interpreted correctly. Reducing misrecognition and the possible confusion it entails remains an open research problem.

The Minds and Tina systems have been built to study speaker-independent natural-language understanding with a larger vocabulary. Minds was the first spoken natural-language system to exploit high-level dialog information to aid speech recognition. As in our system, Minds uses domain- and dialog-specific knowledge to predict user responses. These predictions constrain the search for the next user utterance during speech recognition, improving word recognition accuracy from 82.1 percent to 97.0 percent and semantic accuracy from 85 percent to 100 percent in a test of 200 spoken sentences.

Tina provides a portable spoken natural-language interface to database query systems. Tina uses probabilistic networks to provide expectations for the occurrence of words during parsing. The networks are created dynamically from context-free grammar rules where the probabilities are derived from frequency counts on the rules generated in parsing a set of training sentences selected by the system designer. Tina successfully parsed 84 percent of a 200-sentence test set in a domain with a 965-word vocabulary, and successfully parsed 76 percent of a 560-sentence test set in another domain.

Another avenue of research is selective automatic verification, whereby the computer asks the user, “Did you mean to say X,” where X is the computer’s interpretation based on the recognized words. Verification is only done when the correction cost of the selected interpretation exceeds a given threshold, implying that substantial inaccuracies are likely in the recognized words. Such a system was constructed after we completed our experiment. A system including verification correctly interpreted up to 97 percent of user utterances, although it also verified 22 percent of the utterances that had actually been interpreted correctly. Reducing misrecognition and the possible confusion it entails remains an open research problem.

Acknowledgments

This research was supported by National Science Foundation Grant Number NSF-IRI-88-03802, and by Duke University. Other researchers who have contributed to this project include Alan W. Biemann, Robert D. Rodman, Ruth S. Day, D. Richard Hipp, Dania Egedi, and Robin Gambill. I am also grateful to the anonymous referees, whose comments helped greatly in preparing this article.
References


Ronnie W. Smith is an assistant professor in the Department of Mathematics at East Carolina University in Greenville, North Carolina. His research interests include natural-language generation and dialog, experimental evaluation of AI systems, human-computer interaction, and computer science education. He received his PhD from Duke University in 1991, and his BS in mathematics and computer science from Vanderbilt University in 1983. He is a member of ACM and the Association for Computational Linguistics.

Readers can contact Smith at the Department of Mathematics, East Carolina University, Greenville, NC 27858; Internet: masmith@ecuvax.cis.ecu.edu