(Brennan, S. E. & Hulteen, E. (1995). Interaction and feedback in a spoken language system: A theoretical framework. *Knowledge-Based Systems, 8,* 143-151.) Copyright © 1995 Elsevier Science B.V. All rights reserved.

INTERACTION AND FEEDBACK IN A SPOKEN LANGUAGE SYSTEM: A THEORETICAL FRAMEWORK

Susan E. Brennan
Department of Psychology
State University of New York
Stony Brook, NY 11794-2500
brennan@psych.stanford.edu

Eric A. Hulteen
Apple Computer, Inc.
1 Infinite Loop, MS: 301-3H
Cupertino, California 95014
hulteen@apple.com

KEYWORDS: *Speech, dialog, feedback, conversational repair, agents, telephony.*

ABSTRACT

In human-computer interaction, as in conversation, neither partner is omniscient. To facilitate repairs when problems arise, an interface needs to enable both user and system to coordinate their separate knowledge states. We present a conversational feedback model for human-computer interaction, based on a collaborative theory of human communication (Clark & Schaefer, 1989) and use this model to systematically provide context-sensitive feedback messages from an application-independent spoken language system. We then describe a simulation, an informal user study, and a working prototype that use this model in a telephone agent application that allows dialing by voice.

TOWARD A MORE ROBUST SPEECH INTERFACE

Traditional approaches to improving the performance of spoken language systems have focused on improving the accuracy of the underlying speech recognition and natural language processing technology. The assumption is that if a system can translate exactly what the user said into text and then map this onto an application command, speech will be a successful input technique. With this approach, improving speech recognition accuracy requires asymptotic effort that ultimately is reflected in the cost of the technology.

We argue that perfect performance by a speech recognizer is simply not possible, nor should it be the goal. There are limiting factors that are difficult or impossible to control, such as variability in the acoustic environment. Moreover, many words and phrases in English are homophones of other words and phrases, so in some situations, both human and machine listeners find them ambiguous. People frequently have trouble discerning, remembering, or guessing the grammar and vocabulary that a system expects and then limiting themselves to it - this has been dubbed "the vocabulary problem" by Furnas, Landauer, Gomez, and Dumais, 1987. In addition, ambiguous input is a problem on the syntactic, semantic, and pragmatic levels. Finally, because people have many other demands on them while they are speaking such as performing tasks, planning what to say next and monitoring their listeners and the environment, they frequently do not produce the kind of fluent but constrained speech that a speech recognizer has been trained to process. Human utterances frequently contain speech errors, and yet this is rarely a problem for human listeners. The intrinsic limits on the well-formedness of utterances and on the accuracy of speech recognition technology suggest that to solve the problem, we must first redefine it. Let us start by considering how people handle these problems in conversation.

THE COLLABORATIVE THEORY OF CONVERSATION

Having a conversation is *not* simply the encoding and decoding of well-formed messages. Conversation is collaborative. When there is a misunderstanding, both conversational partners participate in repairing it, and in doing so, they try to expend the *least collaborative effort* (Clark & Wilkes-Gibbs, 1986). That is, one partner often puts in extra effort in order to minimize the effort both partners expend collectively. In addition, shared meanings in conversation are constructed incrementally. People coordinate their individual knowledge states moment by moment, by systematically seeking and providing evidence about what has been said and understood; this is the process of *grounding* (Brennan, 1990; Clark & Brennan, 1991; Clark & Schaefer, 1987; 1989; Clark & Wilkes-Gibbs, 1986). The amount of effort that both partners expend at one point in a conversation before moving on is governed by their *grounding criteria* (Clark & Wilkes-Gibbs, 1986). The higher the grounding criteria, the more evidence conversational partners will require before concluding that an utterance is indeed part of their shared knowledge or common ground. For example, a speaker expects more explicit evidence about what a listener understands when they are discussing something important and less evidence when the purpose is casual.

Recognizing and repairing problems in communication requires the right kind of evidence at the right moment. It is important for a conversational partner to provide evidence when she notices that something is amiss; we will call this *negative evidence* of understanding. Many approaches to modeling language use have assumed that negative evidence is sufficient to keep a conversation on track (see, for instance, Steedman & Johnson-Laird, 1980). But evidence of what *has* been understood — *positive evidence* — is necessary as well (Brennan, 1990; Clark & Brennan, 1991). Positive evidence in human conversation includes precisely timed short utterances that lack what is typically considered propositional content of their own, such as "uh huh," "hm," and other backchannels (Yngve, 1970), explicit acceptances such as "ok," clarification questions, relevant next turns (as well as relevant actions), and even continued eye contact. Positive evidence is necessary because in a dialog neither partner is omniscient. At any moment, a speaker and a listener are likely to have different takes on what has just been said. It is possible that the addressee didn't hear the utterance, or that he didn't parse or interpret it as the speaker intended, or that he didn't even realize he was being addressed. Part of the work of repairing a misunderstanding or ambiguity is in recognizing that one has occurred. A misunderstanding or ambiguity cannot be reliably identified unless conversational partners seek and provide positive evidence (when things are going smoothly) as well as negative evidence (when a partner recognizes a problem or potential problem).

When the situation involves delegating an action to an agent, it is especially important that the speaker have evidence about whether the action was successful. This feedback is just as necessary when the currency of interaction is speech, textual natural language, or a command language as it is when the currency is icons, desktops, and windows. The collaborative view of human language use has been used as a basis for modeling structure and feedback in human-computer dialog for natural language interfaces (Brennan & Cahn, 1994) and for menu-based graphical interfaces (Payne, 1990).

When two partners interact, neither has *direct* access to the other's beliefs or mental state. A speaker may start by expressing an intention that she expects a listener to recognize (Grice, 1967). Upon hearing the utterance, the listener displays evidence about his intentions or mental state for the speaker to interpret. The speaker depends on such evidence from the listener to hypothesize that she has been understood, and will seek such evidence if the listener doesn't provide it spontaneously (Brennan, 1990). In a model of collaborative contributions to conversation proposed by Clark and Schaefer (1987, 1989), there are four possible states that an addressee, B, can inhabit with respect to a speaker, A, and an utterance, u:

State 0: *B didn't notice that A uttered any u*
State 1: *B noticed that A uttered some u (but wasn't in State 2)*
State 2: *B correctly heard u (but wasn't in State 3)*
State 3: *B understood what A meant by u*

The need for conversational repair arises either when a listener believes he has failed to reach one of the successive states, or when he believes he has reached a particular state, but the positive evidence he provides leads the speaker to conclude that he hasn't. In the first case, a listener may recognize there is a problem and initiate a repair himself. In the second case, the speaker may notice the problem first and initiate a repair if the evidence the listener provides is not what she was expecting. In the examples of speech interaction with a spoken language system that follow, we are concerned with both of these situations.

A MODEL OF GROUNDING WITH A SPOKEN LANGUAGE SYSTEM

Most spoken language systems provide some feedback to users, but in a rather ad hoc way. For instance, a system can echo a user's input or ask for confirmation of a command. But users find this cumbersome if it happens all the time. In addition, informative error messages can be produced when certain kinds of processing errors occur. But this is no guarantee that, in the absence of an error message, the user won't be left wondering whether she has been understood. Our goal is to address these problems in a more adaptive, context-sensitive, and systematic way. To do this, we have extended Clark and Schaefer's (1987) model in order to specify an adaptive feedback model for a spoken language system to which a user can delegate actions.

Here is an ordered list of system states, labeled from the perspective of the user:

State 0: Not attending. *The system isn't listening or doesn't notice that the user has spoken.*
State 1: Attending. *The system has noticed that the user has said something, but it hasn't interpreted the words.*
State 2: Hearing. *The system was able to identify some of the user's words, but hasn't parsed the entire utterance*
State 3: Parsing. *The system received what seems to be a well-formed utterance, but hasn't mapped it onto any plausible interpretation.*
State 4: Interpreting. *The system reached an interpretation, but hasn't mapped the utterance onto an application command.*
State 5: Intending. *The system has mapped the user's input onto a command in its application domain, but hasn't acted yet.*
State 6: Acting. *The system attempts to carry out the command. It is not known yet whether the attempt will have the desired outcome.*
State 7: Reporting. *The system may or may not have been able to carry out the user's command, and reports any evidence available from the application domain.*

Our States 0-2 correspond to Clark and Schaefer's. We have divided Clark and Schaefer's State 3 into our

States 3 and 4, since some modular language-processing systems can break down distinctly in their syntactic or semantic modules (see, for instance, Brennan, 1988). States 5-7 are necessary extensions for dialogs involving delegation to an agent. Studies of grounding in task-oriented human conversation (e.g., Brennan, 1990; Clark & Schaefer, 1989; Schober & Clark, 1989) lead us to forecast that feedback at these states would be useful in speech interaction with a machine; these states are intended to mirror the points where human speech processing and action can break down. However, not every language interface and every application will require distinctive or meaningful feedback at all these states. For instance, a text-based natural language interface would not need a state corresponding to State 2, hearing. Likewise, a spoken language system without a parser (one that simply maps a pattern onto a command) would not distinguish States 2 and 3, hearing and parsing. For an application that doesn't involve delegation (such as a question-answering system), feedback at States 5 and 6 would probably not be interesting (nor would feedback at these states be provided, as we shall see). The states of the model can be adapted to the particular spoken language system so that only feedback that is potentially useful to the user is provided.

We adopt the simplifying assumption that a state depends on the states numbered below it. There are of course exceptions to this; if an agent discovers that the goal it intends to act on has already been achieved, or if it guesses or predicts a user's intention without having correctly heard the user's command, then the agent need not pass through these states in sequence. Ideally, however, evidence of success at a particular level obviates the need for evidence at lower levels, with exceptions that we discuss later.

Next we will walk through the model and show how it performs in principle, using as an example an application consisting of a remotely accessible telephone agent. We will demonstrate feedback for each state in the model through a series of examples. Then we will describe two prototypes that we have built in order to test the model.

FEEDBACK AT DIFFERENT STATES

The model assumes that positive evidence is just as important as negative evidence. To illustrate each state, we will begin with examples of negative evidence a system gives when it recognizes itself to be in that state but unable to proceed to the next state. Then, we discuss how and when a system should give positive evidence of having reached a state. Many user interfaces provide negative evidence (error messages) when a problem is identified. But without positive evidence, it is much harder for users to identify problems that the system cannot.

State 0: Not Attending

While our telephone agent is processing an utterance, it briefly stops attending to input from the user. Since it cannot identify the "error" that occurs if the user tries to address it when it's not attending, there is no error message associated with this state.

In user studies of spoken language systems that we and others have conducted at Apple, we have discovered that users sometimes have trouble knowing when to speak. This can happen even when the system is used in a push-button-to-talk mode. We are currently investigating the use of audible cues (such as an unobtrusive ambient sound) to give positive evidence whenever the system is not attending (or alternatively, when it *is* attending)

Example of State 1: Attending

When the agent detects sound but cannot recognize an utterance, it asks the user to repeat.

User: *"Call Susan."*
System: *"What was that again?"*

When the system is attending but fails to hear any speech, it can provide a prompt (after a time-out) such as: "I'm still listening." Since repeated prompts can be annoying, it is better to convey the system's attentional state with ambient cues. For instance, the PlainTalk^{TM} speech interface available on the Macintosh Quadra 660AV and 840AV machines provides several different styles of continuously present graphical cues that the system is listening (Strong, 1993; Apple Computer, Inc., 1993).

Example of State 2: Hearing

User: *"Call Lewis."*
System: *"Sorry, I didn't get that - call who?"*

The important thing about providing feedback about this state is to make it clear to the user what part of an utterance has been detected and what part is missing. By responding "Call *who*?" the system clearly specifies that it got the command "Call;" the system uses the type of the missing argument to fill in an appropriate wh-word. In the context of a repair initiated by a State 2 error message, the system can accept either a repeat of the whole utterance or an elliptical response ("Lewis"). In a non-repair context, the utterance "Lewis" would elicit a different State 2 error message, e.g.. *"What was that about Lewis?"* Error messages like this were used in an early spoken language interface called "Put That There," developed at M.I.T. (Schmandt & Hulteen, 1982). With very little trouble, users were able to help that system repair utterances that it had partially recognized.

Note that the style of the messages provided by this feedback model need not be anthropomorphic. "Still listening" or "ready for input" may be preferable to "I'm still listening." Here, we have chosen to emulate the ways in which human conversationalists typically mark problems at different stages of language processing. See Brennan and Ohaeri (1994) for an empirical comparison of anthropomorphic, non-anthropomorphic, and telegraphic message styles in language interfaces.

Examples of State 3: Parsing

User: *"Call Eric."*
System: *"Do you mean Eric Hulteen or Eric Stubbs?"*

When the system receives a well-formed utterance that it can't interpret unambiguously, it initiates a repair by expressing the alternative interpretations as a spoken menu of choices. Another problem that calls for negative evidence at this state is when the user asks it to call a person it doesn't have any number for:

User: *"Call Eric."*
System: *"sorry, Eric isn't in my directory."*

If the telephone agent were to include the ability to add new entries to its directory, the system's could take more initiative for the repair: "Do you want to create a new entry for Eric?"

Positive evidence of parsing is provided by echoing the user's input. In the next example, the system provides positive evidence of appearing to find itself in State 3, while the user recognizes that it has heard part but not all of the intended utterance (so the system is actually in State 2):

User: *"Call Lisa."*
System: *"I heard you say call Lewis."*
User: *"No! I said, call Lisa."*

When the speech recognizer has been misrecognizing many of a user's utterances, it is particularly useful to provide positive evidence at State 3. We track this need by keeping a record of repairs in the discourse history. Informative feedback provided at an earlier state gives the user more time to interrupt and initiate a repair. Interrupting can be done in a variety of ways; with our system, the user interrupts by pressing a key on a keyboard or telephone keypad designated for this purpose. The user may be able to interrupt vocally if the speech recognizer is always listening for input. Positive evidence at State 3 has turned out to be particularly crucial for our telephone agent application.

Examples of State 4: Interpreting

User: *"Find Chris."*
System: *"Sorry, I don't know how to find Chris."*

Here, the user's utterance has been heard, parsed, and mapped onto an interpretation. In this case, the system cannot map its interpretation onto a command that it can perform. It is possible that when the system gives an error message at this state, the actual problem was at an earlier state (the utterance may have been misrecognized). Giving explicit feedback in the error message about exactly what was interpreted enables the user to recognize a problem that the system cannot.

If it is critical to avoid errors in a particular task operation, then the system can check its interpretations explicitly by asking for confirmation:

User: *"Call Lisa."*
System: *"Do you want me to call Lewis?"*

We have chosen to express positive evidence at this state as an *explicit* request for confirmation since expressing it as a statement ("You want me to call Lewis") is an *implicit* request. Requests for confirmation can be tedious, but they do ensure that the system will not place any embarrassing calls. In general, State 4 feedback should be provided sparingly, only for those task operations that are potentially problematic - that is, when there is significant potential cost to misinterpreting what the user intends, and when there is significant likelihood of misinterpretation. This amounts to the system adopting a higher grounding criterion than it does with operations that can be undone, that are not destructive, or that don't have social consequences. State 4 feedback should reflect information from the recent discourse history about misrecognized commands. For instance, if the system reached State 5 in processing a recent command, but was interrupted by the user before placing a call, it should give positive evidence at State 4 (ask for confirmation) the *next* time it interprets a command to call. This kind of explicit feedback about a system's interpretation is typically given in both command and direct manipulation interfaces, e.g. when users are asked to confirm before quitting or before a file is deleted.

Examples of State 5: Intending

Negative evidence for State 5 is necessary when the user asks the agent to perform an interpretable action that it cannot carry out, often because the application is in an incompatible state. Our telephone agent provides negative evidence for State 5 when the user's two-line telephone already has two calls on hold and no line is available for making a third call.

User: *"Call Lisa."*
System: *"I can't call Lisa - there is no available line."*

Positive evidence that the system is about to act but has not yet acted is important whenever a user's command is potentially destructive or has the potential to involve other people. So our system often provides positive evidence of reaching State 5, even when it is not arrested at this state. Here is an example of positive evidence:

User: *"Call Lisa."*
System: *"Ok, I'll call Lisa."*

With this response, the system tells the user it is about to act on her command; it also explicitly repeats her command as part of the message. A similar strategy was employed to give users feedback about how a natural language interface had interpreted pronouns (Brennan, 1988). A State 5 message implies that the system has also passed through States 0-4 successfully (attending, hearing, parsing, and interpreting) and that the system is confident about its performance so far. If the command is what the user intended, the user need not respond. But consider this exchange:

User:	*"Call Lisa."*
System:	*"Ok, I'll call Lewis."*
User:	*"No! I said call Lisa."*

Even with experienced users and a state-of-the-art speech recognizer, the system may have misrecognized all or part an utterance. Here, the evidence the system provides shows that it identifies itself to be in State 5, but the user can tell by this evidence that in fact it is not. Without positive evidence of what the system understands at this point, the user will not recognize that there is a problem until she gets the wrong person on the phone. Again, positive evidence gives the user time to interrupt and repair the problem.

Example of State 6: Acting

Evidence is particularly informative during system actions that take a noticeable length of time to carry out, or during actions that contain intermediate steps that can go awry. We found that our telephone agent didn't need an error message at State 6. It does, however, provide positive evidence of acting:

User:	*"Call Yin Yin."*
System:	*(sounds of touch-tone dialing)*

Positive evidence about this state informs the user that the system is in the process of carrying out a command. At that point it is still possible to interrupt the system. Feedback at State 6 is not necessary when the action is carried out instantaneously (faster than the time it takes to display the feedback). Feedback at State 6 is quite common in direct manipulation interfaces, where there is immediate positive evidence of the effects of a user's actions. Direct manipulation systems sometimes offer explicit opportunities to interrupt during what we would call State 6; for instance, the system may inform users of its progress at printing or copying files, while providing a "Cancel" button option.

Example of State 7: Reporting

User:	*"Call Lewis."*
System:	*"I'm sorry, the line's busy. Shall I keep trying?"*

In this case, the system has acted on the user's command and it reports the outcome. This state is particularly important in an interface to which the user can delegate actions that can either succeed or fail. In our telephone application, the agent evaluates whether the outcome of its attempt to place a call is a success or a failure; such an evaluation is not possible (or relevant) in all tasks. When that is the case, the system simply reports whatever evidence is available from the application, leaving it up to the user to conclude success or failure. Feedback about this state need not be speech; for spoken language systems that control graphical applications, the feedback can be a visual result (Brennan, 1988).

KNOWLEDGE SOURCES

Our approach is intended to provide a systematic way to specify feedback to the user while supporting both domain-independence and a variety of system architectures (from sequentially modular to blackboard-style). Three sources of knowledge - the dialog history, the physical environment, and the task model - are used to determine what kind of feedback messages to provide and when to provide them. These sources of knowledge come together in a single parameter that is frequently updated: the system's grounding criterion. The dialog history is a data structure that is updated turn by turn, every time the system or the user makes a contribution to the dialog. The dialog history has other uses in addition to determining feedback, such as resolving referring expressions. The physical environment we represent simply as two possibilities: a noisy or quiet speech input line. The task model lists the set of commands available in the telephone agent application and maps each of them onto an appropriate relative adjustment to the grounding criterion that is triggered when an utterance gets mapped onto a command (in State 5, intending).

In the next section, we describe our initial scheme for setting the system's grounding criterion, which in turn determines the amount and placement of feedback. It is conceivable that additional parameters, rules, or decision processes should also affect the grounding criterion, for instance, in the generation of multimedia responses (see Brennan, 1988; Hovy & Arens, 1993). For a discussion of how the grounding process differs according to communication medium, see Clark & Brennan, 1991.

HOW MUCH EVIDENCE TO PROVIDE? ADAPTING FEEDBACK WITH THE GROUNDING CRITERION

We have argued that a spoken language system needs to give positive evidence of understanding. But how much evidence should it give, and when, exactly? Consider this set of positive messages associated with each state at which the system can be successful at processing the user's utterance:

State 1:	*"I'm still listening."*
State 2:	*"I heard something."*
State 3:	*"I heard you say 'Call Lewis'."*
State 4:	*"Do you want me to call Lewis?"*
State 5:	*"OK, I'll call Lewis."*
State 6:	*(telephone dialing sounds)*
State 7:	*"Here's your call."*

Of course, it would be tiresome for a system to provide evidence for every state that it passes through successfully; for one thing, this would violate Grice's maxim of quantity (Grice, 1967). If we assume that reaching a particular state presupposes that the lower-numbered states have also been reached, then it may be enough to simply provide positive evidence about the most advanced state of processing that the system appears to have reached. This is what our system does when its grounding criterion is low.

Next we give examples of how the grounding criterion is adjusted according to information from the three knowledge sources mentioned earlier: the task model, the dialog history (which includes an implicit model of the user), and a model of the physical environment.

The Task Model

The grounding criterion is low when the system has mapped the user's command onto an ordinary task. Ordinary tasks include those that are not destructive, do not involve other people, or can be undone. For tasks that have a high probability of error or that have potentially destructive or embarrassing consequences, the model allows for giving more feedback than usual — that is, feedback about earlier states — by raising the system's grounding criterion. When the task at hand is placing a call, the grounding criterion is incremented with the result that additional feedback is provided at an earlier state (as in the examples for State 5) to give the user the opportunity to interrupt before the call goes through.

Consider the situation where there are several phone numbers associated with one individual:

User:	*"Call Chris."*
System:	*"OK, I'll call Chris DiGiano at his office."*

Here the agent provides explicit evidence of how it has interpreted the command (in this case it has chosen the number where the callee is most likely to be during business hours). If the agent has no heuristic for choosing between several possibilities, it asks for more information from the user in order to proceed from State 3 to State 4:

User:	*"Call Chris."*
System:	*"Shall I call Chris DiGiano at home or at his office?"*

Of course, to implement this successfully requires that the interface designer have a good idea of just what can go wrong. We acquired this information by prototyping the telephone agent and observing people trying to use it.

The Dialog History And User Model

The telephone agent adapts its feedback both to the user and to how the dialog is going. The grounding criterion starts out high until the user and system have managed several smooth exchanges in the dialog. A smooth exchange is counted when the system recognizes an utterance of the user's and maps it onto an interpretation, an intention, an act, and a report (e.g., reaches State 7), without having to initiate a repair and without the user having to initiate a repair. If the user initiates a repair, the dialog history records that exchange as having a repair nested in it; this record includes whatever information is available about the utterance's surface form and the type of command it was mapped onto (see Brennan & Cahn, 1993).

Most repairs in our system are of misrecognitions; these problems must be noticed and repaired by the user. How does the system determine the appropriate feedback to give? Consider these options:

User: *"Call Susan."*
System: *(One of these)*
"I heard you say call Susan. Do you want me to call Susan? (Highest criterion: feedback at States 3 and 4)
"Do you want me to call Susan?"
(High criterion: State 4 feedback)
"OK, I'll call Susan."
(Medium criterion: State 5 feedback)
(telephone dialing sounds).
(Low criterion: State 6 feedback)

In this example, the system needs to give feedback at a state earlier than State 7 because placing calls is a high-criterion action. If this exchange is early in a new user's dialog with the system, the grounding criterion is high and the system gives feedback at State 3 displaying the utterance it parsed and at State 4 that requests confirmation. If the user's response to the request for confirmation is "yes," then that interchange is recorded as a smooth one in the dialog history and the grounding criterion is lowered slightly. After a second "yes" to a request for confirmation, the criterion is lowered a bit more; the consequence is that the next time this user asks to place a call, the system will not ask for explicit confirmation, but will provide feedback at State 5, giving the user time to interrupt if she detects a misrecognition. Because placing a call is a high-criterion action and the dialog history is seldom error-free, the system would not normally lower its criterion enough to provide only State 6 feedback for placing a call (although it does this for low-risk actions).

At present, we use simple heuristics to adjust the grounding criterion. Currently the dialog history tracks turns in which the user interrupts or corrects the system (e.g., when a user responds "no" to a clarification question that counts as State 4 feedback). Then, when the system arrives at an interpretation that contains a command that has been problematic, it temporarily adjusts the grounding criterion upward. While we intend the states and grounding criterion in our model to be generalizable to other kinds of applications, we do not yet have an algorithm for adapting the grounding criterion that we believe is generalizable to all tasks and all users. We plan to investigate this further as we try out the model in other application domains.

Finally, some of our users have more trouble than others in getting their utterances recognized by the system (particularly if they speak English with a foreign accent). When the system detects unusually frequent repairs of misrecognitions or is arrested repeatedly in States 2 or 3, it uses a higher grounding criterion and consistently asks for confirmation.

The Physical Environment

Sometimes there is noise in the user's environment or in the audio input to the speech recognizer. If the system detects noisy input or frequent misrecognitions, then it adopts a higher grounding criterion and gives feedback both at State 3 (parsing) in addition to feedback at a later state (in this example, State 4):

User:	*"Call Eric."*
System:	*"I heard you say 'Call Eric'. Do you want me to call Eric?"*

This level of feedback is reassuring in the face of noise or repeated error, but is quite tedious when the dialog is going smoothly. Currently we lower the grounding criterion after two smooth contributions to the dialog history (e.g., when the users has responded "yes" to two requests for confirmation and allowed the system to proceed). This results in feedback at States 3 (echoing the user's utterance) and 6 (dialing sounds).

TESTING THE FEEDBACK MODEL

We tested the feedback model initially by building two prototype systems and conducting an informal user study, consistent with the philosophy expressed as "user-centered design" (Norman & Draper, 1986; Gomoll, 1990; Hulteen, 1991). User studies differ from controlled laboratory experiments in that they try to get as much information about usability as possible for the least amount of time and effort invested, immediately feeding back changes inspired by users to the prototypes themselves. Our goal at this stage was to test and iteratively tune the system concept as a whole and to discover any areas where implementation or use might show our model to be problematic

The first prototype was a simulated speech interface to a telephone agent that we studied using a "Wizard-of-Oz" paradigm; the second was a working prototype of a telephone agent.

Wizard-of-Oz Study

First, we developed a HyperCard-based simulation of the telephone application to support a Wizard-of-Oz study where we simulated the telephone agent with its spoken language interface. The simulation was connected to a speech synthesizer and a text parser. It represented a two-line telephone system and provided the experimenter with a control panel for generating telephone sound effects and text-to-speech feedback. Simple functions were supported, including: dialing by name, dialing by number, holding, switching lines, picking up calls, and hanging up. Users were 12 employees of Apple Computer, Inc. The users role-played several telephone-based tasks. They were instructed to imagine calling their "phone agent," e.g. the voice interface to their personal office telephone, from a pay phone. They were instructed to press the # key when they wanted to address or interrupt the voice interface and to speak into the telephone handset. There were no instructions about the kind of language the agent would understand. Subjects were aware that this was a simulation. As they spoke, the experimenter rapidly typed their input into the text parser and used the buttons on the control panel to provide appropriate telephone sounds and synthesized speech messages from the telephone agent in response. The subjects' input and the system's responses were logged for later review. After the session, subjects filled out a questionnaire and discussed their reactions with the experimenter.

The tasks included: calling a coworker, placing that call on hold and calling another coworker, terminating that call and returning to the first call, answering an incoming call, and placing a call to someone at a car phone.

For this brief initial test, we used a high grounding criterion, since our users were inexperienced with the speech recognizer. This meant that after each of the user's utterances, the system gave State 3 feedback ("I heard you say...") in addition to feedback at a higher level. The tasks were planned so that most of the time the user's utterances would be "understood" by the system. However, several errors were scripted into the system's behavior during the task, to see if users could use natural conversational strategies to get back on track. These errors included missing one word of the user's command and completely misunderstanding the user's command (that is, hearing a different command). When the system missed one word of the user's command, it gave feedback at State 2: "I heard you say call somebody - who do you want to call?" Most of the users (83%) responded immediately with an elliptical answer (e.g., "Susan"); the rest reissued the whole command. When the system got the wrong command, it responded with feedback at States 3 and 6 ("I heard you say call Joy; calling" [dialtones]). Although all the users had been instructed to interrupt the system when necessary by using the # key on the handset, only two thirds of them did so; the rest tried to interrupt verbally. Even those who remembered to use the # key felt that it would be more natural to interrupt verbally.

When the simulated speech recognizer was consistently successful in recognizing utterances, many users reported that messages from State 3 were annoying. On the other hand, after misrecognitions, users reported that they found messages about this state helpful and came to depend on them for evidence that their input had been interpreted as they intended. This preliminary study supports the prediction from the model that feedback should be provided concerning the highest possible state the system reaches, except when it is error-prone at an earlier state.

Working prototype

We implemented a second prototype in LISP, for demonstration purposes. We used a custom-built state-machine to track and control the telephone agent application, speech recognizer, text-to-speech synthesizer, and dialog history manager.

This prototype used PlainTalk™, a speaker-independent, connected-speech recognizer developed at Apple Computer, Inc. (Lee, 1993). This speech recognizer uses a limited grammar and is remarkably fast and successful at recognizing the utterances of users who are experienced enough to conform to its vocabulary and grammar. But it performed considerably less well with some of the inexperienced users of our demonstration system, who had no idea of the vocabulary or the grammar that had been defined for it.

In addition, as we expected after the Wizard-of-Oz study, users of the working prototype had problems determining when the system was listening for input. We conclude from this that if a speech recognizer cannot listen all the time for input, there needs to be some cue (audible or graphical) about whether or not it is listening. Feedback from a spoken language system need not be in the form of speech; we are currently experimenting with sounds and graphics to be used along with verbal messages.

Contributions of this and future work

The contribution of this work is to provide a general solution to providing feedback for speech applications. The data structures to support several of the states in our feedback model have been incorporated into the PlainTalk™ speech interface that is part of the Macintosh Quadra 660AV and 840AV machines (Strong, 1993; see also Apple Computer, Inc., 1993). This interface includes a feedback window with graphics that are customizable via the "Speech Setup" control panel. Graphical feedback styles available via the control panel include an ear icon, several anthropomorphic characters, and some abstract colored lights. Positive evidence is presented visually using these styles for three of the states we have described here, corresponding to attending, hearing, and acting (States 1, 2, and 6). Negative evidence in the form of an audible error message (e.g., "pardon me?") is presented only when the system hears speech addressed to it that it is unable to parse or interpret. Graphical feedback at State 7 is automatically provided by the direct manipulation-style Macintosh desktop.

While that implementation is only partial, a fuller version of the feedback model, tailored to specific applications, is possible. In future work, we plan to build two more prototype applications that use the PlainTalk™ speech recognizer with our feedback model and test these prototypes with human subjects. For the first application, we plan a set of the commands to an "intelligent" hotel room that a user could communicate with via speech over a telephone handset, in order to retrieve information and request services (such as wake-up calls). The second application will be a scheduling application. We plan to carry out more controlled experimentation to compare the adaptive feedback approach to (a) a version of the system that provides feedback consistently at the same state, and (b) a version that provides only negative feedback. We would also like to incorporate information from the speech recognizer about the confidence level of each recognition match in setting the grounding criterion; these prototypes will be better at conveying information about their states to the user. A response to an utterance with a low confidence value will include more explicit evidence about the system's interpretation than will a response to an utterance with a high confidence value. When the system is in doubt, more of the responsibility for identifying problems and initiating repairs will shift to the user. This flexible shifting of responsibility, we believe, is the key to truly collaborative interaction between people and systems.

CONCLUSIONS

Applying the collaborative view of conversation to human-computer interaction provides a good theoretical framework for generating feedback from a spoken language system. The approach we have taken here enables us to take account of two important insights about conversation. First, because neither partner in a dialog can read the other's mind, communication problems tend to arise asymmetrically. That is, one partner realizes that there is a problem before the other one does. Second, to initiate a repair, a partner needs to take different actions depending on where the problem lies.

Conceptualizing understanding by both the system and the user as a succession of states provides a systematic way to design and program feedback that adapts to the user, the task, and the discourse history. While the behavior embodied in these prototypes could probably be achieved by hand-tailoring each message to a particular situation, a systematic approach is more promising and generalizable. Such an approach also ensures that the user will receive the necessary positive evidence about the system's state as well as the usual error messages. We should also emphasize the importance of trying out the prototypes of these systems on users. Informal user studies have been, and we expect will continue to be, extremely valuable in tuning this feedback model.

In conclusion, spoken language systems can be greatly improved by focusing on issues of dialog and feedback, rather than by focusing exclusively on the accuracy of the underlying speech recognition technology. In fact, improvements to the technology should be driven by the need to provide an adequate technical basis for feedback and for enabling partial interpretations and the incremental construction of meanings. Finally, to be considered a useful interface technology, a speech interface should provide adaptive feedback and makes use of the collaborative strategies from human conversation that users bring with them to human-computer interaction.

ACKNOWLEDGMENTS

We thank our collaborators at Apple Computer, Inc.: Lewis Knapp, Lisa Stifelman, and Chris DiGiano for their creative input and participation in this work; Kai-Fu Lee for insightful comments; Bob Strong, Rick Parfitt, and Matt Pallakoff for technical expertise; and other members of Apple Computer's Human Interface Group and Speech and Language Technologies Group for their technical and intellectual support. This work was partially funded by NSF grant # 92-02458 to Susan Brennan. An abstract of this paper appeared in the AAAI-93 Fall Symposium on Human-Computer Collaboration: Reconciling theory, synthesizing practice. We thank all the participants of that workshop, especially Loren Terveen.

REFERENCES

Apple Computer, Inc. (1993). Using speech with your Macintosh. In: Getting started with your Macintosh Quadra 840AV (manual). Cupertino, CA: Apple Computer, Inc., pp. 75-88.

Brennan, S. E. (1988). The multimedia articulation of answers in a natural language database query system. In Proc., Second Conference on Applied Natural Language Processing, pages 1-8. Association of Computational Linguistics, Austin, TX.

Brennan, S. E. (1990). Seeking and providing evidence for mutual understanding. Unpublished doctoral dissertation, Stanford University, Stanford, CA.

Brennan, S. E. (1991). Conversation with and through computers. User Modeling and User-Adapted Interaction, 1:67-86.

Brennan, S. E. and Cahn, J. (1993). Modeling the progression of mutual understanding in dialog. Manuscript.

Brennan, S. E. and Ohaeri, J. O. (1993). Effects of message style on user's attributions toward agents. Proceedings Companion, CHI '94. Boston, MA: Association for Computing Machinery.

Clark, H H. and Brennan, S. E. (1991). Grounding in communication. In J. Levine L.B. Resnick and S.D. Teasley (Eds.), Perspectives on Socially Shared Cognition. APA, Reading, MA.

Clark, H. H., and Schaefer, E. F. (1987). Collaborating on contributions to conversations. Language and Cognitive Processes, 2:1-23.

Clark, H. H., and Schaefer, E. F. (1989). Contributing to discourse. Cognitive Science, 13:259-294.

Clark, H H., and Wilkes-Gibbs, D. (1986). Referring as a collaborative process. Cognition, 22:1-39, 1986.

Furnas, G.W., Landauer, T.K., Gomex, L.M., & Dumais, S.T. (1987). The vocabulary problem in human system communication. Communications of the ACM, 30, pp. 964-971.

Gomoll, K. (1990). Some techniques for observing users. In B. K. Laurel (Ed.), The art of human-computer interface design. Reading, MA: Addison-Wesley.

Grice, H.P. (1967). Logic and conversation (William James Lecture, Harvard University). Reprinted in P. Cole & J. Morgan (eds.), 1975, Syntax and semantics 3: Speech Acts. Academic Press, New York, pp. 41-58.

Hulteen, E. A. (1991). User-centered design and voice-interactive applications. Proceedings, AVIOS '91: Voice I/O Systems Applications Conference, pp. 3-8.

Norman, D. A. and Draper, S. W. (1986). User centered system design. Erlbaum, Hillsdale, NJ.

Lee, K. (1993). Towards conversational computers: An Apple perspective. Proceedings, EuroSpeech. Berlin, Germany.

Payne, S.J. (1990). Looking HCI in the I. Human-Computer Interaction INTERACT '90. North Holland: Elsevier Science Publishers B.V.

Schmandt, C. M. and Hulteen, E. A. (1982). The intelligent voice-interactive interface. Proceedings, Human Factors in Computing Systems, 363-366.

Steedman, M. J. and Johnson-Laird, P. N. (1980). The production of sentences, utterances and speech acts: Have computers anything to say? in B. Butterworth (Ed.), Language production: Speech and talk. London: Academic Press.

Strong, R. (1993). Casper: Speech interface for the Macintosh. Proceedings, EuroSpeech. Berlin, Germany.

Yngve, V. H. (1970). On getting a word in edgewise. Papers from the sixth regional meeting of Chicago Linguistic Society, 567-578.