Proc. Natl. Acad. Sci. USA
Vol. 92, pp. 10031–10037, October 1995
Colloquium Paper

This paper was presented at a colloquium entitled "Human–Machine Communication by Voice," organized by Lawrence R. Rabiner, held by the National Academy of Sciences at The Arnold and Mabel Beckman Center in Irvine, CA, February 8–9, 1993.

User interfaces for voice applications

CANDACE KAMM

Bellcore, 445 South Street, Morristown, NJ 07962-1910

ABSTRACT **This paper discusses some of the aspects of task requirements, user expectations, and technological capabilities that influence the design of a voice interface and then identifies several components of user interfaces that are particularly critical in successful voice applications. Examples from several applications are provided to demonstrate how these components are used to produce effective voice interfaces.**

As speech synthesis and speech recognition technologies improve, applications requiring more complex and more natural human–machine interactions are becoming feasible. A well-designed user interface is critical to the success of these applications. A carefully crafted user interface can overcome many of the limitations of current technology to produce a successful outcome from the user's point of view, even when the technology works imperfectly. With further technological improvements, the primary role of the user interface will gradually shift from a focus on adapting the user's input to fit the limitations of the technology to facilitating interactive dialogue between human and machine by recognizing and providing appropriate conversational cues. Among the factors that must be considered in designing voice interfaces are (*i*) the task requirements of the application, (*ii*) the capabilities and limitations of the technology, and (*iii*) the characteristics of the user population. This paper discusses how these factors influence user interface design and then describes components of user interfaces that can be used to facilitate efficient and effective human–machine voice-based interactions.

Voice interfaces provide an additional input and output modality for human-computer interactions, either as a component of a multi-modal, multimedia system or when other input and output modalities are occupied, unavailable, or not usable by the human (e.g., for users with visual or motor disabilities). One motivation for human-computer interaction by voice is that voice interfaces are considered "more natural" than other types of interfaces (e.g., keyboard, mouse, touch screen). That is, speech interfaces can provide a "look and feel" that is more like communication between humans. The underlying assumption is that by presenting this "more natural" interface to the user the system can take advantage of skills and expectations that the user has developed through everyday communicative experiences to create a more efficient and effective transfer of information between human and machine (1).

A successful human–machine interaction, like a successful human–human interaction, is one that accomplishes the task at hand efficiently and easily from the human's perspective. However, current human–computer voice-based interactions do not yet match the richness, complexity, accuracy, or reliability achieved in most human–human interactions either for speech input [i.e., automatic speech recognition (ASR) or speech understanding] or for speech output (digitized or synthetic speech). This deficit is due only in part to imperfect speech technology. Equally important is the fact that, while current automated systems may contain sufficient domain knowledge about an application, they do not sufficiently incorporate other kinds of knowledge that facilitate collaborative interactions. Typically, an automated system is limited both in linguistic and conceptual knowledge. Furthermore, automated systems using voice interfaces also have an impoverished appreciation of conversational dynamics, including the use of prosodic cues to appropriately maintain turn taking and the use of confirmation protocols to establish coherence between the participants.

A well-designed voice interface can alleviate the effects of these deficiencies by structuring the interaction to maximize the probability of successfully accomplishing the task. Where technological limitations prohibit the use of natural conversational speech, the primary role of the interface is to induce the user to modify his/her behavior to fit the requirements of the technology. As voice technologies become capable of dealing with more natural input, the user interface will still be critical for facilitating the smooth flow of information between the user and the system by providing appropriate conversational cues and feedback. Well-designed user interfaces are essential to successful applications; a poor user interface can render a system unusable.

Designing an effective user interface for a voice application involves consideration of (*i*) the information requirements of the task, (*ii*) the limitations and capabilities of the voice technology, and (*iii*) the expectations, expertise, and preferences of the user. By understanding these factors, the user interface designer can anticipate some of the difficulties and incompatibilities that will affect the success of the application and design the interaction to minimize their impact. For optimal results, user interface design must be an integral and early component in the overall design of a system. User interface design and implementation are most successful as an iterative process, with interfaces tested empirically on groups of representative users, then revised as deficiencies are detected and corrected, and then retested, until system performance is stable and satisfactory.

USER INTERFACE CONSIDERATIONS

By considering the interdependencies among the task demands of the application, the needs and expectations of the user population, and the capabilities of the technology, an interface designer can more accurately define a control flow for the application that will optimally guide the user through the interaction and handle the errors in communication in a way that facilitates task completion.

Task Requirements

Current speech applications incorporate tasks ranging from very simple ones that require the system to recognize a user's

The publication costs of this article were defrayed in part by page charge payment. This article must therefore be hereby marked "*advertisement*" in accordance with 18 U.S.C. §1734 solely to indicate this fact.

single-word, binary-choice response to a single question (e.g., "Say *yes* if you will accept this collect call; say *no* if you won't") to more complex speech-understanding systems in which the user's input is a complex query that may address a task within a limited domain (e.g., "What times are the flights between Dallas and Boston on Monday?"). In addition, the interfaces to these applications can be highly structured, with either the computer or the user primarily responsible for controlling and directing the interaction, or more conversational, with the computer and the user frequently switching roles from the directive to the more passive participant in the dialogue. These applications differ in the amount and type of information that needs to be exchanged during the interaction and also in the size and composition of the application vocabulary. In addition, applications may vary in the input and output modalities available to the user as well as in the costs of interaction failures.

Information Elements. Clearly, the information requirements of the application are a central component in developing any user interface. For those applications that would otherwise occur as human–human dialogues, understanding how the dialogue typically progresses between humans can help identify the information that must be elicited from the user in order to accomplish the task. Studying the human–human interaction can also point out additional information that is likely to be provided by the user but that is either not critical to task completion or can be obtained from another source. The user interface designer can use this knowledge to determine a more natural order for eliciting the information elements and how much effort should be expended in eliciting each element (2). The observation that users interact quite differently with machines than they do when interacting with a human agent argues against trying to model human–human dialogue precisely. However, analysis of the human–human interaction may help define the application's vocabulary, suggest appropriate wording for prompts, and identify the probable syntax of frequent responses.

Other applications do not have a human–human analog but rather were originally designed as automated systems using strictly nonvoice modalities. Although analysis of such systems can provide useful insights for designing voice interfaces, a strict conversion of the nonvoice interaction to the voice-based counterpart may not be optimal. For example, an automated bank-by-phone application using input from the telephone keypad may use an audio menu to direct the user to "For checking (account balance), press 1; for savings, press 2." Early implementations of voice interfaces for such systems directly translated this request into "For checking, say 1; for savings, say 2," rather than eliciting the more natural command words (e.g., "Do you want your account balance for checking or savings?"). Although the vocabulary choices in this application were probably the result of limitations in the availability of templates for speaker-independent ASR rather than poor interface design decisions, using vocabularies that are meaningful and memorable to the user should facilitate the efficiency of an interaction. Speech recognition systems that easily create application-specific speaker-independent vocabularies are becoming more common, so the technological necessity for direct modeling of the vocabulary for voice-based interfaces on their key-based counterparts should soon disappear.

Task Modalities. The set of input and output modalities available to the user affects the design of the user interface. In addition to voice, input modalities can include keyboard, mouse, touch-screen, or a telephone keypad. The availability of multiple input modalities allows the user to select the most efficient modality for entering input and also allows the system recourse to request an alternative input modality if it determines that the modality being used is not resulting in progress toward task completion. For example, if, in a telephone application, the presence of extremely high levels of background noise consistently results in incorrect recognition, the system may ask the user to switch to the telephone keypad so that the task can be completed successfully.

For output of information from the system to the user, either audio (speech) or textual modalities, or both, may be available and the appropriate modality depends on the task. For example, most voice dictation systems use textual output on a computer's monitor to provide concurrent feedback as the system recognizes words so that the user is made aware of errors or uncertainties in the system and can take corrective action. (Often, the corrective action may be taken using either voice or keyboard input, at the user's discretion.) Using voice output for feedback in this application would be highly ineffective, as it could severely tax the user's auditory memory, disrupt the user's train of thought, and possibly also interfere with reception of the user's speech by the system.

In cases where voice is the only modality available for output (as in most current telephone applications), the interface designer must take into account the temporal nature of speech output and its concomitant demands on user memory. For example, if voice menus are used, they should be constructed with only a very few options at each node, and users must be given a way to request repetition of information whenever they need it.

Cost of Interaction Failures. Errors inevitably occur in human–computer interactions, just as they do in interactions between humans. These communication failures may be the result of the human providing unexpected or inappropriate input, the system misrecognizing or misinterpreting the user's speech, or a combination of the two. The costs of recognition or understanding errors can be direct (e.g., incorrect deposits in automated banking transactions or increased holding time in automated telephone services) or indirect (e.g., user dissatisfaction that leads to reduced use of a service or product). Depending on the cost of errors, the interface can be designed to be more or less conservative in its responses to input it finds ambiguous or incomplete. For example, in an information retrieval system, the cost of an error may be only the time spent making the query and hearing the incorrect answer, as long as the incorrect answer provides sufficient information for the user to determine that the original query was misunderstood. If a confirmatory subdialogue has a time cost similar to that of an incorrect response, it may be more efficient to permit the system to act on the user's voice input without first explicitly confirming that the request was accurately understood. In this case the system expects the user to be able to distinguish between inappropriate responses and correct responses and to inform the system of its error or reconstruct the query. On the other hand, if the application involves an action that, if performed incorrectly, has a high cost or greatly impedes progress toward task completion, it may be desirable to request explicit confirmation of the user's input prior to performing that action, or at least to provide feedback about the impending action and to permit the user to issue a command to cancel it. An example of this type might be a voice-activated dialing application, where a recognition error could result in charges for a phone call the user did not intend to make. For applications where the cost of errors is very high, systems should be designed to redirect the user to a nonspeech input modality or to a human agent in the event that it becomes apparent that repeated interchanges between the user and the system have not resulted in progress toward task completion.

Technological Capabilities and Limitations

Voice Input. Current speech applications demonstrate a wide range of voice input capabilities. The requirements of the application typically dictate whether a system uses (*i*) speaker-trained, speaker-adaptive, or speaker-independent recognition technology and (*ii*) isolated-word, word-spotting, or con-

tinuous speech recognition. Prototype spoken-language-understanding systems with vocabularies of 300 to 1000 words can accept spontaneous continuous speech input without user enrollment (3, 4). These systems allow limited interactions in restricted domains, such as travel planning and providing directions, and currently have an understanding rate of about 70% (5). Several commercially available voice dictation systems incorporate speaker-dependent or speaker-adaptive isolated-word recognition technology for 5000- to 100,000-word vocabularies (6). In contrast, most telephone applications have been restricted to isolated-word or word-spotting technology for small vocabularies in order to maintain acceptable recognition accuracy in the more adverse and variable electroacoustic environment of the public switched telephone network. If the user's input is restricted to the recognizer's vocabulary and speech style limitations, recognition performance in the latter applications can be quite high.

The capabilities of the voice input technology chosen for an application influence the priorities in the design of the user interface. As will be discussed in a later section, user expectations and needs may at times clash with the capabilities of the technology. To create a viable application, the short-term solution is to force the user to comply with the technology's requirements. For example, when the technology can accept only restricted, highly structured input, the user interface must focus primarily on guiding the user to speak in a style appropriate to the technology's limitations. The longer-term solution is to improve the technology to require less adaptation of the user's preferred behavior in accomplishing the task. As the technology allows more natural conversational input, the primary role of the user interface could shift from directing the user about how to speak toward providing more graceful resolution of ambiguities and errors.

Voice Output. Two kinds of voice output technologies are available for voice interfaces: prerecorded (typically digitized) speech and synthetic speech. Prerecorded speech involves recording and storing natural speech for later combination and playback. Prerecorded speech has the quality of natural speech and, provided that care is taken to maintain natural prosody in combining recorded elements, it generally has more natural prosody than synthetic speech. If the voice output requirements of the application are known fully and are stable over time, prerecorded speech can be an appropriate technology for voice output.

However, if the voice output requirements are extensive and changeable, synthetic speech may be required. Speech synthesis technology converts textual information to speech using sets of rules for converting letters to phonemes and for converting phonemes to acoustic events. As a result, messages can be constructed as they are needed. However, although several synthetic speech systems demonstrate high segmental intelligibility, they do not yet match the intelligibility or naturalness of human speech (7). Furthermore, there is evidence that perception of synthetic speech imposes greater processing demands on the listener than perception of natural speech (8). User interfaces that use synthetic speech for voice output should employ techniques aimed at facilitating listener comprehension. Providing messages that include syntactic and semantic context to aid comprehensibility and, where possible, using messages with relatively simple structure will reduce cognitive and memory demands on the user. In addition, applications with low-context messages (e.g., the delivery of proper names in a telephone reverse directory inquiry) should provide the user with a mechanism for requesting repetitions of the message and for the spelling of words that may be difficult for the user to understand.

System Capabilities. Several system capabilities not directly related to speech technology per se have been shown to affect the success of voice interfaces. Some systems are incapable of "listening" to speech input while they simultaneously produce speech. With these systems it is impossible to interrupt a prompt, and the user must wait until the prompt is completed before responding. The user interface for these systems must be designed to detect and discourage violations of this protocol.

Systems that detect speech during prompts and truncate the prompt in response to user input provide more natural interfaces, but adding this capability may also lead to increased ambiguity regarding which prompt the user's input was intended to address. As a result, the user interface for a system with prompt-talk-through may need a more complex strategy for determining whether responses that interrupt prompts are delayed responses to prior prompts or anticipatory responses to the interrupted prompt.

System response time can influence user satisfaction (9). Interfaces to systems with slow response times for speech- and language-processing components can play interim messages (e.g., "Please wait. Your request is being processed.") to reduce perceived response time and increase user acceptance. With continuing advances in processor speed and in the efficiency of recognition search and language-processing algorithms, near-real-time system response is becoming feasible even for complex speech-understanding tasks, so system response time may soon cease to be a significant interface issue.

In most applications the speech system is only a small component of a larger system. Constraints of this larger system can have a major impact on the user interface for an application. For example, in telephone network applications, system capabilities and resource limitations may dictate whether a customer can gain immediate access to a speech recognition subsystem when the customer picks up the telephone or whether a special code must be dialed to access the recognizer. Other system constraints that can influence the design of the user interface include whether a human agent is available in the event of recognition failure and whether relevant information and databases are available throughout the interaction or only during fixed portions.

User Expectations and Expertise

Conversational Speech Behaviors. Because users have so much experience with human–human speech interactions, it is not surprising that new users may expect a human–computer voice interface to allow the same conversational speech style that is used between humans. If the speech recognition technology used in an application cannot perform accurately with normal conversational speech input, the user is typically instructed to speak in the manner that the recognition system has been designed to accept. However, many speech behaviors are overlearned, and it is difficult for users to overcome these habits. Three such behaviors are (*i*) speaking in a continuous manner (i.e., without pausing between words), (*ii*) anticipating responses and speaking at the same time as the other talker, and (*iii*) interpreting pauses by the other talker as implicit exchange of turn and permission to speak.

Continuous speech. Speaking phrases with pauses between each word is unnatural, and the tendency to speak continuously is difficult to overcome, particularly when the user's attention is on completing a task. In an early telephone application that used isolated-word speech recognition technology, users were required to recite the seven-digit number of the telephone being used to place the call, pausing briefly after each digit. In one study 35 percent of the customers who spoke seven-digit numbers did not speak the digits in an isolated manner, demonstrating the difficulty people have overcoming the natural behavior of speaking without interword pauses, even when they are instructed to do so (10). In another telephone study of digit recognition, only 37% of the customers responded with appropriate interword pauses (11).

In addition to the tendency to speak in a continuous manner, extraneous speech and hesitation sounds (like "uhh" or "um") are common in interactive dialogues. The inability to reliably coax humans to speak in isolation, without any nonvocabulary words or sounds, has been a driving force for the development of word spotting and continuous recognition algorithms. Word spotting allows accurate recognition of a small vocabulary in the presence of nonvocabulary utterances, even when they are modified by coarticulation effects introduced by continuous speech.

Talk-over. In conversational settings, participants often begin to respond as soon as they understand the speaker's request. In an analysis of 50 conversations between telephone service representatives and customers, Karis and Dobroth (12) observed that simultaneous speech occurred on 12.1% of the utterances of service representatives and 14.4% of the customers' utterances. This simultaneity rarely impairs the information flow in human–human dialogues, and, as a result, most users expect that their speech will be heard and understood even if it is spoken while the other participant in the dialogue is still talking. Many answering machine messages end with the admonition "Wait for the beep"—attesting to the difficulty people have conforming to a limited response interval.

Conversational dynamics. In a human–human dialogue, subtle cues carried primarily in the speech signal are used by the participants to indicate that they want to take or release control of the speaking role or to confirm that they understand or have been understood by the other party. Violating these cues can cause difficulties in user interfaces. Turn-taking permission is often conveyed by prosodic pauses. In an early implementation of an automated billing service for collect calls, customers were responding prematurely (i.e., before the system was ready to accept speech input) but consistently at a particular phrase break (i.e., brief pause) during the system's prompt (13). The pause at the phrase break was interpreted by these customers as an invitation to take the floor and respond, and so they did. The pause duration in the prompt was shortened, and the resultant user interface was more successful, as users became much more likely to wait until the appropriate response window to respond. Alternatively, a user interface may take advantage of the knowledge that turn taking is likely to occur during pauses, as does the interface described by Balentine and Scott (14). In this system the recognizer is activated during pauses between items, in case the user responds immediately after hearing the menu item he/she wants. If the recognized word matches the menu item presented immediately before the user's input, the system proceeds to that selection. If the recognized word is ambiguous, the system asks for confirmation of whether the menu item presented before the interruption is the desired selection.

Cues for confirmation or signaling of potential errors are often provided by repeating the information to be confirmed, along with specific intonation patterns. For example, falling intonation can imply acknowledgment, while rising intonation of the confirmatory phrase signals uncertainty and potential error (12). Currently, such intonational cues are not recognized reliably by automated systems, but advances in understanding the contribution of intonation patterns to dialogue flow will provide a powerful enhancement to user interfaces, not only for confirmation and error recovery but also for detecting user hesitations and repairs (i.e., self-corrections a user makes within an utterance). A better understanding of how intonation is used to convey intent would also be quite useful in constructing more prosodically appropriate messages for voice output (12).

Novice vs. expert users. Users have different expectations and needs depending on their experience with a system. First-time or infrequent users are likely to require instructions and/or guidance through a system to allow them to build a cognitive model of how the system works and how to interact with it. In contrast, experienced users who interact frequently with the system want ways to bypass the instructions and move through the interaction more efficiently. As humans become more experienced with the specific requirements of a human–human collaborative dialogue, they begin to increase the efficiency of the interaction by abbreviating portions of the dialogue to the minimum sufficient information necessary to accomplish the task (1). For example, confirmation utterances become shorter or occur less frequently, and unnecessary function words may be omitted (e.g., "There are 48 pages in volume I, 42 in volume II, 36 in III"). Successful user interfaces for automated systems should accommodate both the needs of novices and the preferences of expert users.

USER INTERFACE DESIGN STRATEGIES

Once the interface requirements for the application have been identified through the analysis of task demands, user characteristics, and technological factors, there are several aspects of user interfaces that have often proved effective both for alleviating system limitations and for improving an application's usability. These include dialogue flow strategies using appropriately directive and informative prompts and the use of subdialogue modules to provide access to instructions, to confirm that the user's input has been correctly recognized, and to detect errors and recover from them.

To demonstrate how far astray an interaction can go without these features, Table 1 shows a transcript of a human–computer dialogue from a very early application of ASR. The application used isolated-word speaker-independent recognition technology. Employees of the company that implemented this application called the system and spoke an authorization code. The system was supposed to recognize the spoken codes, check the access privileges associated with that code, and either permit or prohibit the placement of long-distance telephone calls, based on the user's access privileges. In reviewing the transcript in Table 1 (and those of other similarly unsuccessful interactions), it is apparent that the users did not know how to use the system. (In one interaction a user even asks "What am I supposed to do?") If users received written instructions to speak the digits of the authorization code with pauses between each digit, they either had not read them or did not remember them. On dialing into the system, the user heard the synthesized message "Authorization number please." The user began to comply by speaking a continuous string of numbers in response to this prompt. The system interrupted her and asked her to repeat, which she did, again speaking in a continuous manner but using a slightly louder voice. However, the problem with her initial utterance was not her speaking level but rather the fact that the system could not handle continuously spoken digits. (In addition, whether the words "sixty-one" and "hundred" were in the system's vocabulary is unclear.) In response to the user's repetition, the system spoke the number "four," presumably as a confirmation request for an erroneously recognized digit. When it became apparent to the system that the continuous input was not likely to be a digit, the system asked for another repetition. The user then responded with a reasonable question but, again, not an utterance the system was able to handle. At this point the user

Table 1. Transcript of an unsuccessful voice interaction

System	Authorization number, please
User	231 sixty-wa-
System	Please repeat
User	(Speaking more loudly) 231 sixty-one hundred
System	Four (pause) please repeat
User	How many times do you want me to repeat it? [hangs up]

Table 2. Suggested interface revisions

System	Authorization number, please
User	Two three one sixty-wa-
System	[Detects unusually long utterance] Please speak the digits of the authorization number, pausing after each digit
User	Two [pause] three [pause] one [pause] six [pause] one [pause] oh [pause] oh [pause]
System	I heard two three nine six one oh oh. Is that correct?
User	No, two three *one*
System	[Detects unusually long utterance, assumes negative response] Please answer yes or no. Is the first digit two?
User	Yes
System	Is the second digit three?
User	Yes
System	Is the third digit nine?
User	No
System	Please say the third digit again
User	One
System	Is the third digit one?
User	Yes
System	[Even if the recognizer still misrecognizes the input as "nine," the system does not accept nine as a candidate and selects the next best choice.] Is the authorization number two three one six one oh oh?
User	Yes
System	Checking authorization number two three one six one oh oh
System	Authorization number accepted. You may place your call now

hung up. Other transcripts from this application demonstrated that the system had two modes of completing an interaction: one that gave the response "Thank you" and the other issuing the word "Sorry." The latter occurred only after a nonproductive series of iterations through a loop consisting of an inappropriate input from the user, followed by another "Please repeat" prompt from the system.

What are the major problems with this user interface? First, the users seem not to be aware of what they are expected to do. This could be alleviated by using a directive prompt explicitly requesting that the digits be spoken with pauses between them. Second, when the system discovers a problem in the input (because it fails to recognize any digit with high certainty), the error message it reports ("Please repeat") does not provide the user with any indication of what the problem might be. Even though the system may not be certain about the source of the problem, it might be possible to make a reasonable guess based on the length of the input utterance. If the length of the utterance that is detected is significantly longer than would be probable if the input utterance was a single digit, the error message might instruct the user to speak the digits one at a time, or even explicitly say "Please say the next digit of the authorization number now." Third, the confirmation messages themselves were confusing, particularly when the recognized digit was not a number in the user's input utterance. It is not clear that the user realized that the word "four" was an attempt at confirmation, and it also was not apparent whether the user had any capability of alerting the system to an incorrect recognition. The confirmation prompt could have been directive and informative—for example, "Was the first digit four?", aimed at eliciting a yes or no answer. However, requesting confirmation after every digit results in a very tedious interaction. If the recognition performance is good enough, it would be better to reduce the frequency of confirmation requests by chunking confirmation messages to contain sets of three or four digits at a time. Finally, the user's only recourse to difficulties with the system seems to be to hang up. The user is apparently afforded no other control of the interaction. All of these suggestions will make the interaction lengthier, but at least the user would have a better chance of understanding how to use the system and of accomplishing the task. (See Table 2 for a sample revised interaction conforming to the technology limitations of isolated-word recognition.) The transcript in Table 1 shows how a user interface that does not incorporate instructions, confirmation, and error recovery can render an application virtually unusable. The importance of these elements is discussed in more detail in the following sections, using examples that demonstrate their use in more successful voice interfaces.

Dialogue Flow

Underlying every user interface is a plan for obtaining the information elements necessary to complete the task. In more structured interactions the order of acquiring these elements may be explicitly defined, whereas in more conversational interfaces the order in which the information is gathered may be controlled more by the user. Directive prompts that explicitly instruct a user are useful for guiding an interaction. They are especially critical when the technology requires input that is restricted (e.g., to isolated-word input, to a small vocabulary, or to a circumscribed time interval), but they are also useful for resolving ambiguities in systems where more natural input is permitted.

Directive prompts can significantly increase user compliance to system restrictions. As shown in Table 3, a Bellcore study of customer responses to alternate billing prompts demonstrated that the percentage of acceptable inputs to an isolated-word recognition system with a two-word vocabulary (the words "yes" and "no") increased from about 55 percent to 81 percent when the prompt was changed from "Will you accept the charges (for a collect call)?" to "Say *yes* if you will accept the call; otherwise, say *no.*" Other studies of similar applications have demonstrated similar prompt-related effects (13, 15).

Directive prompts can also be used after ambiguous or error conditions to attempt to tell the user how to modify the next input to improve the chances of a successful input. For example, if the system (*i*) does not allow prompt talk-through, (*ii*) detects high energy at the beginning of the input to the recognizer, and (*iii*) has low certainty about the accuracy of the recognized word or phrase, the system might assume that the user began speaking before the prompt was over and would reprompt the user for the input, augmenting the reprompt with "Please speak *after* the beep" (with stress on the word "*after*").*

Feedback and Confirmation

One way to limit erroneous actions by the system is to give the user feedback about the application's state and to request

*In fact, this strategy has been implemented in a voice-activated dialing application developed by NYNEX.

Table 3. Effect of prompt wording on customer responses

Customer response	Proportion of responses (T)	
	Prompt: Will you accept the call?	Prompt: Say *yes* if you accept the call; otherwise, say *no*
Isolated yes or no	54.5	80.8
Multiword yes or no	24.2	5.7
Other affirmative or negative	10.7	3.4
Inappropriate response (e.g., "She's not here," "It's for you")	10.4	10.2

confirmation that the system's interpretation of the input or impending action on that input is what the user intended. However, providing feedback and eliciting confirmation on each piece of information exchanged between the user and the system often results in a lengthy, tedious, and inefficient interaction. As mentioned above, if the cost of errors is minimal, and if the user is provided sufficient information to ascertain that the system's response was appropriate, it may be reasonable to forgo many of the confirmation interchanges for some applications. For example, in a research prototype stock quotation service developed at Bell Northern Research (16), the system immediately searches the stock database and provides the user with a stock quotation for any input utterance (except utterances that are recognized as command phrases in the application). If the system retrieves a different stock than the user requested, the user can say "Incorrect stock" to retrieve stock market information for the stock that was the next best match to the input utterance, or, if the user is not aware of that option, the user can repeat the stock name. This strategy provides a way for the user to take control to indicate errors and backtrack to a stable point in the dialogue.

Many applications issue requests for confirmation only when there is some level of ambiguity about the input (e.g., when the most likely recognition candidate has a relatively low likelihood or certainty or when an input utterance to a spoken language system cannot be parsed). In these cases, whenever possible, the confirmation request should not be a simple request for repetition of the input utterance, since that utterance has a history of not being handled successfully. Rather, the confirmation request should be a directive prompt that attempts to resolve the ambiguity and progress toward task completion. One strategy is to request confirmation by phrasing the request as a yes/no or forced-choice question about the accuracy of the best guess of the recognized utterance. Katai *et al.* (17) observed that, even when the yes/no confirmation strategy is used, users often respond with a repeat of the command, suggesting that it might be useful to extend the response set for the confirmation question to include the particular word or phrase that is being confirmed.

Providing feedback for confirmation is also critical in interfaces that do not provide voice output. For example, some voice dictation systems provide the user with a short list of the most likely word candidates, so that the user can indicate the correct alternative by identifying its position in the list or by spelling it. In any application the user interface must allow the user to initiate corrective actions.

Instructions

To use a system efficiently and effectively, the user must know what is expected by the system and what information the system can provide. For simple tasks the instructions to the user may consist of a single prompt. For more complicated interactions with a variety of options, the instructions may be quite lengthy and detailed. Novice users will need more help, and experienced users may not need any guidance to use the system (unless an error occurs). An optimal user interface should provide the option of accessing instructions or may automatically offer instructions if the user's behavior suggests that they are needed. An example of the latter strategy is implemented in a prototype voice-activated directory service used at Bolt, Baranek and Newman. In this system no initial prompt is given when the user is connected to the system. Expert users know to say the first and last names of the person they want to reach. However, if the system does not detect the onset of speech within 2 seconds, it provides a brief prompt instructing the user (J. Makhoul, personal communication).

The Bell Northern Research stock quotation application mentioned above (16) has extensive instructions describing the capabilities of the system. The brief welcoming prompt to this service tells the user to say the name of the stock or "Instructions" to hear the instructions. Thus, the expert user can bypass the instructions and progress directly to the primary task. New users, however, are provided with the opportunity to access the instructions. At the beginning of the instructions, the user is told how to exit the instructions at any time and return to the initial prompt. This informs the user that he/she can gain control of the interaction at any time and how that can be accomplished.

Error Recovery

Since errors are inevitable, error recovery procedures are also an inevitable requirement in a user interface. The goal of error recovery procedures is to prevent the complete breakdown of the system into an unstable or repetitive state that precludes making progress toward task completion. Obviously, error recovery requires the cooperation of the user, and both the system and the user must be able to initiate error recovery sequences. The feedback and confirmation dialogues described previously are the first step at detecting errors.

In the Bell Northern Research stock quotation application (16), the user initiates error correction by saying "Incorrect stock." The system then provides the stock information for the next best match to the original input and asks the user if that is the correct stock. The system will propose a few alternate candidates and, if the user confirms none of them, will ask the user to repeat the stock name. If no new candidates are highly likely, the system suggests that the user check to be sure the requested stock is really offered on that stock exchange.

In other applications the system flags the error or uncertainty and initiates error recovery. For example, in the voice dictation task, the system signals recognition uncertainty by taking control of the interaction and suggesting a set of possible alternatives for the user to confirm. By requiring the user to indicate the correct alternative by providing its item number on the list or by spelling it, the error recovery module uses a much more restricted vocabulary than the general dictation task to increase the likelihood of correct recognition of the user's corrective response.

When provided with error messages, users do modify their subsequent behavior. In a Wizard of Oz study where a human listener entered accurate transcripts of the user's speech to the natural language component of a spoken-language-under-

standing system for travel planning, Hunnicutt *et al.* (18) observed that users were able to make use of the system's error messages to rephrase their queries and reliably recovered from errors about 80% of the time. Additional studies by the same authors showed that, in a fully automated system that occasionally hypothesized incorrect utterances, users were generally able to identify misleading responses from the system and successfully recovered from them, even when system error messages were not very specific. These results suggest that, if a system can make a reasonable assumption about the probable cause of the error, directive error messages (e.g., "I didn't hear your response. Please speak louder" after a system time-out) are useful. In addition, even if the system's assumption is wrong (e.g., the user didn't speak too softly but rather not at all), the user generally has sufficient information to be able to disregard inappropriate messages arising from the system's incorrect hypotheses.

Finally, some voice interactions will not succeed, no matter how clever the error detection and recovery strategies. In these cases the user interface should inform the user of the problem and its probable cause and direct the user to an alternative solution—for example, a human agent, if there is one available, or an alternate input modality that may be more reliable. As a general principle, user interfaces should also allow the user to initiate a graceful exit from the application at any stage in the interaction.

EVALUATING TECHNOLOGY READINESS

This paper has stressed the interdependencies among application requirements, user needs, and system capabilities in designing a successful voice interface. Often, voice interface developers are asked whether speech technology is "good enough" or "ready" to accomplish a particular task. Although the query is stated simply, a thorough answer must extend beyond the focus on the technology alone to address other questions that include:

1. Would a typical user judge the automated interaction to be satisfactory—that is, is the user successful at completing the task, and is the interaction easy and efficient from the user's viewpoint?

2. Is the degree of successful automation sufficient to make its use financially attractive for the application provider? Do the benefits and savings accrued from the automation effort offset the costs of implementation?

The "readiness" question should encompass not only the technological capabilities but also the context of the specific application, its users, and the entire system within which the application is instantiated. The optimal way to estimate "readiness" is in field trials of the application, using prototype systems and users who represent the population that the application is intended to serve. Performance measures for assessing the success of such field trials would include task completion rates and user reactions. If a field trial demonstrates that the application is successful for the user, a cost/benefit analysis can then be applied to determine whether the application is "ready" for deployment (i.e., is economically justifiable).

CONCLUSION

Whether the voice application is relatively simple (e.g., recognizing only a limited set of key words) or quite complex (e.g., speech-understanding systems that permit more natural conversational interactions), a well-designed user interface will always be an essential element in successful human-machine communication by voice. The designer of a voice interface must consider the interrelationships among the task requirements, technological capabilities, and user expectations in implementing an interface that will facilitate "ease of use" and naturalness of a human–machine interaction. Voice interfaces can be made less brittle by incorporating directive prompts in the main task dialogue as well as in confirmation requests and by providing user control to access instructions, to indicate and correct errors, and to exit the system as necessary. Iterative testing with representative users is necessary in developing a robust voice interface. With improved understanding of how prosody and message content cue conversational dynamics for confirmation, error recovery, hesitations, and repair, and with the development of reliable techniques for automatically recognizing and providing these cues, voice interfaces can begin to realize the promise of a "more natural" interface for human–computer interactions.

I would like to thank Mary Jo Altom, Susan Dumais, Dan Kahn, and Sharad Singhal for helpful comments on early drafts of this paper.

1. Leiser, R. G. (1989) *Appl. Ergonom.* **20,** 168–173.
2. Mazor, B., Zeigler, B. & Braun, J. (1992) in *Proceedings of the American Voice I/O Society* (San Jose, CA).
3. Appelt, D. & Jackson, E. (1992) in *Proceedings of the DARPA Speech and Natural Language Workshop*, ed. Marcus, M. (Kaufmann, San Mateo, CA).
4. Pallett, D. (1992) in *Proceedings of the DARPA Speech and Natural Language Workshop*, ed. Marcus, M. (Kaufmann, San Mateo, CA).
5. Cole, R., *et al.* (1992) *Oregon Graduate Institute Technical Report No. CS/E 92-014* (Oregon Graduate Inst., Beaverton).
6. Makoul, J. & Schwartz, R. (1995) *Proc. Natl. Acad. Sci. USA* **92,** 9956–9963.
7. Spiegel, M. F., Altom, M. J., Macchi, M. J. & Wallace, K. L. (1988) in *Proceedings of the American Voice I/O Society* (San Jose, CA).
8. Luce, P., Fuestel, T. & D. Pisoni, D. (1983) *Hum. Factors* **25,** 17–32.
9. Shriberg, E., Wade, E. & Price, P. (1992) in *Proceedings of the DARPA Speech and Natural Language Workshop*, ed. Marcus, M. (Kaufmann, San Mateo, CA).
10. Wilpon, J. G. (1985) *AT&T Tech. J.* **64,** 423–451.
11. Yashchin, D., Basson, S., Lauritzen, N., Levas, S., Loring, A. & Rubin-Spitz, J. (1989) *Proc. IEEE ICASSP*, 552–555.
12. Karis, D. & Dobroth, K. M. (1991) *IEEE J. Selected Areas Commun.* **9,** 574–585.
13. Bossemeyer, R. W. & Schwab, E. C. (1990) *J. Am. Voice I/O Soc.* **7,** 47–53.
14. Balentine, B. E. & Scott, B. L. (1992) *J. Am. Voice I/O Soc.* **11,** 46–60.
15. Spitz, J. & Artificial Intelligence Speech Technology Group (1991) in *Proceedings of the DARPA Speech and Natural Language Workshop*, ed. Price, P. (Kaufmann, San Mateo, CA).
16. Lennig, M., Sharp, D., Kenny, P., Gupta, V. & Precoda, K. (1992) in *Proceedings of the International Conference on Spoken Language Processing* (Univ. of Alberta, Edmonton, AB Canada), pp. 93–96.
17. Katai, M., Imamura, A. & Suzuki, Y. (1991) *American Voice I/O Society* (Atlanta).
18. Hunnicutt, S., Hirschman, L., Polifroni, J. & Seneff, S. (1992) in *Proceedings of the International Conference on Spoken Language Processing* (Univ. of Alberta, Edmonton, AB Canada), pp. 196–220.