Proceedings, 1982 Conference on Cybernetics and Society, IEEE, Seattle, 1982

VOICE INTERACTION: PUTTING INTELLIGENCE INTO THE INTERFACE

Christopher Schmandt

Architecture Machine Group
Massachusetts Institute of Technology

Although the natural nature of speech suggests its use as an input medium to computer systems, current speech recognition hardware does not perform adequately for general purpose use with large vocabularies. A methodology is described for design of an interface module consisting of both hardware and software mediating between the speaker and application processes. The goal is communication, with its implication of intelligence and interaction, by this context sensitive interface.

INTRODUCTION

A broad spectrum of computer system users are showing growing interest in speech recognition hardware. Speech is a natural input channel, requiring few special skills and freeing hands from keyboard responses. There remains, however, prudent doubt of the performance of speech recognizers; early users have realized that recognition degrades dramatically with larger vocabularies and outside laboratory conditions. Admitting these limitations, this paper presents a model for voice interactive system design, and catagorizes a number of elements toward implementation of such a model.

Real performance evaluations of speech as an input medium are difficult, as they must be more concerned with user satisfaction than with raw statistics of "percentage accuracy". Equipment with even 95% field recognition may be only marginally useful if it consistently fails to respond to a single important word, such as the vocal equivalent of carriage-return; user frustration rises rapidly in such situations. Unfortunately, squeezing out another percent or two may prove difficult, and in fact addresses satisfaction only indirectly.

The hardware oriented "black box" approach, the limited view of speech input as a "speech in/data out" accessory to be plugged into some application, has not been convincing; recognizers rarely fail gracefully under such circumstances, and test statistics are misleading. Marketing battles between manufacturers only obscure real issues. In fact, there is no such thing as a general purpose speech recognizer robust enough for large vocabulary tasks in real world operating environments.

Yet, there is a growing community of satisfied users in the research community. Successful approaches can be broadly classified as "systemic" or "holistic". The solution is not to make speech i/o replace a few buttons or indicator lamps, but rather to fully integrate speech into the whole context of communication, i.e. exchange of information, between the operator and computer. The tools are not so much recognition as understanding, with the implication of an intelligent system interacting with an intelligent user. Essential to this approach is emphasis on an interface module, consisting of both i/o oriented hardware as well as software aware of the task context, which mediates between the human and the application software. Without such an interface component, complex voice interaction will not be possible.

SOURCES OF FRUSTRATION

The origins of this paper are actual implementations using speech recognizers in various application tasks, and the frustrations generated by their real performance. A large portion of this experience is derived from work on a project called "Put That There" at MIT's Architecture Machine Group. But much additional insight has been gained by observing details of others' work, mostly unpublished. The intent of this paper is to categorize these experiences and solutions in a manner that can

0360-8913/82/0000-0199 $00.75 © 1982 IEEE

aid other designers of voice controlled systems.

The communications oriented model of voice interaction presented in the following section is both a "wish list" as well as an abstraction of the essential qualities of these successful applications. A multitude of techniques illustrate the implementation of this model; some are universal ; others are dependent on the capabilities of the particular speech recognition hardware chosen. Although the majority of them were used in Put That There, which will be used where appropriate as an example, the lessons should be general purpose.

Put That There is a voice and gesture interactive system which allows a user to build and modify a graphical database on a large screen video display. The system responds either graphically, by drawing on the screen, or vocally, using either speech synthesis or digital audio speech playback. The particular application is the manipulation of shipping on a map of the Caribbean, allowing the user to create, move, copy, delete, name, or change the shape or color of ship icons, or request different maps, query the system about the current map, or annotate it. Details of its operation have been described elsewhere (1,2).

The dominant lesson from this work was that utilizing speech input for an application should be thought of as trying to find the signal in a very noisy input channel. This necessitated an interface component which had to strive toward understanding the user, before any application routines could be called. In early versions of Put That There speech response was absent, and we resorted to a number of techniques to expand on the actually recognized speech, including where the user was pointing, knowledge about the database, and building in assumptions. It was only when speech output was introduced, however, that any serious ability to respond to the user was an option; at this point the system was redesigned to respond in an intelligent and useful style. Hence the emphasis in this paper on voice INTERACTION.

COMMUNICATION BARRIERS

Speech could become a widely available avenue of access to computers by a non-programming population. This could involve a human telling the machine what to do, or, in other circumstance, the computer informing its user of some piece of information. Responsive systems will be interactive rather than static, reflecting changing user needs. In either case, communication is a minimal prerequisite; the person must know what the computer has decided it is supposed to do, and the computer must know what information is of any interest to the user -- any barrier to utilization of these resources. An optimal voice interactive computer would indeed pass a "Turing test" of listening and responding as convincingly as an intelligent person. With this theme in mind we begin by first identifying some of the constraints hindering this complete accessibility.

Language Constraints. The most obvious constraint on communication with a computer to which one speaks is its small vocabulary. Although current speech recognition hardware is of course limited to a finite vocabulary, there is no reason why the content of that vocabulary should not be user selected, or task dependent, and in fact dynamically reconfigurable in the midst of the speech driven application.

Grammar is another language constraint. The difference between knowing how to write papers in English and being able to program in PL/1 are obvious, and in fact the inability to grasp the unnatural syntax of a programming language is one of those incredible barriers separating the "naive user" from the computers he or she may be using. Voice interactive systems should strive for as natural a grammar as possible. Every syntax rule of a command language is an unnatural limit which only helps underline the fact that the user is talking to a fallible machine. Of course, recognizing totally naturally spoken English is a distant goal, but the more unconstrained the language, the more accessible speech systems will be.

Accuracy Constraints. The accuracy or error rate of speech recognition hardware is the next obvious constraint to its use. Undetected or misrecognized words are a very concrete limit to the effectiveness and acceptance of speech input technologies.

It must be noted here that the crucial factor is the error rate as perceived by the user. This may imply cataclysmic or benefical perceptions of system useability. If a recognizer does superbly on every word except the one needed to enter each command string, any sane user will feel constrained to the point of removing the microphone and demanding an alternative input device (which should probably be there anyway). By the same token, however, context sensitive software may be able to resolve or interpolate around some of the hardware errors, dramatically enhancing system performance from the operator's perception. Evaluation of speech responsive systems in actual use will really be based on their understanding rather than hardware recognition.

Intelligence Constraints. A third major barrier/facilitator to communication is intelligence, a minimal prerequisite for communication. Intelligence is embodied in responsiveness or feedback, through which the interface indicates what it has understood or needs to know, and without which the user can have only marginal confidence that the system is even listening. Additionally, intelligence implies sensitivity to context, an awareness of the task at hand and inputs and responses which are germaine to it. This linkage to context can become a powerful tool to assist recognition, as well as providing the guidelines to direct dialog aimed toward task completion.

Failure Constraints. There will always be recognition errors, and some of those errors will not be recoverable; what happens then? The most simple and most constraining failure mode is that the system and user remain in a loop, the system not responding until the right word is recognized, and the frustrated user repeating this magic word until by some coincidence it is recognized. The user may or may not even realize what the system is waiting for him to say! More encouraging are helpful system responses, which in some manner coax the user to supply missing information on as direct a path as possible to completing plausible application commands. In this manner, failure may not even be irritating, having been as fully absorbed in the general flow of interaction between the human and the computer.

IMPLEMENTATION TECHNIQUE

Any serious attempt to design an interactive system enbodying the system qualities just described necessitates more than merely plugging a speech recognizer directly into an application, much as one might a use a terminal. Inherent is a communication module, involving both hardware access and software, and clearly sensitive to the context of the particular application. This module mediates between the user and the application (to which it may be thought of as a "front end"), manifesting intelligence to the user and passing control to the application only when fully satisfied that the user's intent is well understood. Such an interface component will clearly sustain a dialog with the person, and may well also query the application on its current state or history, to aid its understanding of the speech.

Speech Analysis.
Among the first components of this mediating intelligent interface are syntactic and semantic analysis. Although perhaps an intimidating first step, it is important to realize analysis is to be made in the context of a limited vocabulary and the known context of the application task. Svntactic analysis in Put That There was accomplished by mapping each word into a class, and instance of that class, e.g. the class may be "command" and the instance "move". Only with speech analysis can one hope to allow any reasonable subset of natural language, with flexible word order, and the ability to distinguish the meanings of the command in "make an oil tanker there" versus "make that red".

Equally important to what has been understood is what is missing. The analysis phase determines what parts of the most likely syntactic template are missing, and passes control to other types of analysis, depending on the missing operands. If a speech recognizer can also return a "second guess" or series of guesses in decreasing probablity order, analvsis may be able to select the correct order of the permutations of these choices by analyzing which is most probable, according to syntactic rules (3).

Redundant Input Channels. One additional source of information beyond speech is input which may be arriving on other perhaps redundant, channels. It is for this reason that speech is seen as an addition, rather than a replacement, for other channels. In the case of Put That There, gesture was the second source; pointing at an object will usually designate it at least as well as talking about it.

Knowledge Based Assumptions. More of the missing speech may be deduced from constraints of the particular application at hand. A knowledge based system may well be able to pare down the universe of syntactically possible commands dramatically, and indeed may even be able to make reasonable assumptions of user intent within the context of what it is possible to do and what steps may be prerequisites to others. A trivial example is knowing that the only command one can do to a non-existent object is create it. A more sophisticated one is keeping track of what a user has done to each object, to perhaps be able to guess what she might want to do to it later.

Non-vocal Feedback. An obviously vital component of a responsive, communicative interface is feedback. Non-vocal feedback is graphical feedback in the case of graphics application, for example. In Put That There, an example of such feedback is that a ship changes whenever the system thinks the user is talking about it. The goal of this style of feedback is to communicate back to the user conclusions the system has made (what it thinks is being talked about) as soon as possible, at the locality of the user's attention.

Feedback should be immediate; even if the system will take some time to perform a task, some response should be the first step. The earlier that feedback is provided, the sooner errors can be detected or corrected by the user. But equally important, any feedback gives the user confidence that he is being heard, and motivation to continue the dialog.

Voice Response. A rich domain for interaction, this is a special class of feedback because it occurs in the same medium as the speech input. It seems only natural for a computer to which one speaks to talk back; the issue is what it might say. When the software is ready to contemplate some response, it already gleaned some useful information about the user's speech; the syntactic model indicates what words are missing or ambiguous. Clearly the best feedback is to actually do the task; the worst is to say nothing, or, nearly as bad, to respond with "I didn't understand".

Well phrased questions should be direct and avoid generalities, indicating as much as possible about system understanding. In Put That There, we attempted to ask precise questions, each query directed toward filling in a single missing word. For example, "What object?" conveys a missing operand, while "Which one?" shows that it has found more than one object matching the description.

A noteworthy though unexpected benefit, particularly for connected speech hardware, fell out of this approach of querying for single word replies. Connected speech recognition is much more difficult than discrete, having to cope with coarticulation distortions at both ends of each word. If a word in the middle of a sentence has been missed, then asking for repetition of the sentence has to cope with same coarticulation. A single word response, however, is not subject to any of the distortion of connected speech and is in fact much more likely to be recognized. This is significant for user satisfaction; the worst time to make a recognition error is immediately after making a recognition error!

Another interesting component of this style of dialog is the expectation of what the answer may be. Some speech recognizers allow "syntax control"; each word is assigned to one of a small number of classes, and the hardware may be told to listen for certain classes only. As decreasing the universe of possible replies to a known question can only improve recognition probability, this syntax control could dramatically enhance continuity of intelligent dialog, by limiting acceptable recognitions to plausible responses. In effect, the interface software provides some context to the otherwise context free hardware. Note, however, that in addition to allowing recognition of direct replies to the previous question, the "quit" or "start over" word should also be recognizable; a well phrased machine question may indeed indicate that serious enough misrecognition has occurred to warrant just trying again.

Of course, error tolerance can usually be controlled by setting the recognizer to report recognition of words only when above some threshold. If this threshold is too loose, many false recognitions occur; if too tight, then fewer correct recognitions are made. If the confidence (which would be compared to the threshold) of each recognition is reported to the host, discrimination may be done in software. This allows more interaction around words in the grey region of marginally acceptable confidence. If a recognition is syntactically possible but perhaps doubtful, voice response can query "Did you mean...?" rather than trying to get the user to repeat the offending word. This is especially powerful with above described syntax control, as the question clearly has only two replies.

Dynamic Vocabularies. Clearly for some time speech recognizers will have to work with limited vocabularies. The task is to make such a limitation as little a constraint as possible. As already suggested syntactic analysis is a tool allowing synonyms as well as multiple meanings for the same word as a function of usage. Still, it seems reasonable for a user to proclaim "I'd rather not say it that way". Analysis removes constraints on word order, by migrating words to their appropriate location in the internal representation. Dynamic vocabularies could allow addition of even different words.

Of course, it is always possible to change the vocabularies offline. Really dynamic vocabularies imply the ability to modify them in real time, in the midst of and under the operation of the application. This can be done with recognition hardware which has two features; words need to be trainable by speaking them once, and recognizer functions must be computer controllable. In Put That There, when the user said "Designate copy", the recognizer was put in selective training mode for a particular word slot, and the new utterance, e.g. "duplicate" was trained in and associated with "copy" by software. Henceforth "duplicate" was understood by the system as "copy". Thus, anyone using the system could quickly modify it to be responsive to her own idiosyncratic manners of speaking.

Similarly, if computer access to the recognizer's internal vocabulary template memory is fast enough, dynamic context-sensitive "virtual vocabularies" may be possible. This may be helpful if the application can be broken into clearly defined tasks, with some continuity of usage of groups of tasks. If one were to use a speech responsive system to make phone calls, for example, the recognizer's vocabulary could be quickly loaded with frequently called names. When the same person wished to log into his computer to read the morning's electronic mail, those names could be replaced with operating system commands to allow verbal file system access. In short, exactly as with virtual memory, an apparently very large recognition vocabulary could be presented by quite limited hardware.

<u>Memory</u>. A final technique to improve system responsiveness and intelligence is memory. Software can keep track of what it has done, as well as histories of the objects it is manipulating. This can facilitate the knowledge based considerations discussed above. It can also allow such commands as "undo that", or putting something back where it was, for example. Both these modes embody elegant error recovery, by conveying much information with few words.

UTILIZATION

This paper has presented a vision of speech interaction under goals of minimizing constraints to maximize user access to computers and their efficient operation. The above catalog of techniques are relevant to this vision only to the extent that they enhance its realization. At least some will be useful to any particular application environment; the more that can be properly employed, the more intelligence the system will manifest.

No claim is being made that the above list of interactive techniques is exhaustive; far from it. Our experience has certainly shown optimistic results, that very useable speech systems can be constructed with current technology. This should foreshadow a hopeful future for human/computer communication.

ACKNOWLEDGEMENTS

The foremost acknowledgement I must make is to Eric A. Hulteen, currently with Atari Systems Research. Over the course of several years of working together Eric programmed at least half of Put That There, sharing the development of these models of voice communicative systems, and of course aiding in this writing. Dr. Richard Bolt initiated the first concepts of voice and gesture coupled interaction, which started our work on Put That There, which work was funded by the Defense Advanced Research Projects Agency, Cybernetics Technology Division, under contract number MDA-903-77-C-0037.

REFERENCES

1. Schmandt, C. and Hulteen, E.A., The Intelligent Voice Interactive Interface. <u>Proceedings, Human Factors in Computer Systems</u>, Gaithersburg, MD, 1982. National Bureau of Standards / ACM. pp. 363-366.

2. Bolt, R.A., Voice and Gesture at the Graphics Interface. <u>Computer Graphics, Proceedings of ACM SIGGRAPH '80, Vol. 14 No. 3, 1980. pp. 262-270.</u>

3. Levinson, Stephen E. and Liberman, Mark Y. Speech Recognition by Computer. <u>Scientific American</u>, April 1981. <u>pp. 64-76.</u>