Conversational Interaction with Speech Systems

Alexander I. Rudnicky   Alexander G. Hauptmann

4 December 1989
CMU-CS-89-203 2

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

This paper discusses design principles for spoken language applications. We focus on those aspects of interface design that are separate from basic speech recognition and that are concerned with the global process of performing a task using a speech interface. Six basic speech user interface design principles are discussed: 1. User plasticity. This property describes how much users can adapt to speech interfaces. 2. Interaction protocol styles. We explain how different interaction protocols for speech interfaces impact on basic task throughput. 3. Error correction. Alternate ways to correct recognition errors are examined. 4. Response Time. The response time requirements of a speech user interface is presented based on experimental results. 5. Task structure. The use of task structure to reduce the complexity of the speech recognition problem is discussed and the resulting benefits are demonstrated. 6. Multi-Modality. The opportunity for integration of several modalities into the interface is evaluated. Since these design principles are different from others for standard applications with typing or pointing, we present experimental support for the importance of these principles as well as perspectives towards solutions and further research.

The research described in this paper was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 5167, monitored by SPAWAR under contract N00039-85-C-0163. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or of the US Government.

Table of Contents

1. Introduction

1.1. Background

Recent advances in speech recognition technology (e.g., [Lee 89]) have made it possible to build "spoken language" systems that combine several desirable properties. Recognition of *continuous speech* allows users to use a natural speech style. *Speaker independence* allows casual users to easily use the system and eliminates training as well as its associated problems (such as drift). *Large vocabularies* make it possible to create habitable languages for complex applications. Finally, a *natural language* processing capability allows the user to express him or herself using familiar expressions.

Despite advances in basic technology, we know little about building a successful speech interface, beyond such obvious characteristics as the need for high recognition accuracy. Recent work on spoken language systems, however, has allowed us to identify properties that are critical to the usability of a speech interface.

Our understanding of complex problems is often enabled or at least facilitated through the use of the proper metaphor or model. In the case of spoken language interfaces, we believe that the proper metaphor is that of the conversation. Conversation, from our point of view, is characterized by two important properties: it constitutes an extended interaction over time which involves continuity of topic. In addition, it is a joint activity, with each participant contributing his or her share to the goal of making the interaction progress.

We have identified six properties of interaction that support the conversational model of speech interaction. Some are not unique to speech and can be said to characterize other interaction modalities as well; nevertheless, each property is essential to the usability of the speech interface. The properties are as follows:

1. **User plasticity.** As they do in conversation, computer users adapt their speaking style to the perceived characteristics of the system they are dealing with. This is a natural response and, at least in the case of human-human interaction, happens quite effortlessly.
2. **Appropriate interaction protocols.** The response characteristics of a particular system govern interaction protocol procedures. An interaction protocol regulates the communication between two participants by providing an agreed upon procedure for taking turns and making corrections or confirmations. For example, an inaccurate system (or a noisy communication channel) necessitates explicit verification of input correctness. As the accuracy improves or as the context of interaction changes, the optimal interaction protocol changes accordingly.
3. **Error correction.** As speech recognition is an errorful modality, facilities need to be provided that allow the user and the system to quickly and efficiently recover from input errors. Suitable error repair strategies are essential for the usability of an errorful input modality.
4. **Response Time.** Recent empirical data suggest that even subtle delays in system response characteristics affect how users perform tasks. The presence of delays in recognition may create usability problems for speech systems.
5. **Task structure.** Many computer-based tasks have a distinct structure which can be effectively exploited by a recognition system. Exploiting local or global consistency in interaction structure allows the system to make predictions and educated guesses about what is spoken.

6. **Multi-Modality.** Human interaction takes advantage of many different modalities, making use of visual, prosodic, and gestural information. The integration of different modalities help the user achieve a more natural interaction style.

In the course of our ongoing work on spoken language systems at Carnegie Mellon, we have had occasion to explore a number of these properties. Subsequent sections in this paper will discuss our findings.

1.2. Speech recognition systems

To support natural interaction, a speech recognition system needs to embody certain basic properties, without which it cannot be called a spoken language system. Speaker independence, continuous speech, and large vocabularies constitute desirable properties along the classic dimensions used to describe speech systems [Reddy 76]. In addition, some degree of natural language processing is essential for natural interaction. In this section we will describe why each of these is a crucial property for a spoken language system (SLS).

Speaker independence means that the system can accept and recognize with high accuracy the speech of many talkers, including voices that were not part of its training set. Speaker independence is important for a SLS because it makes the system accessible to casual users. Speaker dependent systems require that all speakers spend a period of time enrolling their voice, for example, by recording a corpus of training sentences. Such a system will, at any one time, be responsive to only one speaker, precluding its use in various public applications or even within a group of users. Furthermore, such a system is susceptible to changes in voice characteristics. These include transitory changes (e.g., a speaker suffering from a cold) as well as phenomena such as *speaker drift*, the change over time in the quality of an individual's speech. The characteristics of a voice change slowly over time, to the extent that the system may no longer accurately recognize the speaker's voice a few weeks after the initial enrollment. When this happens the user needs to retrain the system, which adds to the cost of using the system. Speaker dependence creates a barrier to the use of spoken language systems, one that does not exist for other input modalities, such as typing. There, once the appropriate skill is learned, the user is able to use any number of systems, in many different situations.

Continuous speech recognition is the property that allows a system to deal with connected words, as normally spoken by humans. This is in contrast to discrete word recognition, where the speaker needs to separate the words in an utterance by short pauses. This process greatly simplifies the recognition problem, since the system can now unambiguously segment word-sized units from the utterance. However, it also prevents users from speaking in a natural manner and forces them to attend to their speech, a diversion of attention that might deteriorate task performance for complex applications or stressful situations. While claims have been made to the effect that producing discrete speech is not an onerous burden [Gould *et al.* 83], it is clear that in most applications (for example, dialing telephone numbers) discrete speech is at a strong disadvantage compared to a keyboard.

A speech recognition system must also have the ability to deal with a ***large vocabulary***, that is, to reliably distinguish a large number of different words. Note that a large vocabulary capacity is not *per se* essential for a usable system. It is possible to design an acceptable and highly useful digit-entry system with as few as a dozen words. On the other hand, some vocabularies, such as those needed for a medical diagnosis system, require as many as 50 000 words. Nevertheless, the capacity for a large vocabulary is indicative of a system's ability to reliably perform fine distinctions between words, a prerequisite for high accuracy recognition.

A final requirement often associated with the term "speech understanding" is the capacity to interpret ***natural language***. Using a system in a natural way means that the user should be able to use familiar expressive strategies for communication. While it is much easier for the interface designer to specify a rigid command syntax or menu set, these will be unnatural to speak, produce more errors and be awkward to use. The interface must allow the user to use reasonably natural utterances, which are natural English phrases instead of abbreviations or codes [Hauptmann and Rudnicky 88]. Interpreting such natural language communication requires the capability for mapping conventional linguistic constructs onto the internal representations for an intended action. Many of the principles of typed natural language processing can be applied to speech as well. However, the necessary modifications and the added complexity of the speech recognition problem will force the natural language understanding in spoken language systems to be more limited than for typed input. A complete English language understanding system, in the spirit of the 'listening typewriter', is clearly beyond reach, but a limited natural language understanding capacity is feasible and useful for many application domains.

1.3. Special problems posed by spoken language systems

Much of the work in speech recognition has concentrated on processing read speech. That is, speakers are given lists of sentences or words to read. This speech is recorded and then used for training and testing a speech system. This procedure serves a number of useful functions: The input is known, since it is prespecified and can be re-recorded if there is any question as to its exactness or quality. The speaker has an opportunity to rehearse each utterance and can produce it in a fluent, even manner. This guarantees consistency within a recorded corpus. As a result, it is possible to develop and test algorithms under controlled circumstances, an important advantage.

Speech encountered in a "natural" situation will be an entirely different matter. The focus of the user's attention in natural situations is on the task at hand, not on the qualities of the speech that is being produced. When engaged in complex problem-solving activity, or even a simple activity such as dialing a telephone number, the users should not have to pay special attention to the quality of their speech. Thus the speech interface should not only avoid special restrictions (such as discrete-word speech) but also be prepared for speech characteristics common in situations of cognitive engagement, such as hesitations, filled pauses and restarts. Collectively these phenomena are referred to as "spontaneous speech" and represent a violation of many of the assumptions made by designers of existing speech recognition systems. In spontaneous goal-directed speech, utterances are not necessarily complete, acoustically contiguous events. They may contain pauses as the user reformulates an utterance he or she has already begun to speak. They may contain extraneous words, either due to thought processes or because of a shift of attention, for example, to another event in the environment. Furthermore, the user is likely to produce ungrammatical utterances, either as a result of the above processes or through disregard of the particular grammatical constraints imposed by the system.

A spoken-language system which is embedded in a problem-solving environment, where the goal is to accurately and rapidly perform some task, must also provide the user with a number of facilities, such as error repair and a new-word capability, which allow a task to proceed.

Error repair is the process of recovering from a recognition error. A critical determinant of system usability is the ease with which users can recover from the consequences of unintended or misinterpreted inputs. The simplest system provides no such facilities and the user's only recourse is to repeat the input until it is recognized correctly (perhaps first undoing the effects of a misrecognition). A somewhat better system allows for some form of undoing of misrecognitions, though of course the system's ability to do so depends on its ability to accurately detect the request for an "undo" operation. In the case of speech input, it may be reasonable to give the user the option to switch to a different modality (e.g., keyboard) to reliably correct an error. It has been shown in [Rudnicky 89] that such an option indeed reduces total task completion time. Since the goal is to make the system reliable and efficient, the interface designer must combine suitable recognition error repair facilities with some form of fault-tolerant interaction protocols that isolate recognition errors from unintended system actions.

Complex problem solving systems differ in the degree to which they represent *open* or *closed* domains. A calculator is an example of a closed domain, since all activity takes place in the context of a pre-defined universe of discourse (e.g., numbers and arithmetic operations). An application such as a spreadsheet represents an open domain, since the activity involves the introduction of task-specific terms (e.g., to identify items in a calculation). The range of terms we need to understand for a financial planning spreadsheet is different from those for restaurant inventory control. Yet, the functionality of the underlying system, a spreadsheet, is the same in both cases. Users of open systems need to dynamically define symbols for their problem space. One study of users performing spreadsheet problems by voice (see [Rudnicky, et al. 88]) shows that users naturally use symbolic referents that are meaningful within their current problem space. Thus, if the problem is financial, the user will refer to terms (and thus their locations in the spreadsheet) by financial terms: "tax", "income", etc. Complex problems need to have user-defined "variables" since each problem is different and the user naturally wishes to define the problem using semantically applicable terms.

The user must be able to rapidly introduce new words into the system vocabulary and have these be accurately recognized. Unfortunately, introducing new words into the recognition system is a difficult process, particularly for systems that derive their accuracy from training on multiple tokens of a given word. In a spoken language system, in addition to introducing a new recognition item, the system needs to know with the syntactic and semantic role of a novel item. Introduction of a new word therefore involves not only modification of the recognition component but also changes the parsing component, as well as updating the semantic interpretation component of the system.

In an ideal system, the introduction of a new term is dealt with like in a human conversation: the word is introduced, defined, and used thereafter with the expectation that it will be properly understood. Such a flexible new-word facility is beyond the capabilities of contemporary systems.

Together, these facilities will endow a system with the properties of an intelligent interlocutor. The spoken language system acts as a *conversational other*, performing its share of conversation maintenance activities, with the goal of completing the task at hand. Guidelines for spoken language systems are necessary if we are to make applications maximally user-friendly. *Cooperative problem solving* is the key, though the principles we develop should be as independent of specific problem tasks as possible.

2. Principles for Spoken Language System Design

The following principles were derived from experience with several speech applications developed at Carnegie Mellon and from empirical research in speech interaction.

2.1. User plasticity

One of the most striking things about human speech is its fluid, unruly character. A human conversation involves a dynamic interplay, with talkers interrupting each other, correcting themselves, and making use of various speech effects. From the point of view of speech recognition, such speech is highly problematic, since few proven techniques exist for dealing with it (but see [Ward 89]). Fortunately, unconstrained human-human interaction is not the most accurate model for human-computer interaction, as suggested by a recent study [Hauptmann and Rudnicky 88]. In this study, subjects were asked to use an electronic mail program to perform some simple correspondence and database retrieval tasks. The study involved what is known as a "wizard" experiment, where a human operator takes the place of a speech recognition system, and translates spoken instructions into the appropriate system commands. Subjects were led to believe that they would be using a speech recognition system to interact with the electronic mail system. Other subjects were assigned to another condition, in which they were to relay their requests to a human operator who would then type in the requested command. Of interest for the current discussion is the difference in speech style observed for the two conditions.

Figure 2-1: Transcripts of human-human and human-computer speech

Human-Human speech:

```
type message one eighty two
list all headers with the word mckean honda
type one eighty three
forward this message to rudnicky on the a
.
.
.
```

Human-Computer speech:

```
type message one eighty two
search all mckean honda
mail alex at g
send
.
.
.
```

Excerpts from two transcripts of users performing the same task are shown in Figure 2-1. The difference is quite striking. When subjects think they are communicating with a computer, their language is regular and to the point. This is quite contrary to what has been observed in human-human communication in other studies as well [Grosz 77]. In human-computer speech, unessential words are eliminated from the utterance. We believe that this phenomenon reflects the user's intuitive assumptions about how one ought to speak to a computer. Two important observations can be made here: Speech addressed to a "computer" is more orderly than speech addressed to a human, even in a formal experimental situation, with minimal personal contact. Our evidence shows that humans will make some effort to speak clearly to a computer. This implies that many of the spontaneous speech phenomena observed in human-human interaction will be attenuated in a human-computer situation and therefore the speech language interpretation problem is more tractable than might have been assumed on the basis of observing human-human interaction. The second important observation is the apparent ease with which users slip into a "computer speech" mode. They did so without prompting on the part of the experimenter, nor did they have any reason to do so as a result of error feedback from the system, since a human interpreted the speech and the appropriate action was always executed, no matter how the command was expressed. While individuals may have different notions of what constitutes "computer speech", and this is a topic that needs to be investigated, all naturally gravitated to some regular form of speech. In itself this is not an unusual thing for humans to do, since we in fact continually change our speech style depending on context. Linguists have referred to contextually conditioned changes in speech style as changes in *register* ([Lyons 81]). Talking in different styles is a natural human characteristic and there is no reason to expect that humans will not be willing to add a register for computers, which is a crucial advantage for the successful development of speech systems.

2.2. Interaction Protocol Styles

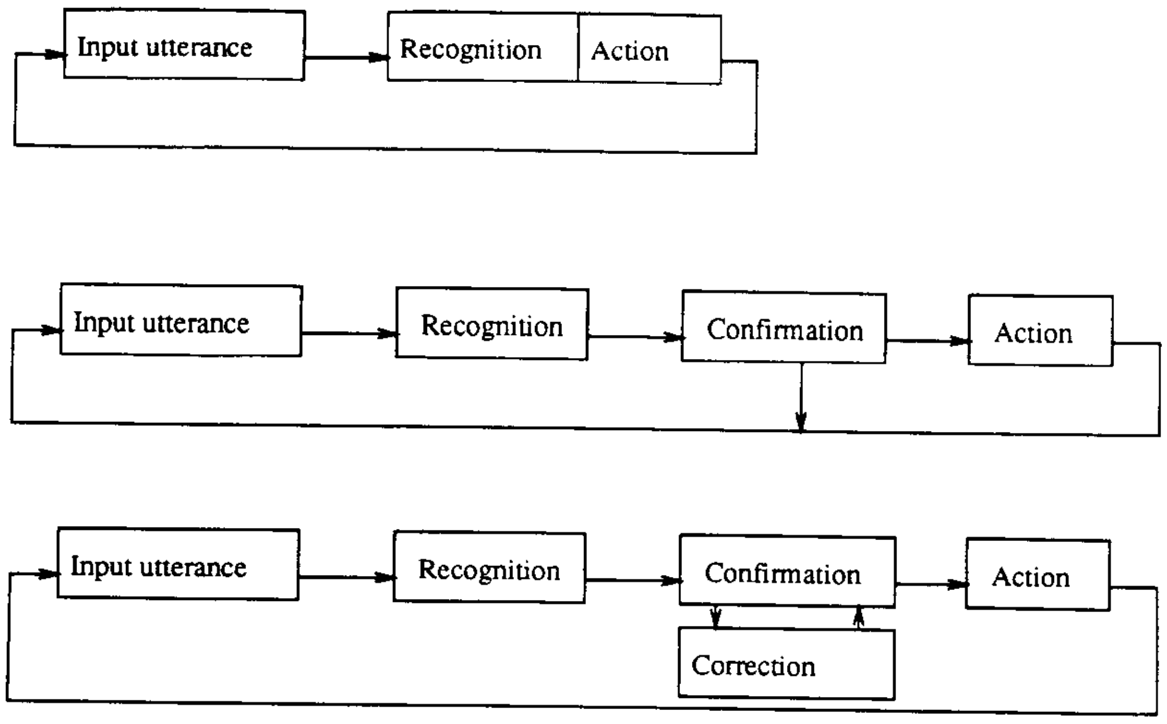
Turn-taking in human conversation is controlled by a mutually-understood protocol [Sacks *et al.* 74]. By giving certain verbal or non-verbal clues, a participant can signal his intention to contribute to the conversation, end his turn, give his conversation partner a chance to speak, ask for clarification from the partner, confirm what was said by the partner, and so on.

Simple protocols exist for human-computer interfaces using keyboards and pointing devices. For example, a user types in a line of text, possibly editing it, then transmits it by entering a terminator character (e.g., a carriage return). Other keyboard interaction protocols are based on single-character "turns": control is passed to the computer after each keystroke. This latter style is frequently encountered in form-filling or menu-based interfaces. Variations of these protocols include pointing devices to select and click on objects. The success and simplicity of these protocols hinges on the dependability of the keyboard interface.

The speech interface, on the other hand, is errorful. A particular input, an utterance, will not always be interpreted correctly by the system (often as many as 1 out of 4 utterances fail in systems that we have built). Fortunately, techniques exist that allow us to control the impact of unreliable recognition, through modification of the interaction protocol.

Figure 2-2 shows three of the speech protocols we have studied. The protocol shown at the top of the Figure is the simplest to implement, but places a heavy burden on the user to catch and correct errors. If the system acts on a misrecognition, the user is forced into a (potentially) complicated process of undoing unintended actions. Even if the misrecognition is relatively benign or merely produces an error message, the user is compelled to retry the utterance and possibly repeat the same error over and over. The advantage of the protocol is its simplicity, the disadvantage is that it can only be used for very high accuracy recognition applications, where the cost of occasional errors can be absorbed or for applications

Figure 2-2: Three simple interaction protocols for speech systems.



where speech input alone provides an overwhelming advantage.

The second protocol introduces an explicit confirmation step; the user must acknowledge the correctness of a recognition before it is acted upon, otherwise the input is flushed. This protocol gives the user control over which inputs are actually translated into action. No special consideration is given to correction of erroneous inputs. Thus it is an acceptable protocol for unreliable recognition situations. A disadvantage is the need to recognize a confirmation word (e.g., "OKAY") with near perfect accuracy, though this may be circumventable by using a keystroke as the confirmation. A more serious disadvantage is the doubling of interactions, since each input utterance must be followed by a separate confirmation input.

The third protocol is similar to the second, but instead of a simple accept/reject decision at the confirmation step, the user is given the opportunity to somehow correct the input. In one study [Rudnicky 89], keyboard correction of misrecognitions produced a 21% reduction in task time. One can envision error correction using only the speech mode, but mixed mode protocols, using both keyboard/pointing and speech might ultimately provide the most effective interfaces.

The choice of an appropriate protocol depends both on the response characteristics of the recognition system and on the requirements of the task being performed. For a situation in which individual inputs need to be correct or system accuracy is low, the lower two protocols with confirmation in Figure 2-2 appear to be well suited. Of course they incur the cost of a separate confirmation step, but presumably at a lower overall cost compared to the correction of unintended system action. If we have a high-accuracy system, we can try to eliminate this cost, while still retaining a form of confirmation, by using the following protocol:

- **Disconfirm Incorrect Recognitions.**

  1. *User speaks utterance.*
  2. *Computer recognizes utterance and displays recognition.*
  3.
     a. *If recognition is correct, user speaks next utterance.*

     b. *If recognition is not correct, user disconfirms recognition using keyword.*
  4.
     a. *If the computer recognizes the disconfirmation, proceed with Step 1.*

     b. *If no disconfirmation was given, computer acts on recognition.*
  5. *Proceed with step 2, using the new utterance.*

The advantage of the above protocol is that it eliminates a separate confirmation step, but still provides the option to intercept an incorrect action. We can therefore approach the efficiency of the simple "no verification" protocol: as the speech recognition system approaches perfect recognition, few errors are made and few 'disconfirmation' steps will be executed. When an error is encountered, suitable special error correction strategies can be implemented within this framework.

The obvious disadvantage is the less intuitive feel of the interaction for the user, partly because of the need to check the accuracy of each recognition and partly, because the break in the interaction cycle does not fall into a "natural" unit: the computer carries out the action of the *previous* utterance and displays the recognition of the *current* utterance in what amounts to a single turn. All misrecognitions still require a separate disconfirmation step in addition to the repetition/retrying of the step.

To develop a better insight into whether these protocols are sufficiently natural and how well users can adjust to them, we implemented them for a digit-string entry system. In addition to one "Explicit Confirm" and one "Explicit Disconfirm" protocol, we included two others. The goal of the two additional protocols was to retain the advantage of explicit confirm / disconfirm while eliminating the need for an extra interaction with the system. While these protocols certainly had the potential for greater efficiency, it was not clear whether they represented natural modes of interaction to the users.

- **Pack Confirmation into Next Utterance**

  1. *User speaks utterance.*
  2. *Computer recognizes utterance and displays recognition.*
  3.
     a. *If recognition is correct, user confirms recognition with a special word and speaks next utterance immediately.*

     b. *If recognition is not correct, user repeats the utterance.*
  4.
     a. *If the computer recognizes the confirmation portion of the utterance, it carries out the action for the old recognition and proceeds to Step 2 using the rest of the utterance.*

     b. *If the computer does not recognize a confirmation, proceed with Step 2.*

The advantage of this protocol is that a separate step is not required for either correct or incorrect recognitions. The user still has explicit control over what action is actually carried out by the computer. However, the disadvantage is that a confirmation word needs to be spoken for each correct recognition. The cognitive load on the user is higher, since the confirmation word must be spoken together with the next utterance, and the results of the previous action will be displayed with the results of the current recognition. Every correct utterance still needs a special word, though not a whole separate confirmation step.

- **Pack Disconfirmation Into Next Utterance**

  1. *User speaks utterance.*
  2. *Computer recognizes utterance and displays recognition.*
  3.
     a. *If recognition is correct, user speaks next utterance immediately.*
     b. *If recognition is not correct, user speaks special disconfirmation word and repeats the utterance.*
  4.
     a. *If the computer recognizes the disconfirmation portion of the utterance, it proceed with step 2, using the rest of the utterance.*
     b. *If the computer does not recognize a disconfirmation it carries out the action for the old recognition and proceeds to Step 2 using the new utterance.*

In some respects, this last protocol is the most desirable. The user retains control of what actions are actually carried out, preventing disastrous errors due to misrecognitions. In addition, the extra word is only necessary when an error occurred, which, in an accurate system, would happen only rarely. Under perfect recognition circumstances, this protocol again resembles the simplest "no verification" protocol, without the drawbacks of possible erroneous actions. Note, however, that it still presents the user with an unnatural segmentation of the interaction cycle, both the previous utterance action as well as the current utterance recognition happen in a single turn.

2.3. Correction facilities

The presence of recognition errors in speech systems governs the choice of interaction protocols and creates the need for a confirmation step, as discussed in the previous section. The usability of a system depends not only on making the confirmation procedure simple and consistent with the requirements of a task, but also depends on the provision of quick and intuitive error correction procedures in case a recognition is disconfirmed. In this section, we describe a number of these procedures and report some studies of their effectiveness.

The primary requirement for an effective error correction strategy is that it be "cheap and easy" in the sense that an error should not unduly interrupt the flow of interaction. That is, it be neither time-consuming nor require a significant shift of attention for the user. Additionally, the strategy should reflect familiar correction procedures that humans already use in other situations. We have identified a number of strategies that approach this ideal to a greater or lesser extent:

Contrastive Emphasis. In human-human communication, the simplest correction mechanism is repetition of the utterance. This has also been found to be a natural reaction to speech recognizer errors: Repeat what you just said, and hope for the best. When users repeat an utterance to the system, we have found that they use changes in emphasis to highlight the misrecognized portions. This use of contrastive emphasis can be detected through cues such as amplitude, duration, silences and pitch contours. Preliminary studies, reported in [Rudnicky and Hauptmann 89] demonstrate that contrastive emphasis is a potentially useful method for avoiding multiple errors when the same utterance is repeatedly recognized. Our experiments have shown that these cues can reliably identify the incorrect portion of the utterance.

We envision a speech interface that uses this information to correct errors. The system has the following information available:

- The person spoke utterance1 to communicate a text.
- The speech system recognized recognition1 which was NOT correct.
- The person spoke utterance2 as a repetition of utterance1.
- The system tentatively recognizes a recognition2 for utterance2.
- We also have word scores for recognition1 and recognition2.
- The analysis of contrastive emphasis identifies a specific portion of utterance1 to be incorrect.

The system now can re-analyze utterance1 based on the new evidence. It can verify the analysis by ensuring the recognition is consistent with the one for utterance2. Different speech systems may provide different capabilities for doing this, but the path is fairly clear. The correct recognition should now be obtainable from utterance1 and utterance2, and is based on using all the available information.

Repetition of an incorrect word in context. While contrastive emphasis analysis allows the system to pinpoint the error region on its own, the user could also actively provide the context. One procedure for doing this is to have the user repeat the misrecognized word together with its left and right contexts. Thus, if the user spoke the words: "`What is Badger's maximal speed?`", and the system misrecognized the utterance as "`What is Badger's minimal speed?`", the user would correct this error by saying "`Badger's maximal speed`". This gives the system a known left context, a correction of the erroneous part of the utterance and a right context for the correction. It should then be possible for the system to identify the misrecognized part in the previous utterance, and correct it based on the correction utterance. The original misrecognized speech can be used as a further reference for the correction.

Select correct words from near misses. An alternate strategy for error correction is to provide the user with a word lattice of near-misses in the recognition. Even though the system has identified its best guess at a recognition, it also displays other word candidates which were close matches. The user then merely selects the corrections to the misrecognition by pointing into the word lattice. No speech is involved in this correction and only a small set of candidates will be presented to the user, making a quick choice possible. Variants of this strategy are already available in some (discrete word) speech systems. There are some drawbacks, however. The strategy involves a mix of speech and manual input, creating a distraction. Another drawback is the possibility that the correct word is not included in the word lattice of near-misses presented to the user. A complete repetition, or other form of correction would then be necessary.

Selection and repetition of the misrecognized portion of the utterance. A mixed-modality strategy that suggests itself is the combined use of pointing and speaking. The user would manually select the part of the (displayed) utterance that was mis-recognized, and repeat it. The system then is able to determine exactly what was wrong and correct the error based of the new input. Again, this strategy assumes that an alternative input mode is available to the user.

Correction by spelling, synonyms or rephrasing. Another alternative for correcting errors is to change what was actually said to the system. If, for example, the system has difficulty recognizing a particular word or expression, it may be advisable to use a different word or expression. It should always possible to spell out a word. As we have discussed in the section on interaction protocols, given the proper protocol the system can infer that the mis-recognized utterance is to be ignored. A drawback of this strategy is the potential inability of the user to rephrase a particular entry. In a very limited system, such as a calculator, we may not have alternative ways to express a number or arithmetic operation. We should also consider the added cognitive load on the user to make a rephrased utterance sufficiently different from the failed utterance.

Typed Correction. One obvious correction strategy for misrecognized speech is to fall back on typed input. Keyboard correction is better than a simple repetition of input, presumably because a large portion of the input was correctly recognized and only certain parts need to be edited. The advantage of this strategy is particularly evident for complex or extended inputs, where the cost of speech and editing is noticeably less than that of the equivalent typed input. The typed correction may engage the facilities of an appropriate editor to correct the displayed misrecognition or the user may type in the complete utterance. This was the strategy examined in an experiment described below, and was shown to be reasonably effective. Of course, this strategy assumes the user has the typing channel available. This strategy typifies the principle of using a different modality for the correction or re-entering of incorrectly recognized speech.

2.3.1. An experiment in correction

While the need for correction facilities in speech recognition systems is clear, the nature of the most effective procedures is not evident. To demonstrate how the ability to correct errors affects task performance, we present the results from a digit-entry experiment in which we asked subjects to enter a list of digit strings into the computer. The digits strings were given to the subject on a sheet of paper placed next to the keyboard and monitor. Three data entry modes were included in the experiment.

- In the first mode, **speech only**, subjects could only use speech to enter a digit string. Subjects read a string, then observed the recognition result on their display. If the recognition was not correct, the subject was instructed to repeat the digit string into the microphone. This procedure was repeated until the recognition displayed was correct. At this point, the subject was to give a confirmation word ("OK" or "ENTER"). The computer system would then store the number that was entered and the subject would proceed to the next number on the list.

- In the **speech with keyboard correction** mode, the subject would again read the digit string into the system. If the recognition was not correct, the subject was instructed to use the keyboard to enter the correct digit string terminated by a carriage return. If the recognition string was correct, or after the keyboard correction was performed, the subject typed a carriage return to store the string.

- In the **keyboard only** mode, subjects typed in the digit string, which was then also displayed for confirmation and correction. If they had miskeyed the string they could correct it again using the keyboard. Once the correct digit string was displayed on the screen, subjects would hit the enter key to store the digit string and proceed to the next string.

The SPHINX speaker-independent, continuous-speech recognition system [Lee 89] was used as the front-end. The speech recognition vocabulary consisted of the 13 words ZERO through NINE, OH, ENTER and OK. When a spoken digit string was recognized, the system displayed the result using numerals (i.e. ZERO, OH $\rightarrow$ 0; ONE $\rightarrow$ 1; TWO $\rightarrow$ 2; ... NINE $\rightarrow$ 9). Typed input was displayed without alteration on the same line as the spoken input.

To determine the efficiency of data entry under the different conditions, we measured the total time a subject needed to enter a number correctly. This aggregate cycle time includes the time elapsed before the subject began speaking after the system had displayed its prompt, the time required to produce the utterance or to type the digit string and (for speech) excess recognition time. The value computed is the result of adding these times for each initial recognition attempt, each correction attempt and the final confirmation cycle. Thus this time reflects the average time to enter a digit string *correctly*, including all correction and verification time.

To highlight the differences between the conditions, we analyzed the aggregate cycle time in terms of its component times: the time for the initial attempt to enter the digit string, the time for the correction cycles and the time necessary for the confirmation cycle. The aggregate cycle time data analyzed by components for this experiment are in Figure 2-3, which also presents the times for different digit string lengths.

In terms of overall performance: In the voice condition, subjects averaged 5.20 seconds for the initial digit string entry, 1.86 seconds for all corrections to the string and 4.06 seconds for the confirmation transaction. In the voice with typed correction mode, we measured an average of 5.32 seconds for the initial digit string entry, 1.70 seconds for all corrections to the string and 2.57 seconds for the confirmation transaction. In the typing condition, the initial digit string entry transaction lasted for an average of 6.23 seconds, while all corrections to the string counted for an average of 0.31 seconds and the confirmation cycle required 2.41 seconds.

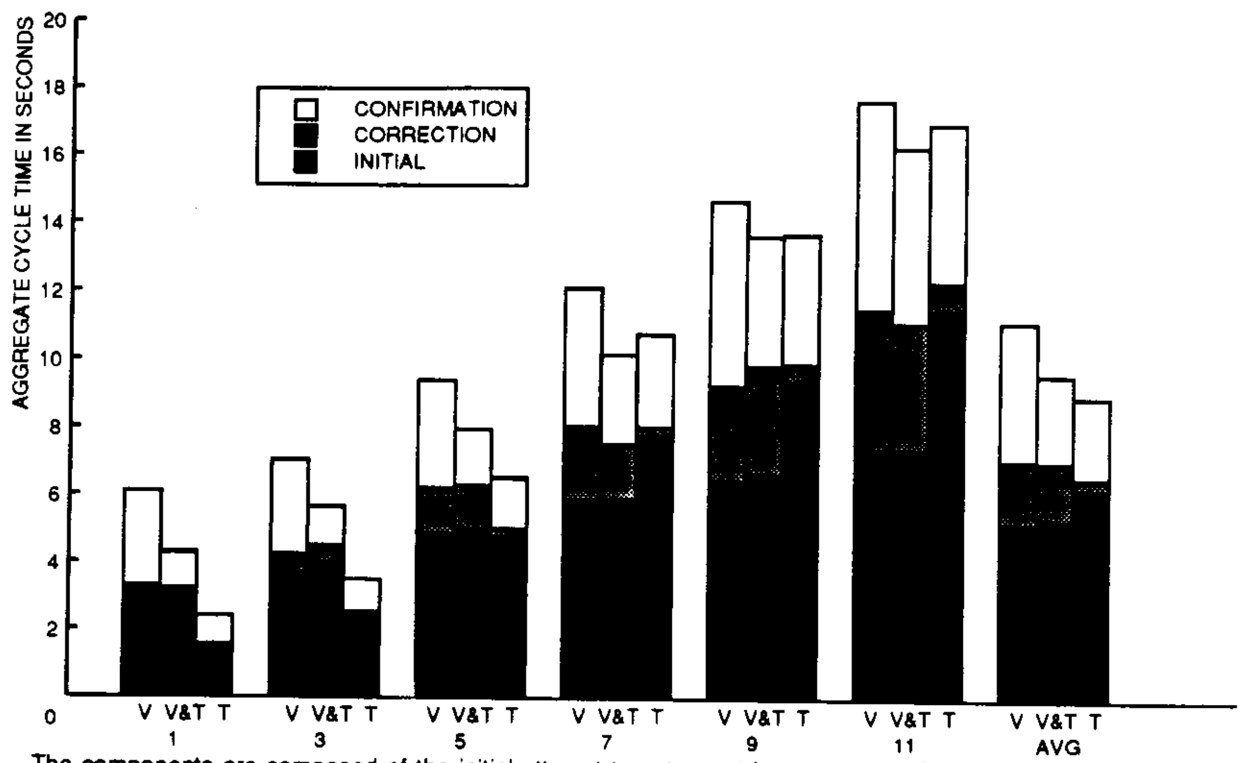
Considering the components of the aggregate cycle time in Figure 2-3 we find relatively fast initial entry times for speech, better than the equivalent time for typing. Speech had a strong disadvantage, especially for longer strings that needed many corrections. The speech mode loses the race due to correction time. Typing accuracy avoids almost any correction, and speech loses most of its ground there. In the confirmation transaction, typing is again very fast, but speech is about a constant amount slower. We also found that confirmation times increase with string length, a reflection of the extra effort needed to verify long strings.

2.4. Response Time

System response time is a critical parameter for effective interaction. In the limit, system response time (SRT) has a decisive impact on the ability to carry out a task on the computer. If too long, users are unable to keep track of the task at hand and cannot execute it properly.

Interestingly, even very short SRT delays, on the order of less than 1 second can affect how a task is performed.

Figure 2-3: Aggregate cycle time broken down by components

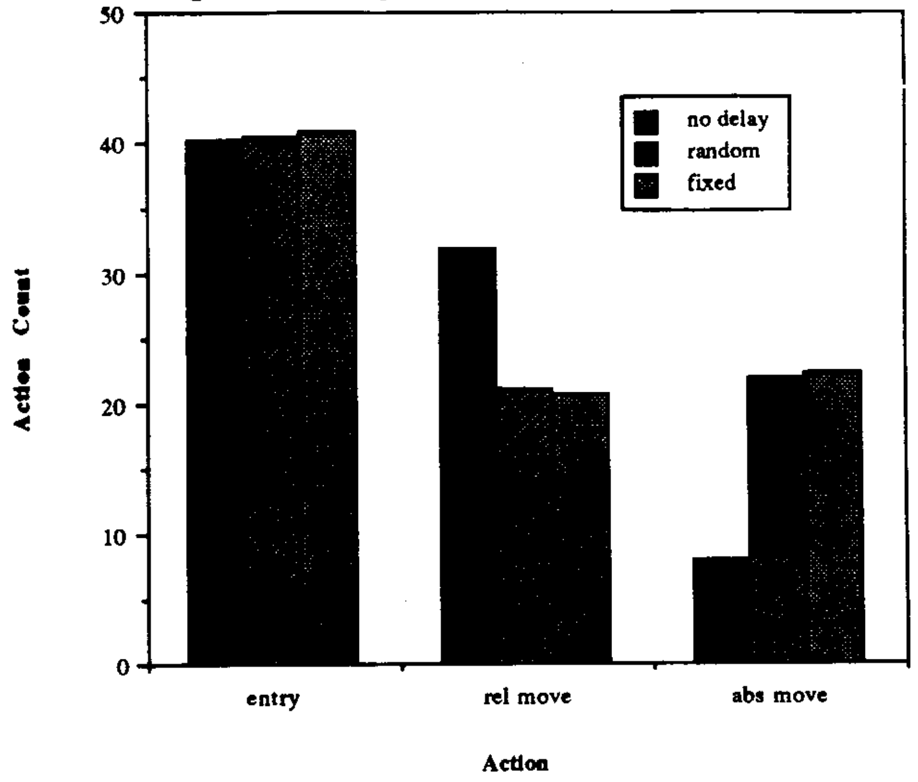


The components are composed of the initial attempt to enter a string, any corrections and the final confirmation that the string is correct. The modes are abbreviated as V=Voice, V&T=Voice with typing, T=Typing. Each digit string length is plotted separately; AVG denotes the overall averages for a condition.

This can be seen in an experiment in which users were asked to carry out a series of simple spreadsheet tasks that involved movement about a spreadsheet and the entry of numeric information. Three different versions of the spreadsheet program were used: an unaltered one, a version with a fixed 820 ms delay after the entry of each command, and a version with a random delay averaging to 703 ms. In the spreadsheet used, movement could be carried out in one of two ways: the use of "arrow" keys that moved the cursor one cell at a time (or several, depending on a prefix value), and a "go to" command that would move the cursor to a specified location, either a cell coordinate (e.g., "B6") or a symbolic location (e.g., "SALARY"). Users were given complete freedom over the choice of command for movement, the outcome of interest being the user's preference for either one or the other movement type. Table 2-4 shows histograms of the proportion of each command type for the three different conditions. Quite clearly, users favor the relative movements in the "no delay" condition but shift to the other category when using the versions that introduce a delay. A similar shift was observed in [Rudnicky, et al. 89], where users used either a (standard) non-delay keyboard version of the spreadsheet or a speech-input version that included processing delays.

The shift to absolute movement is understandable, given the overall time that it would take to reach a destination with relative movement under the two circumstances: relative movement would be very slow compared to absolute movement. Nevertheless, it should be noted that the total time users devote to movement in the two cases was about the same. Given that this is the case, one might wonder why absolute movement was not used more in the no delay condition. After all, the total amount of time spent on that activity would have been about the same. The answer appears to be that users chose the

Figure 2-4: Histogram of Movement action choices



cognitively *less complex* action when given a choice. Relative movement is clearly less complex—it consists of repeatedly moving in the desired direction, stopping at the point where the cursor reaches the desired cell. Absolute movement involves determining either the coordinates of the destination or determining its label, then typing in a suitable command, consisting of a directive followed by the label. All this requires additional effort on the part of the user and it appears that other things being equal, he or she is not willing to put out this effort if a structurally simpler alternative is available.

This result is particularly important for speech interface design (and for interface design in general), since it shows that users are extremely sensitive to small differences in response characteristics and to perceived differences in the relative effort needed for different strategies. The presence of delays disposes users to pack more information into a single interaction with the system, even if they are not naturally inclined to do so. Packing more information into a command unfortunately means that the cognitive load is increased for that interaction. We can now see that this is a problem for speech systems that introduce a processing delay into the interaction cycle. Such systems inherently increase the difficulty of interaction and put the interface at a disadvantage compared to other systems that do not introduce a delay. If delay turns out to be unavoidable in speech systems, then it will work against the supposed advantage of a speech interface—its naturalness and the ease of interacting by speech.

The causes of delay are numerous. Some of these can presumably be dealt with through suitable implementation. A sufficiently fast special-purpose can provide real-time or sub-real-time performance. This is currently true for moderate-sized tasks, but might require substantial effort to achieve for the larger systems currently being contemplated (with vocabularies on the order of 5000 words or more). It is likely that other characteristics of spoken language, particularly its "unruly" aspects (ungrammaticality, restarts, extraneous noises, etc) might require processing strategies that are inherently incapable of generating results in real-time. On the other hand, it is quite clear that users are very willing to modify their behavior (i.e., be "cooperative") in order to produce acceptable system behavior.

2.5. Task specific dialogue structure

Interaction with a spoken language system normally takes place in the context of some meaningful task. Knowledge about the structure of the task and about the ways that people go about carrying it out can be used to derive constraints for speech recognition. The MINDS system developed at Carnegie Mellon provides a demonstration that task knowledge can be used to improve recognition. The MINDS system uses knowledge of dialog structures, user goals and "focus" in problem solving situations. The different kinds of knowledge are combined to form predictions which translated into dynamically generated semantic networks. These semantic networks are then used to constrain and guide the parse of the speech recognizer. Several experiments with the system showed that the use of these multiple knowledge sources improved system recognition accuracy. The more knowledge was brought to bear, the better the performance.

The system represented an attempt to maintain information on what had been talked about and what was likely to be talked about next. While a full description of the system can be found in [Young *et al.* 89], we will emphasize here the most important aspects and their implications for speech interface development. MINDS tracks the dialog-phase or "script" in which a user currently operates. Within this current part of a script, the dialog was restricted to the concepts a person could still refer to and these restrictions were based on the specific task structure the system was designed for. From the set of script and concept restrictions, a semantic network was compiled for each new utterance. This network was then used to parse the incoming speech signal. Thus the recognizer selected the utterance recognition only from those words that were given as plausible choices at each point in the semantic network.

Figure 2-5: Excerpt of a typical MINDS task script.

```
Badger is disabled.

What capabilities did it have.

What was Badger's speed?

Show me its mission area ratings.

Which frigates have harpoons?

Phalanx?

What are their other capabilities?

What is the speed of the Kirk?

What are the mission readiness ratings for Kirk?
```

The effectiveness of this system was demonstrated in an experiment [Hauptmann *et al.* 88] in which speakers read three sets of sentences that represented typical task scripts in the domain of naval logistics planning. An excerpt of one of these scripts is shown in Figure 2-5. By using task-based constraints, the system was able to increase its recognition accuracy from 44.2 percent correct to 66.3 percent accuracy for words, and from 32.1 percent sentential meaning accuracy to 58.4 percent sentential meaning accuracy. Sentential meaning accuracy only counts sentences as correct when the appropriate database query was created from the utterance.

Speech interface design can benefit from the lessons of MINDS in two ways. Recognition accuracy can be improved by reducing the search space of the speech recognizer to the minimum necessary. The fewer words in the active recognition set, the higher the recognition accuracy possible. As a result overall system performance is improved and more complex tasks can be realistically implemented. Second, the structure of speech systems can be designed in such a way as to enhance the decomposability of a domain. For example, a system that consists of distinct subtasks or that supports activities with distinct separable contexts can take advantage of MINDS-like control of the recognition process.

Tasks are susceptible to decomposition to a greater or lesser extent. Some tasks operate on very little context. A telephone directory or a simple calculator might be examples of this. The constraint in these systems is inherent in the syntactic structure of single utterances or perhaps the constraints imposed by well-formedness (e.g., the need to balance parentheses in a formula, even if specified over several utterances). Other tasks, particularly ones that require the assembly of data for decisions, or more simply, the filling of fixed-format forms, can make use of dynamic constraint application as in the MINDS model. An example of the former might be travel arrangement, an example of the latter might be catalog ordering. A MINDS-like approach is beneficial because even though the complete structure of the interchange cannot be predicted in advance, it can be dynamically constructed over the course of the interaction. An understanding of how task constraints might be used is critical to proper system design.

2.6. Multimodal Interaction

A frequent complaint about menu-based interfaces centers on the effort involved in switching between the keyboard and the pointing device. While each modality has clear advantages in some situations, the integration of the two modes is not ideal.

Bolt [Bolt 84] and the Media Lab at MIT have combined speech and pointing into the very impressive "Put That There" system. This system allows a user to point at items on the screen and select them. Using voice commands, the selected items on the screen are moved to new locations or used in certain actions. While we are not aware of any empirical evaluations of the ease-of-use of this system, it represented a milestone for multimodal interaction. "Put That There" took two imperfect channels of communication, speech and gestures, and combined them to provide a much better means of communicating between human beings and computers.

At Carnegie Mellon we performed a related experiment on the use of gestures and speech for graphic image manipulation [Hauptmann 89]. In this experiment subjects were asked to manipulate an object on the screen using either voice, gestures or voice and gestures combined. The manipulations consisted of scaling, rotating and translating operations. The results of this experiment showed an amazing uniformity in the gestures as well as simple speech forms, with a small vocabulary need. Subjects overwhelmingly (58.8 percent) preferred the combined use of speech and gestures for the interface over each modality used alone. In those trials where a choice was possible, subjects actually used both speech and gestures for the manipulations 70.6 percent of the time when both modes were available: This was especially true for the more complex rotation operations. Subjects also tended to perform better when both modes were available. They were able to specify the manipulation more accurately using the two modes together than with either of the modes alone.

The gesture study allows us to see that speech is actually only one component of a complete user interface. While speech, because of its role in human communication, might be the most natural and therefore primary input modality, there a certain forms of communication, particularly those concerned with manipulation of physical or graphic objects, that are expressed more naturally through gestures. A challenge for interface design is to find not only that range of modalities that are best suited for a particular task but also to understand how system functions can be optimally assigned to different modalities.

3. Conclusion

Current developments have created the opportunity to incorporate spoken language systems into the computer interface, in a way that allows for natural use, as opposed to the stilted interactions previously available. These developments have in turn created a real need to understand the nature of spoken language interaction and the characteristics of corresponding human-human interaction. As a result, we have identified those properties that we believe are essential to the usability of spoken language systems and have described some empirical data that help us begin to understand the nature of these properties.

Humans have evolved the capability to use speech, it is a natural communication modality. It does not require the learning of a special skill as the keyboard does and it provides a more direct link between an intention and the external expression of that intention. In this paper we have presented six properties of spoken language interaction that we believe are a key to its successful use. Some of these work to the benefit of the system designer. The willingness of people to adopt a distinct speech style when interacting with a machine means that designers can expect users to accommodate to *reasonable* rules of interaction. Other properties, such as the provision of real-time response are amenable to already available technical solutions, e.g., special-purpose hardware, but provide very clear-cut enhancements to interaction. For others, such as confirmation protocols and error correction strategies, we have merely scratched the surface in our understanding of them, and it will require additional work before we can develop systematic guidelines for their use. Finally, properties such as multi-modal integration represent the ultimate goal for spoken language interfaces, the creation of a complete communications medium.

References

[Bolt 84] Bolt, R.A. *The Human Interface.* Lifetime Learning Publications, Belmont, CA, 1984.

[Gould *et al.* 83] Gould, J.D., Conti, J. and Hovanyecz, T. Composing letters with a simulated listening typewriter. *Journal of the Association for Computing Machinery* 26(4):295 - 308, 1983.

[Grosz 77] Grosz, B.J. *The Representation and Use of Focus in Dialogue Understanding.* Technical Note No. 151, SRI Stanford Research Institute, Stanford, CA, 1977.

[Hauptmann 89] Hauptmann, A.G. Speech and Gestures for Graphic Image Manipulation. In *CHI-89: Proceedings of the Conference on Computer Human Interfaces.* Austin, TX, May, 1989.

[Hauptmann *et al.* 88] Hauptmann, A.G., Young, S.R. and Ward, W.H. Using Dialog-Level Knowledge Sources to Improve Speech Recognition. In Gregg, L.W. and Steinberg, E.R. (editor), *Proceedings of AAAI-88, The seventh National Conference on Artificial Intelligence,* pages 729 - 733. American Association for Artificial Intelligence, 1988. Saint Paul, MN.

[Hauptmann and Rudnicky 88] Hauptmann, A.G. and Rudnicky, A.I. Talking to computers: an empirical investigation. *International Journal of Man-Machine Studies* 28(6):583 - 604, 1988.

[Lee 89] Lee, K.-F. *Automatic Speech Recognition: The Development of the SPHINX System.* Kluwer Academic Publishers, Boston, 1989.

[Lyons 81] Lyons, J. *Language and Linguistics.* Cambridge University Press, New York, 1981.

[Reddy 76] Reddy, D.G. Speech Recognition by Machine: A Review. *Proceedings of the IEEE* 64(4):501 - 531, April, 1976.

[Rudnicky 89] Rudnicky, A.I. The design of voice-driven interfaces. In *Proceedings of the DARPA Workshop on Spoken Language Systems,* pages 120-124. February, 1989.

[Rudnicky and Hauptmann 89] Rudnicky, A. I. and Hauptmann, A. G. Errors, repetition, and contrastive stress emphasis in speech recognition. In *AAAI Spoken Language Symposium.* March, 1989.

[Rudnicky, et al. 88] Rudnicky, A.I., Polifroni, J.H., Thayer, E.H., and Brennan, R.A. Interactive problem solving with speech. *Journal of the Acoustical Society of America* 84:S213(A), 1988.

[Rudnicky, et al. 89] Rudnicky, A.I., Sakamoto, M.H., and Polifroni, J.H. Evaluation spoken language interaction. In *Proceedings of the DARPA Workshop on Spoken Language Systems.* October, 1989.

[Sacks *et al.* 74] Sacks, H., Schegloff, E. A., and Jefferson, G. A Simplest Semantics for the Organization of Turn-Taking for Conversation. *Language* 50(4):696-735, 1974.

[Ward 89] Ward, W.H. Modelling Noise Events with HMMs. In *Proceedings of the DARPA workshop on spoken language systems.* October, 1989.

[Young *et al.* 89] Young, S.R., Hauptmann, A.H., Ward, W.H., Smith, E.T. and Werner, P. High Level Knowledge Sources in Usable Speech Recognition Systems. *Communications of the ACM* 32(2):183 - 194, 1989.