The design of voice-driven interfaces

Alexander I. Rudnicky

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

This paper presents some issues that arise in building voice-driven interfaces to complex applications and describes some of the approaches that we have developed for this purpose. To test these approaches, we have implemented a voice spreadsheet and have begun observation of users interacting with it.

Introduction

Interaction with a complex system imposes a substantial cognitive load on a human user. Voice, being a natural communication modality for humans, allows the user to spend less time on translating requests into keyboard actions and more time on problem solving activities. The benefits of voice input will very likely be realized through such observable consequences as shorter completion times, fewer errors in finished work, and increased user satisfaction. The design of interactive systems that use continuous speech for complex problem-solving activity, however, is not well understood. This purpose of this paper is to describe an approach to the problem of carrying out such designs.

To study the variables that affect the behavior of a user performing a task using voice, we must construct a suitable environment. The choice of task is critical: simulations, for example, while the allow the experimenter to control all aspects of the environment might not produce findings that are easily translatable to other applications. On the other hand, some "real" tasks may not necessarily produce behavior that is "interesting" from a spoken language system perspective. Simple data entry tasks might fall into this category. We believe that the spreadsheet environment has the desirable characteristics we are seeking in that it will permit us to study a wide range of behavior. We will describe it in greater detail below. Before doing so, we would like to review the issues that must be addressed in the course of building an interface.

Speech Interface Design

Designing a functioning spoken language system requires attention to a number of factors. This section reviews these and describes some of the approaches we have taken.

Language design

Language design refers to the creation of a spoken language suitable for human interaction with an application. The goal is to create a *habitable* language that captures the range of expression to be expected from the typical user and maps it into the capabilities of the application. From the perspective of the system we have been exploring, we can make the further distinction between *core* language and *task* language. Core language is a component that is not expected to change from one circumstance to another and encompasses certain common modes of expression. A good example of such a core is the *numbers*. There are many ways to express numeric information and in fact capturing these leads to the construction of a fairly complex grammar. However, once created, it is clear that the number grammar fits not only into, say, the spreadsheet task but can be embedded in any number of tasks that require the specification of numeric information. A less general language, but one that is still a core is the language that expresses the basic commands available in a particular application. Thus, a calculator core language might include arithmetic, assignment, and function statements. Finally, we can talk about non-core language that is task-specific. An example of this is the financial planning task described below, which brings with it an entire range of words (e.g., TAXES) that are specific to that task only.

Our approach to developing a task language has been to first study how humans speak in the task environment in question when given no constraint on how they express

themselves. To do this, we have been conducting a series of protocol collection experiments in which users are asked to perform a task using only voice, but are given only minimal restriction on what they may say. This is meant to encourage the participant to express intentions in the most direct fashion and not feel constrained by any perceived limitation in the capability of the interface (which in the case of these experiments consists of a human operator translating the subject's verbal instructions into appropriate command sequences). To this time, we have performed two series of experiments, one for a data-entry task embedded in a spreadsheet and the second for a "planning" task involving the satisfaction of a series of constraints given a financial summary. The first study is described in more detail in (Rudnicky, et al., 1988). Figure 1 shows an excerpt from one of the protocols generated in this experiment (a total of twelve were collected, from different people).

We found that users, although they produce a seemingly bewildering variety of expression, tend to, as a whole, to speak a language that is amenable to analysis and can be described with sufficient precision to allow the construction of a grammar. Figure 2 show one of the categories recovered through our analysis. As can be noted, the grammatical structures used are fairly straightforward. We have used these as a guide in implementing the language used in the system to be described below.

Figure 1: Sample protocol

.
.
.
under cable, enter fifteen dollars even [*]
oka:y, go to the next screen [*]
oka:y unde:r . movies .. in the entertainment, section, subsection .
enter twenty-eight fifty [*]
oka:y .. u:h in the . FOOD subsection under HOSting [*]
enter . hundred eighty dollars [*]
on the line above that, enter a hundred seventy-five dollars [*]
okay, go down to the oh wait
on the CLOthing . entry [*]
ente:r . sum of . eighty-five, eighty-nine, twenty-four [*]
[click] okay
and, under gasoline, enter twenty-two fifty [*]
okay . uh . under child support slash day care [*]
enter ..... uh FORTY times [rustle] . three: time four [*]
okay
u:m .. [click] okay, go down to the: uh, next screen [*]
.
.
.

Figure 2: The DATA ENTRY category

Numeric
sixty-five hundred dollars

Absolute Locate and Enter
uh go down to line six and enter four thousand five hundred

Relative Locate and Enter
on the next line enter seventy-five ninety-four

Symbolic Locate and Enter
go to the cell for my salary and enter the amount six thousan five hundred under credit card enter the amount zero

Symbolic Locate and Enter (no keyword)
food two seventeen and eighty-five cents plus two hundred fo

Absolute Locate and Enter (reverse order)
enter six thousand five hundred in column b row 6

Relative Locate and Enter (reverse order)
enter seven hundred and forty-eight fifty-seven there

Symbolic Locate and Enter (reverse order)
fifteen dollars under bank charges

Fluent Interaction

Defining a suitable spoken language is only one aspect of building a spoken-language system. In addition, we need to provide facilities in the interface that encourage *fluent interaction*. Fluency is achieved when a system creates for the user the sense of communicating with a competent interlocutor. Some aspects of this competence are derived from the ability to model the progress of a dialog and building up a model of user characteristics. Other aspects, and the ones of concern in the present case, have to do with providing facilities that allow interaction to proceed smoothly. There are three aspects to this: Giving the user sufficient feedback such that there is a sense of control over the recognition process, providing error repair facilities that allow naturalistic recovery from errors, and allowing the user to define new words and constructs for purposes of a task.

Interaction control involves facilities such as being able to control whether the recognizer transmits nonsense to the application, providing feedback to the user to confirm that actions have been taken, and allow the user to undo the effects of unintended commands or misrecognized utterances.

Error repair provides the user with facilities for recovering from recognition errors. Ideally this facility should function much like a human listener, providing the user with a variety of repair strategies. At the same time, the user should have access to an absolute override mechanism. We have explored aspects of error correction based on repetition (Rudnicky and Hauptmann, 1989) and intend to explore a number of strategies based on voice editing. In our current implementation, users have only two ways of correcting an error: repeating the utterance and hoping it is recognized correctly, or manually editing a partially parsed version of the utterance.

A **new word mechanism** allows the user to define new terms to be used in the course of the interaction. There are a number of compelling reasons for providing this facility. Humans find it much more natural to use symbolic labels to refer to objects in their workspace; there is even some quantitative data that indicates that this is the case (Lerch, 1988). Certainly this was the case for the individuals in our protocol study. An application such as the spreadsheet is not tied to a particular domain and in fact needs to be programmed for each new task. The ability to enter appropriate symbolic information is therefore critical to the full usability of a voice spreadsheet. A new word facility is useful even for systems that do not necessarily deal with new information, but wish to provide the user with the ability to create synonyms for existing constructs.

Recognition

As necessary as a well-constructed language and a fluent interface is the need for a high-performance recognition system. For our present purposes, this means high accuracy and real-time. High accuracy is necessary since without it, the user spends an unacceptable amount of time recovering from recognition errors. At some point, the cost of error recovery exceed that of using some other, presumably less desirable but more reliable, input mode such as the keyboard and causes the user to abandon voice input. Real-time is essential because the user will be unwilling to wait for speech to be understood, especially if a faster modality is available (the keyboard, depending on the nature of the input; for a highly optimized minimal-keystroke system such as a spreadsheet, the margin is very thin; for involving extended text strings, the margin is more forgiving but nevertheless there.)

Although SPHINX is a speaker-independent system, it is nevertheless a task-dependent one, in that the phonetic models used in the system need to be retrained for each new task. In creating a number of small tasks, we found that the proportion of models that can be reused from task to task is about two thirds. Thus, if we were to take the triphone models trained for the Resource Management task and build word models for the spreadsheet task, we would find that only about two thirds of the necessary triphones were available, and we would be forced to use monophone models for the remainder of the vocabulary. This may even understate the lack of coverage, as relative word frequencies may be such as to emphasize the triphone-poor words in the new vocabulary, leading to poorer than expected performance.

To increase system accuracy, we undertook to record several datasets that would provide suitable training material for our needs. To this date (February 1989) we have collected approximately 4300 utterances, spanning three domains: programmable calculator, spreadsheet command set, and financial planning. Tables 1 and 2 present some results we have obtained for recognition on two sets of utterances, using word-pair grammars. The effects of three sets of training data re shown: Resource Management (RM), (nominally) 1000 calculator utterances, and (nominally) 2000 calculator, spreadsheet, and financial planning sentences. The first Table shows results for a spreadsheet language, the second table is for a language augmented by terms specific to financial planning. The Spreadsheet dataset included 145 utterances with an average utterance length of 5.2 words. The test-set perplexity is rather high, 50.3, reflecting the relative lack of constraint in the spreadsheet command language. The Financial Planning dataset included 96 utterances, with an average utterances of 2.8 words, comparatively short. The test-set perplexity is 34.5 (since it emphasized the task-specific vocabulary and used a different, more constrained grammar). Note that these results do not reflect training on all the data we currently have available, only about half. We expect performance to improve as models based on the full dataset become available.

A voice spreadsheet system

To understand how the various factors affecting system usability interact, we implemented a voice-based spreadsheet system and studied its behavior under different conditions.

The Carnegie Mellon system uses a UNIX program "sc" as the back-end and the SPHINX recognition system as the front-end. To link the two, we built an interface, show in the diagram in Figure 3.

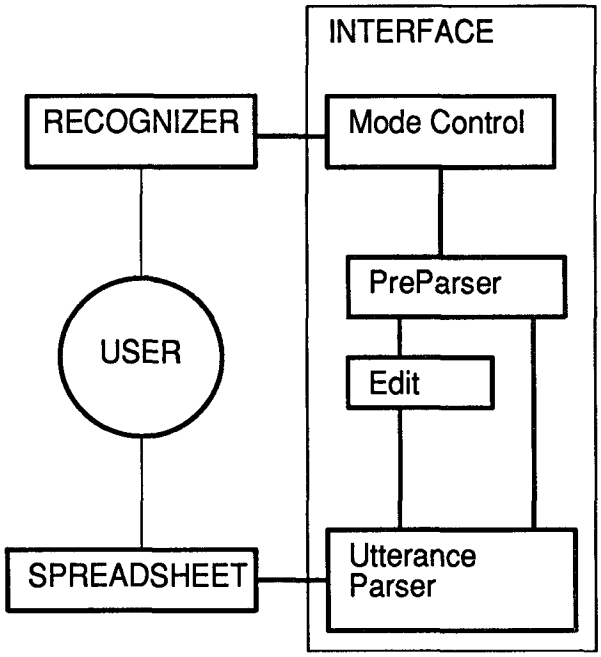
Table 1: Effect of training on SPREADSHEET (word-pair grammar, 119 word vocabulary)

[error rate]	Word %	Sentence %
RM	21.4	68.5
1000 utts	13.0	32.1
2000 utts	4.4	21.7

Table 2: Effect of training on FINANCIAL PLANNING (word-pair grammar, 272 word vocabulary)

[error rate]	Word %	Sentence %
2000 utts	8.1	15.6

Figure 3: Voice spreadsheet system



The interface components include the following: a **mode control** that handles mode changes and controls feedback functions; a **preparser** that both determines whether the input is syntactically legal,parses core subvocabularies (such as numbers), and collapses synonyms; an **utterance parser** that maps the spoken command into the symbolic form required by the back-end. There is also an **edit** buffer that (optionally) allows the user to manually edit the partially parsed input. The **mode control** performs interaction control functions. The functionality of some of the components is minimal at present. For example, the mode control will eventually incorporate the new-word component of the system, while the manual editor will be supplemented with or replaced by voice-driven error correction.

A preliminary evaluation of usage

To gain some understanding of how humans might interact with a voice spreadsheet, we performed an experiment in which participants performed a financial data entry task using the system. The tasks used were the same as those in the protocol gathering experiment described earlier. The operations that had to be performed to complete the task included not only movement and numeric entry, but also the entry of arithmetic expressions, including addition and multiplication.

We have currently collected data from three subjects. All the participants had prior experience with spreadsheets, having used them in their own work. Other than witnessing demonstrations, none had used a speech recognition system before. Three input modes were defined: *keyboard only*, *voice with (keyboard) editing*, and *voice only*. In the first condition, users keyed in all information, using the spreadsheet as they might under conventional circumstances. In the voice with editing condition, they spoke the commands and had an opportunity to edit each preparser string before it was acted upon.

At the start of the session, the subjects were given a general description of the tasks they were to perform and asked to read a short summary of spreadsheet commands. They were then asked to do a short exercise involving movement, data entry, and arithmetic. They did this using the keyboard, then repeated the same exercise using voice. The voice condition used for the exercise was the same as the first one they would actually encounter.

The subjects were given only minimal guidance as to the proper way to formulate commands and were encouraged to speak in a manner they were comfortable. Despite this, the subjects had a tendency to adopt a "speaking to a computer" speech style. For example, one user persisted in entering numbers as digit strings, even when reminded that this was not necessary.

Three different scripts were used, presented in the form of a sheaf of slips, each slip containing some information corresponding to a single spreadsheet cell. This format allowed us to introduce situations in which the user had to "backtrack" and modify already entered information. Each script contained an average of 38 data, six of these required the use of arithmetic operations. The orders of presentation for scripts and conditions were counter-balanced according to a greco-latin design.

The voice spreadsheet system was instrumented to generate a variety of information: the command sequences used and their corresponding spoken version (if applicable). Certain events were time-stamped, allowing us to recover the duration for different components of the interaction. Table 3 presents mean time-to-completion, the number of separate commands given by the three participants in each of the conditions, and recognizer performance. As can be noted from the table, keyboard input was fastest, with voice-only input the slowest. The latter result is distorted by the time posted by one of the subjects. This individual had a cold and moreover persisted in certain behaviors that interacted unfavorably with the recognition system. The last column lists *utterance accuracy*, the number of utterances that contain at least one word error. The error rates are comparable for the two voice conditions we examined and (as might be expected) are higher than those reported for recordings of read speech (see Table 2).

We are continuing to run subjects in this experiment and expect to develop a more stable picture of the relationship between the different conditions.

Table 3: Evaluation results

	time	commands	SR error
keyboard	12:39	187.3	--
voice & keybd	15:16	112.6	29%
voice	19:21	134.0	25%

Conclusion

This paper has presented a methodology for developing spoken language systems. It's main points include:

- The observation of users performing the task by voice in an unconstrained situation and the development of a language model based on such observations.
- The design of an interface that includes various facilities that promote fluent interaction, error repair, and the capability to introduce new (task-specific) words.
- Task-specific training that allows the creating of models phonetically appropriate for a task.

An implementation of a voice spreadsheet based on these principles was produced and it was shown that people can accomplish useful work with it. In our current work, we are extending our study of the voice spreadsheet system to encompass additional metrics, more users, and different tasks (in particular planning).

Acknowledgments

A number of people have contributed to the work described in this paper. Robert Brennan did the initial implementation of the voice spreadsheet program, Joseph Polifroni conducted the protocol studies, Janet Patterson and Robert Weide collected the training data.

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 the US Government.

References

Lerch, F.J., Mantei, M.M., & Olson, J.R. (1988). *Skilled financial planning: The cost of translating ideas into action* (Tech. Rep.). Carnegie Mellon University: Center for the Management of Technology,

Rudnicky, A. I. and Hauptmann, A. G. (March 1989). *Errors, repetition, and contrastive stress emphasis in speech recognition.* AAAI Spoken Language Symposium.

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