Universalizing Speech: Notes from the USI Project

Stefanie Shriver, Roni Rosenfeld, Xiaojin Zhu, Arthur Toth, Alex Rudnicky, Markus Flueckiger

School of Computer Science
Carnegie Mellon University, Pittsburgh PA USA
sshriver@cs.cmu.edu

Abstract

This paper discusses progress in designing a standardized interface for speech interaction with simple machines – the Universal Speech Interface (USI) project. We discuss the motivation for such a design and issues that must be addressed by such an interface. We present our current proposals for handling these issues, and comment on the usability of these approaches based on user interactions with the system. Finally, we discuss future work and plans for the USI project.

1. Introduction

As one of the most common modes of human-*human* interaction, speech holds great promise as a medium for human-*computer* interaction. Speech is natural, and the vast majority of humans are already fluent in using it for communication. Furthermore, technology now exists for reliably allowing machines to process and respond to basic human speech and it is currently being used as an interface medium in several commercial applications, such as dictation systems (e.g. IBM ViaVoice™, L&H Voice Xpress™, Philips SpeechPro™), web browsers (e.g. Conversay), and information servers (e.g. TellMe, BeVocal).

However, the characteristics of human speech and language still pose many problems for designers of speech-based interfaces. A key feature of spoken language is its unbounded variability, but speech recognition systems perform best when the speaker uses a limited vocabulary and syntax [1]. Machines are also limited in their ability to understand the underlying meaning of words: humans do not simply identify words as they hear them – rather, they extract semantic and pragmatic meanings from a string of words based on their order, their prosody, and the context (both spoken and situational) in which they were uttered.

Additionally, humans tend to follow certain rules when engaging in conversations with others. Grice, for instance, proposes that humans follow certain conversational maxims, such as being brief, being “orderly,” and making one’s contribution to the conversation no more and no less informative than is required by the situation [2]. Clark further stresses that language is a joint activity in which both participants must work to make a conversation succeed, especially with respect to problems that could (and do) arise over the course of the conversation [3].

Well-designed speech interfaces must take all of the above issues into account. In short, they must be able to

- handle problems that result from speech recognition errors,
- give user input a fine enough interpretation to perform the appropriate task, and
- play the proper role of a participant in a conversation.

This last item includes making the user aware of problems that may have arisen, apprising them of the state of the system, and facilitating the progress of the conversation by conveying just the right amount of information. Given the limitations of speech recognition and language processing, the interface should also convey to the user the fact that their conversational partner is simply a tool, in order to discourage the user from ascribing too much intelligence to the system and exceeding its functional capabilities.

We have been working on designing the Universal Speech Interface (USI), which we hope addresses many of the above issues for speech user interface design. Our general approach to these issues is to have the user and the machine meet halfway. That is, rather than allowing unrestricted natural dialog (which is hard for the system, but easy for the user) or requiring adherence to strict command-and-control sequences each of which is unique to a single application, we ask the user to adapt a bit to a universal style which makes it easier for the system to handle the issues noted above.

2. Current Systems

Our first application was a USI MovieLine – a telephone-based application providing information on movie theaters and current films in the Pittsburgh, Pennsylvania, area. This work was first reported in [4]. Since part of our goal is to investigate in what ways universal interfaces can reduce learning time and increase user efficiency and proficiency across applications, designing USI interfaces for multiple applications has been central to this project.

We have recently developed our second application, a USI ApartmentLine, which provides listings of available rental properties in Pittsburgh neighborhoods adjacent to Carnegie Mellon University. We chose to make our second application another telephone-access database application, as we felt that this would require the least amount of hardware and backend adjustment from our MovieLine setup, and would allow us to focus our efforts on porting the interface design. Indeed, porting the USI to another application was generally quite simple, and required only very minor modifications to the USI code.

The most challenging issue we faced when developing the ApartmentLine was the realization that, although it is still based on a queryable database like the MovieLine is, the type of information that users retrieve from the ApartmentLine is quite different. In the MovieLine, users specify a simple piece of information that they would like to retrieve from the system, such as movie names (given theater *x*) or show times (given theater *x* and movie *y*). In the ApartmentLine, this kind of simple information is not generally helpful – we would not have much use for a simple list of addresses (given a constraint like ‘2 bedrooms’) or a list of neighborhoods (given price range *x* and a constraint like ‘unfurnished’).

Instead, the type of information we are likely to find useful from the ApartmentLine is complex, summary information: a description of each available property that meets a certain set of constraints. This structural difference led us to revise our list navigation strategies, and reiterated the importance of developing multiple applications in this project.

3. Basic Design Principles

The principal features of the USI approach are a set of universal keywords and interaction guidelines. We have approached the development of the USI as an iterative process, with feedback from user interactions guiding the design of the system. The keywords have been discussed previously in [4] and have remained fairly static to this point; we will review them briefly here before discussing in more depth the current interaction structure of the USI. Fig. 1 shows an example USI ApartmentLine interaction; in the rest of this paper, the notation (n) will refer to line numbers in this figure.

1 **Neighborhood is Shadyside**
2 *Shadyside*
3 **Now what?**
4 *Distance to campus is, rent is, furnished is, <ellsig>*
5 **Rent is less than 700 dollars**
6 *Less than 700 dollars*
7 **Furnished is now what?**
8 *Furnished can be yes, no, partially furnished*
9 **Furnished is no, number of bedrooms is two. Go!**
10 *Unfurnished, two bedrooms. 4 matches.*
At 5555 Forbes Ave, apartment, $650,
0.5 miles from campus <ellsig>
11 **Start over**
12 *Starting over*
13 **Neighborhood is Regent Square**
14 *Wilkinsburg*
15 **Scratch that. Neighborhood is Regent Square**
16 *Regent Square*
17 **Distance to campus is less than 3 miles, number of bedrooms is at least 2**
18 *Hmmm… at least 2 bedrooms*
19 **Distance to campus is less than 3 miles**
20 *Less than three miles*
21 **Rent is less than $1200**
22 *Hmmm… I didn't understand 'rent is less than Oakland'*
23 **Rent is under $1200**
24 *Less than $1200*

Figure 1. Sample ApartmentLine interaction.
User utterances in bold; system responses in italics.

3.1. Keywords

The USI keywords are designed to provide regular mechanisms for performing interaction universals, which were derived by analyzing several applications and application categories prior to developing the USI vocabulary. These universals include help, orientation, navigation, error correction, and general system interaction. Our goal is to limit the regular set of USI keywords to around ten. Currently the system incorporates the following keywords:

- `NOW WHAT?` allows the user to find out what can be said next at any point in the application (3,7).
- `GO!` sends the user's command to the application (9).
- `SCRATCH THAT` cancels the user's last utterance (15).
- `START OVER` erases all accumulated context (11).
- `REPEAT` replays the system's last utterance.
- `RESTATE` tersely restates the accumulated context.
- Navigation keywords allow users to move through lists of information. Our current set includes `MORE`, `NEXT`, `PREVIOUS`, and `STOP`.

3.2. Interaction structure

By incorporating a standard interaction structure as part of the USI design, we feel that users will be able to work more efficiently and accurately with USI applications. We also hope that introducing a structure that is intended to be used with many different applications will mitigate any negative effects that may otherwise be associated with learning unique, specialized command languages. The USI uses standard interaction structures for both input and output. By standardizing user input we hope to reduce the negative effects of variability on system complexity, since a less complicated grammar and smaller vocabulary can be used. Additionally, by standardizing the output of the system, we believe we can improve the learnability of the USI across applications by improving the user's ability to understand the information presented by the machine. Standardization in the USI takes the form of principles governing the regularities in the interaction and includes the following items.

3.2.1. Phrases

USI input is always provided in phrases, each of which conveys a single information element. Phrasal units map to `<slot> + <value>` pairs in a form-filling paradigm. Users can enter a single phrase per utterance (1), or they can string together several phrases in one turn (17).

In our current applications we have been using a strict `<slot> is <value>` syntax for our phrases. The form `<slot> is what?` allows the user to get specific, as opposed to summary, information from database applications (for example, in a flight information system, a user might say `departure gate is what?` in order to find out what gate a specific flight is leaving from).

In general however, the USI gives each application designer the flexibility to specify the syntax for phrases within an application, and we plan on experimenting in the future with freer, more natural phrase structures. At this point we do allow common synonyms in phrases, however, and have found that these are often intuitively used by users (21,23). For instance, in the USI MovieLine, users can and often do say `movie is what?` or `titles are what?` even though `title is what?` is the only form that is ever given by the system as an example or prompt.

3.2.2. Command Execution

After entering a string of phrases, a keyword (currently `GO!`) is used to send the command to the application (9). We considered as an alternative design having a query phrase (i.e. `<slot> is what?`) signal command execution, since it seems natural for questions and their accompanying intonation to signal the user's desire for the command/question to be answered – indeed, we have found that users often tend to forget to say `GO!` in those situations where they end their commands with a query phrase.

However, by delaying command execution until it is explicitly specified by the user, we accomplish several things. First of all, we can allow user utterances to contain less than a full command. This in turn allows novice users to proceed more slowly and allows users who are experiencing speech recognition problems to better pinpoint where errors are occurring and to correct them in a stepwise manner. We would also like to allow users some freedom in the order in which they enter their phrases, rather than requiring the query phrase to fall at the end of an utterance.

Using an execution keyword also enables the USI to accept different types of commands. One example is commands that do not require or need a query phrase: for instance, in a non-database situation such as setting the radio station on a car stereo, or when querying for summary rather than discrete information in a database application.

Furthermore, at times a user might want to include more than one query phrase in a command in order to receive a matrix of information (for example, in the USI MovieLine: `movie is Casablanca, theater is what, show times are what?`), and this is easily accommodated by using an execution keyword.

3.2.3. Output, general

In many speech interface environments, no visual display is available, so extra care must be given to the design of audio output to ensure that the system is able to convey information and express concepts to the user clearly yet concisely. We believe, and others have noted as well [5, 6], that often information can and should be conveyed tersely.

Unnecessary verboseness and repetition in a system can become tiring; since we propose the USI as a universal interface, we envision that people might interact with USI applications many times each day, which would magnify the effect of unnecessary verboseness. Therefore, the general paradigm for USI output is to convey information as succinctly as possible, and to prompt the user for information only when absolutely necessary.

3.2.4. *Confirmation and Error Detection & Correction*

In keeping with our general strategy of concise output, we have adopted a succinct confirmation strategy in which the system only confirms that input which it has understood and therefore does not have to distinguish between different types of errors. There are several different contexts in which confirmation and error detection can occur.

In general, the USI confirms each entered phrase with a paraphrase of its value (2,5,24). In (13-14) a parsable recognition error occurs, and so it is up to the user to correct this using `SCRATCH THAT`. In (17-18), one phrase fails to parse (in this case because of a recognition error, since the input uses the correct syntax and vocabulary), and the user can simply repeat the phrase which was left out of the confirmation. Note that whenever a problem phrase is left out of the confirmation, the system precedes the confirmation with a signal to the user that something was missed – a "hmmm…" in our current applications.

The real exception to the confirmation strategy occurs in cases like (21-22), where only a single phrase has been spoken and it generated an error. Rather than respond with silence in this case, we tell the user what the system "heard," in order to help them figure out where things went astray.

3.2.5. Lists

Particularly in database applications, information that is returned to the user often takes the form of a list. We have implemented standard USI structures for the output and navigation of two varieties of lists: simple/simple, and simple/complex, as shown in Fig. 2. We have also considered strategies for the third type of list information, complex/simple, but have not yet implemented or tested our approach for this.

```
1. SIMPLE LISTS OF SIMPLE INFORMATION
   theater is the Squirrel Hill theater,
   movies are what? go!
      Squirrel Hill theater, 6 movies.
      Traffic, Gladiator, Chocolat, <ellsig>
2. SIMPLE LISTS OF COMPLEX INFORMATION
   rent is under $700, neighborhood is
   Shadyside, go!
      Under $700, Shadyside, 4 matches.
      At 5942 S Aiken Ave, one bedroom
      unfurnished apartment, $650, <ellsig>
3. COMPLEX LISTS OF SIMPLE INFORMATION
   theater is the Manor theater, movies
   are what? show times are what? go!
      Manor theater, 4 movies. Traffic:
      4:50, 6:55, 8:20 <ellsig>
```

Figure 2: List categories with examples.

In keeping with our philosophy of presenting just as much information as is useful, our general strategy is to output information in small, manageable chunks. Therefore, in simple/simple lists, we output three to four items at a time. In simple/complex lists, such as those which occur in the USI ApartmentLine output, the long, summary information for each item is split into chunks. We use the `MORE` keyword with lists to access additional information of the same type – i.e. the next chunk at the same level of information. We use an auditory icon (specifically a short triple-beep, represented by <ellsig> in our text examples (4,10)) at the end of each non-final chunk to signal that users can say `MORE` at those points. In simple/complex lists the user can jump to the next item in the list (as opposed to the next chunk) by using the keyword `NEXT` (in simple/simple lists this keyword simply acts the same as `MORE`).

Splitting complex output into chunks not only helps avoid information overload, but also enables the `REPEAT` keyword to act on current, smaller segments of information that the user might be interested in hearing again, rather than having to repeat *all* the summary information from the very beginning. For instance, in the USI ApartmentLine, this can be particularly useful for the contact information segment, which comes at the very end of the summary information, and which users are likely to want to hear repeated.

3.2.6. Tutorial

An important component of our approach is that given an initial training investment of about five minutes, users should be well-enough acquainted with the keywords and structure of the system to be able to successfully use new USI applications. Currently our systems include two training components: a generic tutorial and application-specific introductions.

The generic tutorial is intended to be a more substantial introduction to the USI approach and is currently implemented as a series of web pages explaining the system, using examples from fictitious USI flight and hotel information systems. Currently users read and click through the web pages on their own, preferably with a project member nearby to answer questions (which helps us further refine the tutorial content) and to act as a sort of wizard-of-oz with which users can test their understanding of the system. Our plan is to make this generic tutorial even more interactive, first as a web implementation that can accept and correct typed user input, and then as a speech-based automatic tutorial that will do the same.

The application-specific introductions consist of approximately 90-second-long recordings of a hypothetical user's interaction with an application. The user in the examples generally speaks in a confiding, casual, "thinking" voice to explain a feature to the listener (e.g., "If I make a mistake, I can always say `scratch that` to cancel my utterance"), and then issues an example command in a more assertive, "command" voice. The listener also hears the system's response to each command. These introductions are played at the beginning of each interaction, following a brief welcome and experimental-system disclaimer, and functions primarily as a reminder of that particular application's functionalities and of the general USI paradigm. More advanced users can skip the introduction by barging in.

We have found that these application-specific introductions are not quite adequate for pure novice USI users. In user studies, we have found that new users introduced to the USI only via the application-specific introductions often need to listen to the introduction more than once in order to feel comfortable with the system, and key features of the USI (such as using the execution keyword `GO!`) are frequently missed. We suspect that the inadequacy of this method for first introductions is largely related to the lack of user interaction in it, and we hope that the further development of the interactive general tutorial will solve or at least ameliorate this problem.

4. Observations

From our observations of user interactions with the USI MovieLine, we have discovered several positive features of our design. Nearly all users were able to grasp the basic concepts of the USI such as phrase structure and the use of major keywords like `NOW WHAT` and `SCRATCH THAT` almost immediately. Also, our earlier designs did not include an explicit confirmation strategy, which created problems for error correction since users were never quite sure of the cause or location of the error. Our current confirmation strategy is much more user-friendly; most users who have tried the current implementation have commented on the resulting transparency of the system as one of its strongest assets.

Following our latest refinements and the introduction of the second application (ApartmentLine), we conducted the following pilot study. We asked six new users to try the USI ApartmentLine after having gone through the generic web tutorial (and listening to the application-specific introductions). Two of these users also used the MovieLine application between the generic tutorial and the ApartmentLine interaction; another used the MovieLine after finishing the ApartmentLine interaction. Users were first asked to simply interact with the system to find out some information that they thought might be useful. Then they were asked to accomplish specific tasks, such as "find out how many 2-bedroom apartments are available in Shadyside." Our goal in these initial observations was not to make quantitative judgments about task completion rates and times, but rather to see if the overall interaction patterns in the ApartmentLine applications matched our earlier observations with the MovieLine.

The results were positive. Users issued correctly formed phrases nearly all the time, even generalizing correctly to slot types they had not seen before. Interestingly, users also over-generalized, by constructing USI-correct phrases for slots that were not actually queryable, such as "parking." `NOW WHAT` and `SCRATCH THAT` were again used readily, and users did not forget to say `GO!` as frequently as in the MovieLine – perhaps because of the non-specific nature of the ApartmentLine queries, or because of the increased emphasis on `GO!` in the general tutorial.

5. Conclusions & Further Work

Our work to date on the USI project seems to indicate that the USI approach is indeed viable and holds promise as an interface style for many different applications. In addition to general design refinement, we have plans for investigating a number of different extensions to and aspects of the USI. Among these are:

- Implementing USI for non-database applications, specifically device control and "interactive guidance systems," by which we mean systems that lead the user through some task external to the USI application, such as changing a tire, driving to a given address, or cooking.
- Designing an effective, speech-based interactive tutorial to introduce users to USI systems, which can be used for non-application-specific user training.
- Creating a toolkit allowing application developers to design USI interfaces for their projects with a minimum of effort and without much knowledge of speech interface issues and speech recognition technology.

6. References

[1] Helander, M. "Systems Design for Automatic Speech Recognition," *Handbook of Human-Computer Interaction*. Elsevier Science BV, Amsterdam, 1988. 301-319.

[2] Grice, H. "Logic and Conversation," *Syntax and Semantics, Vol. 3: Speech Acts*. Academic Press, New York, 1975. 41-58.

[3] Clark, H. "Managing Problems in Speaking," *Speech Communication 15*, 3-4 (December 1994), 243-250.

[4] Rosenfeld, R., et al. "Towards a Universal Speech Interface," in *Proceedings of ICSLP '00*, October 2000.

[5] Marx, M. and Schmandt, C. "MailCall: Message Presentation and Navigation in a Nonvisual Environment," in *Proceedings of CHI 96*.

[6] Stifelman, L., Arons, B., Schmandt, C. and Hulteen, E. "VoiceNotes: A Speech Interface for a Hand-Held Voice Notetaker," in *Proc. INTERCHI '93*, Apr. 1993. 179-186.