TOWARDS A UNIVERSAL SPEECH INTERFACE

Roni Rosenfeld, Xiaojin Zhu, Arthur Toth, Stefanie Shriver, Kevin Lenzo, Alan W Black

School of Computer Science
Carnegie Mellon University
{roni, zhuxj, atoth, sshriver, lenzo, awb}@cs.cmu.edu

ABSTRACT

We discuss our ongoing attempt to design and evaluate universal human-machine speech-based interfaces. We describe one such initial design suitable for database retrieval applications, and discuss its implementation in a movie information application prototype. Initial user studies provided encouraging results regarding the usability of the design, as well as suggest some questions for further investigation.

1. INTRODUCTION

Speech recognition technology has made spoken interaction with machines feasible. However, no suitable universal interaction paradigm has yet been proposed for humans to communicate effectively, efficiently and effortlessly by voice with machines.

On one hand, natural language applications have been demonstrated in narrow domains, but building such systems is data-, labor- and expertise-intensive. Perhaps more importantly, unconstrained natural language severely strains recognition technology, and fails to delineate the functional limitations of the machine. On the other hand, directed dialog systems using fixed menus are commercially viable for some applications, but are inefficient, rigid, and impose high cognitive demands.

The optimal paradigm, or style, for human-machine speech communication arguably lies somewhere in between these two extremes: more regular than natural language, yet more flexible than hierarchical menus. The Universal Speech Interface (USI) project at CMU is designing and evaluating such styles. In essence, we are trying to do for speech what Graffiti™ has done for mobile text entry. A crucial aspect of the design is uniformity across applications. In that regard, we are trying to do for speech what the Xerox/Macintosh revolution has done for GUIs. As in the latter case, uniformity also means that toolkits can be used by application developers to facilitate compliance and dramatically reduce development time. Another crucial aspect of the design is learnability: like Graffiti™, our style must be learned in no more than a few minutes, then be immediately useful and transferable to all other applications. For a more detailed discussion of the motivation behind the USI approach, see [1]. For current information on the USI project at Carnegie Mellon, see http://www.speech.cs.cmu.edu/usi .

This paper discusses one such design, one that is most suitable for information retrieval from a database. We chose to demonstrate the design in a prototype movie information application. Our database is the one used by an existing natural language interface, the Carnegie Mellon MovieLine [2]. It contains information about movies and movie theaters in the Pittsburgh area and is updated weekly. We chose this as our first application for three reasons: a database was readily available; interfacing with an information server allowed us to focus mainly on the design of the interface while still creating a fully functional system; and the existing MovieLine interface facilitates head-to-head comparisons of natural language and USI interactions. We plan to implement and test USI systems for a variety of applications types, and eventually distribute a development toolkit to allow others to create USI interfaces for their own systems.

2. A SAMPLE INTERACTION

The following sample interaction with the USI movie line will form the basis for a discussion of our interface design.

User (U) wants to know where Casablanca is playing:

1 *U:* Movie is Casablanca, theaters are what?, **go!**
2 *Movieline (M):* Two matches: Showcase East, Waterworks Cinema.

User would like to find a comedy in Squirrel Hill:

3 *U:* Neighborhood is Squirrel Hill, **now_what?**
4 *M:* Title is <dadada>, theater is <dadada>, genre is <dadada>, <ellsig>
5 *U:* Genre is **now_what?**
6 *M:* Comedy, drama, foreign, <ellsig>
7 *U:* Comedy, titles are what?, **go!**

User inquires about the movie October Sky:

8 *U:* Movie is October, **go!**
9 *M:* <oksig> movie is <confsig> October
10 *U:* October Sky, theaters are what?, **go!**

User wants to know what time Casablanca is showing at Waterworks Cinema:

11 *U:* Title is Casablanca, theater is Watergate, **scratch_that!**
12 *M:* Scratched.
13 *U:* Movie is Casablanca, theater is Waterworks, **ok?**
14 *M:* Okay.
15 *U:* Times are what?, **go!**
16 *M:* Six matches: 1:15, 2:45, 4:00 <ellsig>
17 *U:* **More**
18 *M:* 5:45, 7:50, 10:00

3. INTERFACE DESIGN

3.1. Syntax

The USI uses as its basic utterance a series of *phrases* followed by a *terminator* keyword. Each phrase specifies a slot name and its value. Thus, in line 1, "movie is Casablanca" is a phrase specifying "movie" as the slot and "Casablanca" as its value; "go!" is the terminator.

The use of slot+value phrases simplifies the work of the parser and conforms to natural speaking patterns. Phrases are order independent and synonyms are permitted when appropriate in the slots and values (e.g. "movie" and "title" in lines 1 & 3). In our current implementation we restrict phrase syntax to "*slotname/s* is/are <value>", but in general the grammar for each slot type may be quite elaborate. We use the Phoenix parser [3] developed at Carnegie Mellon, to define and parse the utterances. Some grammar variations we have considered include allowing prepositions in slots (e.g. "at [theater] Showcase East, on [day] Tuesday"). For database applications, the USI uses "what?" to indicate the slots to be queried, as in line 1.

The burden of processing is also eased by the use of terminators: the ASR engine simply watches for one of the terminators, and upon finding one sends the preceding string as a completed structure to the parser. From the user's point of view, a terminator allows them to take as much time as needed to formulate a query. "Go!" was implemented as our basic terminator and signals that the user is ready to have their query executed. We have also incorporated a fallback timeout feature.

3.2. Vocabulary

The vocabulary of a USI-enabled application consists of two parts: a set of universal USI keywords, and an application-specific lexicon. The keywords are used to perform basic functions in all USI applications and are discussed individually in this paper. The lexicons are specified by developers of individual applications.

For the USI to be truly universal, it must use a small set of words that non-technical users will feel comfortable with. Therefore, we have attempted to restrict our list of keywords to simple, everyday words and phrases such as "ok" and "scratch that" rather than more technical terms like "enter" and "execute".

It is also essential to keep the number of USI keywords to a minimum, to reduce the perplexity of the language and the burden of learning it. We have tried to limit the number of essential keywords in the USI to 7-9.

The size of the application-specific lexicon is naturally determined by the functionality and complexity of each application, and will generally be quite a bit larger than the USI keyword set (the movie line lexicon includes 791 words; however, 58% of these are movie names). To add flexibility, synonyms are allowed where appropriate, as noted above. Although this increases the size of the vocabulary, it actually reduces the burden on the user's memory.

3.3. Help/Orientation

An essential component of any interface is a simple, effortless help function. This is particularly important when the system has no visual component, as the user in this case must not only be able to remember how to access the help but must also be able to retain and use from short term memory the information that the help function provides to them.

We consider six types of help requests:

1. what the machine is/does;
2. local help while issuing a query;
3. how to use a keyword or command;
4. step-by-step help in issuing a query (a "wizard");
5. help finding the appropriate keyword or command for performing a task;
6. more information about something in the application.

We address the first situation by playing a short introduction at the beginning of each USI interaction. This introduction also includes a short sample of dialog appropriate to the application which is intended to instruct new users (or remind more experienced users) how to perform basic actions in the USI system.

The main mechanism for getting help in the USI is the keyword "now_what?", as shown in lines 3-7 of the sample dialogue. When a user says "now_what?", the system responds with a list of all the things that could come next at that point in the user's query; the specific form and content of the list is determined by the context in which it is said.

In line 3, the user has asked "now_what?" at a phrase boundary, so the response is a list of all the phrases that could be used to continue the query. In line 5, the user has asked "now_what?" inside a phrase, at a point where they are expected to specify a value, so the USI responds with a list of possible values.

Another principle of the USI design is that machine prompts should be phrased so as to entrain the user. Therefore the machine's response in line 4 is structured in phrases just like the user is expected to use, rather than returning something like "ask about a title, theater, or genre." "Lexical entrainment" [4] such as this helps promote more efficient interaction with the machine with no added computational complexity (in fact it is probably often simpler than generating an appropriate, grammatical rephrasing like "at what time?"). As shown in line 6, the response to a mid-phrase "now_what?" also uses the exact words that the user is expected to say. In some cases however, the class of possible responses is too large, and a description of the response is given instead:

U: Location is **now_what?**
M: State the name of a neighborhood or city.

The <dadada> notation in line 4 indicates a fill-in-the-blank marker, and is currently implemented as a fast, low-stress "da-da-da." The <ellsig> notation in lines 4, 6, and 17 is intended to indicate that the list continues beyond this. Since recitation of a very long list of items does not generally allow the user adequate time to process and retain each item, and because we want to encourage turns to be as brief as possible, USI lists are output in groups of three or four. The USI movie line currently implements the <ellsig> lexically, as the phrase “and more.” Experiments have indicated that using audio signals or natural prosody to convey non-finality of lists is also effective, and we continue to explore this and other non-lexical alternatives [5].

Since the user can ask it at any point and get help specific to that context, “now_what?” addresses the second help situation. It also covers the third situation, since when it returns information it is also telling the user exactly how to use it.

For the fourth type of help, a user could move through a query one step at a time by repeatedly asking “now_what?” As a shortcut for this process, the user could say “lead_me” and be guided through the query in essentially the same way. With “lead_me,” control of the dialog rests with the system, so that a query segment is elicited from the user, and then the next prompt is given by the machine. The user can of course resume control of the dialog at any time. (This keyword has not been implemented yet.)

At the very beginning of an interaction, saying “now_what?” will result in a list of all possible phrases; this could help the user in the fifth help situation who knows what they want to do but is not sure how to do it. A more efficient solution is a simple keyword search. If a user has something in mind that they want to do, they can simply say "how_do_I <do something>." Each application will include an index of words that might be associated with each of its main functions. The “how_do_I” function will search through this index to find items corresponding to the words in the user's utterance string and will report the matches back to the user as a list in a manner similar to the response to “now_what?”

The sixth type of help is handled with the keyword “explain.” A user can say “explain <USI keyword>” or “explain <application term>,” and the machine will respond with a brief, USI- or developer-specified description of what that item does or represents.

3.4. Errors

Our initial design includes mechanisms for alerting users to errors and also for helping users avoid errors in the first place. An example of the first case is shown in lines 8-10 of the sample dialogue. The user has intended to ask about the movie *October Sky* but instead has only said “October.” However, as far as the system knows, there is no movie called *October* and therefore it cannot occur as a value for the movie slot, so it signals an error.

In general, a USI error can result from a failed parse (which could be due to a recognition error or to an ill-formed query, as above), invalid data (e.g. "February thirty-first"), or possibly as a result of a low confidence score from the ASR component.

We handle errors by conveying to the user which part of the query was understood, and in which part the error occurred. In line 9 of the sample dialog, the <oksig> indicates that the system understood “movie is,” and the <confsig> indicates that the system did not understand “October.” The part of the query that was understood correctly is retained by the system, and the user can correct and continue their query from the point of the error. Currently, our design is deliberately left-to-right, so the processing stops as soon as an error is encountered.

The current version of the USI movie line implements the <oksig> and the <confsig> lexically, so that the actual error message for the above situation would be “I understood ‘movie is,’ but I didn’t understand ‘October.’” Experiments with non-lexical signals have indicated that simply repeating “movie is October?”, where October is spoken with a rising, stressed, “confused” prosody is also a reasonable error alert for users, although it is not simple to implement [5]. We plan to conduct further user tests of noises and other non-lexical signals for their effectiveness as error alerts.

Another error strategy is shown in line 11 of the sample interaction. Here, the user recognizes that they have misspoken a word and uses the terminator “scratch_that” to clear the query and start over. In addition, the keyword “rather” allows the user to make a correction without starting over:

U: Title is Casablanca, theater is Watergate, rather, theater is Waterworks, **go!**

The USI also includes two other of confirmation terminators. “Ok?”, as shown in line 13, directs the system to parse the current utterance and respond with an “okay” if there are no parsing or data errors. “Restate” does the same thing, except that the machine responds with a listing of all the slot+value pairs parsed since the user's last “restate,” so that the user can be sure the slots have been filled correctly.

4. SYSTEM ARCHITECTURE

Our implementation is modular, with the various components residing on multiple machines spanning two platforms (Linux and Windows NT). The dialog manager consists of an application-independent *USI engine* and an application-specific *domain manager*. The two interact via a USI API. The USI engine calls on the Phoenix parser, and the domain manager interacts with a commercial database package. These components together constitute a standalone text-based version of the system, which can be developed and tested independently of the ASR, synthesis, and telephony control.

Recognition is performed by CMU’s Sphinx-II engine [6], using acoustic models developed for the Communicator testbed [7]. For speech synthesis, we recorded a voice for unit-selection based limited-domain synthesis using the Festival system [8]. All the components are integrated using a VB framework borrowed from the CMU MovieLine, and a socket interface where needed.

Finally, new movies must be added to the application at least weekly. For each such movie, one must update the database, the grammar, the language model, the pronunciation lexicon and the synthesis database. To reduce costs and errors in development and maintenance, we are automating this process.

5. PRELIMINARY USER STUDIES

We conducted preliminary user studies to gauge how well new users understood the basic concepts of the interface. 15 subjects were asked to listen to a 100-second recorded introduction and sample dialog for the movie line application. They were then asked to call the system and use it to get answers to five questions such as "Find the first showing of *Chicken Run* after 2:00 at the Galleria." In addition, before listening to the introductory recording, half the subjects were given approximately two minutes of personal instruction covering USI basics such as phrases, terminators, and the format of error messages. All users were asked to return three days later, listen to the introductory recording again, and use the USI movie line to answer a different set of five questions.

In general, users assimilated the interaction style quite well. Ten subjects issued a correctly formed USI query on the first or second try; an additional three users issued correct queries within five to seven tries. Only two users had critical problems formulating a query; after some additional help from the experimenter they were able to answer most of the questions (one with the aid of the USI basics "cheat sheet" which was used in the personal instruction sessions). All participants used "scratch_that" and "more" at least once. We found that only a small number of participants used "now_what?"; the rest guessed the necessary slot names, usually successfully. This is likely to be the case with intuitive, self-suggesting slot names, but "now_what?" may still be useful in other cases.

Our user tests also provided support for the need for synonyms in the USI vocabulary. On the first day of testing, 11 out of 15 subjects used "movie" instead of "title" in their queries – even though "title" was the phrase presented in all the introductory material.

Two of the issues we had anticipated as possible problems did indeed surface in the user tests: error correction and "go!" We found that many users had difficulty correcting errors at the appropriate location. Currently, our system expects the user to correct the problem at the point of the error and move on, but our testing showed that many users simply started the entire query over again, or at least restarted it at a phrase boundary. This inevitably led to further errors, since the slot+value structure of the query was disturbed.

While some users overused "go!" by adding it to other terminators like "now_what" and "more," almost all users failed at least once to say "go!" to send their query to the system. This is not unlike the situation with novice computer users, who often forget to hit "Enter." Although we believe that, as in the latter case, this is a habit that is easily acquired, we plan to experiment with shorter and/or user-adjustable timeouts and possibly eliminate "go!" from the set of terminators altogether.

Another finding that deserves further study is that some users tried to answer more complex questions with multiple "what?" phrases in a single query. We would like to allow this functionality, but we have yet to determine the best way to present the resulting matrix of information.

6. FUTURE WORK

We plan to conduct more user studies to inform our future designs. In addition to addressing the issues noted in section 5, we hope to investigate when and how confirmation should be used, how learnable new USI applications are for those who have used the USI movie line, and how to introduce users to more advanced USI features. We also plan to run side-by-side user studies comparing the USI movie line interface with the CMU Communicator's natural language interface.

7. ACKNOWLEDGEMENTS

We are grateful to Rita Singh and Ricky Houghton for help with acoustic modeling issues, and to Alex Rudnicky for much appreciated advice. This research was sponsored in part by the Space and Naval Warfare Systems Center, San Diego, under Grant No. N66001-99-1-8905. The content of the information in this publication does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

8. REFERENCES

[1] Ronald Rosenfeld, Dan Olsen and Alexander Rudnicky, "A Universal Human-Machine Speech Interface," *Technical Report CMU-CS-00-114*, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, March 2000.

[2] Constantinides, P., Hansma, S., Tchou, C. and Rudnicky, A., "A schema-based approach to dialog control", ICSLP 1998.

[3] Ward, W. "The CMU Air Travel Information Service: Understanding Spontaneous Speech," *Proceedings of the DARPA Speech and Language Workshop*. 1990.

[4] Boyce, S., Karis, D., Mané, A., and Yankelovich, N. "User Interface Design Challenges," *SIGCHI Bulletin* Vol. 30 (2) p. 30-34. 1998.

[5] Shriver, S., Black, A., and Rosenfeld, R. "Audio Signals in Speech Interfaces," *ICSLP 2000*.

[6] Huang, X.D., Alleva, F., Hon, H.W., Hwang, M.Y., Lee, K.F. and Rosenfeld, R. "The SPHINX-II Speech Recognition System: An Overview," *Computer, Speech and Language* Vol. 2 p. 137-148. 1993.

[7] Rudnicky, A., Thayer, E., Constantinides, P., Tchou, C., Shern, R., Lenzo, K., Xu W., Oh, A., "Creating natural dialogs in the Carnegie Mellon Communicator system," Proc. Eurospeech, 1999, 4, 1531-1534 .

[8] Black, A., Taylor, P. and Caley, R. The Festival Speech Synthesis System. http://www.cstr.ed.ac.uk/projects/festival.html. 1998.