Spoken language interfaces: The OM system

Jean-Michel Lunati and Alexander I. Rudnicky
School of Computer Science, Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

The intrinsic properties of speech communication (e.g., the presence of malformed utterances) and the characteristics of current recognition technology (inaccurate recognition) pose special problems for the design of a speech interface. We are interested in understanding these problems and in identifying an interface structure that allows speech to be a useful form of computer input. Ultimately, our goal is to understand how to turn speech into a conventional input modality, well integrated into a multimodal interface that includes keyboard and mouse. To fully exploit the advantages of spoken communication, a spoken language system must afford the user the following forms of flexibility: natural production, natural language, and a natural flow of interaction. The Carnegie Mellon Spoken Language Shell (CM-SLS) attempts to provide such flexibility through the use of speaker-independent continuous-speech recognition, natural language processing, as well as rudimentary "conversational skill" heuristics.

A good interface design embodies a clear functional decomposition which in turn simplifies system implementation and allows for independent development of different components. The CM-SLS design has proven useful, allowing us to implement several different recognition systems while maintaining modularity. The design decomposes a recognition system into functionally independent units, each corresponding to a necessary function in the speech interface. Note that we have not created novel elements. Each of these functions are implicit in all existing recognition systems but typically are not explicitly identified or recognized as separable components of the interface. The present decomposition provides an explicit separation of these functions, simplifying the exploration of issues that correspond to each component. Figure 1 shows the functional components of the spoken language interface.

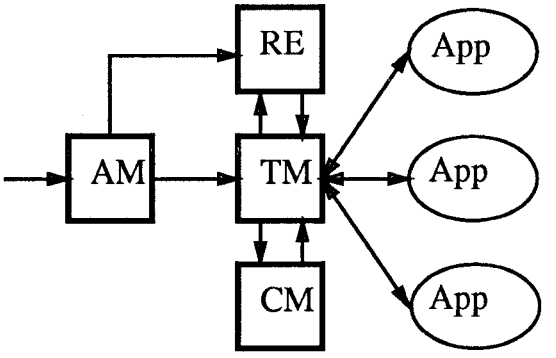

Figure 1: *Components of the spoken language interface.*

ATTENTION MANAGER (AM)

The signal processing component produces a continuous stream of coded speech. The Attention Manager segments utterance-sized units from this stream and routes these utterances to the recognition engine. The AM implements a range of strategies. At one extreme, the user explicitly controls the signal acquisition process by indicating to the system both the start and the end of an utterance (**Push to Talk** and **Push and Hold** modes). At a more complex level of function, the system determines one or both of these points through automatic end-point detection (**Push to Start** and **Continuous Listening** modes). Ideally, the AM should be capable of determining not only the bounds of a true utterance but also know to reject unintentional utterances (and noise) and be able to determine whether the user is actually ad-

We would like thank the following individuals for their contributions to the Office Manager project, including Kathryn Baker, Eric Thayer, Robert Weide, Paul Arceneaux, and Alexander Franz.

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-C0163. 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.

dressing the computer (as opposed to another agent in the environment).

RECOGNITION ENGINE (RE)

The Recognition Engine decodes the input utterance into an ASCII string. In its present implementation, the RE functions as a dedicated server and allows multiple clients to share the same recognition facilities. Recognition imposes a high computational load and it is often impractical to have this process reside on a computer on which several applications (themselves potentially requiring substantial resources) are active. Ideally, the RE would be implemented as a specialized co-processor. Currently, we use a separate computer for this purpose. A critical attribute of a recognition engine is its ability to decode speech in real-time. Real-time response (or rather response that is within a 200-300 msec delay of the end of an utterance) maintains the rhythm of interaction.

TASK MANAGER (TM)

Speech recognition systems are often built as monolithic processes. While this approach is adequate for a computer that runs a single speech application, it is inefficient for a computer that is meant to support a variety of speech-enabled applications. In the latter case it becomes more efficient to centralize speech resources and to allocate them dynamically to individual applications. The purpose of the Task Manager is to supervise, in the context of multiple voice-addressable applications, the assignment of the speech channel to the proper application. In our implementation, the actual services performed by the TM also include the maintenance of context information and the communication of this information to the RE, where it governs the selection of recognition knowledge bases.

CONFIRMATION MANAGER (CM)

The errorful nature of speech recognition compels the introduction of a component not normally found in other interface technologies, the Confirmation Manager. The CM allows the user to intercept and edit a recognition before it is acted upon by the application. In terms of human communication, the CM performs the error repair necessitated by breakdowns in the communication channel (such as might be caused by a noisy telephone line or a loud interruption). It does not concern itself with the consequences of errors due to some misunderstanding on the part of the user (although it does offer an opportunity for the immediate undoing of a just-spoken utterance).

THE OFFICE MANAGER SYSTEM

The Spoken Language Shell has been used to implement the Office Manager, a system that provides voice access to several common office applications. The Office Manager (OM) domain is interesting for the following reasons: It provides a range of interaction requirements, from tight-loop (e.g., calculation) to open-ended (e.g., database retrieval); it supports meaningful problem-solving activity (e.g., scheduling, information search); and since the applications are directly usable in real-life settings, the activity of actual users can be studied over extended periods.

OM includes a personal database, an appointment calendar, a mailer interface and a calculator. OM itself understands a 36 word vocabulary and provides control functions, such as starting up and switching between applications, help invocation, etc. The current implementation of the system includes a database of names and addresses of 172 conference participants. It is used by the Voice Mail and Personal Information Database components of OM. Database customizing tools are provided, allowing the non-expert user to create or modify entries in existing databases and to create new databases. Changes to a database result in automatic updates to the recognition knowledge base, allowing users who lack a speech background to easily extend the system.

All components of the system, with the exception of the Recognition Engine are implemented on a NeXT computer, using Objective-C and the Nextstep interface. The RE server is implemented in C as a parallel program on a 4-processor Apollo 10040. Other platforms (DEC DS5000 and IBM R6000) are currently in use as RE's. Word recognition accuracy for a typical database query, of perplexity about 105, is 92%.

CONCLUSION

Our future work includes the development of techniques for structuring recognition and parsing knowledge bases along "object" lines, to permit individual applications to inherit language characteristics from their environment (the OM), and to encourage the modularization and reusability of language components. The goal is to simplify the process of creating languages for particular applications by providing the developer not only with standard interface components but also with standard language components.

Meaningful study of spoken language interaction requires a system that will be used on a daily basis and whose utility will persist past the initial stages of play and exploration. We believe that the Office Manager is such a system. Systems that do not have this persistence of utility will ultimately have little to tell us about spoken communication with computers.