BUILDING 10,000 SPOKEN DIALOGUE SYSTEMS

Stephen Sutton, David G. Novick, Ronald Cole, Pieter Vermeulen, Jacques de Villiers, Johan Schalkwyk and Mark Fanty

Center for Spoken Language Understanding, Oregon Graduate Institute
P.O. Box 91000, Portland, OR 97291, USA

ABSTRACT

Spoken dialogue systems are not yet ubiquitous. But with an easy-enough development tool, at a low-enough cost, and on portable-enough software, advances in spoken-dialogue technology could soon enable the rapid development of 10,000 or more spoken dialogue systems for a wide variety of applications. To achieve this goal, we propose a toolkit approach for research and development of spoken dialogue systems. This paper presents the CSLU toolkit, which integrates spoken-dialogue technology with an easy-to-use interface. The toolkit supports rapid prototyping, iterative design, empirical evaluation, training of specialized speech recognizers and tools for conducting research to improve the underlying technology. We describe the toolkit with an emphasis on graphical creation of spoken dialogue systems; the transition of the toolkit into the user community; and research directed toward improvements in the toolkit.

1. INTRODUCTION

In the past decade, research in spoken language technology has enjoyed strong support in the United States. However, spoken language systems are not yet ubiquitous, and there are many problems that need to be solved before they can become so. These problems include high expertise requirements, a lengthy and expensive development process, and the lack of portability of spoken language technology. To reach a future with ubiquitous spoken dialogue systems, then, the field needs to make it possible for non-expert developers to build portable spoken-language applications and interfaces rapidly. This approach should enable developers, authors and even end-users to create and adapt speech technology to the tens of thousands of specialized domains in which they have their own expertise.

To achieve this vision, we propose to address the barriers to ubiquity through a toolkit approach. This paper presents the CSLU toolkit, which supports rapid-prototyping, iterative design, empirical evaluation of spoken language systems, training of specialized speech recognizers, research into spoken language technology, and provides a modular and flexible environment which allows the sharing of resources (e.g. telephony cards) over a computer network.

2. BARRIERS TO UBIQUITY

While many aspects of spoken-language technology—such as recognition accuracy, out-of-vocabulary rejection and mixed-initiative dialogue—continue to pose serious difficulties, a number of spoken-language systems are already in successful use. These systems are characterized by limited vocabularies, simple grammars, and well-defined tasks. But the number of such systems remains small relative to their promise; we attribute this to three principal barriers:

Lack of expertise. Development and research of spoken language systems currently requires technical expertise across several subject areas. Because of these requirements, development and research is currently limited to a few specialized laboratories.

High development costs. The development process is lengthy and expensive, often requiring months or even years to produce a spoken language application. Data collection for training recognizers and for building language and dialogue models is costly and often must be done via "Wizard-of-Oz" simulation, with humans attempting to mimic the performance of a spoken language system.

Lack of portability. Current spoken language systems technology is not very portable [1, 4]. That is, the technology cannot support adequately the development of new applications with acceptable performance without significant engineering for each new task.

The combination of these problems severely hinders application development and limits the role that spoken-dialogue technology can play in key areas such as research and education.

3. THE CSLU TOOLKIT

The CSLU toolkit is made up from number of layers, as shown in Figure 1 [6]. At the highest layer, a direct manipulation interface (CSLUrp) enables authors to design graphically a spoken language system. This graphical specification is translated into Tcl Scripts which are then executed inside a programming shell (CSLUsh). This shell is made up from a collection of core libraries, written mostly in C. The libraries can also be used without the shell for developing stand-alone C applications and third-party applications.

The CSLU toolkit supports the complete life-cycle of a spoken-language system. It enables research by providing the essential tools and infrastructure for advancing speech technology, as well as other

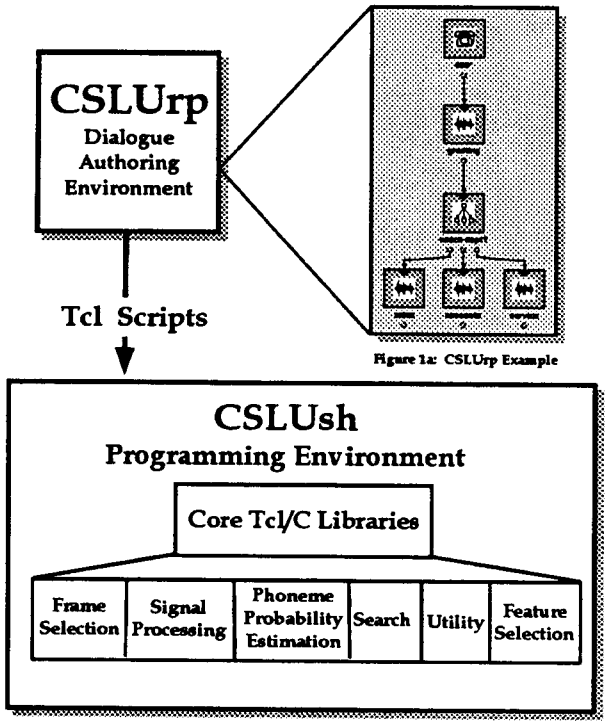

Figure 1: Overview of CSLU toolkit

aspects of human-computer interaction such as dialogue design. It allows quick and easy development of spoken-language system prototypes. It supports applications development through activities such as collecting speech data, training of recognizers, and includes integrated tools for browsing and labeling speech. Finally it serves as a valuable educational tool offering the opportunity for "hands-on" learning.

We now describe the main components of the toolkit—CSLUsh and CSLUrp—in more detail.

3.1. The CSLU Shell

The base component of the toolkit is the CSLU Shell (CSLUsh, pronounced "slush"), which is based on the extensible scripting language Tcl. We have added a number of modular, integrated and dynamically loadable packages which add new commands to the language, designed to support research and development of spoken language systems. Basic functions include manipulating wave files, performing signal analysis (e.g. FFT, mel cepstrum), extracting features, training and utilizing artificial neural networks, and doing speech recognition for isolated words and continuous speech with finite-state grammars. It includes a general-purpose (vocabulary-independent) recognizer and a number of special-purpose recognizers for common vocabularies such as digits and alphabets.

Fundamental to CSLUsh is the use of objects, which are essentially C structures hidden to the CSLUsh developer. Access to the objects is provided by the new Tcl commands. Communication between modules is performed by passing object handles. This works across the network as well as across platforms. For example, it is quite easy in CSLUsh to send an object to a CSLUsh server running on another machine, perform some computation and receive a result object back. CSLUsh automatically takes care of repacking the underlying structure to take care of platform dependencies such as byte order.

To give the flavor of CSLUsh programming, here is the transcript of an interactive session in which the user reads a speech file, scales it so that the max sample is 15,000 and write it back to disk. The identifiers "wave:0" and "wave:1" are object handles [6].

```
% wave read NU-2234.zipcode.wav
wave:0
% wave info -max wave:0
{4158 5168 646.0}
% wave scale wave:0 [expr 15000.0/4158.0]
wave:1
% wave info -max wave:1
{14999 5168 646.0}
% wave write wave:1 new.wav
wave:1
```

In addition to technology created at CSLU, CSLUsh incorporates implementations of many common and widely-used algorithms essential for creating spoken language systems.

3.2. The CSLU Rapid Prototyper

CSLUsh is a good development and research environment, but building a telephone dialogue using CSLUsh still takes several steps that may be intimidating for non-experts. The CSLU Rapid Prototyper (CSLUrp, pronounced "slurp") is a graphically-based authoring environment built on top of CSLUsh and incorporates all the steps necessary for building and executing simple spoken-dialogue systems. The main strengths of CSLUrp include: (a) the speed with which application prototypes can be created; (b) an easy-to-use interface; (c) strong support for authors who lack specialized technical expertise in speech recognition; (d) the ability to create a wide range of real-world applications; and (e) suitability for a broad community of users.

CSLUrp includes a graphical palette of dialogue objects and a simple drag-and-drop interface. The dialogue objects serve as visual-programming building blocks. During the design phase, the author selects and arranges appropriate objects, linking them together to create a finite-state dialogue model. Then, during the run phase, CSLUrp provides a real-time animated view of the dialogue. The author can alternate between the design and run phases, enabling the incremental development and iterative refinement of spoken language systems. The set of objects in the palette covers a range of fundamental spoken language system functions including answering the telephone, speaking a prompt, recording speech input, recognizing speech input, and identifying DTMF tones.

The interface is designed to require minimal technical expertise on the author's part and to simplify the design and specification process. For example, specifying a speech recognizer is largely automated; all that is required of the author is to enter the recognition vocabulary by typing or saying—for speaker-dependent

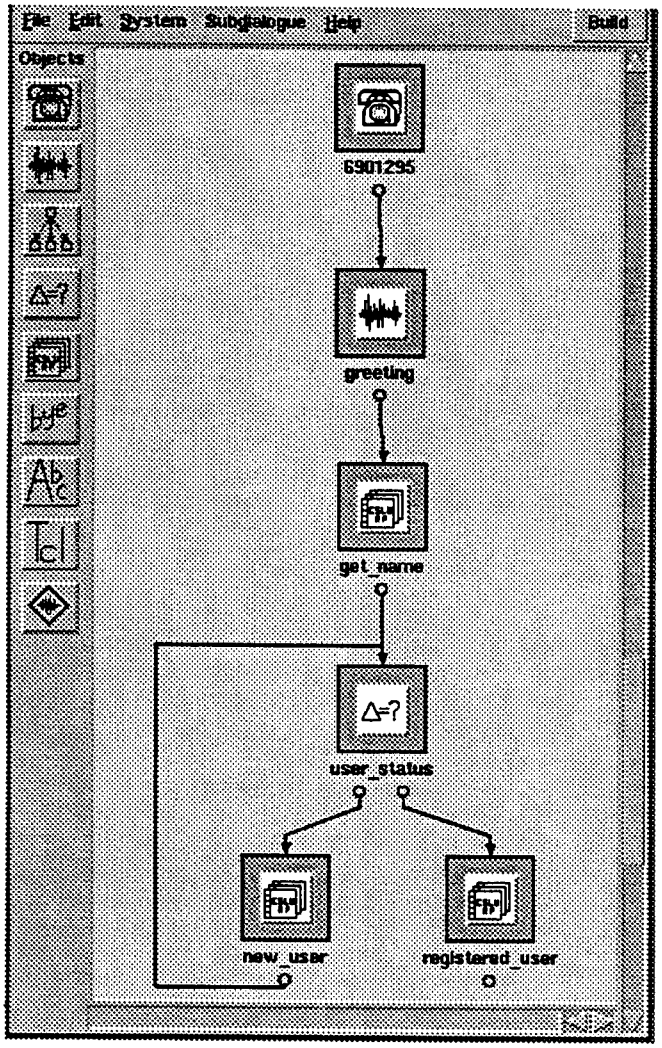

Figure 2: Prototype system being developed using CSLUrp.

recognition applications—words or phrases. The system prompt which is produced immediately prior to performing recognition can be either recorded, or entered as text and spoken by a speech synthesizer. Also, in the case of the latter, the text can optionally be generated automatically according to a set of pre-specified prompt styles [3]. Figure 2 shows a prototype spoken-dialogue system being developed using CSLUrp.

CSLUrp is fully integrated into the local environment. For instance, after designing a prototype the author simply clicks the "build" button followed by the "run" button, after which the system is ready to be called at the phone number displayed. Additionally, the author has the option of selecting from the set of available phone numbers which are provided by CSLUrp after it queries the telephone server.

CSLUrp inherits the power and flexibility of CSLUsh through its foreign code feature. This enables authors to design prototypes that access the CSLUsh level and execute arbitrary CSLUsh code from within CSLUrp at run time. Thus, as authors become more experienced and familiar with CSLUsh's capabilities, they can move beyond the scope of CSLUrp's initial set of functions and take advantage of the CSLUsh level to develop a wide range of interesting applications such as speech interfaces for existing text-based applications, speech front-ends to replace existing DTMF (touch-tone) interfaces, voice-response questionnaires and applications for spoken access to the world-wide web.

In summary, the CSLU toolkit is designed to address many of the problems associated with the lack of portability of spoken language system technology discussed above: the need for multidisciplinary expertise, substantial infrastructure requirements, and the effort required to develop systems for each new task. These problems are addressed (a) by providing a toolkit that incorporates most of the infrastructure needed to design, develop and investigate spoken language systems (most research at CSLU is now performed within the toolkit); (b) by making all of CSLU's speech corpora available to university researchers free of charge and providing tools to train new networks; and (c) by packaging other public-domain language resources within the toolkit.

3.3. Platform Portability

To insure portability and ease technology transition, the CSLU toolkit includes complete specifications for putting together turn-key systems. It also specifies the hardware and software requirements to make porting to new platforms as easy as possible. Our principal target platform is an Intel x86 (Pentium or better) processor running Solaris, a Dialogic telephone board, and DECtalk[1] text-to-speech software (which we have ported to x86 Solaris in cooperation with DEC, and which can be commercially licensed). The toolkit provides a generic interface for text-to-speech engines, including a common set of embedded commands, such as for changing pitch. The toolkit also provides a generic interface for speech I/O and a guide for writing servers for new devices. It supports the microphone-speaker of a Soundblaster in a PC running Solaris x86 and the standard microphone-speaker on a Sun Sparcstation. Dialogic and Linkon telephony boards are supported, and device drivers exist for several platforms. These devices can be shared between platforms in a client-server fashion. Multi-platform porting is facilitated by the toolkit's C/Tcl/Tk implementation.

4. TECHNOLOGY TRANSITION

One of the major objectives of the CSLU toolkit is to make spoken-dialogue technology less exclusive and more accessible by promoting technology transition. To encourage sharing of applications and technological enhancements, the toolkit is designed to make it easy to add new software, to be as platform independent as possible, to include no proprietary software and to incorporate or be compatible with the most useful and widely-used software development tools.

Our goal is to create a toolkit that will be embraced by a generation of students, developers and researchers simply because it is useful. We anticipate that such a toolkit approach will create and benefit

1.DECtalk is a trademark of Digital Equipment Corporation.

from a *multiplier effect*. By providing the capability to build spoken language systems, speech interfaces will be introduced in a growing number of applications, the limitations of the technology will become apparent, and more effort will be expended to improve the technology to enable better applications. Better applications will yield more value, more use and more interest in improving the technology, and the multiplier effect will move the development of spoken language interfaces out of the laboratory and into the public sector. The CSLU toolkit is designed to support the many activities needed to make this happen—system design and deployment, research and training.

We are providing the CSLU toolkit to the academic community free of charge. We are also aiding the process of sharing software and ideas by developing and maintaining a World-Wide Web site where developers, researchers and users can contribute and obtain useful software, such as recognizers, subdialogues and working systems. We plan to incorporate the most useful of these into new releases of the toolkit. We have also set up a mailing list for reporting problems and requesting help.

Finally, we are supporting technology development and transition by offering short courses in which we teach people to use the toolkit for designing and developing spoken language systems. An earlier version of the toolkit has already formed the basis of a short course described in [2]. We are working with other universities to transfer the toolkit to their sites and use it to develop laboratory courses in spoken language technology, to be incorporated into their undergraduate curricula.

5. FUTURE WORK

We are committed to continuing to develop the scope of the toolkit and its underlying technology. In addition to incorporating more advanced speech recognition capabilities, such as interpreting spontaneous speech using robust parsing techniques, we are investigating fundamental advances in dialogue technology. For instance, we would like to support more fluid and usable human-computer dialogues by moving beyond structured dialogues with simple finite-state grammars and fixed vocabularies. Research is needed to develop a more general basis for building flexible spoken language systems using high-level representations of knowledge, such as the goals and expectations of the user and the system. Improved dialogue representations should capture the dynamics of mixed-initiative interaction and provide authors with dialogue control structures suitable for tracking the dialogue focus, monitoring the mutuality and coherence of contributions, and providing a basis for performing automatic repair.

Specifying a spoken-dialogue system from scratch requires a great deal of expertise both about the domain and about the nature of human-computer interaction. This burden can be lessened through the reuse of knowledge. While it may be difficult to generalize interactions across domains, capturing pockets of knowledge for specific domains and tasks is a real possibility. Certain tasks and subtasks may recur in different applications, such as getting someone's name and address, obtaining an order, retrieving messages, and scheduling meetings. Research is needed to discover suitable knowledge for reuse and to explore the building of libraries of dialogues, subdialogues, metadialogues, and referent objects. CSLUrp's subdialogue objects are designed to support this effort, representing entire task-oriented subdialogues as a single icon.

6. CONCLUSION

As demand increases for ubiquitous spoken-dialogue applications, there is a critical need to make spoken-dialogue technology less exclusive, more affordable and more accessible. An important step towards satisfying this need is to be able to place development of spoken-dialogue systems in the hands of the real domain experts rather than limit it to technical specialists. The technology will have succeeded when a large number of spoken-dialogue systems are developed and used. To reach this goal, we need technology that will make it possible to build 10,000 spoken-dialogue systems. To address this need, we have developed and are distributing the CSLU toolkit.

7. ACKNOWLEDGEMENTS

This research was supported by U S WEST, the National Science Foundation, the Defense Advanced Research Projects Agency, the Office of Naval Research, and the member companies of the Center for Spoken Language Understanding. We thank Azdine Tadrist for his help in creating CSLUrp.

8. REFERENCES

1. Cole, R. A., Hirschman, L., et al. "The challenge of spoken language systems: Research directions for the nineties," *IEEE Transactions on Speech and Audio Processing*, 3(1), 1-21, 1995.

2. Colton, D., Cole, R., Novick, D., and Sutton, S. "A laboratory course for designing and testing spoken dialogue systems," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP 96)*, Atlanta, GA, 1129-1132, 1996.

3. Hansen, B., Novick, D., and Sutton, S. "Systematic design of spoken prompts," *Conference on Human Factors in Computing Systems (CHI 96)*, Vancouver, BC, 157-164, 1996.

4. Hirschman, L., et al. *Summary report from the workshop on toolkits for language interface portability: The toolip workshop*, Technical Report MP-95B0000173, MITRE, Bedford, MA, 1995.

5. Schalkwyk, J., Colton, D., Fanty, M. *The CSLU toolkit for automatic speech recognition*, Technical Report No. CSLU-011-96, Center for Spoken Language Understanding, Oregon Graduate Institute of Science & Technology, 1996.

6. Sutton, S., Vermeulen, P., de Villiers, J., Schalkwyk, J., Fanty, M., Novick, D. and Cole, R. *Technical specification of the CSLU toolkit*, Technical Report No. CSLU-013-96, Center for Spoken Language Understanding, Oregon Graduate Institute of Science & Technology, 1996.