A LABORATORY COURSE FOR DESIGNING AND TESTING
SPOKEN DIALOGUE SYSTEMS

Don Colton*      Ron Cole      David G. Novick      Stephen Sutton

Center for Spoken Language Understanding, Oregon Graduate Institute of Science & Technology
P.O. Box 91000, Portland, Oregon 97291-1000 USA
don.colton@cse.ogi.edu   http://www.cse.ogi.edu/CSLU/cslurp/course/

ABSTRACT
Spoken Dialogue Systems Laboratory at OGI gives students hands-on experience developing spoken dialogue systems (SDSs) in a rapid prototyping setting. The CSLU rapid prototyper (CSLURp) allows students to quickly build and operate SDSs for dialogues of arbitrary complexity. CSLURp consists of a graphical user interface that allows users to create SDSs with speech recognition, speech generation and arbitrary computation. When an application is designed, CSLURp configures the system from the appropriate libraries in CSLUsh, the CSLU shell. The course explores CSLURp in enough depth that students can craft and integrate their own low-level components (such as special purpose recognizers). In their final projects, students successfully built and demonstrated SDSs to do a variety of interesting tasks such as voice mail and directory assistance. Students found the course helpful and rated it highly.

1. INTRODUCTION
Spoken Dialogue Systems Laboratory [1] is a three-credit course at Oregon Graduate Institute (OGI). It comprises fifteen hours of lecture and thirty hours of directed laboratory assignments that teach students to build spoken dialogue systems using the Center for Spoken Language Understanding (CSLU) rapid prototyping tool, CSLURp (pronounced “slurp”).

Rapid Prototyping: The short-term goal of the course is that students will produce working spoken dialogue systems quickly. At the end of the course, each two-person team creates a working, realistic system and demonstrates it to class members and visitors. This happens within a single day and shows the speed with which systems using real dialogues can be prototyped and demonstrated.

Research Enablement: The long-term goal is that students will conduct their research by refining prototyped systems in any of their details. It is important to us that refinements and research take place in the context of working systems. Hence students are taught the fundamentals of speech recognition. Automatic speech recognition research in our connectionist setting involves learning many techniques, including waveform capture, signal representation, feature extraction, network training, pronunciation modeling, and search, among others. (Before this course, it was difficult to teach new students each of the tools and procedures needed to do productive research or create working systems. Students worked on narrow research problems and were unable to evaluate the results of their research in working systems. To address this problem, CSLU created this integrated research environment.)

Vision: This course and its accompanying software are intended to provide students at interested colleges and universities the opportunity to build, use, experiment with, and improve spoken dialogue systems. CSLU expects to offer this as a short course for teachers and to make the software and course notes freely available through the World Wide Web, by anonymous ftp, and in turn-key (ready to use) systems. The tools described below will also be offered free of charge to researchers and educators.

The software is described in section 2. The hardware is described in section 3. The course is described in section 4. The course culminates in team projects that consolidate important knowledge, described in section 5. Preliminary indications of success are given in section 6.

2. SOFTWARE
Overview: The architecture of the software used in the short course can be viewed as a number of layers, as shown in Figure 1. At the highest layer, CSLURp serves as a direct manipulation interface to enable authors to design a spoken dialogue system graphically. Once the system is designed, authors click on a menu option to start their application, which then conducts the specified dialogue with a user by telephone or microphone/speaker. This graphical specification is translated into Tcl scripts by CSLURp. These Tcl scripts are then executed inside a programming shell called CSLUsh.

CSLUsh: [2] The CSLU shell provides the operating environment for CSLURp. CSLUsh (pronounced “shush”) is a specially designed superset of the Tcl language with extensions for automatic speech processing (e.g., speech recognition and speaker verification), system building, and research. It has a collection of core libraries written mostly in C. It provides a modular, integrated command set that incorporates basic functions, such as manipulating wave files, extracting features, training and utilizing artificial neural networks, and searching for utterances subject to word model and grammar constraints. It provides distributed processing across a network, and special-purpose hardware support. It includes a general-purpose (vocabulary-independent) recognizer and special-purpose recognizers for common vocabularies such as digits and alphabets.

Tel: [3] (tool command language, pronounced “tickle”)
and Tk (its graphical interface) by John Ousterhout are freely available for many computational platforms. The Usenet newsgroup comp.lang.tcl provides much support. CSLUsh and Tcl are written in C.

**CSLUrp**: CSLUrp presents to the author a canvas (Figure 2) on which an SDS is constructed. It includes a palette of graphical dialogue objects and a simple drag-and-drop interface. The dialogue objects serve as visual program building blocks, and support a range of fundamental SDS tasks including answering the telephone (or listening to a microphone), speaking a pre-recorded prompt, generating and speaking a text-to-speech prompt, recording speech input, recognizing speech input, identifying DTMF touch tones, transferring or ending the call, creating subdialogues, and running arbitrary CSLUsh/Tcl commands.

During the design phase the author selects and arranges appropriate objects, linking them together to create a finite-state dialogue model. Mouse actions are used to position icons and connect them by arcs, which show the paths the dialogue can take. Keyboard entry is used to specify prompts and recognition vocabulary. CSLUrp configures a speaker-independent speech recognizer with the active vocabulary at each recognition point. The pronunciation model for each word is retrieved from a dictionary or generated using a text-to-speech algorithm.

At run time the dialogue progresses the system highlights icons to show the progress of the call. Branching is based on word recognition results (or DTMF input or arbitrary computation). Each outbound arc has a specified vocabulary. Control passes through the arc with the word or phrase that best matches the caller utterance. Wizard mode lets each recognition be reviewed and corrected by a human monitor. Dialogue repair, when enabled, identifies low-confidence recognitions and invokes a customizable “sorry, please repeat” subdialogue.

**Lyre** is part of the OGI Speech Tools [4] available by anonymous ftp. OGI students and staff use Lyre to add time-aligned phonetic labels to speech.

### 3. Hardware

CSLUrp runs on DEC Alpha or Alpha PC under OSF, or Sun4 workstation under Solaris. Speech input is by microphone or by telephone connection to an audio server using Linkon boards and running on an Intel PC under Solaris 2.4. T-1 digital lines are also supported. Workstations and servers are connected by a tcp/ip (Internet) local area network. Lyre uses local audio delivered through headphones.

### 4. The Course

The course was given in Summer quarter 1995 at OGI. To accommodate students from other parts of the country, the course was taught in an intensive, five-day format, rather than the more traditional ten-week format. Twenty students attended; enrollment was limited due to availability of hardware.

#### 4.1. Student Preparation

Students who enrolled in the course at OGI were first-year graduate students in computer science or electrical engineering: about one third were MS students, one third were PhD students, and one third were out-of-town persons from industry taking this single course.

Students are required to learn and use a small amount of Tcl during the course. Aptitude is demonstrated by prior programming experience in some computer language. Students are expected to work successfully in a UNIX operating environment.

#### 4.2. Course Outline

The course is organized into ten units. For the intensive short course, each unit is assigned a half-day of intensive attention. For the full-quarter course, each unit is assigned a full week of normal attention. (A sixteen-unit version of the course is contemplated.) Units build in a logical sequence, introducing new material within a framework of existing knowledge.

The course manual [1] gives all student handouts, lecture transparencies, and provides an index. The main parts of the manual, as well as information about the course and its contributors, are available on the World-Wide Web.

Each unit consists of ninety minutes of lecture and three hours of lab. The lectures present the relevant theory and practical background, and are given by local experts in the subject area. Labs follow immediately and are scripted to enable the student to experience a particular technique, hands on, and then to challenge the student to extend the technique beyond the scripted instructions in some creative way. Generally we provide suggested enhancements, as well. The course culminates with student projects.

The units shown here were crafted for the intensive format.

**Unit 1**: Lecture on the vision of universal access to information and services, and the challenges of automatic speech recognition. A follow-on lecture covers the syllabus, course expectations, and other housekeeping issues. Students begin building systems right from the first lab. They construct an application that answers their telephone call
and says “Hello, world!” They then go on to build a pizza-ordering system that incorporates speech recognition to discriminate among three types of pizza. Generally the students finish well within the three hour allotment.

**Unit 2:** Lecture on data collection and corpus development, with a view to building and training special-purpose speech recognizers. The lab introduces students to data collection and has them write short Tcl scripts (generally one- and two-liners) to organize data that is collected. Data includes utterance waveform files and recognition results. Data collection is done both in the context of an actual application (the pizza-ordering system) and as a prompted list of utterances (the months of the year).

**Unit 3:** Lecture on phonetic labeling [5] discusses the Worldbet [6] labels used at OGI. Students are taught to use Lyre. The lab consists of labeling some of the collected month-name utterances. Lyre provides the capability to listen to any section of the waveform, and to see the waveform in both the time domain and the frequency domain, as well as neural network outputs and time-alignment labels (generated automatically or by hand). Labeling is slow work, so we do not label all of the corpus that the students collected. If time permits, we lecture on automatic labeling by forced alignment.

**Unit 4:** Lecture on artificial neural networks and training by conjugate gradient descent methods. In the lab, students train a special-purpose recognizer for month names, using the data collected and labeled earlier in the course. It is understood that there is insufficient data for adequate training but our purpose here is to train students rather than nets. To be frank, the nets have fairly abysmal performance characteristics, due to the small data collection, but students come to understand the importance of speech corpora if it was not clear before.

**Unit 5:** Lecture on word modeling, grammars, and Viterbi search. The lab consists of reviewing the labeled transcriptions in the OGI Numbers corpus [7] for various pronunciations of the digits (zero through nine, plus oh). Each team constructs state charts giving pronunciation variations for at least two of the digits. Teams are assigned different digits as starting places for their research.

**Unit 6:** Unit 5 lecture continues. The lab consists of browsing incorrect recognitions as follows. Students build a system to recognize certain words and write appropriate files to disk. When recognition fails, the student is to pinpoint the major source of failure. Does the word model allow the actual pronunciation? Does the neural network correctly specify the phonemes at each frame? Is there too much confusion from similar vocabulary?

**Unit 7:** Lecture on dialogue building helps students understand what errors are likely and how to spend their efforts to make better spoken dialogue systems. Students start their class projects.

**Unit 8:** We review and solve any problems students have encountered while planning or building their applications. Students commit to the application they will build, and briefly describe what they hope to accomplish. Locally the session is attended by the software developers, who assist in problem resolution. For lab, the class projects continue into the evening. Pizza (appropriately) is provided at about supper time. Teams finish their projects the same day.

**Unit 9:** Student project demonstrations are given for ten to fifteen minutes each. Students explain their projects, demonstrate them, and then take questions from the audience. We held the demonstrations open to visitors.

**Unit 10:** We conclude with an in-depth technical wrap-up. Students evaluate the tools, the lectures, and the lab assignments, and suggestions are developed into a list for future improvement. Some comments from the course evaluations are given below in section 6.

## 5. STUDENT PROJECTS

Here are excerpts from five of the student project summaries. These demonstrate not only the creativity of the students but also the breadth of activity supported by this tool and this course. It is important and exciting to note that these projects actually worked!
Voice Activated Network Control Service Simulation Prototype: "... simulates the control of two real telephone network services, 'call forwarding,' and 'cancel call waiting.' Upon receiving an incoming call, the system provides a list of options for the caller ... The system will playback a message to confirm the caller's intention and asks the caller to speak a name from a voice dialing list. Once the system recognizes the name, it will play a final confirmation message ..."

Directory Assistance: "The Directory Assistance system serve as a switchboard operator at OGI after the hours. First, the system answer a phone call, ask the caller if he or she have the extension number. If yes, the caller can directly dial the number. If not, the system will listen to a name (spelled), confirm the name, retrieve the associate telephone number from OGI extension database, inform the caller of the number, and forward the call to the right person."

Smart Voice Mail: "We ... [permit] the user to process messages randomly, by selecting a message by message number. We also give the user the option of listing messages by subject or by caller name. These items are requested from the caller when the incoming message is recorded. We assume that our users become experts after a few sessions ... We find that novice users adapt quickly to the terseness of the system. The quick access also reduces the user frustration."

E-Mail by Phone: "This system reads a user's e-mail over the phone. First, the user is prompted for a login name and password. When these are successfully recognized, the user enters the menu state. From the menu, the user can read the current message, get the date and time of the current message, go to the next or previous message, go to a specific message (referenced by number), or quit."

Adventure: "For our demonstration of the rapid prototyping speech tools, we connected to the classic Adventure game by Will Crowther and Don Woods. By providing recognition for the following vocabulary, it is possible to explore a small portion of this large game: walk north, walk south, ... go building. The text from the game is fed directly to the speech synthesizer allowing the player to 'see' where they are ..."

6. CONCLUSION

The overall course evaluation by the students was an enthusiastic 3.8 out of 4.0. (Typical ratings in other courses range from 3.0 to 3.5.) Here are two of the comments:

- (a student from industry): "BTW it was a great course, probably the best summer short course I have ever taken."
- (a full-time student): "I feel it would be an excellent course to introduce speech technology to development engineers."

After the course, several students continued to develop their projects, and to use them occasionally as practical devices for getting work done (such as monitoring computer jobs by telephone). The consensus of opinion is that the course was greatly appreciated and enjoyed, and that it will prove effective.

Revision is under way for a full-quarter version of the course, which will require some adjustments in lecture/lab ratio and in homework expectations. Future plans are to expand from ten units to sixteen. This will allow use of the course in semester-oriented settings, and allow greater flexibility of topic emphasis in quarter-oriented settings. Additional topics may include perceptual coding (PLP), utterance verification, dialogue repair, and more emphasis on CSLUs (Tel) programming.

A number of software improvements were suggested during the first technical wrap-up session. Many of these improvements have been made and more are under way.

6.1. Availability

We are pleased with this teaching approach and intend to make the course and its tools available to every interested college and university. Please contact us directly for up-to-date information on availability of software and course materials.

7. ACKNOWLEDGEMENTS

The students on our maiden voyage are especially acknowledged for their patience, enthusiasm, and constructive suggestions. Guest lectures were provided by CSLU faculty and staff: Ron Cole, Mike Noel, Terri Landers, Etienne Barnard, Mark Panty, and David G. Novick. CSLUr and CSLUs were developed by Pieter Vermeulen, Jacques Villiers, Stephen Sutton, Mark Panty, and others, under the direction of David G. Novick and Ron Cole. A one-day, pilot course taught at Clark Atlanta University was developed by David G. Novick and Stephen Sutton.

REFERENCES


