Connected Digit Recognition Experiments with the OGI Toolkit's Neural Network and HMM-Based Recognizers

Piero Cosi*, John-Paul Hosom**, Johan Shalkwyk**, Stephen Sutton**, and Ronald A. Cole**

*Institute of Phonetics – C.N.R.
Via G. Anghinoni, 10 - 35121 Padova (ITALY),
e-mail: cosi@csrf.pd.cnr.it www: http://www.csrf.pd.cnr.it

**Center for Spoken Language Understanding (CSLU)
Oregon Graduate Institute of Science and Technology (OGI)
P.O. Box 91000, Portland Oregon 97291-1000 USA
e-mail: mailto:{hosom,johans,sutton,cole}@cse.ogi.edu www: http://www.cse.ogi.edu/CSLU/

ABSTRACT

This paper describes a series of experiments that compare different approaches to training a speaker-independent continuous-speech digit recognizer using the CSLU Toolkit. Comparisons are made between the Hidden Markov Model (HMM) and Neural Network (NN) approaches. In addition, a description of the CSLU Toolkit research environment is given.

The CSLU Toolkit is a research and development software environment that provides a powerful and flexible tool for creating and using spoken language systems for telephone and PC applications. In particular, the *CSLU-HMM*, the *CSLU-NN*, and the *CSLU-FBNN* development environments, with which our experiments were implemented, will be described in detail and recognition results will be compared.

Our speech corpus is OGI 30K-Numbers, which is a collection of spontaneous ordinal and cardinal numbers, continuous digit strings and isolated digit strings. The utterances were recorded by having a large number of people recite their ZIP code, street address, or other numeric information over the telephone. This corpus represents a very noisy and difficult recognition task.

Our best results (98% word recognition, 92% sentence recognition), obtained with the FBNN architecture, suggest the effectiveness of the CSLU Toolkit in building real-life speech recognition systems.

1. INTRODUCTION

Since the early nineties, the Center for Spoken Language Understanding (CSLU) has been working on the development of new tools for creating spoken language systems. The result of this effort is the CSLU Toolkit, an integrated set of software and documentation that represents the state of the art in tools for research, development, and learning about spoken language systems [1]. The CSLU Toolkit is freely available for non-commercial use and may be downloaded from http://cslu.cse.ogi.edu/toolkit.

Usually, the development of spoken language systems is a lengthy and expensive process requiring months or even years to design, test, and deploy systems for useful applications. With the help of the Toolkit, an increasing number of inexperienced users are able to rapidly prototype, test, and deploy spoken language systems, and experienced researchers are provided with an environment for performing research and for testing and showcasing research advances.

In the following sections, a brief description of the Toolkit is given[1], implementations of a speaker-independent continuous-speech digit recognizer using three different architectures are described, and finally, the recognition results obtained by applying these recognizers to the OGI 30K-Numbers corpus [2] are compared.

2. CSLU TOOLKIT

The CSLU Toolkit has been developed to support speech-related research and development activities for a wide range of users and uses. Among various other topics, the Toolkit is designed to:

- enable domain experts, who may be naive about spoken language technology, to rapidly design spoken language systems for real applications, even in languages other than English, with easy-to-use authoring tools;
- generate state-of-the-art spoken language systems automatically from high level design specifications;
- learn about spoken-dialogue systems through coursework incorporated into the tools;
- easily perform research on human computer interaction in many tasks using spoken-dialogue systems;
- perform research on the underlying technologies, and incorporate research advances into working systems for evaluation in real applications.

[1] For a more detailed description see the www on-line Toolkit manual at http://cslu.cse.ogi.edu/toolkit.

This toolkit is a comprehensive software environment that integrates a set of core technologies including speech recognition, speech synthesis, and facial animation. It also features authoring and analysis tools that enable quick and easy development of desktop and telephone-based speech applications. The architecture of the Toolkit, as shown in Figure 1, has three main components: a set of libraries containing core technology modules specific to speech recognition, speech synthesis and facial animation, an interactive programming shell (CSLUsh [3]) and a graphically-based Rapid Application Developer environment (RAD).

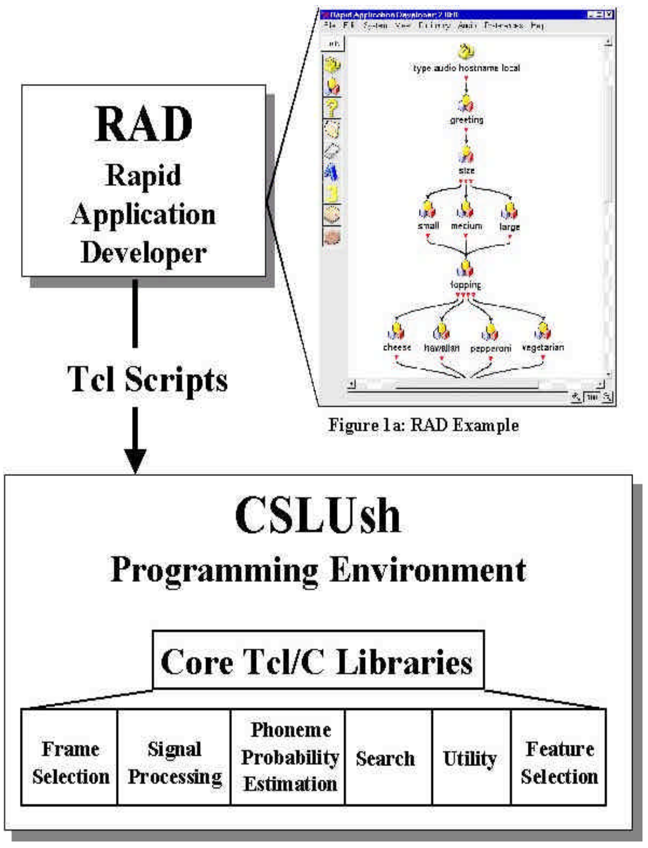


Figure 1a: RAD Example

Figure 1. Architecture of the CSLU Toolkit

2.1 Core Technology Modules

The core of the Toolkit consists of a set of modules that implement technology fundamental to all aspects of speech recognition, speech synthesis, and facial animation. These modules are written in C and form an application programming interface (API) that is independent of hardware and operating systems. The modules contain routines useful for signal processing, training Neural Networks (NN) and Hidden Markov Models (HMM), pipelined speech recognition with a Viterbi search, a telephone interface, and a vendor-independent text-to-speech interface. The modules provide a good deal of flexibility since they can be linked directly into a C program or individually loaded into a programming shell as needed.

2.2 CSLUsh: programming shell

The main application level of the Toolkit is an interactive Tcl/Tk-based [4] programming shell called CSLUsh (pronounced "slush"). CSLUsh incorporates the core technology modules (described in the previous section) with the well-known, freely available and widely ported Tcl/Tk scripting language. The functionality of each C-API is made available as a scripting command via a standardized calling convention and data are referenced as objects that can travel a network and can be saved to disk in a device-transparent way. The code that implements this consistent interface is made available to allow for extension of both the underlying C-API and additional `glue' functionality that may be lacking. An application is built by gluing together the modules based on the C-API and using additional application functions such as an event loop servicing file events, network events, and graphical user interface events. The application can be run across multiple platforms connected on a local LAN or the Internet by using the client-server capabilities built into the Tcl application interface. This includes TCP, UDP, the ability to run as a daemon, remote execution of Tcl scripts, and transfer of data objects.

The C-API modules are grouped into functional libraries that are dynamically linked and loaded at run time, allowing the application to scale in terms of resources. In addition, it allows for the extension of the Toolkit without changing the Toolkit itself.

2.3 RAD: Rapid Application Developer

The third component of the Toolkit is a Rapid Application Developer environment called RAD. RAD seamlessly integrates speech recognition, speech synthesis, facial animation and visualization tools into a graphical-based authoring environment for building and executing simple spoken language systems. RAD includes a palette of graphical 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, RAD 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 telephone touch-tone input (DTMF).

The interface is designed to require minimal technical expertise on the author's part and to simplify the specification process. For example, specifying a speech recognizer is as simple as typing in the words or phrases to be recognized. Similarly, the computer's speech output is specified by typing in words for the system to speak or by attaching a recorded voice. The animated face, Baldi, is automatically synchronized with either synthetic or recorded speech. At installation time, RAD is fully integrated into the local environment. After designing a speech application, the user selects from a list of input devices, which can include a microphone or telephone. Then the user presses the "Build" button followed by the "Run" button and the system executes. RAD inherits the power and flexibility of the underlying programming environment through its foreign code feature, enabling authors to create sophisticated applications that execute programming commands from within RAD. As authors become more experienced and familiar with the underlying programming environment, they can move beyond the scope of RAD's initial set of functions to build speech front-ends for existing applications such as voice handling of e-mail.

3. CSLU-ASR DEVELOPMENT ENVIRONMENT

Three different development environments are available for training speech recognition systems using state-of-the-art techniques. The first environment, CSLU-HMM, is used for developing Hidden Markov Model (HMM) recognizers. The second environment, CSLU-NN, is used to develop neural-network-based (NN) recognizers. The third environment, CSLU-FBNN, is an extension of the CSLU-NN environment that allows training neural-network recognizers with the forward-backward algorithm (FBNN) [5]. These three environments are referred to collectively as the CSLU-ASR development environment. They are designed as extensions to the CSLU shell (CSLUsh) and use a wide variety of pre-existing modules for distributed computing, speech signal processing, mathematical operations, and various miscellaneous modules to provide a complete recognition development environment.

As shown in Figure 2, the complete CSLU-ASR development environment is a collection of modular building blocks which aim to provide the user with an easy to use, powerful research and development environment.

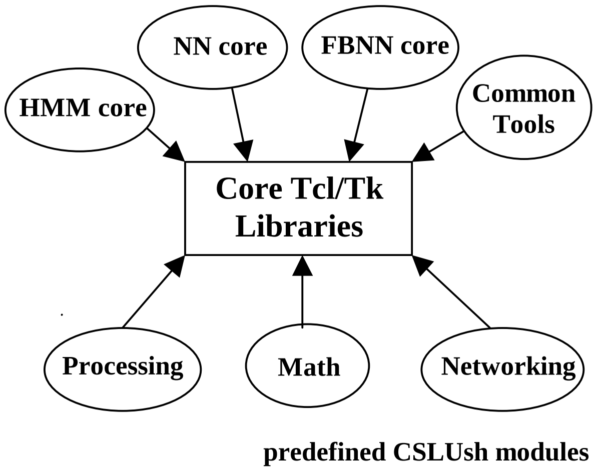


Figure 2. Architecture of the CSLU-ASR development environment comprising HMM, NN and FBNN tools.

Based on CSLUsh, which uses Tcl/Tk to provide a scripting environment, CSLU-ASR provides a flexible environment in which the user can shape the existing modules to meet specific needs. Implemented in Tcl/Tk and C, these development tools support a flexible environment for various modeling strategies. Great care was taken to design all of the core components to operate in as efficient and consistent a manner as possible, with special attention given to modularity, portability and extendibility.

3.1 CSLU-NN Development Environment

This tool contains various Tcl/Tk and C functions that implement the general steps needed to create a neural-network-based recognizer, including:

- specifying the phonetic or sub-phonetic categories that the network will recognize;
- finding many samples of each of these categories in the speech data;
- training a network to recognize these categories and evaluating the network performance using a test set.

As shown in Figure 3, the recognizers that are developed use a frame-based approach with a neural network for estimating posterior probabilities. The steps performed during recognition are:

- The waveform is divided into frames;
- Features are computed for each frame. These features describe the spectral envelope of the speech at that frame and at a small number of surrounding frames;
- The features in each frame are classified into phonetic-based categories using a neural network. The outputs of the neural network are used as estimates of the probability, for each phonetic category, that the current frame contains that category;
- The matrix of probabilities and a set of pronunciation models is used by a Viterbi search to determine the most likely word(s).

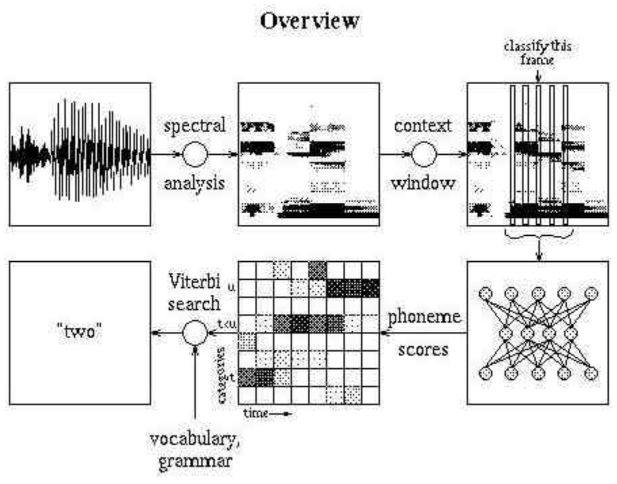


Figure 3. Overview of frame-based speech recognition using neural networks.

3.2 CSLU-HMM Development Environment

The current version of CSLU-HMM has tools for both standard and advanced HMM training methods [6,7,8,9]. The standard techniques such as

- model initialization using vector quantization (VQ);
- model training based on the expectation/ maximization (EM) algorithm;
- standard embedded model parameter re-estimation.

Also supported are more advanced techniques, such as:

- maximum a-posterior (MAP) training;
- maximum likelihood linear regression (MLLR).

Parameter tying may be done at either the model, state, mixture component, mean and/or covariance levels. Moreover, to facilitate the construction of HMM recognizers for medium to large vocabulary tasks, CSLU-HMM supports decision-tree state clustering.
The outline of the typical process of training an HMM model for speech recognition is shown in Figure 4, where for each block the associated CSLU-HMM scripts are indicated.

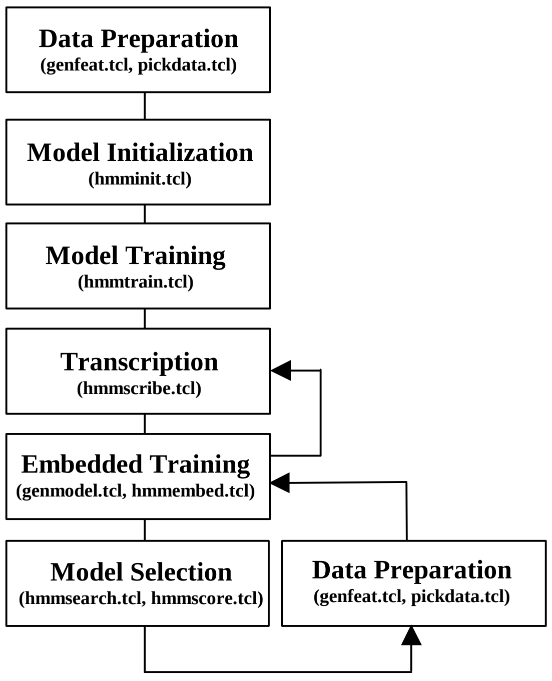


Figure 4. Outline of typical HMM training.

3.3 CSLU-FBNN Development Environment

The CSLU-FBNN tool is an extension of the CSLU-NN tool that allows the user to train neural-network recognizers using the forward-backward procedure commonly employed when training HMMs. A neural network is used as a state emission probability estimator and the conventional forward-backward algorithm is used for estimating continuous targets for the NN training patterns. The network is not trained on binary target values, but on the probabilities of each category belonging to each frame [5].

4. CONNECTED DIGIT RECOGNITION EXPERIMENTS

For investigating the performance of the CSLU-ASR development environment, three continuous-speech, speaker-independent digit recognizers were trained using each of the three development environments. The recognizers were trained on telephone-band speech using the OGI 30K-Numbers corpus [2], available from http://www.cse.ogi.edu/CSLU/corpora/.
The performance of each recognizer was evaluated on a development set and a test set. The digits portion of the OGI Numbers corpus contains many thousands of utterances of continuous and isolated (uttered with pauses between each word) digit strings recorded by having a large number of people recite their ZIP code, street address, or other numeric information over the telephone, under various conditions. As a result, many aspects of "real-life" speech are present in the data, including noise, widely-varying energy levels, and dialect differences. Three-fifths of the available data were randomly chosen and allocated (in a speaker-independent way) for training, and one-fifth each were allocated for development and testing.

4.1 NN experimental settings

As illustrated in Figure 3, a three-layer neural network was trained to estimate the probability of 165 context-dependent phonetic categories at every 10-msec frame.

4.1.1 Segmentation

For creating context-dependent categories, each phoneme was split into one, two, or three parts, depending on the length of the phoneme and how much the phoneme was thought to be influenced by coarticulatory effects. One-part phonemes are context independent. Two-part phonemes have the left (first) half dependent on the preceding phoneme and the right (last) half dependent on the following phoneme. Three-part phonemes have a left third that is dependent on the preceding phoneme, a middle third that is context independent, and a right third that is dependent on the following phoneme. Phoneme states were trained for different preceding and following phonetic contexts, and some phonetic contexts were grouped together to form a broad-context grouping. For example, the dental fricatives /s/, /z/, /th/, and /T/ were combined into one broad-context grouping, so that the left half of /I/ following /z/ was the same category as the left half of /I/ following /s/. The broad-context groupings were done based on acoustic-phonetic knowledge.

4.1.2 Training

As a first step in training the neural network, Perceptual Linear Prediction [10] (PLP) features (including energy) and Mel-Frequency Cepstrum Coefficients [11] (MFCC) are computed at non-overlapping 10-msec frames. At each frame, a 130 dimensional vector of PLP+MFCC

features is constructed using five surrounding frames; we use 13-PLP+13-MFCC features from frames at -60, -30, 0, 30, and 60 msec relative to the frame of interest.
The training data are searched to find all the vectors of each category in the hand-labeled section of the OGI-30-Numbers corpus. The neural network is trained using the back-propagation method with 130 inputs, 200 nodes in the single hidden layer, and one node for each context-dependent category in the output layer (for a total of 165 output nodes). Training is done for 70 iterations, and the iteration with the best performance on the development set is chosen to be the final baseline neural network. (This network will be referred to as network *B*, for baseline.)

4.1.3 Forced Alignment

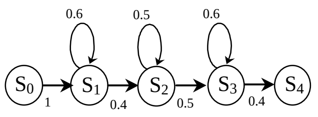
Often, the corpus we want to train on has text transcriptions but no time-aligned phonetic labels. In this case, we can create either phonetic labels or category labels using a process called "forced alignment". Forced alignment is the process of using an existing recognizer to recognize a training utterance, where the grammar and vocabulary are restricted to be the correct result. The result of forced alignment is a set of time-aligned labels that give the existing recognizer's best alignment of the correct phonemes or categories. If the existing recognizer is good, then the labels will have more consistent time alignments than the hand labels. These labels can then be used for training a new recognizer. Even if the existing recognizer produces some alignment errors, this process can be used to determine an initial set of labeled training data. The best network previously trained (*B*) is, in fact, utilized to force align all of the training data in the OGI 30K-Numbers corpus (not all of the training set has been phonetically hand labeled), and a new force-aligned network is trained for 30 iterations using all of these new phonetically labeled data. The best force-aligned network (as evaluated on the development set) is chosen to be the final force-aligned neural network, called *FA*.

4.1.4 Recognition

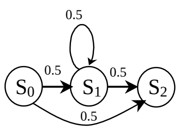
For recognition of an utterance, PLP+MFCC vectors are computed in the same way as for training. These PLP vectors are input to the neural network, which computes for each frame the probabilities that the current frame contains each of the specified categories. As illustrated in Figure 3, the result of classification is therefore a C x F matrix of probabilities, where C is the number of categories and F is the total number of frames. This matrix is then used by a Viterbi search algorithm to determine the most likely sequence of words. The Viterbi search uses minimum and maximum durations of each category to constrain the possible word choices, but these are not "hard" limits. If the duration of a hypothesized category falls beyond one of the specified limits, a penalty is applied; this penalty is proportional to the time difference between the specified limit and the hypothesized duration. Initial values for these limits are taken from the durations of the categories that were used to train the baseline network. These values are refined during the development stage by taking durations of the categories that were created during forced alignment.

4.2 HMM experimental settings

The same phonetic hand-labeled subset of the data used in the previous section for training the NN digit recognizer is used to create initial "seed models" and train a continuous phone-based HMM digit recognizer. As illustrated in the example of Figure 5, each phone is represented as a 3-state left to right model with a discrete Gaussian mixture.



```
prototype mono numstate 4 mixtures 3 transp
 0.000  1.000  0.000  0.000  0.000
 0.000  0.600  0.400  0.000  0.000
 0.000  0.000  0.500  0.500  0.000
 0.000  0.000  0.000  0.600  0.400
 0.000  0.000  0.000  0.000  0.000
```



```
prototype onestate numstate 4 mixtures 3 transp
 0.000  0.500  0.500
 0.000  0.500  0.500
 0.000  0.000  0.000

define mono  <z>   <ih>   <r>   <ow>   <w>   <ah>   <n>   <t>
             <uw>  <th>   <iy>   <f>  <aor>  <ay>  <v>   <s>
             <ks>   <eh>   <ey>  <sil>;
define onestate <sp>

tie <sil>.state[2] <sp>.state[1]
```

Figure 5. Example of monophone models created with the CSLU-HMM configuration scripts. The short pause model (/sp/) is tied to the center state of the silence model (/sil/).

For each frame, 13 Mel-frequency cepstral coefficients are computed every 10 milliseconds. Cepstral mean subtraction is performed and the first and second order time derivatives are added to the base cepstral coefficients to form a 39-element feature vector. As in the previous section (see 4.1.3), forced alignment is applied to the whole training set to create new automatically labeled data, with which retraining and refining the initial seed models is done.
Embedded parameter re-estimation, addressing the problem of modeling the interactions between neighboring models (which is not addressed by single model training) is executed, and finally evaluation is performed on the development set. The “search build” procedure uses a triphone lookup table to determine which model to use during cross word expansion, thus the expected triphone models are tied to the present corresponding monophone model. All these steps, such as data preparation (dividing up a corpus, word transcription, phonetic transcription, feature extraction,

data selection), model training (model initialization, single model training, forced alignment, embedded parameter re-estimation) and evaluation are executed by specific Tcl scripts. Results are given for the best experimental setting, obtained with 16 Gaussian mixtures per state.

4.3 FBNN experimental settings

Like most of the other hybrid systems, the neural network in this system is used as a state emission probability estimator. A three-layer fully-connected neural network was used, with the same configuration as the baseline and forced-alignment neural networks. The output categories are the same as in the original NN system. Unlike most of the existing hybrid systems which do not explicitly train the within-phone model transitions, this new hybrid trains the within-phone models to probability estimates obtained from the forward-backward algorithm, rather than binary targets. We call this new system FBNN (Forward-Backward Neural Networks) or NN/HMM. To start FBNN training, an initial binary-target neural network is required. For this initial network, we used the network resulting from forced-alignment training (*FA*). Then the forward-backward re-estimation algorithm is used to regenerate the targets for the training utterances. The forward-backward re-estimation is implemented in an embedded form, which concatenates the phone models in the input utterance into a "big" model and re-estimates the parameters based on the whole input utterance. The networks are trained using the standard stochastic back-propagation algorithm, with mean-square-error as the cost function.

5. RESULTS

The three systems (NN, HMM, and FBNN) are evaluated on the development set and test set randomly chosen from the OGI 30K-Numbers corpus. (The training, development, and test sets are all speaker-independent.) Results are summarized in Table 1.

Data Set	Unit	NN		HMM	FBNN
		B	FA		
Dev. Set	word	97.76	97.76	96.37	97.94
	sentence	87.59	91.19	87.35	91.80

Test Set	word	96.36	97.58	96.42[2]	97.87
	sentence	86.82	91.03	86.08	91.77

Table 1. Word and sentence percent accuracy recognition rates for the various systems. *B* is the Baseline NN, while *FA* is the Force-Alignment NN.

[2] Results obtained with 16-Gaussian mixtures per state.

The FBNN performance is a 12% reduction in word-level error compared to the FA system, and a 41% reduction in word-level error compared to the HMM system. These results represent the best obtained results on the OGI-30K-Number corpus since the Toolkit was designed.

6. CONCLUSIONS AND FUTURE WORK

Encouraging results have been achieved on a very difficult telephone-band digit recognition task, suggesting the effectiveness of the CSLU Toolkit in building real-life speech recognition systems. Currently we are extending this work to an Italian digit corpus.

REFERENCES

[1] M. Fanty, J. Pochmara and R.A. Cole, “An Interactive Environment for Speech Recognition Research”, Proc. of ICSLP-92, Banff, Alberta, October 1992, pp.1543-1546.

[2] R.A. Cole, M. Fanty, M. Noel and T. Lander, “Telephone Speech Corpus Development at CSLU” Proc. ICSLP-94, Yokohama, Japan, September 1994, pp. 1815-1818.

[3] J. Schalkwyk, J.H. de Villiers, S. van Vuuren, and P.Vermeulen, “Cslush: An Extendible Research Environment,” Proc. of Eurospeech 97, Rodhes, September 1997, pp 689-692.

[4] J.K. Ousterhout, *Tcl and the Tk Toolkit*. Addison Wesley, 1994.

[5] Y. Yan, M. Fanty and R. Cole, “Speech Recognition Using Neural Networks with Forward-Backward Probability Generated Targets” Proc. ICASSP-97, Munich, Germany, April 1997, pp. 3241-3244.

[6] L. Rabiner and B-H Juang, Fundamentals of Speech Recognition, Prentice Hall

[7] G.Zavaliagkos, “Maximum A Posteriori Adaptation Techniques for Speech Recognition.” PhD thesis, Northeastern University, Boston, Massachusetts, October 1995.

[8] C.J.Legetter and P.C.Woodland, “Speaker Adaptation of HMM's Using Linear Regression,” Tech. Rep. CUED/F-INGENG/TR.181, Cambridge University Engineering Department, Cambridge, England, 1994.

[9] S.J.Young and P.C.Woodland, “Tree-based state-tying for high accuracy acoustic modeling,” Proc. Human Language Technology Workshop, pp. 307-312, March 1994.

[10] H. Hermansky, “Perceptual Linear Predictive (PLP) Analysis of Speech” J. Acoust. Soc. Am., Vol. 87, No. 4, pp. 1738-1752, April 1990.

[11] S.B. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences", IEEE Trans. Acoust.Speech and Signal Processing, Vol. ASSP-28, No. 4, August 1990, pp. 357-366.