WEB-BASED EXPERIMENTS FOR INTRODUCING SPEECH RECOGNITION BASICS IN A DSP COURSE

Venkatraman Atti and Andreas Spanias

Department of Electrical Engineering, MIDL Lab
Arizona State University, Tempe, AZ 85287-5706, USA
[atti, spanias]@asu.edu

ABSTRACT

In this paper, we describe web-based educational software tools tailored to expose students in an undergraduate DSP course to the basics of hidden Markov model (HMM)-based speech recognition. In particular, we developed Java software that enables on-line computer laboratories on the essential pre-processing, HMM, and Viterbi algorithms as used in a basic speech recognition task. The software is complemented by streaming lectures, a set of on-line demonstrations with animation, and exercises that take the student through HMM training and recognition. The software is being made available to students in the Fall of 2003 and we expect to present assessment results at the conference.

1. INTRODUCTION

Speech recognition is an area in signal processing that is having a major impact in applications that involve a machine-human-interface. Although algorithms and hardware supporting speech recognition applications are still evolving, it is expected that the integration of language modeling and artificial intelligence algorithms in the next 5 to 10 years will enhance considerably speech recognition accuracy, which will in turn enable several new applications and products. A common perception, among the experts in the field, is that the number of practitioners and researchers trained by universities in the field is relatively small. Furthermore, most of the graduating engineers from Electrical and Computer Engineering departments are not exposed to the basic tools supporting this application. At Arizona State University, as part of an effort to introduce undergraduates in DSP to application-oriented content, we are developing a series of on-line modules that include Java™ software, animated demonstrations, computer exercises, and video streamed lectures. This paper describes in particular, the Java software development aspects of this effort. Although HMM-based speech recognition research has been well published [1]-[3] and supported by advanced software tools for the expert, there are no educational tools tailored specifically for the novice. For instance, the hidden Markov model Toolkit (HTK) [4] developed by S. Young *et al.* represents a set of highly sophisticated programs designed primarily for research purposes. The software tools we developed provide hands-on training to DSP class participants. The software consists of roughly 5000 lines of Java code and is accompanied by a series of computer experiments that addresses the various stages of a typical speech recognition unit. These Java speech recognition modules have been integrated in the Arizona State University's award-winning on-line simulation software called J-DSP† [5]-[8]. Hence, the Java software for speech recognition will be described in the context of the J-DSP graphical user interface (GUI).

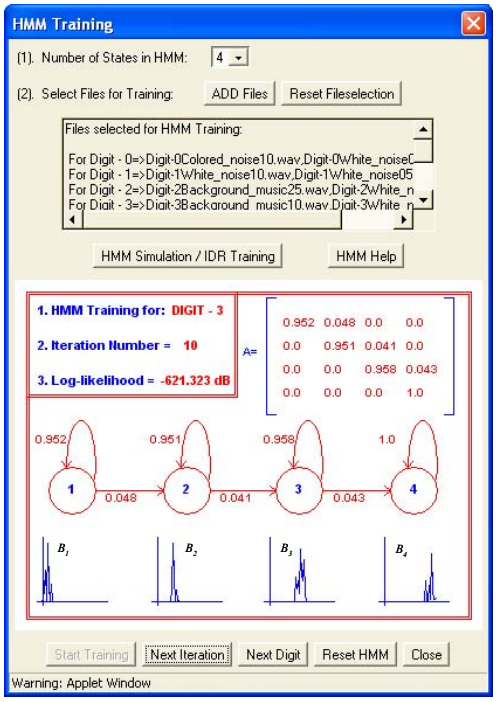

Figure 1. The HMM module in J-DSP

The rest of the paper is organized as follows. Section 2 gives an overview of the J-DSP simulation environment and the various speech recognition modules integrated in J-DSP. Next, Section 3 addresses the front-end analysis modules and the three exercise sets developed. Section 4 reviews the HMM training procedure with an example simulation. Figure 1 depicts the HMM GUI designed in J-DSP to emphasize the HMM training procedure. A brief description on the HMM GUI is also included in Section 4. Section 5 describes the recognition unit. Finally, Section 6 presents the conclusion.

† Work funded in part by NSF-CCLI-0089075. J-DSP has been ranked in the top three learning resources for the year 2003 by the NEEDS.

2. J-DSP SIMULATION ENVIRONMENT AND THE SPEECH RECOGNITION MODULES

Figure 2 shows the simulation tool's editor window. From this figure, it can be seen that all functions appear as graphical blocks. Each one of these blocks is associated with a specific signal processing function. A variety of DSP systems can be simulated by connecting the blocks, and editing their parameters through dialog windows. System execution is dynamic, which means that any change at any point of a simulation will automatically take effect in all related blocks. More detailed instructions on block manipulation can be obtained from [5]. The various speech recognition modules implemented in J-DSP are listed in Table 1. These blocks are divided into three sub-sections, i.e., feature extraction, HMM, and built-in demos, based on their functionality.

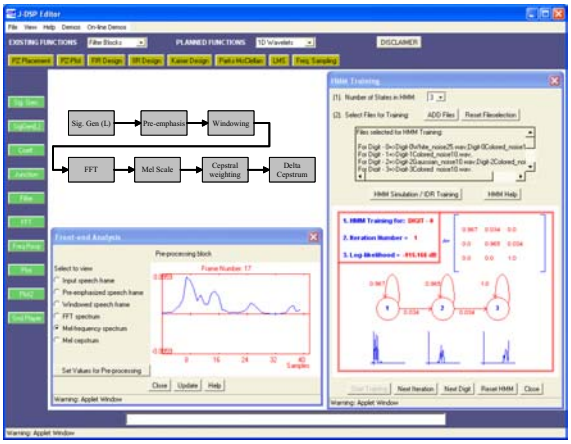

Figure 2. An example speech recognition simulation in J-DSP

Table 1. A list of speech recognition modules in J-DSP

	Block Name	Block Description
Feature extraction	***Pre-emph.***	Pre-emphasis filtering
	Mel-Scale	Computes the mel-spectral coefficients
	Mel-Cepst.	Computes the mel-cepstral coefficients
	Cepst. Weight.	Performs cepstral weighting
	Delta-Cepst.	Computes the delta-mel-cepstrum
HMM	***HMM Training***	Designs HMMs
	Recognition-unit	Performs Viterbi decoding
Demos	***Front-end-Anlys.***	Pre-processing stage demo
	Isolated-dig-Rec.	Isolated digit recognition (IDR) demo
	HMM Basics	Example demos to introduce HMMs

3. FRONT-END ANALYSIS MODULE IN J-DSP

The first stage in a typical speech recognition system constitutes a feature extraction block. In particular, the main objective is to convert the raw acoustic speech into a form that is well-suited for automatic speech recognition (ASR) [1]. Figure 3 shows the front-end analysis GUI (upper dialog window) designed to examine the various aspects of pre-emphasis filtering, mel-spectrum, and mel-cepstrum.

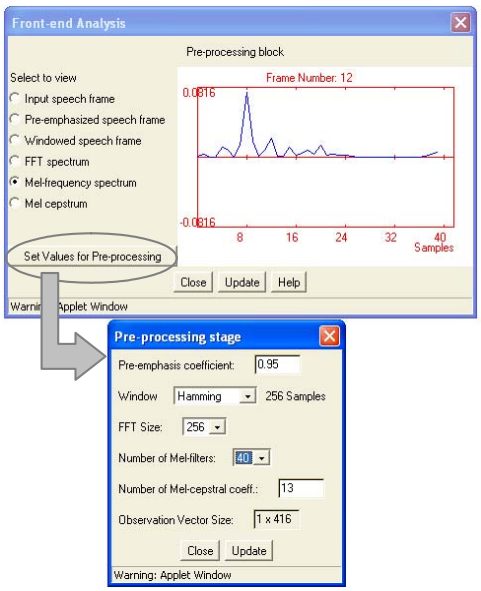

Figure 3. Front-end analysis to obtain the feature vectors

The lower window in Figure 3 provides options to set the parameters for pre-processing. First, the pre-emphasis coefficient, *a*, corresponding to a simple first-order FIR filter (1) is set.

$$H(z) = 1 - az^{-1} \tag{1}$$

Next, a windowing operation is performed to obtain a smoother evolution of the LP filter, thereby providing better spectral estimates. The available window functions are: Hamming, Bartlett, rectangular, and Kaiser. Next, the short-time power spectrum of the windowed speech is computed using either a 256-point FFT or linear prediction (LP) analysis. A set of L triangular band-pass filters, spaced in mel-frequency scale, are applied to the short-time power spectrum. The mel-cepstral coefficients, c_i, are then computed by taking an L-point discrete cosine transform (DCT) as follows:

$$c_i = \sqrt{\frac{2}{L}} \sum_{m=1}^{L} \log(|S(m)|) \cos\left(\frac{\pi i}{L}(m-0.5)\right) \quad i = 0,1,...,C-1 \tag{2}$$

where *S(m)* represents the mel-spectrum energy and C the number of mel-frequency cepstral coefficients (MFCC). Both, L and C are user-defined parameters. From the example simulation in Figure 3, it can be noted that $L = 40$, $C = 13$, and the number of speech frames, $N_f = 32$ (not shown in the figure) that result in an observation vector of size 1 x 416.

3.1. Exercise Sets – I, II, and III

A concise description of the first three exercise sets follows [5]. The first exercise set illustrates the following: a) the effects of pre-emphasis filtering over voiced and unvoiced segments and for different pre-emphasis filter coefficients; b) the significance of pre-emphasis filtering in the automatic formant tracking; and c) the analysis of pre-emphasis filtering in the pole-zero domain. The second exercise set involves experiments with: a) different windows; b) various FFT sizes (typically 64, 128, and 256-point); c) number of mel filters (L=10, 20, and 40); and d) analyze the similarities between the triangular band-pass mel-filtering and the weighted Daniel periodogram. The third

exercise set primarily highlights the aspects of using a DCT in computing the cepstral coefficients. The student will also run a pre-configured simulation in J-DSP to analyze the mel-spectrum obtained from both the FFT-based and the LP analysis-based methods.

4. HMM TRAINING IN J-DSP

This section addresses the HMM functionality and the associated GUI implemented in J-DSP. Figure 4 shows the sequence of steps typically employed in training HMMs. In this figure, a simple isolated digit recognition scenario is considered to highlight these steps; and it should be noted that we will confine to high-level descriptions while explaining the HMM design procedure. An HMM model, λ, is usually characterized by the four parameters given by,

$$\lambda = (N, \pi, A, B) \qquad (3)$$

where 'N' is the number of states in the model, 'π' the initial state distribution, 'A' the state transition matrix, and 'B' the state observation probability distribution.

First, the observation vectors O_t, obtained from the front-end analysis are used to initialize the hidden Markov model. Next, the forward (α) and backward (β) variables are computed based on the initial HMM model. A detailed mathematical development and the design equations for the forward and backward variables can be found in [2]. Using α and β, we compute another set of parameters γ and ξ in order to re-estimate the HMM model (π, A, B) as shown in Figure 4. A log-likelihood, L_l is computed for each iteration from the re-estimated HMM model as follows:

$$L_l = \log(\pi) + \sum \log(A) + \sum \log(B) \quad \text{(dB)} \qquad (4)$$

The training process will be terminated when two adjacent log-likelihood values remain approximately the same, resulting in a final HMM model, λ_d. In the training procedure mentioned here, we considered the continuous HMM, while the simulations involving discrete HMMs and VQ can also be performed in J-DSP. In the example simulation shown in Figure 5, the number of states is $N = 4$ and the HMM model is estimated for 'digit-2'.

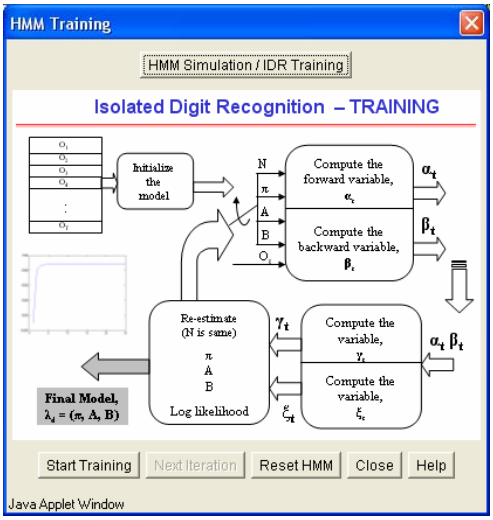

Figure 4. The HMM training process

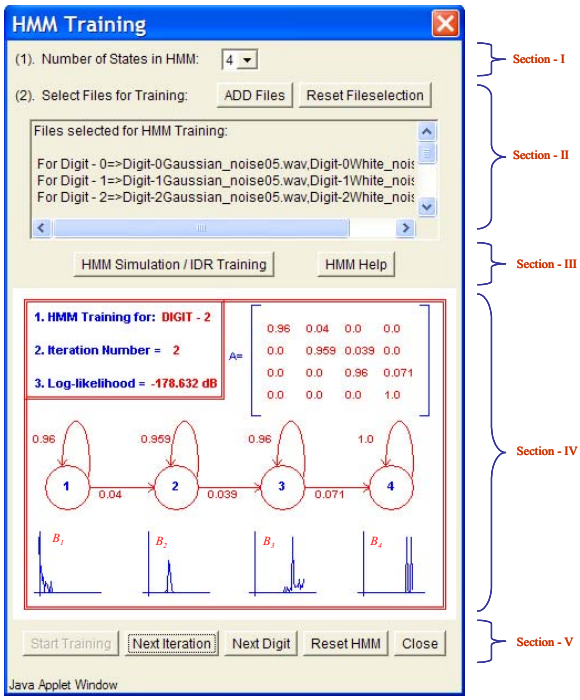

Figure 5. The GUI depicting the HMM training in J-DSP

4.1. HMM GUI Description

The HMM dialog window shown in Figure 5 consists of five sections. The first-section provides options to choose the number of states, N in the HMM model. The second-section primarily helps the user to select the files for HMM training. The dialog window shown in Figure 6 facilitates the file-selection process. Furthermore, options are provided to add noise to the input files (explained later in detail while discussing the exercise set-VI). The third-section provides access to the help topics and the on-line HMM documentation [5].

The fourth-section is the most important part of the GUI that essentially explains the training procedure through graphics. This section consists of four parts – (i) the top-left corner provides information that reflects the simulation status; (ii) the top-right corner presents the changes in the transition probabilities in the form of a matrix; (iii) the middle portion shows the animations of the four states (the circles) and their corresponding transition probabilities; and, (iv) the bottom part presents the plots of the observation probability distribution associated with each state. Finally, the fifth-section includes the push-buttons ('*Start Training*', '*Next Iteration*', '*Next Digit*', and '*Reset HMM*') that control the simulation. Moreover, options are provided to reset the file section or to reset the HMM model at any point of the simulation process. Next, we present the computer experiments designed to review the HMM training procedure.

4.2. Exercise Sets – IV, V, VI and VII

The fourth exercise set deals with the implementation of an isolated digit recognition system using the three modules, i.e., the front-end analysis, the HMM training, and the recognition. Moreover, this computer experiment helps students to get

acquainted with the HMM GUI and understand the basic steps involved in the HMM design procedure. The fifth exercise set is an extension of the previous one and helps students analyze the performance of the HMMs designed by varying the number of states, changing the feature vector size, and various other front-end analysis parameters.

Figure 6. File selection dialog window for HMM training

The exercise sets VI and VII are more advanced relative to the previous experiments and are primarily concerned with evaluating speech recognition in the presence of noise and for different training files. Figure 6 shows the dialog window designed for the file selection in HMM training. In the sixth exercise set, students study the implications of using noisy files in the training process. Different signal-to-noise ratio (SNR) levels can be set by selecting a specific dB-value using the *option-(3)* shown in Figure 6. The *option-(2)* in this figure allows users to choose a noise type from the following: white noise, Gaussian noise, colored noise, and background music. The seventh exercise set is concerned with evaluating the performance of the HMMs designed based on training files chosen from both with-in and outside the training set. Pre-configured HMM demos are developed in J-DSP that expose students to the step-by-step procedure of training HMMs. Moreover, the exercise sets VI and VII have a partial overlap with the Viterbi decoder experiments that are discussed next.

5. RECOGNITION UNIT – THE VITERBI DECODER

The recognition module implemented in J-DSP evaluates the conditional observation probability, $P(O_t|\lambda_d)$ for each HMM model, and chooses the corresponding model that yields the highest probability. Estimating the probability, $P(O_t|\lambda_d)$ involves finding the optimal state sequence, i.e., finding the path (in a trellis of states) that has the highest probability. This can be done efficiently using the Viterbi algorithm [2].

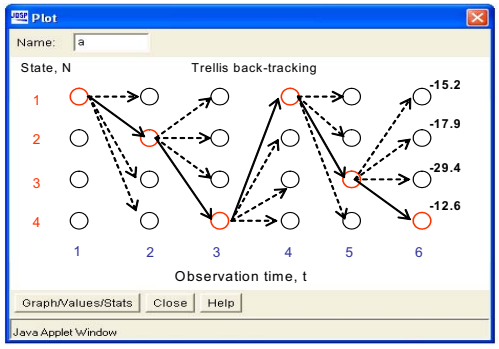

Figure 7. Trellis back-tracking in J-DSP

5.1. Exercise Sets – VIII, IX, and X

The underlying principles of trellis back-tracking are illustrated in the eighth exercise set. More emphasis is laid on illustrating the process involved in estimating the optimal state sequence. Figure 7 shows the GUI developed in J-DSP for studying the trellis back-tracking process. In the exercise sets IX and X, the students test the recognition performance by varying the number of training files and in the presence of noise. The students are asked to report the baseline results for an IDR scenario. The exercises that have been outlined here in this paper as well as additional ones not described here are given to students with detailed write-ups that have a theory section and step-by-step instructions for operating the software.

6. CONCLUSIONS

This paper presented simulation modules and exercises to teach the HMM-based speech recognition techniques in an undergraduate DSP course. The exercise sets I through X are designed such a way that they can be included in a course as one comprehensive term-project that essentially addresses the basic concepts of HMM-based speech recognition.

7. ACKNOWLEDGEMENT

The authors would like to acknowledge all the students (S. Benton, N. Chakravarthy, A. Natarajan, C. Panayiotou, S. Sabesan, Y. Song, T. Thrasyvoulou, and B.Veeramani) of the speech recognition class, a special course offered during the Spring of 2003 at ASU, for their valuable feedback that influenced this work.

8. REFERENCES

[1]. L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE,* 77(2), pp. 257-286, February 1989.

[2]. L. R. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*. New Jersey: Prentice Hall, 1993.

[3]. The ISIP web-page, Mississippi State University: http://www.isip.msstate.edu/projects/speech/index.html

[4]. Hidden Markov Model Toolkit (HTK): http://htk.eng.cam.ac.uk/

[5]. The J-DSP™ web-page, MIDL LAB, Arizona State University: http://jdsp.asu.edu

[6]. A. Spanias, *et al*, "Development of new functions and scripting capabilities in Java-DSP for easy creation and seamless integration of animated DSP simulations in Web courses," in *Proc. of IEEE ICASSP*, Vol. 5, pp. 2717-2720, May 2001.

[7]. A. Spanias, *et al*, "On-line laboratories for speech and image processing and for communication Systems Using J-DSP," in *2nd DSP-Education workshop*, Pine Mountain GA, Oct 13-16, 2002.

[8]. V. Atti, *et al*, "On-line simulation modules for teaching speech and audio compression," in *33rd ASEE/IEEE FIE-03*, Boulder, Nov. 2003