Implementation of Speaker Identification System by Means of Personel Computer

Sheikh Husssain Shaikh Salleh, Ahmad Zuri, Zulkarnian Yusoff, Syed Rahman, Lim Soon Chieh

Fakulti Kejuruteraan Elektrik
Universiti Teknologi Malaysia
81310 Skudai, Johor, Malaysia
Email: hussain@suria.fke.utm.my

Abstract: Speech processing systems are highly complex and teaching students in this subject matter with the underlying technologies can be a challenging task. The aim of this work was to give a hands-on experience via a development of speech processing system based on Hidden Markov Model (HMM) as a teaching aid. In this paper, a method for implementing speaker recognition system using our toolkit was developed as a dedicated laboratory environment for students. For speaker recognition, experiments were performed to evaluate the performance of the system with 30 speakers (22 impostors and 8 clients). The identification error was 2%, the false acceptance rate was 28% and the false rejection rate was 1%.

Teaching Module Speech Recognition (TMSR) toolkit was used in the lab which was part of the courses on digital signal processing (DSP) technology given by the Computer Engineering and Microelectronics Department. Students are given some initial guidance on how to use the toolkit and instructions to carry out the speaker identification experiments. Overall, the laboratory system was a success and plans are taken in the coming academic years to improve and extend the capability of the system.

Keywords
Speaker recognition, speaker identification, speaker verification.

I. INTRODUCTION

Speech recognition has been an important subject for research, and its development has come to a stage where it has been actively and successfully applied in a lot of industrial and consumer applications. The methods used for speech recognition have since been developed and improved, with increasing accuracy and efficiency leading towards a better human-machine interface. Speech recognition is very useful in various applications such as voice-activated systems, industrial control, and also automatic dialer applications. In this paper, we developed a general-purpose speaker recognition system for personal identification as a security system. In such systems, confidentiality is of utmost importance. Thus, a speaker recognition system must be totally reliable in terms of acceptance of client speakers and rejection of impostors.

HMM-based speech recognition system has already been applied successfully in commercial products such as the IBM ViaVoice, Dragon NaturallySpeaking, L&H's Voice Xpress and Philips's FreeSpeech. However, none of these commercially available speech recognition products are made in Malaysia. Besides, there are no speech recognition programs for learning and teaching speech processing developed locally. Thus, we have come up with a user-friendly and flexible speech recognition program that facilitates learning of HMM as a tool for doing speech and speaker recognition experiments. The system is then applied to recognition of isolated Bahasa Melayu digits, that is 'kosong', 'satu', 'dua', 'tiga', 'empat', 'lima', 'enam', 'tujuh', 'lapan', and 'sembilan'. Experiments were done to evaluate the system's performance on speaker recognition, which can be further divided into speaker identification and speaker verification.

II. THE TMSR TOOLKIT.

The Teaching Module Speech Recognition (TMSR) toolkit is built upon established speech recognition algorithms. The speech recognition system that was built consists of the following blocks as shown in Figure 1.0. Detail explanations of these blocks will be given in the next section. As can be seen in Figure 2.0 of the TMSR toolkit, the window shows an outline of the components of the system and how they interact and depend. For each box the corresponding module window can be opened with a mouse click. There are multiple copies of documents, hence the name, multiple-document-interface. The documents are where the sound samples are stored. Each document can store a sample. There are 5 buttons of interest on the toolbar. To start recording a new sample, press the 'Studio' button as shown. A dialog box 'Studio' will pop up. The dialog box contains some buttons:

1. Record - Record a new sample in the active document

0-7803-6355-8/00/ $10.00 © 2000 IEEE

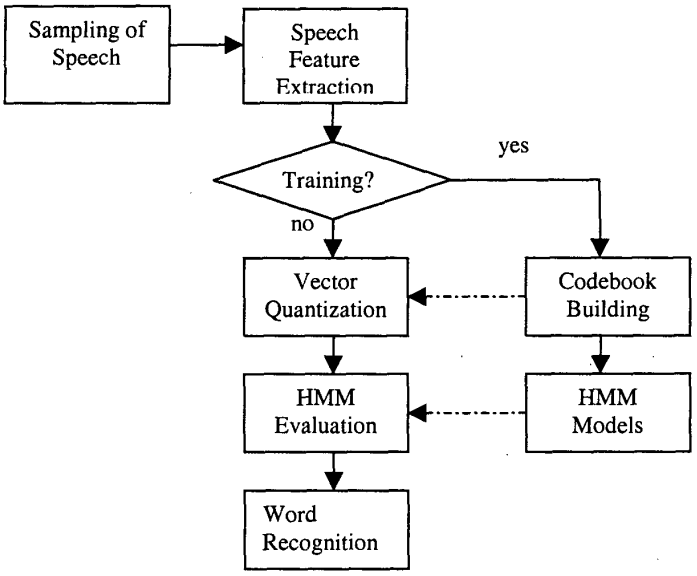


Figure 1.0. The building blocks for the TMSR toolkit.

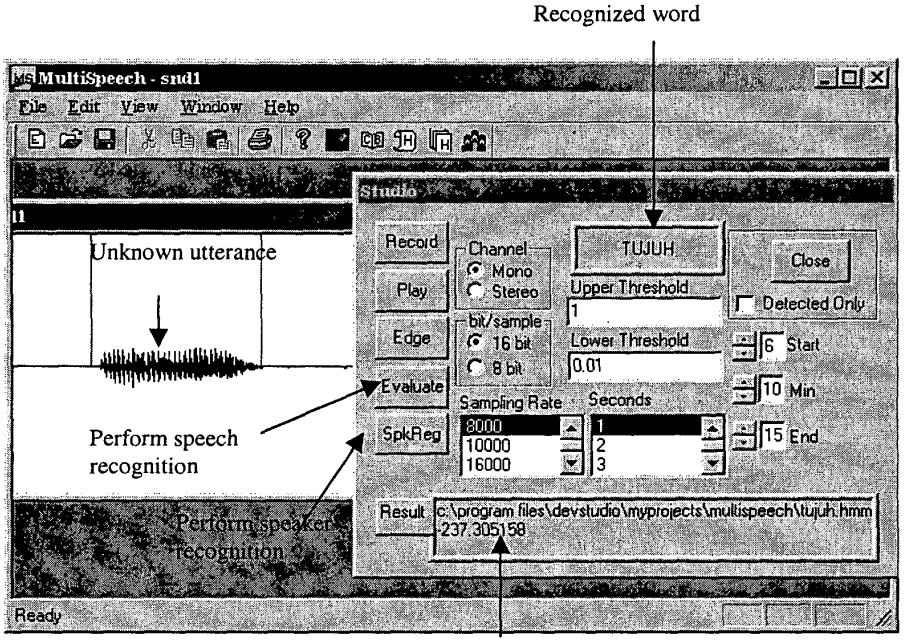


Figure 2.0 A screenshot showing the TMSR toolkit. The speech recognition module (right) and the sound wave recorded will be shown in the active document view screen (upper left)

2. Play - Play the sample in the active document
3. Edge- Perform edge detection on the active document
4. Evaluate- Perform speech recognition on the active Document
5. SpkReg-Perfrom speaker recognition on the active document.

The active document is the topmost document, the one highlighted. It is the document with the sound sample that the user wants to access. Besides the buttons, there are also the options that the user can select to configure his sound sample. The sound wave recorded will be shown in the active document view screen. After that, click on the 'Edge' button to let the program detect the start and end points of an utterance. The 'Play' button can now be pressed to playback the sound just recorded.

The other controls, such as the 'Upper Threshold', the 'Lower Threshold', the Start', the 'Min, and the 'End' are for the edge detection. By tweaking these values, the edge detection algorithm used by the program will detect different endpoints. To save the sample, just click on the 'Save'.

Assuming that the user has recorded samples for different words, each word repeating a few times, the codebook can then be build. Once all the input files have been determined, the name of the codebook must be entered in the 'Codebook Name' box. The 'LPC Order' and 'CEP Order' boxes are to set the LPC order and the Cepstrum order to be performed on the samples. The 'Stage' box indicates the LBG codebook stage to be created. If the user enter '8' means a 256-codewords codebook will be created. Once everything is set, just click the 'Build' button to start building the codebook.

After the codebook development, the next stage is to build the HMM models. Click on the 'Build HMM Model' button in the main screen and the dialog box will appear. When a number of HMM models is created, we are then ready to add the models into the vocabulary of the program. When all word models to be recognized are selected and moved into the 'Vocabulary' box, the program is ready to recognize all the words or speakers in the template.

III. SPEAKER RECOGNITION SYSTEM

Sampling of Speech

The implemented system has a feature for recording/playback sound in real time. The sampling frequency can be selected as either 8khz, 10khz, 16khz, 22.5khz or 44.1khz. The sample resolution can also be selected as either 8 bit or 16 bit per samples. After the sampling stage, a built-in adaptive edge detection algorithm is used to find the start and end points of an uttered word (since this system is meant to recognize isolated word). However, the user can also manually select these points.

Speech Feature Extraction

After the sampling stage, the user can then select either one of the two modes available in our system. The first is to train the system to recognize the word just recorded, and the second is to test the system. In either case, a feature extraction procedure is performed. The purpose of this is to condense and distill the important information of the speech signal. Any form of variability, which is important, must be extracted in order to keep the important characteristics of the uttered word, and the variability, which is not important, must be suppressed and eliminated. In this procedure, the utterance signal is divided into frames of 240 points, with each frame overlapping each other by 160 points shown in Figure 3.0.

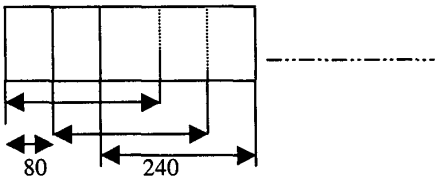


Figure 3.0 Overlapping of speech frames.

Next, each frame is passed through a first order high pass filter in order to spectrally flatten the signal. The high pass filter FIR equation is given by:

$$\bar{s}(n) = s(n) - as(n-1), \; where a = 0.95 \quad (1)$$

Then, each frame is windowed by a Hamming window:

$$\bar{x}(n) = x(n) \times w(n)$$

$$w(n) = 0.54 - 0.46\cos(\frac{2\pi n}{N-1}), \quad where \; N = 240 \quad (2)$$

After that, the Levinson-Durbin algorithm [1][2] is used to find the LPC coefficients of the each signal frame. After this, a P-order LPC vector is obtained. This P-order vector is then converted to a Q-order cepstral vector. The TMSR toolkit allow the user the freedom of setting the order of the LPC and cepstral coefficients. This will enable the user to experiment on the effects of different orders of LPC and cepstral coefficients on speech recognition accuracy.

Vector Quantization

From the previous speech feature extraction stage, a series of Q-order cepstral vectors, representing the whole utterance of speech have been sampled. The next stage is then to convert the vectors into a discrete set of symbols, which can be, used by the discrete HMM model. The method that was

used is the LBG (Linde-Buzo-Gray)[3][4][5] method of vector quantization.

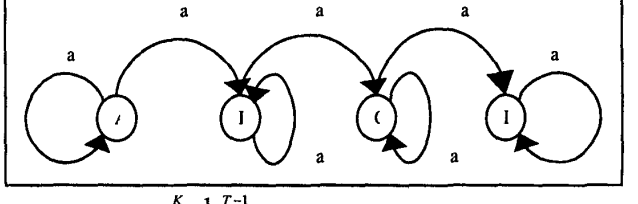
1. First, a codebook must be created before vector quantization can be performed. To build a codebook, a large set of training vectors X_i is needed.
2. For every time step t:
   A) For every training input X_i (i=1,2,3.... Q), calculate its distance to every codeword:
   B) Find the minimum distance and assign it to the codeword cluster

$$d(i,q)=\sqrt{\sum_{j=0}^{p}(W_q^{(j)}-X_i^{(j)})^2},\quad p=\textit{vector dimension},\ q=1,\ldots,2^M \qquad (6)$$

Hidden Markov Model (HMM)

In the training mode, the series of codeword indices obtained from each cepstral vector of each frame represents the uttered word. To recognize an unknown word, the system has to compare and evaluate that unknown word with word models stored in the system. In discrete HMM[1][6], each word model consists of states, with each state corresponding to a short period of time. In each state, there are discrete observations. A typical HMM model is shown below:

Figure 4.0 HMM model for Speaker Recognition.



in the system's vocabulary, is created using the Baum-Welch re-estimation formula on multiple sequences:
where α and β are the forward and backward variables associated with the HMM forward-backward procedure, while a_{ij} (the transition probability of state i to state j) and $b_j(l)$ (the probability of observing symbol l in state j) are the model parameters. P_k is the probability score of the k-th observation sequence $\mathbf{O}=\{O_1,O_2,...,O_T\}$ for time t=1,2,...,T based on the HMM model $\lambda=\{a, b, \pi\}$.

The user has the freedom to select the number of states for a word model that he wishes to create, and also select the inputs for training the model. The inputs are recorded repetitions of the same word utterance.

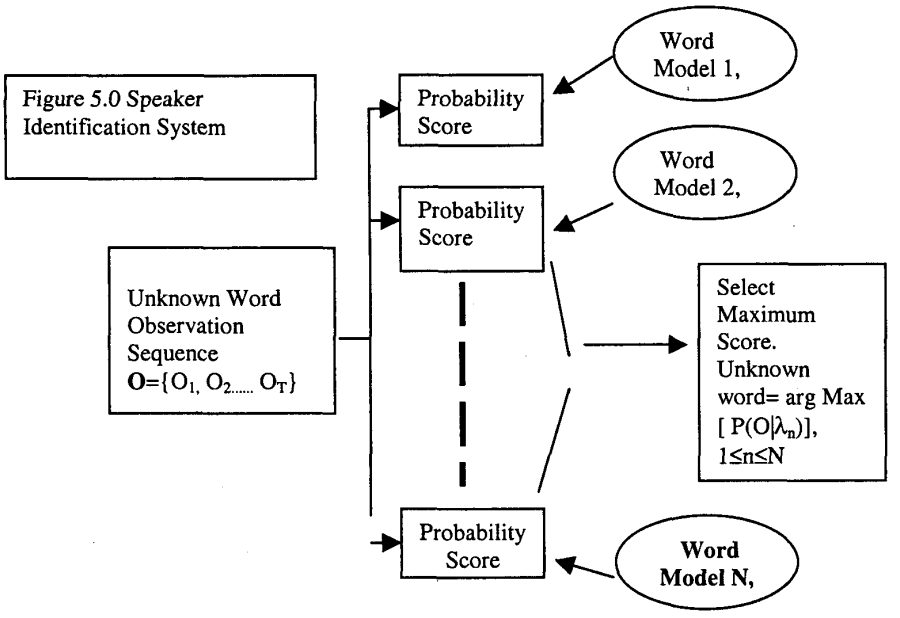
By creating an N-state HMM model for a speaker, a test observation sequence for the time period of T (produced after vector quantization of cepstral coefficient vectors), can be evaluated (recognized by the system) using the logarithmic Viterbi algorithm. P is the final probability score for the whole observation sequence. The logarithmic viterbi algorithm is chosen because it solves the problem of floating point underflow caused by the inability of the computer to calculate real numbers, which are too small.

$$\bar{a}_{ij}=\frac{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=1}^{T-1}\alpha_t^k(i)a_{ij}b_j(o_{t+1}^k)\beta_{t+1}^k(j)}{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=1}^{T-1}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)} \qquad (9)$$

$$\bar{b}_j(l)=\frac{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{\substack{t=1\\ s.t.\,o_t=V_l}}^{T-1}\alpha_t^k\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=1}^{T-1}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)} \qquad (10)$$

A HMM model can be concisely described by 3 model parameters: $\{\pi\}$, the initial state probability matrix; {a}, the state transition probability matrix; and {b}, the observation probability matrix. Each model, which represents each word

Probability scores are theoretically less than 1, and quite often, the scores are very small.



IV. SPEAKER IDENTIFICATION SYSTEM

Besides isolated word speech recognition, the system can also perform speaker recognition. Speaker recognition consists of speaker identification and speaker verification [7].

From the Figure 5.0, it is clear that the whole process is actually a sequence of Word Recognition using HMM:

1. First select client speaker 1's HMM models, and then perform Word Recognition on the unknown speaker's speech utterance. Using HMM, find the word to recognize the unknown utterance. If the correct word is recognized, record the probability score produced. If the word is recognized wrongly, then discard the score.

2. Now select client speaker 2's HMM models, and do the same process again. Repeat for all client speakers' models. Record all maximum probability scores associated with each client speaker if the correct word is recognized.

3. If there is at least 1 client speaker whose models correctly recognized the word spoken by the unknown speaker, we will then have at least 1 probability score in our list. From the list of scores, select the maximum. The client speaker whose probability score is the highest is identified as the unknown speaker. However, if there are no scores recorded, then the unknown speaker is then identified as 'not-in-the-list'.

Speaker Recognition Results

The next step is to evaluate the performance of the system in speaker recognition. There are 8 clients for speaker recognition, which are identified as speaker 1, speaker2 and so on. To find an Equal Error Rate (EER) threshold for each client speaker, first the distribution scores for each client speaker were found. Each client speaker will record his/her digit samples and then evaluate the test samples using the HMM models.

Scores for 10 repetitions of each digit are recorded for speaker1, speaker 2, speaker 5, speaker 6 and speaker 7, whereas only 5 repetitions of each digit are recorded for speaker 3, speaker 4 and speaker 8.

To find the distribution for the impostor speakers, 22 impostors were selected and each impostor will record 1 sample for each digit. Thus, there are 220 samples. For each client speaker, these 220 samples will be tested to find its probability score using that client speaker's models. For example, using speaker 1 digit models, the 220 samples were tested and the scores for each sample recorded. The process is repeated using speaker 2 models, then speaker 3, and so on. For each client speaker, there is 2 set of scores: his own scores and the impostors' scores.

The following results were acquired by testing the system with all the 8 client speakers' samples and the 22 impostors' samples. The top row is the identified speakers. The left is the test speakers from the population. For example, 9 out of 10 times, the test speaker '4' (left column) was identified as '4' (top row), and 1 time identified as '8' (top row).

For speaker recognition, the problem lies with insufficient testing data instead of training data. By assuming the distributions are Gaussiun, a very large number of samples are required to accurately represent the distribution. However, in the experiments, 1 sample per digit per speaker is the distribution of the impostor speakers.

Speaker Recognition Results

speaker	1	2	3	4	5	6	7	8	Rejected
1	10								
2		10							
3			10						
4				9				1	
5					10				
6						9		1	
7							10		
8								9	1
Others	3	3	3	2	7	18	10	10	164

The false acceptance rate is 28%, the false rejection rate is 1%, and the identification error (a client speaker is identified as another client) is 2%.

More importantly, the number of impostor speakers should be increased. The equal error rate threshold is set at equalizing the false acceptance and false rejection rate. Another important factor is that only a single digit is spoken by every test speaker for each time. In a real application, a sequence of digits is required to be spoken by a test speaker before he is identified, such as the person's PIN numbers. It is expected that this can decrease the false acceptance rate.

V. DISCUSSION

The HMM-based speaker recognition system achieved a false acceptance ratio of 28%, while the false rejection rate is 1%, and the identification error (a client speaker is identified as another client) is 2%. Overall, the system is a flexible one suitable for learning and research purposes on speech recognition.

TMSR is an experimental system and some additions are under way which include features extraction, pattern classification and extension of the offered services. The main technique for the recognition phase of the TMSR toolkit only covers the HMM, further development of the system will include other techniques such as Dynamic Time Warping (DTW) and Neural Network (NN). In the work presented, the system was described and the use of TMSR speaker recognition by the students is presented.

VI. REFERENCES

[1] Sheikh Hussain bin Shaikh Salleh (1993), "A Comparative Study of the Traditional Classifier and the Connectionist Model for Speaker Dependent Speech Recognition System", Universiti Teknologi Malaysia: Masters Thesis.

[2] Rabiner L, Juang B. H. (!993), "Fundamentals of Speech Recognition", Englewood Cliffs, New Jersey: Prentice Hall.

[3] Wu, Frank H. and Ganesan, Kalyan (1989) " Comparative Study of Algorithms for VQ Design using Conventional and Neural-net based approaches.", IEEE.

[4] Ashok K. Krishnamurthy, Ahalt, Stanley C., Melton, Douglas E., and Chen, Prakoon (1990) " Neural Networks for Vector Quantization of Speech and Images", IEEE Journal on Selected Areas in Communications, Vol. 6, No 6.

[5] Buck, Joseph T., Burton, David K., and Shore, John E. (1985) "Text Dependent Speaker Recognition Using Vector Quantization", Florida: ICASSP 85, Vol. 1.

[6] Rabinern L, Juang B.H. (1986), "An Introduction to Hidden Markov Models", IEEE ASSP Magazine.

[7] Soong F. K., Rosenberg, A. E., Rabiner L. R. and Juang B. H. (1985) "A Vector Quantization approach to Speaker Recognition", Florida: ICASSP Vol. 1.