Speaker Recognition based on Hidden Markov Model

Sheikh Hussain Shaikh Salleh, Ahmad Zuri Sha'ameri, Zulkarnain Yusoff,
Syed Abdul Rahman Al-Attas, Lim Soon Chieh, Ahmad Idil Abdul Rahman,
Shahirina Mat Tahir

Digital Signal Processing Lab
Faculty of Electrical Engineering
Universiti Teknologi Malaysia
81310 Skudai, Johor, Malaysia
Email : hussain@suria.fke.utm.my

Abstract: In this paper, a method for implementing speaker recognition system using the discrete Hidden Markov Model. This method uses a statistical approach in characterizing speech. The speech utterance is fit into a probabilistic framework, which consist of transition of states and discrete observable sequences. 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. 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%.

1.0 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 and security. 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. In addition, we have come up with a user-friendly and flexible speech recognition program that facilitates learning of HMM and speech recognition, as well as a tool for doing speech recognition experiments.

HMM-based speech recognition system has already been applied successfully in commercial products such as the IBM ViaVoice, Dragon NaturallySpeaking, L&H's Voice Express and Philips's FreeSpeech. However, none of the commercially available speech recognition products are made in Malaysia. Besides, there are also no speech recognition programs for learning and researching HMM recognition developed locally. Thus, we have come up with a user-friendly and flexible speech recognition program that facilitates learning of HMM and speech recognition, as well as a tool for doing speech recognition experiments.

2.0 SPEAKER RECOGNITION OVERVIEW

The system shown in figure 1.0 is built upon established speech recognition algorithms. The speech recognition system consist of the following blocks:

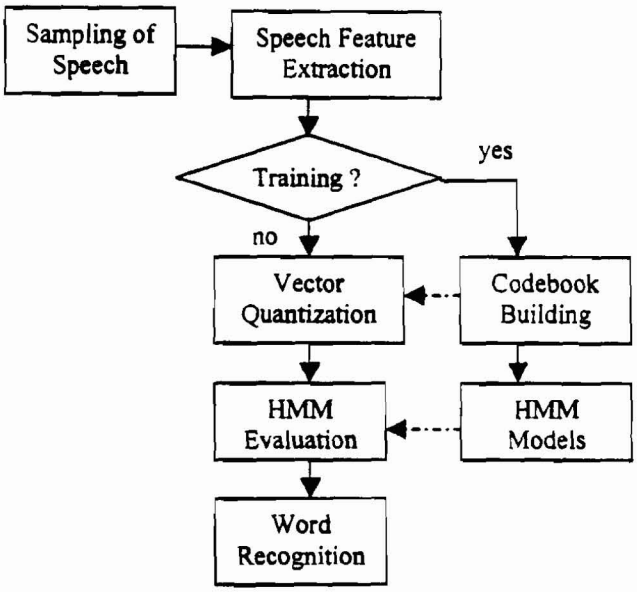

Figure 1.0 Speech recognition system

The implemented system have 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).

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. multiple subcomponents pulse sinusoid centered at time n_i samples and of pulse duration N_i samples. The individual signal component do not overlap in time and this ensured by the following conditions

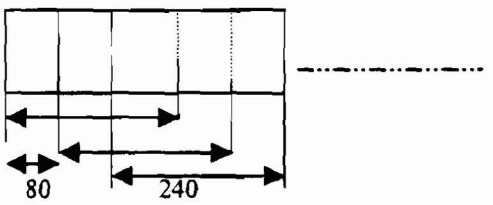

Figure 2.0 Overlapping of frames

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:

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

Then, each frame is windowed by a Hamming window:

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

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

After that, the Levinson-Durbin algorithm is used to find the LPC coefficients of the each signal frame. Assuming that a P-order LPC analysis is performed, the Levinson-Durbin algorithm is as follows:

a) For n=0, E_0=0

b) For step n,

$$k_n = \frac{-1}{E_{n-1}} \sum_{i=0}^{n-1} a_n - 1(i) r_{n-1}$$

$$a_n(n) = k_n$$

for i = 1 to n-1,

$$a_n(i) = a_{n-1}(i) + k_n a_{n-1}(n-i)$$

$$E_n = E_{n-1}(1 - k_n^2) \qquad (3)$$

where k_n is the n-th is the reflection coefficient, or PARCOR, and r_n is the n-th autocorrelation coefficient. a_n is the n-th LPC coefficient.

c) n = n+1, repeat step 2 until n=P

After this, a P-order LPC vector is obtained. This P-order vector is then converted to a Q-order cepstral vector (Q>P) using the conversion below:

$$C_0 = r(0)$$

$$C_i = -a_i$$

$$C_i = -a_i - \sum_{k=1}^{i-1} \frac{i-k}{i} C_{i-k} \alpha_k, \qquad i = 2,.....P$$

$$C_i = -\sum_{k=1}^{P} \frac{i-k}{i} C_{i-k} \alpha_k, \qquad i = P+1,....Q \qquad (4)$$

The Q cepstral coefficients is then passed through a cepstr weighting window given below:

$$\bar{C}m = Wm \times Cm$$

$$w_m = 1 + 0.5Q\sin(\frac{\pi n}{Q}), 1 \le m \le Q, \text{ where Q = 12} \qquad (5)$$

This will produce a Q-order cepstral vector for every signal frame of the sampled utterance.

In the system that has been built, the user has 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.

3.0 VECTOR QUANTIZATION

From the previous speech feature extraction stage, a series of Q-order cepstral vectors, representing the whole utterance of speech are obtained. 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 we used is the LBG (Linde-Buzo-Gray) 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. By using the algorithm below, the codebook is built:

For every training input X_i (i=1,2,3....Q), calculate its distance to every codeword and then find the minimum distance and assign it to the codeword cluster.

$$d(i,q) = \sqrt{\sum_{j=0}^{p} (W_q^{(j)} - X_i^{(j)})^2}$$

$$p = vector\ dimension,\ q = 1,.....,2^M \qquad (6)$$

The large training set of vectors are obtained by recording repetitions of the digits in our vocabulary (the ten Bahasa Melayu digits) and performing feature extraction to get the cepstral vectors.

After obtaining a codebook, any cepstral vector can now be quantized to produce a codeword index. The index is then used as a discrete symbol for the next stage, the HMM.

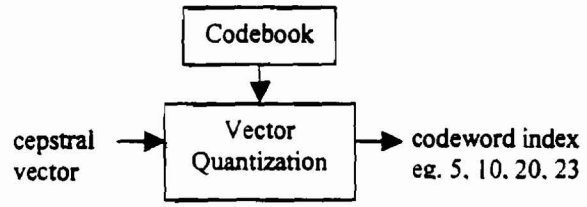

Figure 3.0 VQ system

In the system developed, there is a tool for creating VQ codebooks easily. In the training mode, The user can select the size of the codebook to be created, the inputs to the codebook builder, and also performing VQ on the inputs. The inputs are recorded speech utterances. In the testing mode, the user has to specify initially which codebook to use for quantizing the unknown speech input and the system will then automatically perform VQ every time an unknown speech input is recorded.

4.0 ISOLATED WORD RECOGNITION USING 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, 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:

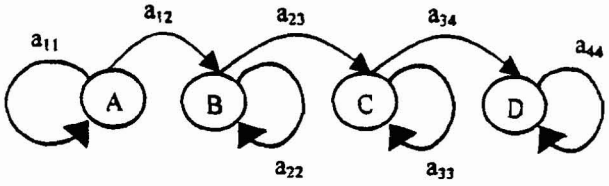

Figure 4.0 HMM model

An HMM model can be concisely described by 3 model parameters: {π}, the initial state probability matrix; {a}, the state transition probability matrix; and {b}, the observation probability matrix. Each model which represents each word in the system's vocabulary is created using the Baum-Welch re-estimation formula on multiple sequences:

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T-1} \alpha_t^k(i) a_{ij} b_j(o_{t+1}^k) \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T-1} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \qquad (7)$$

$$\bar{b}_j(l) = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1,\ s.t.\ o_t=v_l}^{T-1} \alpha_t^k \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T-1} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \qquad (8)$$

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 $O=\{O_1,O_2,...,O_T\}$ 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 to be created 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 word, 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:

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. Probability scores are theoretically less than 1, and quite often, the scores are very small. For example, a number which has a power of -300 is too small to be represented by the computer's floating point number

representation. However, by using the logarithm of the number, this problem is solved.

For each word model, probability score for the unknown observation sequence is computed. The word whose model produced the highest probability score for the test sequence is selected to recognize the unknown speech. The process for word recognition using HMM is shown below:

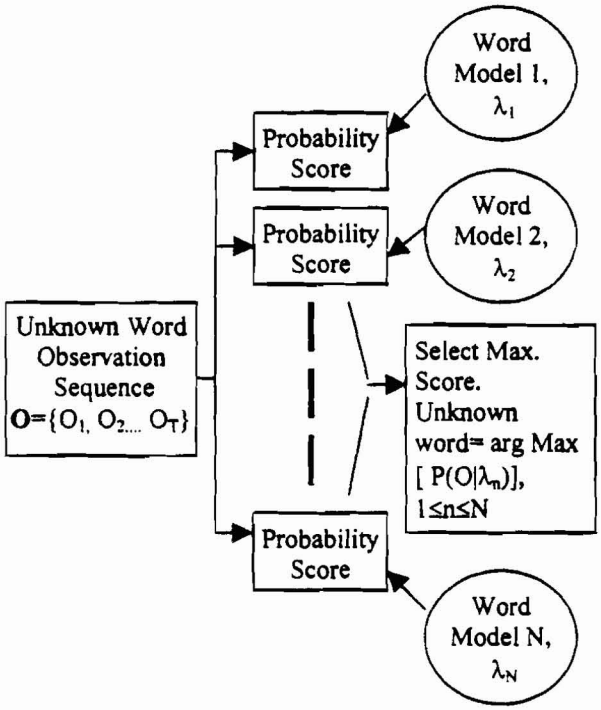

Figure 5.0 Word recognition using HMM

During the testing mode, the system will perform the recognition on the unknown word based on the word models stored in its vocabulary. The user can add or remove word models from the vocabulary. Thus, the system is flexible that it can contain a wide range of vocabularies, such as digit words, English words, Malay words and so on. The goal of the system's recognition algorithm is to find the best matching word model for the unknown word. Even if there is no right word in the vocabulary to identify the unknown word, the system will still select the nearest match. When such a match is found, it will display the result word and the probability score calculated using the Viterbi algorithm. This probability score can be used for further analysis by the user.

5.0 SPEAKER IDENTIFICATION

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

The process of identifying an unknown speaker is actually similar to word recognition, where the word uttered by the unknown speaker is compared against all the HMM models of the vocabulary of every client speaker in the system. The client speaker whose vocabulary contains the model that produces the highest probability score is identified as the unknown speaker. The algorithm is shown next:

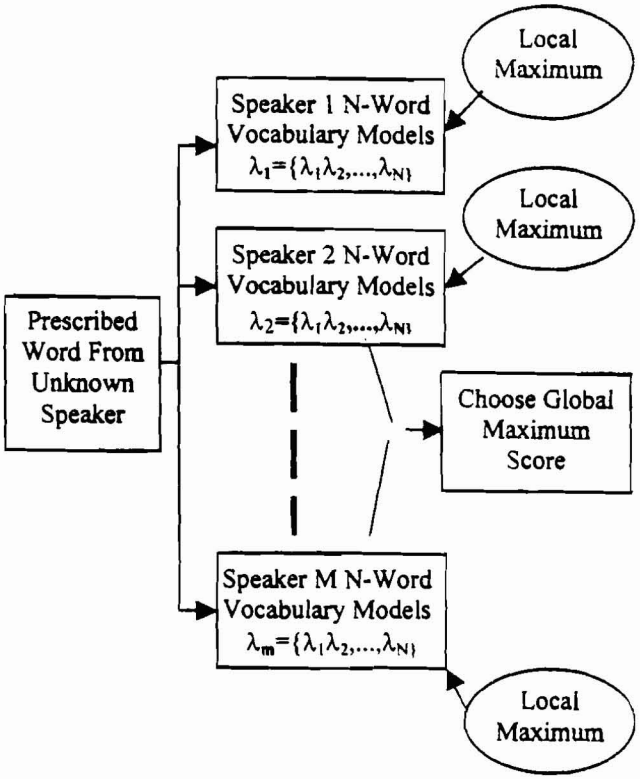

Figure 6.0 Speaker identification

From the figure, 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, 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'.

6.0 SPEAKER VERIFICATION USING EER

Now that a speaker is identified out from R speakers, this is still not enough to say that the identified speaker is really the client speaker. For example, there may be an impostor speaker (one that is not in the client speakers' list) , say A, which has characteristics of speech (intonation, style, sound quality) similar to that of one of the client speakers, say B. So, the speaker identification part might identify the impostor A with B, since speaker identification's objective is to choose the highest matching client speaker, in this case B.

In some cases, where security is concerned, this is unacceptable. Thus, speaker verification is meant to make a decision on whether to accept or reject this speaker. To decide, a threshold is used with each client speaker. If the unknown speaker's maximum probability score (after the speaker identification part) exceeds this threshold, then the unknown speaker is verified to be the client speaker (i.e., speaker accepted). However, if the unknown speaker's maximum probability score is lower than this threshold, then the unknown speaker is rejected completely.

The threshold is determined as follows:

1. For each speaker, evaluate all samples spoken by him using his own HMM models and find the probability scores. From the scores, find the mean μ_1 and standard deviation σ_1 of the distribution.

2. For each speaker, evaluate all samples spoken by a large number of impostors (typically over 20) using the speaker's HMM models and find the probability scores. From the scores, find the mean μ_2 and standard deviation σ_2 of the distribution.

3. For each speaker, calculate the threshold T_i for the i-th client speaker as below:

$$T_i = \frac{\mu_1\sigma_2 + \mu_2\sigma_1}{\sigma_2 + \sigma_2} \qquad (9)$$

Illustrate the idea, see the below graphs which represent the distribution of scores for both the impostors and the client speaker. We assume Gaussian distribution for the scores. The threshold T_i for each speaker, also called the equal error rate, is chosen to be an equal number of standard deviations away from each mean. The threshold is chosen so as to equalize the overlap area of the two distributions , thus equalizing the false acceptance ratio and the false rejection ratio (i.e., number of falsely accepted impostors = number of falsely rejected clients).

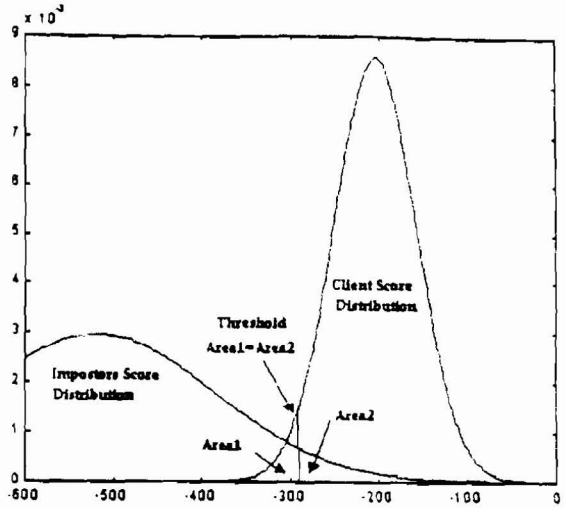

Figure 7.0 Distribution of scores and equal error rate threshold

In the system, each client speaker is represented by a speaker profile. In this profile, information such as the client speaker's name, VQ codebook, word models and speaker verification threshold are stored. Thus, a user has the discretion on selecting the client speakers and setting the verification threshold to suit his application.

7.0 EXPERIMENTAL RESULTS

Speaker Recognition Results

To evaluate the performance of the system in speaker recognition there are eight number of speakers involve as a clients for speaker recognition. To find an 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.

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's digit models, the 220 samples were tested and the scores for each sample recorded. The process is repeated using speaker 2, then speaker 3, and so on. For each client speaker, there are 2 set of scores: his own scores and the impostors' scores.

In the below table, the 1st column is the client speakers' names. The 2nd and 3rd columns represent the mean score and standard deviation for the client speakers' distributions. The 4th and the 5th columns represent the mean score and standard deviation for the impostors' distribution. The 6th column represent the threshold, T_i, or equal error rate, for each client speaker. The equal error rate is found by setting the false acceptance ratio equal to the false rejection ratio.

Table 1.0 Score Distribution And Equal Error Rate

	μ_1	σ_1	μ_2	σ_2	ERR T_i
Sp 1	-223.2	46.0	-495.7	123.9	-296.9
Sp 2	-237.2	52.9	-499.5	127.1	-314.3
Sp 3	-199.1	54.4	-500.4	140.2	-283.4
Sp 4	-204.3	46.5	-518.9	133.8	-285.5
Sp 5	-230.4	57.1	-475.5	123.9	-301.9
Sp 6	-299.8	71.8	-446.2	114.6	-356.2
Sp 7	-261.6	47.9	-451.7	105.2	-334.8
Sp 8	-181.7	40.2	-471.1	123.1	-270.6

* Sp - speaker

The interpretation of the above table is as follows: the higher (more positive) μ_1, and the lower (more negative) μ_2, the better. This is because the distribution of the client speaker is far apart from the impostor speakers. Also, the smaller the standard deviation for both distribution, the better. Then, it is very unlikely for the client speaker to be falsely rejected and the impostors to be falsely accepted. Good distributions (graphs with less overlapping) from the table are speaker 1, speaker 2, speaker 3, speaker 4, speaker 5, and speaker 8. Bad distributions (graphs with more overlapping) are speaker 6 and speaker 7.

Once the equal error rate thresholds are known, the speaker recognition experiment is then ready to carried out using the same client speakers and impostors. 1 sample of each digit from each speaker was recorded as the test samples. The client speaker models and codebooks used were the models trained from the speaker dependent, digit recognition experiment.

As mentioned before, there are 2 stages, the speaker identification stage, and the speaker verification stage. In the speaker identification stage, a sample from the test set is feed into the system, and the system will try to identify which client speaker's HMM model best match this test sample. When such a client speaker is found, the system is said to have identified the speaker of the test sample. The score produced by evaluating this test sample using the identified speaker's HMM models will be compared with his ERR threshold. If the score is higher than the threshold, then the test speaker is accepted. Else, the speaker is rejected.

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

Table 2.0 Speaker Recognition Results

	Sp 1	Sp 2	Sp 3	Sp 4	Sp 5	Sp 6	Sp 7	Sp 8	Reject
Sp 1	10								
Sp 2		10							
Sp 3			10						
Sp 4				9				1	
Sp 5					10				
Sp 6						9		1	
Sp 7							10		
Sp 8								9	1
Other	3	3	3	2	7	18	10	10	164

*Sp - speaker

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%.

8.0 CONCLUSION

Our 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%. One interesting fact to note is that the equal error rate threshold is iniatially set at equalizing the false acceptance and false rejection rate. However, the final results show that the false acceptance rate is much higher than the false rejection rate (28% against

1%). This result is explained by the fact that since the distribution is not accurately represented (caused by lack of training data), the threshold is also not set correctly. 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. Overall, the system is a flexible one suitable for learning and research purposes on speech recognition.

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

[4] Parson, Thomas W. (1986), "Voice and Speech Processing", USA: McGraw-Hill Book Company.

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

[6] 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 Ares in Communications, Vol 6, No 6.

[7] Oppenheim, Alan V. and Schafer, Ronald W.(1989) "Discrete-time Signal Processing", New Jersey: Prentice Hall.

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

[9] 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.

[10] Microsoft Corporation (1991) "Microsoft Windows Multimedia Programmer's Workbook", Redmond, Washington:

[11] Saito, Shuzo and Nakata, Kazuo (1985) "Fundamentals of Speech Processing", Tokyo: Academic Press.