

**People's Democratic Republic of Algeria**  
**Ministry of Higher Education and Scientific Research**  
**University M'Hamed BOUGARA – Boumerdes**



**Institute of Electrical and Electronic Engineering**  
**Department of Electronics**

Final Year Project Report Presented in Partial Fulfilment of  
the Requirements for the Degree of

**MASTER**

**In Electrical and Electronic Engineering**  
**Option: Telecommunications**

Title:

**Text Dependent Speaker Recognition**  
**Using HTK**

Presented by:

- **BOUGUERRA Sara**
- **SAHMI Nacera**

Supervisor:

**Dr. Abdelhakim DAHIMENE**

Registration Number:...../2016



## *Dedication*

*I dedicate this work to my beloved parents whom always help me and stand with me, I also dedicate it to my brothers Rayane and Lounes and my sister Rym, I don't forget my best friends Imene and Nesrine .*

*I want to give special appreciations and the best words to my husband Oussama, I dedicate this project to him and I thank him for his help and his precious advices.*

*Thank you.*

*BOUGUERRA Sara*



## *Dedication*

*I dedicate this work to my family.*

*A special feeling of gratitude to my sisters  
Malha, Fezia, Nabila, Fatiha & Ferroudja  
for standing beside me and help me.*

*I also dedicate it to my nephews and nieces.*

*I dedicate again this work to my friends  
who encourage me and support me, and I  
give special thanks to my best friend Ahlam.*

*Thank you.*

*SAHMI Nacera*



# ACKNOWLEDGEMENT

First and last,

*We thank **ALLAH** for all His blessing and strength that the gives us in  
completing this project*

*We would like to sincerely thank our supervisor Dr A.DAHIMENE, for his  
guidance and support throughout this project, and especially for his confidence  
in us.*

# TABLE OF CONTENTS

Dedication	I
Acknowledgement	III
Table of contents	IV
Abstract	VI
Introduction	1

## CHAPTER 01: SPEECH MECHANISEM

1.1 The process of speech production and perception in human beings	04
1.2 Generation of voiced and unvoiced sounds	06
1.3 Speaker-specific characteristics of speech	07

## CHAPTER 02: VOICE RECOGNITION USING MFCC's

2.1 Voice recognition principle	10
2.2 Mel-frequency Cepstrum	11
2.2.1 Steps in detail	13
2.2.2 Delta and energy	17

## CHAPTER 03 : HIDDEN MARKOV MODELS

3.1 Markov chain	20
3.2 Hidden Markov Model 'HMM'	21
3.2.1 HMM parameters	21
3.2.2 HMM Network topology	22
3.2.3 State-Time Trellis Diagram	23
3.3 Hidden Markov Model Building Blocks	23
3.3.1 Forward-Backward procedure	24
3.3.2 Viterbi Algorithm	25
3.3.3 Baum-Welch algorithm	26
3.4 The relationship between HMMS Feature vectors and Speech	27

## CHAPTER 04 : IMPLEMENTATION IN HTK

4.1 Experiment	31
4.2 Task Grammar	33
4.3 Creating Dictionary	34
4.4 Preparation of the recorded data	35
4.5 Coding the data	36
4.6 Creating monophone HMMs	38
4.7 Fixing the silence models	42
4.8 Recognizer Evaluation	44
4.9 Discussion	46

<b>Conclusion</b>	.....	<b>59</b>
<b>References</b>	.....	<b>60</b>

## APPENDIX A : AN OVERVIEW OF THE HIDDEN MARKOV MODEL TOOLKIT

<b>A.1 HTK software</b>	.....	<b>62</b>
<b>A.2 Available HTK Tools</b>	.....	<b>62</b>
<b>A.2.1 Data preparation tools</b>	.....	<b>63</b>
<b>A.2.2 Training tools</b>	.....	<b>64</b>
<b>A.2.3 Testing tools</b>	.....	<b>65</b>
<b>A.2.4 Analysis tools</b>	.....	<b>66</b>
<b>A.2.5 Standard HTK tool options</b>	.....	<b>67</b>
<b>A.3 HTK files</b>	.....	<b>67</b>
<b>A.3.1 Master Label File (MLF)</b>	.....	<b>67</b>
<b>A.3.2 Configuration File</b>	.....	<b>67</b>
<b>A.3.3 Script files</b>	.....	<b>69</b>
<b>A.3.4 HMM definition files</b>	.....	<b>69</b>

# ABSTRACT

*The objective of the project is to design a text dependent speaker recognition system using HTK, the HTK toolkit is used essentially for speech recognition, after understanding the theory of Hidden Markov Models, and the different processes HTK uses during recognition, the principle of Log likelihood probability, and the language Model are linked together to identify the speaker of an intended sentence chosen by the user and trained by the speaker wanted to be recognized.*

*After several experiments, many threshold options were used, till arrived to the kinds of threshold playing a major role in speaker identification, a method for choosing thresholds has been formulated and it was applied for 10 speakers.*

*The method for speaker identification worked successfully for all speakers, and the goal of recognizing a sentence that belongs to a particular speaker has been realized.*

# INTRODUCTION

The task of speech processing is being developed in the last years. Speech conveys different forms of information to the listener; one of them is the identification of a speaker.

Speaker recognition is the process of automatically recognizing who is speaking on the basis of speaker features, which involves speaker identification and speaker verification. In speaker verification, the voiceprint is compared to the speaker voice model registered in the database wanted to be verified. The result of comparison is a measure of a similarity (score) from which rejection or acceptance of a verified speaker follows. At the identification stage, the voiceprint is compared with model voices of all speakers in the database. The comparison results are measures of the similarity from which the maximal quality is chosen.

These systems can be further categorized as text dependant and text independent. By text-independence, it is meant that speaker can speak any utterance in a particular language, whereas in the case of text dependent systems the speaker is required to speak predefined pieces of text such as specific password.

Several analytical approaches have been applied to the task of speaker recognition, many of which originate from speech recognition. Hidden Markov Models is the one of most common approaches and it is the one considered in this project.

The HTK (Hidden Markov Model Toolkit) is a free and portable toolkit for building and manipulating hidden Markov models system (HMMs) ,developed at Cambridge university engineering department in 1989, and primary designed for speech recognition research. However speaker recognition has co-involved with this technology of speech recognition and speech synthesis because of the similar characteristics and challenges associated with each.

This project seeks to look at the use of hidden Markov models to perform speaker recognition in the text dependent domain. This approach to speaker recognition will be analyzed through the design and implementation of HMMs using hidden Markov model toolkit.



This project is divided into four chapter organized as follow:

- The first chapter describes briefly the process of speech production and perception in human beings.
- The second chapter deals with speech signal processing describing the various steps followed for feature vector extraction representing the acoustical characteristics of the speech signal using Mel-frequency cepstrum technique.
- The third chapter gives an overview on hidden Markov model theory
- The fourth chapter describes the structure of our speaker recognizer and the details of the implementation using HTK software, followed by the discussion of the results.
- Finally we end up with a conclusion along with some comments and suggestions.

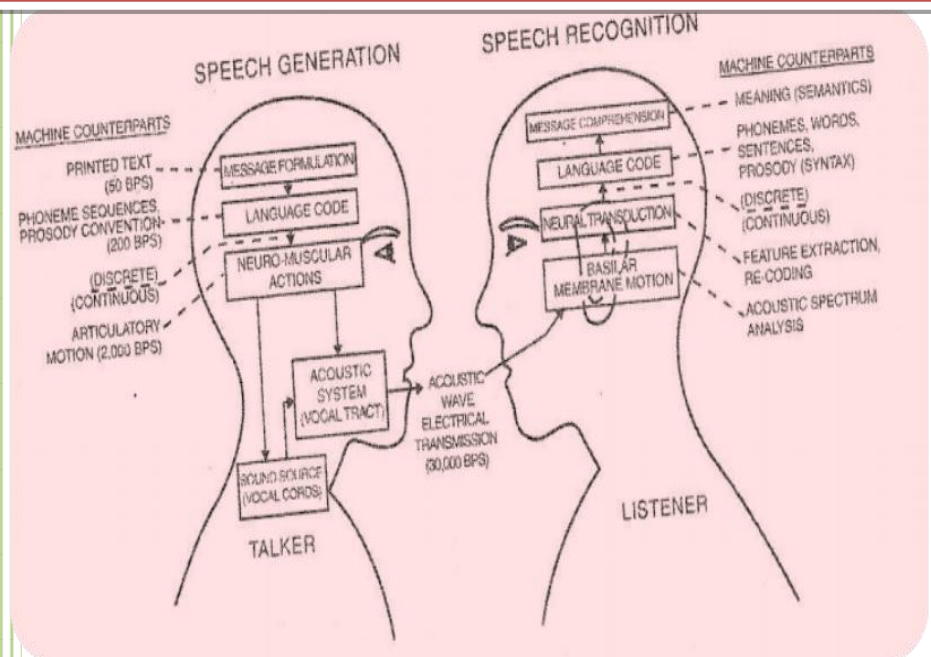
# CHAPTER 1

## SPEECH MECHANISM

**1.1 The process of speech production and perception in human beings**

**1.2 Generation of voiced and unvoiced sounds**

**1.3 Speaker-specific characteristics of speech**



Speech being the natural form of communication is the most basic and the most commonly used communication mean among human beings. For common people, it is just the sound waves coming out of the human mouth and perceived through ears. But there is a complex mechanism behind its production which will be dealt with in this chapter, as it is necessary for the development of our speaker recognition system.

### **1.1 The process of speech production and perception in human beings**

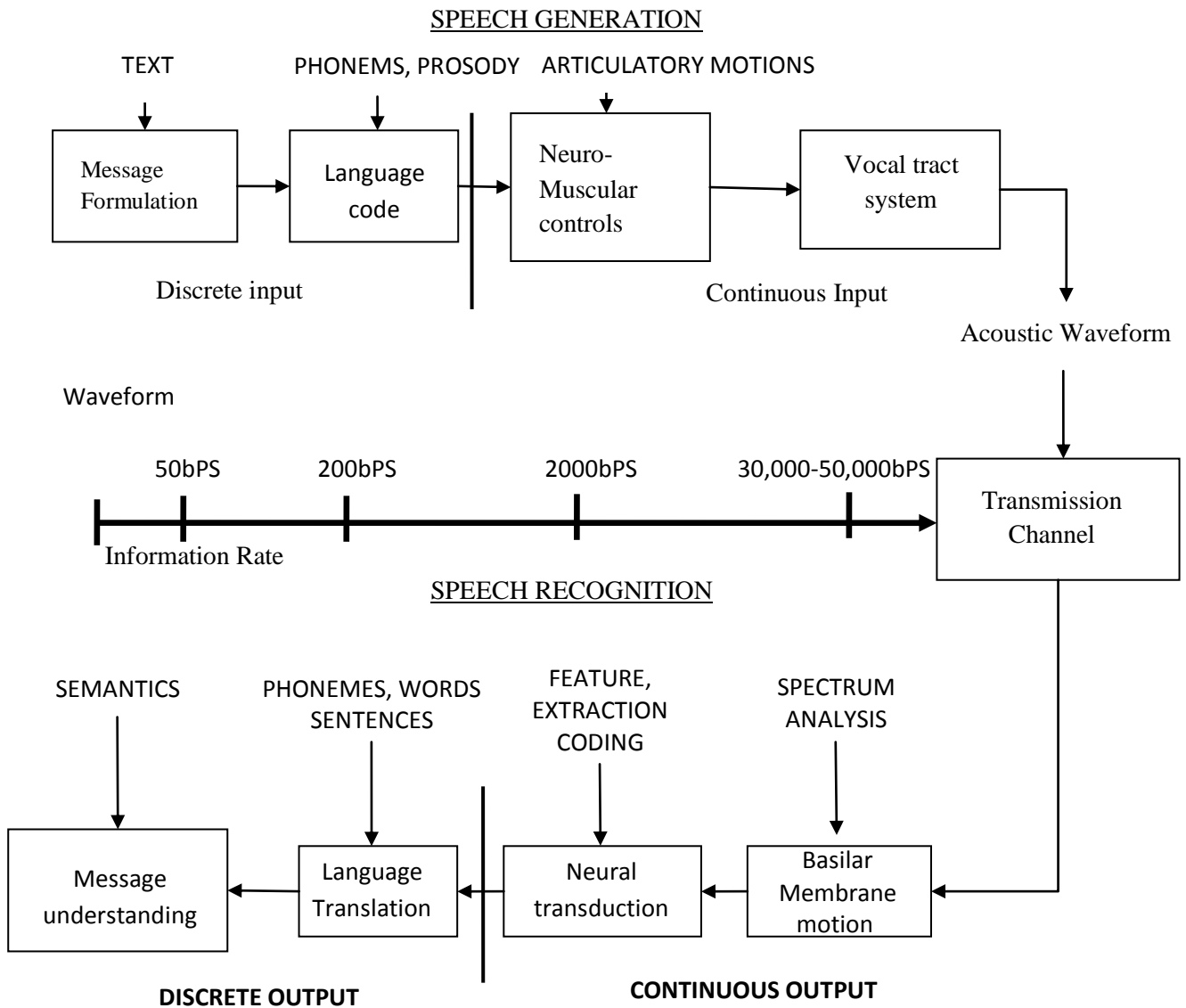
The process of speech production begins when the speaker formulates a message in his or her mind, to communicate with the listener via speech. The next step is the conversion of the message into a language code; this involves converting the message into a set of phonemes, comprising of correct sequence of words along with syntax, duration and loudness of sound. Once the language code is chosen, the speaker executes a series of neuromuscular commands cause the vocal cords to vibrate when appropriate and to shape the vocal tract such that the proper sequence of speech sounds is created, therefore producing an acoustic signal as the final output.

The neuromuscular commands must simultaneously control all aspects of articulation<sup>1</sup> including control of the lips, jaws, tongue, and velum.

Once the speech waveform is generated and propagated to the listener, the speech perception process begins by analyzing the acoustical signal along the basilar membrane in the inner ear, thus spectrum analysis is performed on the incoming signal. Then a neural transduction process extracts features such as duration of vocal cord vibration, intensity of the sound, resonant frequency of the vocal cord. And finally message comprehension is achieved.

---

<sup>1</sup>The movement of organs like tongue, lips, jaws to produce speech sounds, and can be affected by many factors such as health of muscles used to produce the sound or neurological damage.



**Fig 1.1** speech-production and speech perception process

**Fig1.1** shows speech production and perception process and basic information rate of signal at various stages of the process. The discrete symbol information rate is the raw message and is rather low of 50bps (bits per second). After the language code conversion the information rate rises to about 200bps. In the neuromuscular control level the representation of the information in the signal becomes continuous with an equivalent rate of about 2000bps. At neuromuscular control where the movement of articulators and its coordination with brain comes in picture and transfer of information takes place about 30-50kbps at the acoustic signal level.

For speech perception mechanism the information rate in the signal follows an inverse pattern of production process. Thus the continuous information rate at the basilar membrane is in the range of 30-50kbps. The higher level processing within the brain converts the neural signals to a discrete representation, which ultimately is decoded into a low-bit-rate message.[1]

## 1.2 Generation of voiced and unvoiced sounds

Air enters the lungs via the normal breathing mechanism. As the air is expelled from the lungs through the trachea, it causes the tensed vocal cords within the larynx to vibrate producing so-called voiced speech sound, and it results in speech waveform which is quasi-periodic in nature.

### Idea 1.2

The vocal cords are expressed as a simple vibration model, and the pitch of the speech changes according to adjustments in the tension of the vocal cords. When the vocal cords close, their vibration results in voiced sounds. When they are open, this vibration stops, and unvoiced sounds result.

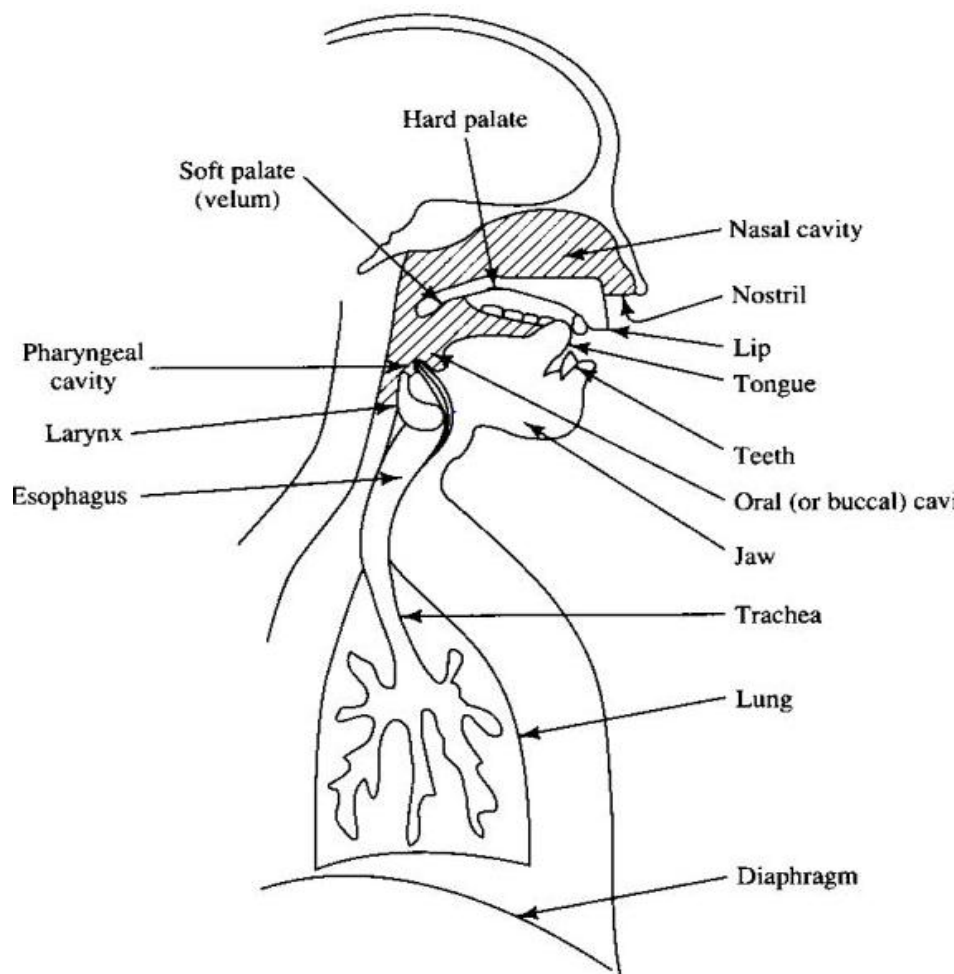


Fig 1.2 A schematic diagram of the human vocal mechanism

In the case of unvoiced sounds, the air flow pass through a narrow space formed by the tongue inside the month. A turbulent flow of air is produced and this is emitted as a noise-like sound, resulting in random speech waveform. [1]

The classification of the speech signal into voiced and unvoiced provides a preliminary acoustic segmentation of speech, which is important for speech analysis.

The analyses of speech signal is again very cumbersome due to its very fast changing parameters every 50 - 100 ms. For this speech analysis is performed in blocks or frames. These blocks are formed by windowing the speech signal for short duration for which the speech parameters are expected to be constant. Various methods are used for short time analysis of speech signal. In our project Mel Frequency analysis is used. [2]

### 1.3 Speaker-specific characteristics of speech

The speaker-specific characteristics of speech are due to differences in physiological<sup>2</sup> and behavioral aspects of the speech production system in humans. The main physiological<sup>2</sup> aspect of the human speech production system is the vocal tract shape, larynx sizes and other parts of the production organs that are unique to each person. Moreover each speaker has his or her characteristic manner of speaking including the use of a particular accent, rhythm, intonation, style, choice of vocabulary and so on. These differences that might go unnoticed by the human ear will be captured by the system in the form of signal statistics and then exploited to distinguish one speaker from another one. Speaker recognition systems use a number of these features in parallel, attempting to cover these different aspects and employing them in a complementary way to achieve more accurate recognition. [3][4]

#### Idea 1.3

Speech conveys different forms of information to the listener. Along with the basic information about the language being spoken and the emotion, gender and the identity of the speaker also is a part of information.

---

<sup>2</sup>**Physiology:** understanding of the higher order mechanisms within the human central nervous system that account for speech production and perception in human beings.



This chapter provides brief explanation of the speech production and perception mechanism in the human being, and illustrates how we can exploit the acoustic properties of the speech signal to identify the basic sound and the speakers, which help us in our recognition system.

# CHAPTER 2

## Voice Recognition Using MFCCs

**2.1 Voice recognition principle**

**2.2 Mel-frequency Cepstrum**



Automatic voice recognition technology is based on the digital processing of the speech signal, the voice is the sound produced by the vocal organs of a vertebrate, especially a human, and it conveys infinite information, to represent the voice signal, two digital processes are used: Feature Extraction and Feature Matching. This chapter covers one of the mostly used feature extraction technique which is Mel Frequency Cepstrum Coefficients (MFCCs). First, the principles of voice recognition will be covered ,then the MFCC method is analyzed by explaining the various steps often used to obtain the coefficients.

## 2.1 Voice recognition principle

To perform speech analysis, an input signal is taken from a microphone then the following operations are performed: Framing, windowing, Mel Cepstrum analysis and Matching (recognition) of the spoken word.

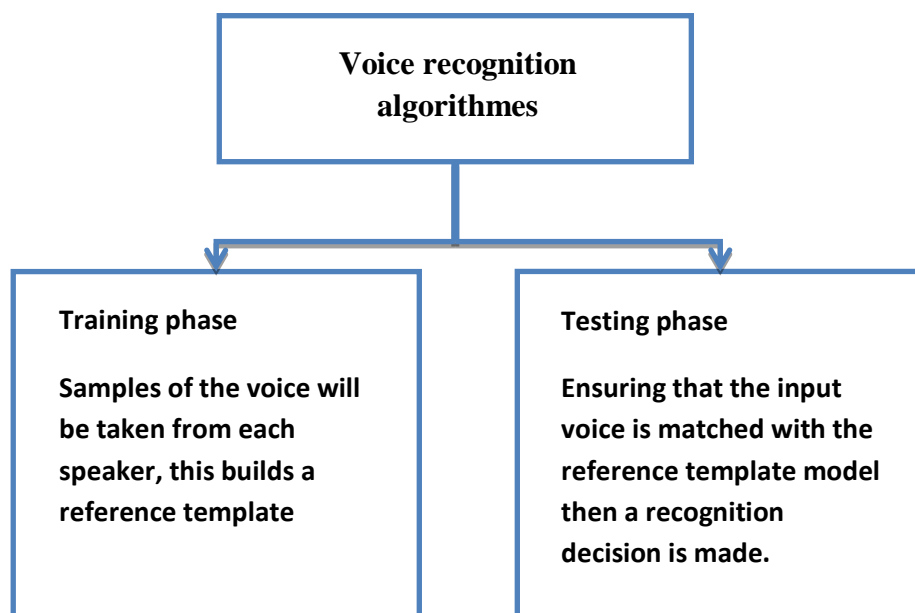


Fig 2.1 Voice recognition algorithms

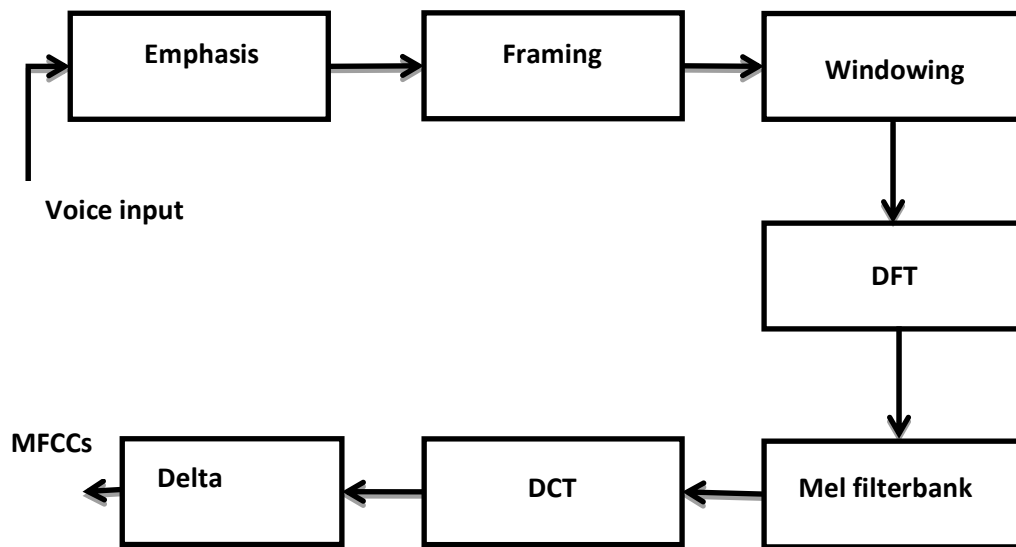
### Idea 2.1

The training phase is the process of familiarizing the system with the voice characteristics of speakers registering, each speaker gives samples of its speech then the system builds a reference model for him after that comes the testing phase which is the actual recognition task, in this session an input voice is entered to the system then it is directly matched to the reference model stored before then a recognition decision is made.[5]

## 2.2 Mel-frequency Cepstrum

Mel frequency Cepstrum is a representation of the short-termed power spectrum of a sound, MFCCs are the Mel Frequency Cepstrum Coefficients form the Cepstrum which is obtained by taking the inverse Fourier transform (IFT) of the logarithm of the spectrum of the signal.

The following bloc diagram shows the different steps to obtain MFCCs.



### Idea 2.2

The sounds that the human generates are filtered by the vocal tract (tongue, teeth,...) the role of the MFCCs is to represent the envelope that the vocal tract forms in the short time power spectrum of the voice signal.[5]

Fig 2.2 Block diagram showing the different steps.

The first step is Sampling of the signal at a given frequency  $f_s$ , and emphasize higher frequencies, this is used to increase the energy of the signal at high frequencies.

We suppose our signal is  $v[n]$

$$x[n] = v[n] - \alpha v[n-1]; \quad 0 < \alpha \leq 1 \quad (2.1)$$

$\alpha$  is the pre – emphasis coefficients

The general steps to calculate the MFCCs can be summarized as follows:

1. Framing the signal into 20-50ms frame.
2. Finding the DFT then the periodogram estimate of the power spectrum.
3. Applying Mel filter-bank to power spectra, then for each filter an estimate of the energy is determined.
4. Taking Log of the filter-bank energies.
5. Computing the DCT of the resulting signal from step 4.

The Mel scale is used to match the human perception to voice, the human ear differentiates small frequencies better than higher ones, this formula represents the relationship between the Mel scale and frequency

$$M(f) = 1127 \ln\left(1 + \frac{f}{700}\right) \quad (2.2)$$

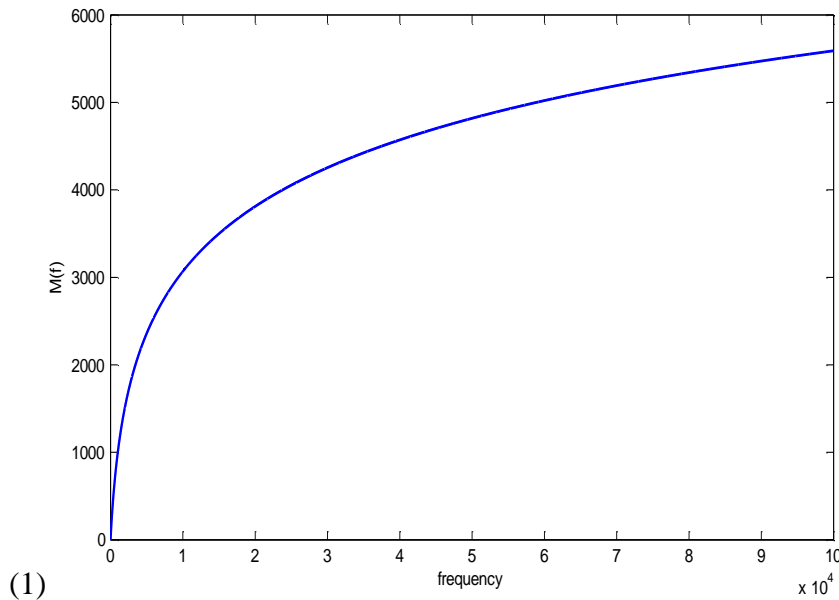


Fig 2.3 Graph of the Mel scale in terms of frequency

### Idea 2.3

The first frame has  $N_w$  samples, the second frame occurs after a frame skip duration that is  $N_s$ , so the second frame begins after the first frame by  $N_s$  samples, this means that they overlap by  $N_w - N_s$  samples.[6]

From the figure above it is noticed that as the frequency increases the slope of the graph increases, means that the distinction between adjacent frequencies becomes weak.

### 2.2.1 Steps in detail

Framing: the signal is framed at a suitable value, generally from 20ms to 50ms, the frame length is measured in samples:

$$\text{Frame length(samples)} = \text{Sampling rate} \times \text{frame duration}$$

$$N_w = f_s \times T_w$$

Fig 2.4 shows the different parameters used in the encoding process applied to speech, framing is within this operation.

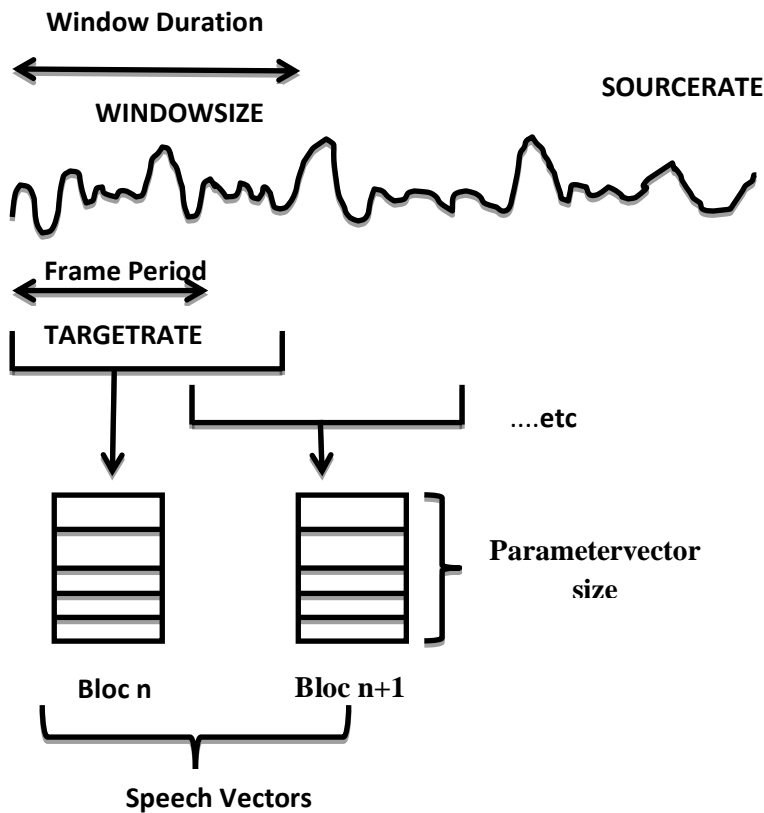


Fig 2.4 Speech encoding process [7]

Let's call the time domain signal  $x[n]$ , the framed signal or the signal  $x[n]$  for the frame length is denoted as  $x_i[n]$

The Hamming window is applied to  $x_i[n]$  let's call  $s_i[n]$  the resulting signal.



$$s_i[n] = x_i[n] \times h[n] \quad (2.3)$$

$$h[n] = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) \quad (2.4)$$

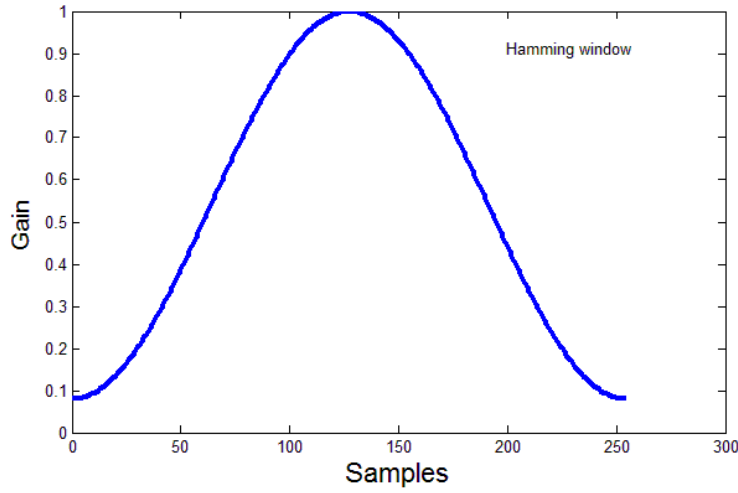


Fig 2.5 Hamming window

DFT of the signal  $s_i[n]$  is computed.

$$S_i[k] = \sum_{n=1}^K s_i[n] e^{-\frac{2\pi kn}{N}} \quad 1 \leq k \leq K \quad (2.5)$$

$K$  is the length of the DFT.

Now the Periodogram estimate of the power spectrum is calculated.

$$P_i[k] = \frac{1}{N} |S_i[k]|^2 \quad (2.6)$$

Now we perform an  $M$  point FFT and keep only the first  $\left(\frac{M}{2} + 1\right)$  coefficients. So that  $M = 2^{\text{next power of 2 of } (N_w)}$

To compute the Mel spaced filter-banks, an upper and lower frequency are chosen, then they are converted to Mel scale (**equ 2.2**), a number of filter banks is determined (let's say 25), for an  $N$  number of filter banks  $N+2$  points are needed, it is to find  $N$  frequency value between  $q_{min}$  and  $q_{max}$  ( $q$  is the frequency in Mels) the  $N$  Mel values are found using the following procedure:

$$q_k = q_{min} + bx \quad b \leq N + 1$$

$$x = \frac{q_{max} - q_{min}}{N + 1} \quad (2.7)$$

$$b = 1, 2, \dots (N + 1)$$

Then the result is  $N$  points linearly spaced between  $q_{min}$  and  $q_{max}$ , the spacing between each point and the other is  $x$ .

Now the resulting Mel scale values are converted to Hertz, and the FFTbins are computed for a given frequency  $f_i$ , the result is  $F_i$ .

$$F_i = \text{floor}\left[\frac{(M \times f_i)}{2f_s}\right] \quad (2.8)$$

$f_s$  sampling rate

Now filter-banks are created using the following function of  $F_i$

$$H_i(k) = \begin{cases} 0 & k < F_{i-1} \\ \frac{k - F_{i-1}}{f_i - f_{i-1}} & F_{i-1} \leq k \leq F_i \\ \frac{F_{i+1} + k}{F_{i+1} - F_i} & F_i \leq k \leq F_{i+1} \end{cases} \quad (2.9)$$

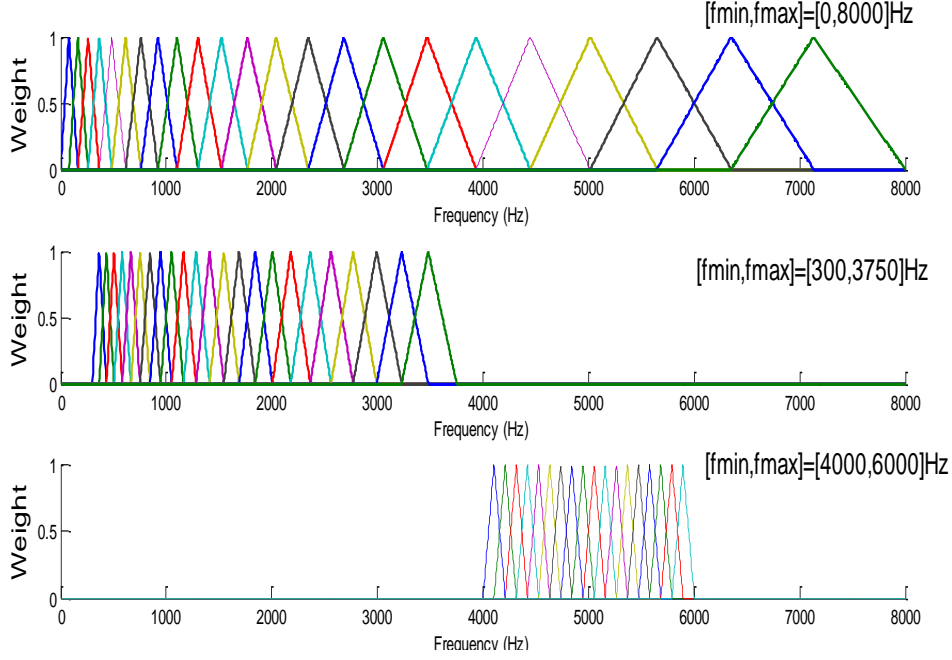


Fig 2.6 Mel Filter banks with different frequency ranges.

$f_i$  is the Mel frequency for  $0 < i \leq N+2$ , the number of filter-banks that can be used is [20-40].

To calculate filter-bank energies, each filter-bank  $H_i(k)$  is multiplied with the periodogram estimate of the power spectrum, and then coefficients are added up, this results in  $N$  numbers giving us an idea of the energy in each filter-bank.

$$B_i[k] = H_i[k] \times P_i[k]$$

1. The Log for each filter-bank energy computed above is calculated.

$$m_i = \log(B_i)$$

2. Finally the Discrete Cosine Transform is taken for each log energy computed above.

The log energy for each filter-bank is  $m_i$  then

$$c_j = \sqrt{\frac{2}{N}} \sum_{i=1}^N m_i \cos\left(\frac{\pi j}{N} (i - 0.5)\right) \quad (2.10)$$

Finally the coefficients  $c_j$ s so that  $j = \{1, 2, \dots, N\}$ , are called Mel Frequency Cepstral Coefficients (MFCCs), the needed coefficients are the first 12 coefficients.[9]

To obtain the Cepstrum  $c[n]$  of the signal  $x[n]$ , DFT of the Log of the power spectrum is taken.[10]

$$c[n] = \sum_{n=0}^{N-1} \left( \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi n}{N}} \right) e^{-j \frac{2\pi n}{N}} \quad (2.11)$$

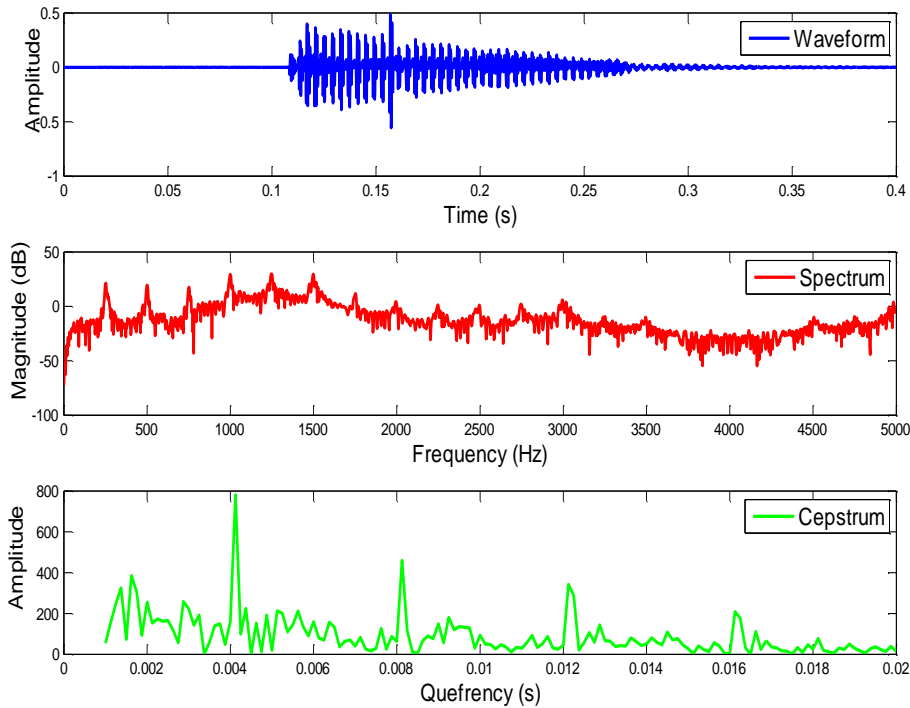


Fig2.7 Audio waveform (letter a) with its Spectrum and Cepstrum

### 2.2.2 Delta and energy

The previous steps resulted in 12 cepstral coefficients for each frame, now the 13<sup>th</sup> feature is added, it is the energy of the signal within the frame, energy changes as the phone changes, for stops like the sounds **p** or **k** (see **idea 2.4**) the energy is low compared to its value when uttering sibilants like **s** or **z**.

In the HTK toolkit the Energy is computed as the Log of the energy of the signal  $x[n]$  in  $N$  samples of a frame  $i$ .

#### Idea 2.4

**Stops** are sounds where there is an occlusion of the vocal tract, in this case there is no nasal air flow so the air flow stops immediately, stops in English include voiceless sounds: **p, t, k** and voiced sounds like **b, d, g**.

**Sibilants** are a kind of fricative sounds producing noisy airflow, the airflow in sibilants generates a groove in the tongue toward the teeth, this happens in the sounds **s** and **z**. [8]

$$E = \log \sum_{n=1}^N x_i[n]^2 \quad (2.12)$$

In reality, speech is not constant from frame to frame, in between, phones can exist and one may lose acoustic informations, this problem is solved by adding 12 delta coefficients (velocity features) and 12 acceleration (double-delta) features. The delta feature for the cepstral value  $c_t$  at a sample of time  $t$  is computed by the following regression formula.

$$d_t = \frac{\sum_{\theta=1}^{\emptyset} \theta (c_{t+\theta} - c_{t-\theta})}{2 \sum_{\theta=1}^{\emptyset} \theta^2} \quad (2.13)$$

$\emptyset$  is set by the user in configuration, if the above formula is applied to delta coefficients ( $d_t$ ) then acceleration coefficients are obtained (12 are needed), the delta energy coefficient is obtained by replacing  $c_t$  (the cepstral coefficient) by the energy of the frame, 2 more coefficients are needed the delta energy coefficient and the double delta energy coefficient.

Now the following feature coefficients are calculated

- 12 mel frequency cepstral coefficients
- 12 delta cepstral coefficient
- 12 double delta cepstral coefficient
- 1 delta energy coefficient.
- 1 energy coefficient.
- 1 double delta energy coefficient.

Now each frame has 39 coefficients having the spectral features of the signal, these coefficient constitute what we call the **feature vector**. [10]

# CHAPTER 3

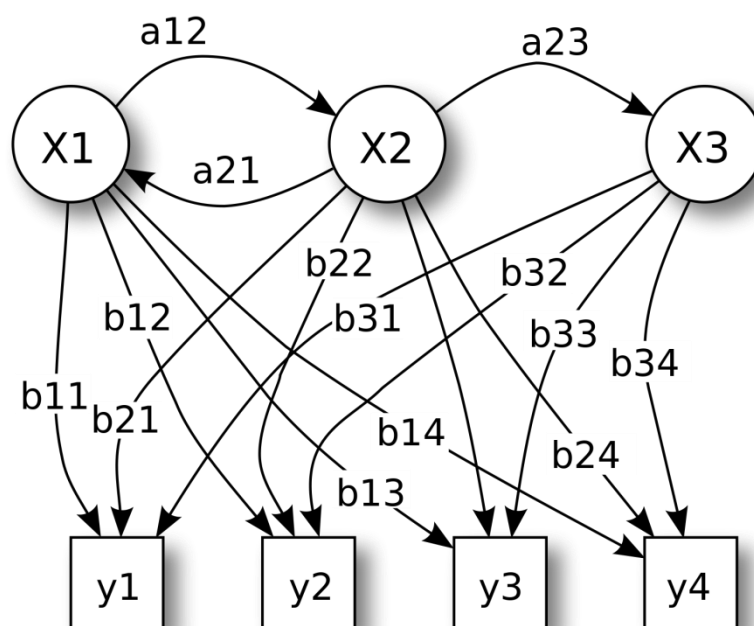
## HIDDEN MARKOV MODELS

### 3.1 Markov chain

### 3.2 Hidden Markov Model 'HMM'

### 3.3 Hidden Markov Model Building Blocks

### 3.4 The relationship between HMMS Feature vectors and Speech





This chapter provides the necessary information to understand the mechanism used for speaker recognition problem. To start, an introduction to the theory of hidden Markov model will be discussed as well as the basic algorithms used for the training and decoding process. In addition, state time trellis diagram will be discussed due to its use in finding the best path.

### 3.1 Markov chain

A Markov chain is a process that consists of a finite number of states, with transitions among these states. A symbol is associated with each state and a probability is associated with each transition; such that the probability of being in a particular state depends only on the previous state.

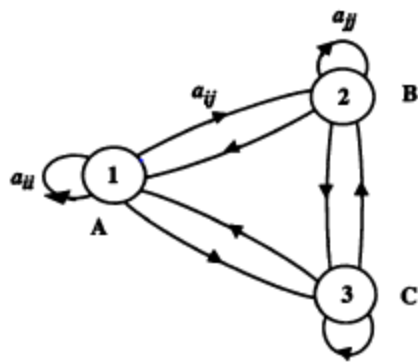


Fig 3.1 A three-state chain

Fig 3.1 shows a three-state Markov chain, with transition probabilities  $a_{ij}$  between states  $i$  and  $j$ . The symbols A, B and C are associated to states 1, 2 and 3 respectively.

For example the transition from state 1 to state 2 generates symbol B as output. These symbols are called output symbols because Markov chain is thought of as a generative model; it outputs symbols as transition from one state to another.

Now, given a sequence of output symbols generated by a Markov chain, we can retrace the corresponding sequence of states completely and unambiguously. An example of symbol sequence B A A C B B A C C C A is

#### Idea 3.1

In Markov model, the probability of each symbol depends only on the preceding symbol not on the entire previous sequence.

produced by transitioning into the following sequence of states: 2 1 1 3 2 2 1 3 3 3 1.

Therefore in Markov chain the transitioning from one state to another is probabilistic, but the generation of the output symbols is deterministic. [11]

### 3.2 Hidden Markov Model ‘HMM’

In the Markov model each state corresponds to one observable event (output symbol). But this model is too restrictive to be applicable to many problems of interest. Therefore Hidden Markov concept extends the model to include the case where the observation is a probabilistic function of the state. Hence for each state a probability distribution is defined, that specifies the probability for every observation symbol to be generated in that particular state. As each state can generate each observation symbol, it is no longer possible to see which state sequence generated an observation sequence as was the case for Markov models, the states are hidden, and hence the name of the model came.

#### Idea 3.2

HMM consists of two stochastic processes. The first stochastic process is a Markov chain that is characterized by states and transition probabilities. The second stochastic process produces emissions observable at each moment, depending on a state-dependent probability distribution.

#### 3.2.1 HMM parameters

A Hidden Markov model can be defined by the following parameters:

- Number of states in the model  $N$ .
- Number of distinct observation symbols  $M$ .
- Observation symbols  $V = \{v_1, v_2 \dots v_M\}$

States will usually be indicated by  $S_i, S_j$ , a state that the model is in at a particular point in time will be indicated by  $q_t$ . Thus  $q_t = S_i$  means that the model is in state  $S_i$  at time  $t$ .

- A probability distribution of transition between states  $A = \{a_{ij}\}$ , where

$$a_{ij} = P(q_{t+1} = S_j | q_t = S_i) \quad 1 < i, j < N \quad (3.1)$$

- An observation symbol probability distribution  $b = \{b_j(k)\}$  in which

$$b_j = P(o_t = v_k | q_t = S_j) \quad 1 < k < C; 1 < j < N \quad (3.2)$$

- The initial state distribution  $\pi = (\pi_i)$  where

$$\pi_i = P(q_1 = S_i) \quad 1 \leq i \leq N \quad (3.3)$$

To indicate the complete parameter a set of a model the notation  $M = (A, B, \pi)$ . [12]

### 3.2.2 HMM Network topology

An HMM topology consists of the number of states with varied connections between states which depend on the occurrence of the observable symbol sequences being modeled

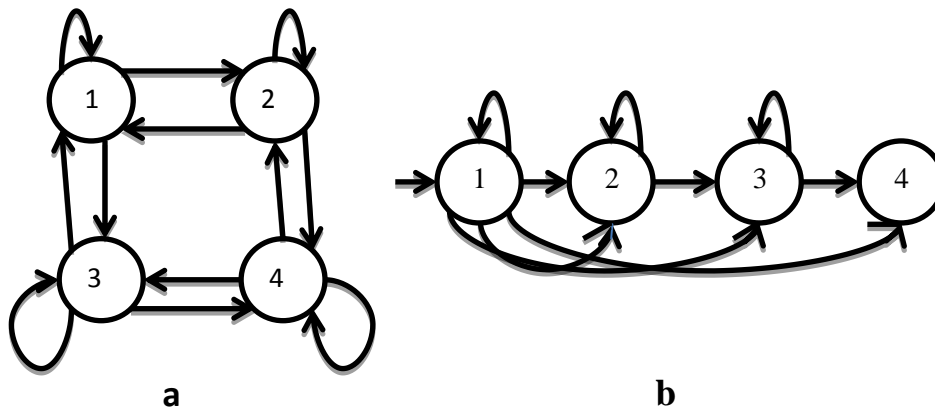


Fig 3.2: a) Ergodic model b) left to right model

The topology of **Fig 3.2.a** is called an ergodic network, since any state can be reached from any other state. This topology is inappropriate for speech recognition because speech consists of an ordered sequence of sounds.

Generally left-to-right model is used **Fig 3.2.b**, where a state cannot be revisited once it is left, a transition can only stay in the same state or go to a higher numbered state. Consequently the transition matrix  $A$  is upper triangular. [13]

### 3.2.3 State-Time Trellis Diagram

A state-time trellis diagram shows the HMM states with all the different paths that can be taken through various states as time unfolds. **Fig 3.3** illustrate a 4-state HMM and its state-time diagram. Since the number of states and the state parameters of an HMM are time-invariant, a state-time diagram is a repetitive and regular trellis structure.

For a left-right HMM the state-time trellis has to diverge from the first state and converge into the last state. There are many different state sequences that start from the initial state and end in the final state. Each state sequence has a prior probability that can be obtained using Markov property.[14]

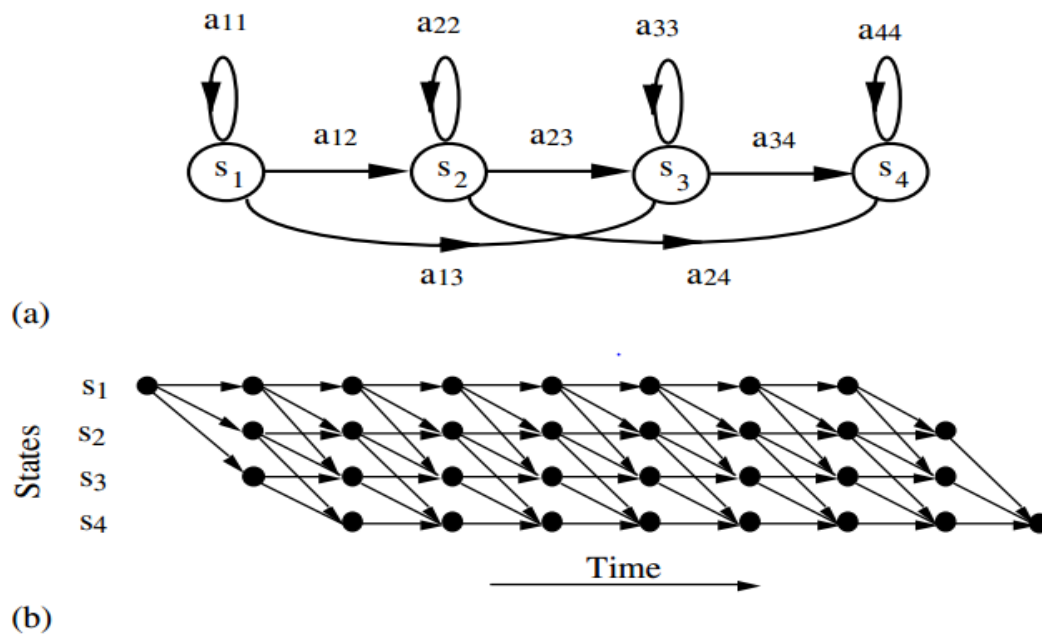


Fig 3.3: a) A 4-state left-right HMM, b) its state-time trellis diagram

### 3.3 Hidden Markov Model Building Blocks

There are three basic HMM problems in finding the probability of speech feature vectors generated from an HMM. Firstly, evaluation, which finds the probability that a sequence of visible states was generated by the model  $M$  and this, is solved by the Forward-Backward algorithm. Secondly, decoding finds state sequence that maximizes probability of observation sequence using Viterbi algorithm. Lastly, training which adjusts model

parameters to maximize probability of observed sequence. This last step is simply a problem of determining the reference speaker model for all speakers. The Baum-Welch re-estimation procedure is used for this case.

### 3.3.1 Forward-Backward procedure

Consider the sequence of observation  $\mathbf{O} = (O_1 O_2 \dots O_T)$  and the forward variable  $\alpha_t(i)$  defined as:

$$\alpha_t(i) = P(\mathbf{O}_1 \mathbf{O}_2 \dots \mathbf{O}_t, q_t = S_i | \mathbf{M}) \quad (3.4)$$

i.e. The probability that the model has generated the partially observed sequence  $O_1 O_2 \dots O_t$  at current state  $S_i$ . This variable  $\alpha_t(i)$  can be solved inductively by the forward procedure:

$$1. \text{ Initialize } \alpha \text{ at time } t=1: \alpha_1(i) = \pi_i b_i(O_1) \quad (3.5)$$

For  $1 \leq i \leq N$ .

$$2. \text{ Induction step: } \alpha_{t+1}(j) = [\sum_{i=1}^N \alpha_t(i) a_{ij}] b_j(O_{t+1}) \quad (3.6)$$

For  $1 \leq t \leq T-1; 1 \leq j \leq N$

Once the variable  $\alpha_t(i)$  is calculated then  $P(\mathbf{O} | \mathbf{M})$  is the sum of this forward variable for the final observation over all the states.

$$P(\mathbf{O} | \mathbf{M}) = \sum_{i=1}^N \alpha_T(i) \quad (3.7)$$

This method is commonly known as the **forward algorithm**.

Another way is to work backward from the ending state. The backward variable can be defined similar.

$$\beta_t(i) = P(\mathbf{O}_{t+1} \mathbf{O}_{t+2} \dots \mathbf{O}_T | q_t = S_i, \mathbf{M}) \quad (3.8)$$

i.e. probability of the sequence of observation after time  $t$  given state  $S_i$  at time  $t$  and the model  $\mathbf{M}$ . again  $\beta_t(i)$  can be solved inductively by the backward procedure:

$$1. \text{ Initialize } \beta \text{ at time } t=T: \beta_T(i) = 1 \quad (3.9)$$

#### Idea 3.3

Given HMM model and observation sequence, forward-backward algorithm finds the probability that the sequence comes with this model.

In speaker identification test, each speaker has a model, and identification is accomplished by finding the best fitting model as determined by the highest probability.

#### Idea 3.4

Forward algorithm starts with the first observation  $O_1$  and work forward through the data sequence until the end reached. This produces a probability which is the sum over all possible state sequence evaluated at  $\mathbf{O} = \mathbf{O}_T$

For  $1 \leq i \leq N$

$$2. \text{ Induction step: } \beta_t(i) = \sum_{j=1}^N a_{ij} b_j(\mathbf{O}_{t+1}) \beta_{t+1}(j) \quad (3.10)$$

For  $t = T-1, T-2, \dots, 1; 1 \leq i \leq N$

Another efficient way of computing  $P(\mathbf{O}|\mathbf{M})$  is [15]:

$$P(\mathbf{O}|\mathbf{M}) = \sum_{i=1}^N [\pi_i b_i \beta_1(i)] \quad (3.11)$$

### 3.3.2 Viterbi Algorithm

Given the HMM  $\mathbf{M}=(\mathbf{A}, \mathbf{B}, \pi)$  and the observation sequence

$\mathbf{O} = (O_1 O_2 \dots O_T)$ , veterbi algorithm finds the sequence of states

$\mathbf{Q} = (q_1 q_2 \dots q_T)$  that is most likely to have generated the given observations sequence. This boils down to maximize  $P(\mathbf{Q}|\mathbf{O}, \mathbf{M})$  or equivalently  $P(\mathbf{Q}, \mathbf{O} | \mathbf{M})$ .

Define variable  $\delta_t(i)$  as the best path ( the path with the highest probability) from the start into some state  $S_i$  at time  $t$ .

$$\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} P(q_1 q_2 \dots q_t = S_i, \mathbf{O}_1 \mathbf{O}_2 \dots \mathbf{O}_t | \mathbf{M}) \quad (3.12)$$

By induction

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] b_j(\mathbf{O}_{t+1}) \quad (3.13)$$

To recover the state sequence, this needs to keep tracks the argument which maximizes the equation (3.13) for each  $t$  and  $j$ , and this is done via the array  $\psi_t(j)$ .

The formal procedure for finding the best state sequence is as follows:

1. Initialisation at time  $t=1$

$$\delta_1(i) = \pi_i b_i(\mathbf{O}_1) \quad (3.14)$$

$$\psi_1(i) = 0 \quad (3.15)$$

For  $1 \leq i \leq N$

#### Idea 3.5

In speaker identification, Veterbi algorithm finds the path through a given speaker's model that result in the highest probability. This algorithm is similar to the forward algorithm, with  $\Sigma$  replaced by max and additional backtracking.

#### Idea 3.6

In most cases, HMMs are implemented using log values to avoid underflow caused by the extremely small probabilities that occur when dealing with large models and large amount of the input data



2. Recursion:

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(O_t) \quad (3.16)$$

$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] \quad (3.17)$$

For  $2 \leq t \leq T; 1 \leq j \leq N$

3. Termination:

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)] \quad (3.18)$$

$$q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)] \quad (3.19)$$

4. Path ( state sequence ) backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad (3.20)$$

For  $t=T-1, T-2, \dots, 1$ .

### 3.3.3 Baum-Welch algorithm

Given finite training observation sequences  $\mathbf{O} = (O_1 O_2 \dots O_T)$  and general structure of HMM (numbers of hidden and visible states), determine HMM parameters  $M=(A, B, \pi)$  that best fit the training data and maximize the probability of the observation sequence given the model  $P(O|M)$ .

Define variable  $\xi_t(i, j)$  as the probability of being in state  $S_i$  at time  $t$  and in state  $S_j$  at time  $t+1$  given observation sequence  $\mathbf{O}$ .

$$\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j | \mathbf{O}) \quad (3.21)$$

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{p(\mathbf{O}|M)} \quad (3.22)$$

In similar manner  $\gamma_t(i)$  defined as the probability of being in state  $S_i$  at time  $t$  given the observation sequence  $\mathbf{O}$ .

$$\gamma_t(i) = P(q_t = S_i | \mathbf{O}) \quad (3.23)$$

$$\gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{\sum_i \alpha_t(i) \beta_t(i)} \quad (3.24)$$

#### Idea 3.7

Application of these steps results in a new HMM  $\bar{M}$  by replacing  $M$  with  $\bar{M}$ , this procedure can be applied iteratively until there is minimal difference between  $M$  and  $\bar{M}$  in successive iteration.

Expected number of transitions from state  $S_i$  to state

$$S_j = \sum_{t=1}^{T-1} \xi_t(i, j) \quad (3.25)$$

Expected number of transitions out of state

$$S_i = \sum_{t=1}^{T-1} \gamma_t(i) \quad (3.26)$$

Using the above formulas, a set re-estimation formulas for the parameter  $\pi$ , A and B are:

$$\pi_i = \text{expected frequency in state } S_i \text{ at time } (t = 1) = \gamma_1(i) \quad (3.27)$$

$$a_{ij} = \frac{\text{Expected number of transitions from state } S_i \text{ to state } S_j}{\text{Expected number of transitions out of state } S_i}$$

$$a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (3.28)$$

$$b_i(k) = \frac{\text{Expected number of times observation } v_k \text{ occurs in state } S_j}{\text{Expected number of times in state } S_j}$$

$$b_i(k) = \frac{\sum_{t=1}^T \mathbf{1}_{S_t(i)=v_k}}{\sum_{t=1}^T \gamma_t(i)} \quad (3.29)$$

[16]

### 3.4 The relationship between HMMS Feature vectors and Speech

The observation sequence for speech recognition is a sequence of acoustic feature vectors (cepstral features). The acoustic feature vector represents information such as the amount of energy in different frequency bands.

Each observation consists of 39 real valued features indicating spectral information.

A reader to HMM theory would think that the hidden states of HMMs corresponds to phone units, and the words are sequences of these phones, but in fact if the frame duration is taken as 20ms, a simple phone like 'ah' or 's' would take about 1second, this means it lasts 50 frames, this amount

of frames cannot have an identical acoustic content, or feature vectors for each frame would not convey the same spectral information so the phone is modeled with more than 1 hmm state.

The sequential nature of speech make strong constraints on transitions between states, the state is allowed to transition either to itself or to 1 succeeding state, the standard model for 1 phone is shown below

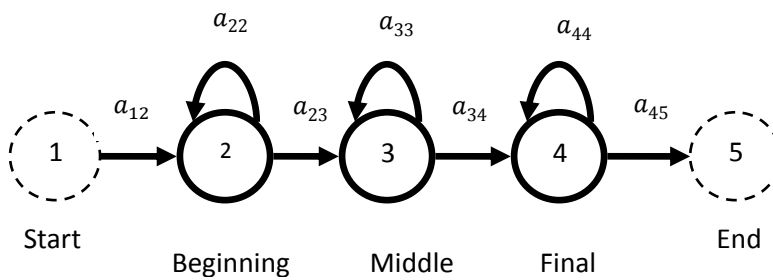


Fig 3.4 standard 5-states HMM model for a phone.

The first and fifth states are non-emitting states, and the 3 others are emitting states.[10]

To construct a sequence for a word having several phones, the start and end states of individual phones are mapped to the other phones. an example of state sequence for the word **GOOD** with succession of phones {G-UH-D} is shown below

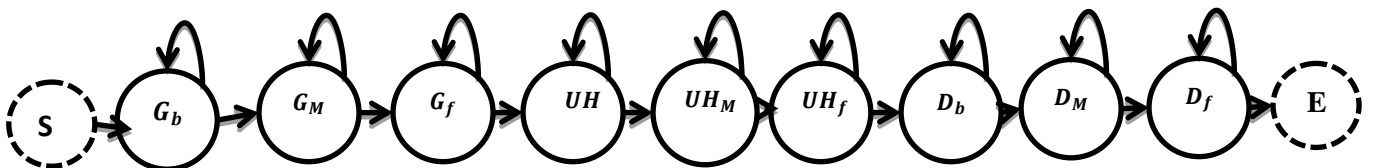


Fig 3.5.a composite word model for the word "GOOD"

This chapter presented background materials on hidden Markov Modeling that will be used to implement a coherent speaker recognition system, in the next chapters feature vectors obtained from the previous chapter are used for speech recognition, and from the results of this, speaker identification can be performed using the acoustic model and the language model in parallel.

# CHAPTER 4

## Implementation in HTK

### 4.1 Experiment

### 4.2 Task Grammar

### 4.3 Creating Dictionary

### 4.4 Preparation of the recorded data

### 4.5 Coding the data

### 4.6 Creating monophone HMMs

### 4.7 Fixing the silence models

### 4.8 Recognizer Evaluation

### 4.9 Discussion

```
S test_sp4.txt -i recout10.mlf -w wdnet -p 500 -t 107 -v 96 -s 5.0 dict.tx
14 physical / 14 logical HMMs
lattice with 5 nodes / 5 arcs
ated network with 26 nodes / 26 links
le: data\mil.mfc
tokens survived to final node of network
le: data\nal.mfc
tokens survived to final node of network
le: data\sai.mfc
tokens survived to final node of network
le: data\ra5.mfc
NL START enjoy_your_vacation SENT_END == [151 frames] -131.9362 [Ac=-21922.4
12000.0] (Act=2.9)

Users\psi\Desktop\proj>HUnit -T 1 -H hmm7_sp4/macros.txt -H hmm7_sp4/hmmdefs.
-S test_sp4.txt -i recout10.mlf -w wdnet -p 200 -t 107 -v 96 -s 5.0 dict.tx
monophones1.txt
14 physical / 14 logical HMMs
lattice with 5 nodes / 5 arcs
ated network with 26 nodes / 26 links
le: data\mil.mfc
tokens survived to final node of network
le: data\nal.mfc
tokens survived to final node of network
le: data\sai.mfc
tokens survived to final node of network
le: data\ra5.mfc
NL START enjoy_your_vacation SENT_END == [151 frames] -138.1103 [Ac=-21654.7
18000.0] (Act=2.9)

Users\psi\Desktop\proj>HUnit -T 1 -H hmm7_sp4/macros.txt -H hmm7_sp4/hmmdefs.
-S test_sp4.txt -i recout10.mlf -w wdnet -p 100 -t 107 -v 96 -s 5.0 dict.tx
monophones1.txt
14 physical / 14 logical HMMs
lattice with 5 nodes / 5 arcs
ated network with 26 nodes / 26 links
le: data\mil.mfc
tokens survived to final node of network
le: data\nal.mfc
tokens survived to final node of network
le: data\sai.mfc
tokens survived to final node of network
le: data\ra5.mfc
NL START enjoy_your_vacation SENT_END == [151 frames] -134.9182 [Ac=-20772.6
14000.0] (Act=3.5)

Users\psi\Desktop\proj>HUnit -T 1 -H hmm7_sp4/macros.txt -H hmm7_sp4/hmmdefs.
-S test_sp4.txt -i recout10.mlf -w wdnet -p 5000 -t 107 -v 96 -s 5.0 dict.t
monophones1.txt
ERROR [+5022] GetChkdFlt: Float arg out of range in p option
FATAL ERROR - Terminating program HUnit

Users\psi\Desktop\proj>HUnit -T 1 -H hmm7_sp4/macros.txt -H hmm7_sp4/hmmdefs.
-S test_sp4.txt -i recout10.mlf -w wdnet -p -2000 -t 107 -v 96 -s 5.0 dict.
monophones1.txt
ERROR [+5022] GetChkdFlt: Float arg out of range in p option

MON [+2835] CrcCPKqqlt: E109t vld one of lunde in b obfrou
monophones1.txt
-g rcg="b4.cxc -i leconclb"mjl -a nquer -b -3000 -c 105 -a de -s 2.8 qicf
eels/bz1/peskrch/blo7\Hnife -i I -H hmm7="b4\mcclos.cxc -H hmm7="b4\mcclos.cxc"

UT ERROR - 1e10tvgfua bloadew Hnife
MON [+2835] CrcCPKqqlt: E109t vld one of lunde in b obfrou
monophones1.txt
-g rcg="b4.cxc -i leconclb"mjl -a nquer -b -2000 -c 105 -a de -s 2.8 qicf
eels/bz1/peskrch/blo7\Hnife -i I -H hmm7="b4\mcclos.cxc -H hmm7="b4\mcclos.cxc"

MON H1 (b4c-3.2)
NL START enjoy_your_vacation SENT_END == [151 frames] -134.9182 [Ac=-20772.6
14000.0] (Act=3.5)
```

In this chapter, the implementation of a speaker recognition system using HTK will be presented. The main purpose is to construct a speech recognizer that can identify the speaker. The speaker of interest is the one uttering the training data.

#### 4.1 Experiment

In our experiment we have chosen the following English sentence: “enjoy your vacation”, having the following monophones in order:” eh n jh oy y ao r v ey k ey sh ah n”

But for pronunciation purposes, silence models and short pauses must be included.

- We have chosen 10 persons to pronounce the preceding sentence.  
The information about persons and the wav files corresponding to them are listed in the table below

**Table I : the speakers with the audio files corresponding to them (the files highlighted are the ones used for training)**

Speaker	name	gender	age	audio file name	content
1	Rym	F	11	mi1.wav	Enjoy your vacation
				mi2.wav	
				mi3.wav	
				mi4.wav	
				mi5.wav	
				mi6.wav	
2	Sarah	F	23	sa1.wav	Enjoy your vacation
				sa2.wav	
				sa3.wav	
				sa4.wav	
				sa5.wav	
				sa6.wav	
				8.wav	Nice day
3	Nacera	F	24	na1.wav	Enjoy your vacation
				na2.wav	
				na3.wav	

				na4.wav	
				na5.wav	
				na6.wav	
4	Rayane	M	12	ra1.wav	Enjoy your vacation
				ra2.wav	
				ra3.wav	
				ra4.wav	
				ra5.wav	
				ra6.wav	
				mon2r.wav	Monday
				mon3r.wav	
5	Aziz	M	52	va1.wav	Enjoy your vacation
				va2.wav	
				va3.wav	
				va4.wav	
				va5.wav	
				va6.wav	
				va7.wav	
				mondayvava.wav	Monday
				monvava.wav	
				sunvava.wav	Sunday
6	wardiya	F	44	WAR1.wav	Enjoy your vacation
				WAR2.wav	
				WAR3.wav	
				WAR4.wav	
				WAR5.wav	
				WAR6.wav	
				MOTWARD.wav	Beautiful
7	mama	F	45	mama1.wav	Enjoy your vacation
				mama3.wav	
				mama4.wav	
				mama5.wav	
				mama6.wav	
				mama7.wav	
				mama8.wav	

				mama9.wav	
				wordmama.wav	Hello
				word2mama.wav	My sister
				word3mama.wav	My mother
8	oussama	M	26	ouss1.wav	Enjoy your vacation
				ouss2.wav	
				ouss3.wav	
				ouss4.wav	
				ouss5.wav	
				ouss6.wav	
				ouss7.wav	
				oussword.wav	I am very intelligent
				oussword2.wav	Amazing
9	Achour	M	40	ach1.wav	Enjoy your vacation
				ach2.wav	
				ach3.wav	
				ach4.wav	
				ach5.wav	
				ach6.wav	
				achword.wav	Je m'appelle zazou
10	Samira	F	35	sami1.wav	Enjoy your vacation
				sami2.wav	
				sami3.wav	
				sami4.wav	
				sami5.wav	
				sami6.wav	
				samiword.wav	Thafath joue ballon

## 4.2 TaskGrammar

- The following content is written in a text file we called “gram”



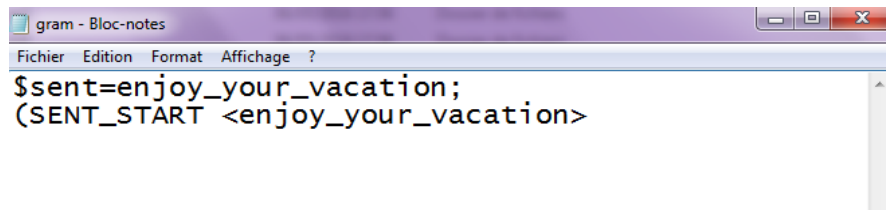


Fig 4.1 gram.txt

>Hparse gram.txt wdnnet

### 4.3 Creating Dictionary

- The dictionary is constructed manually, it contains the sentence defined in the grammar with the proper pronunciation, the script file is called **dict.txt**, then the monophones occurring in it are placed in a new script file called **monophones0.txt**

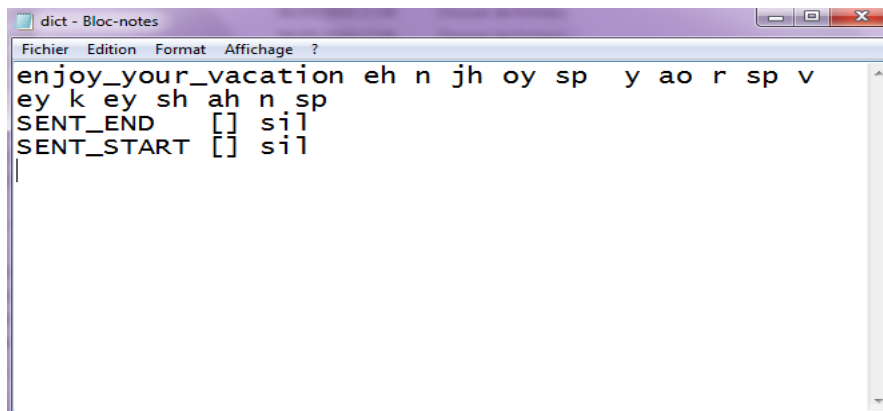


Fig 4.2 dict.txt

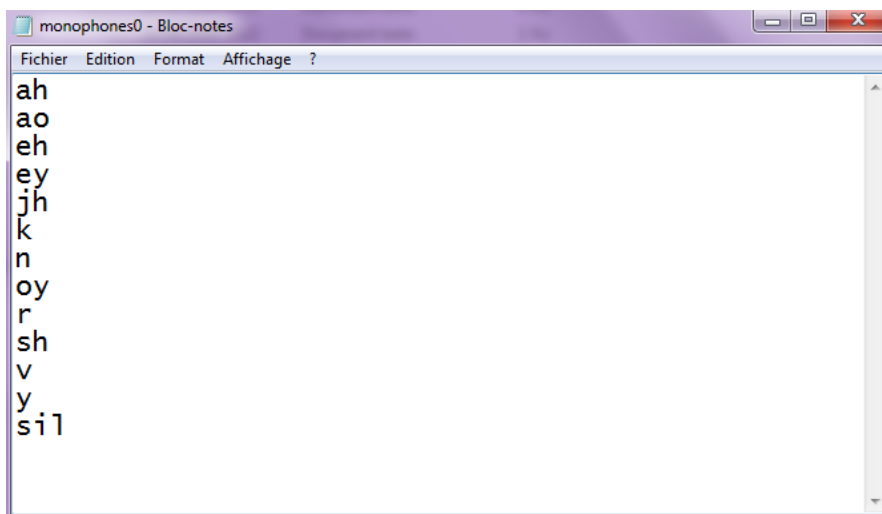


Fig 4.3 monophones0.txt

#### 4.4 Preparation of the recorded data

- A word level script called **words (speaker number).txt** is constructed; it contains the transcriptions of what is said in each **wav** file used for training.

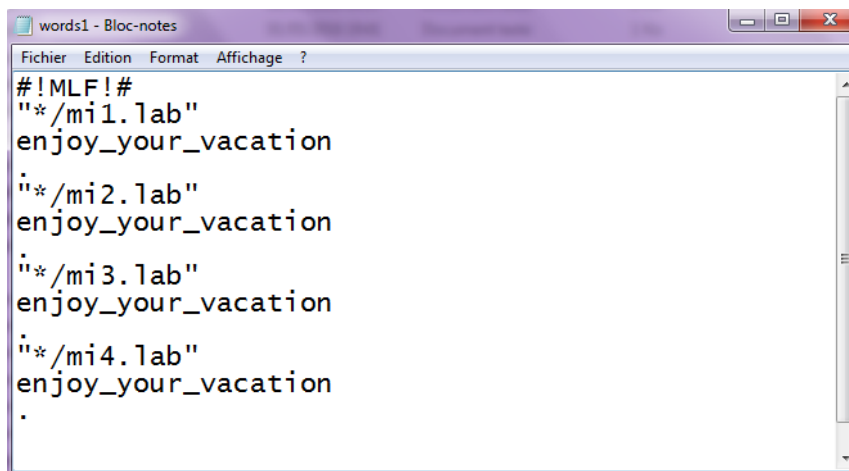


Fig 4.4 words1.txt

- The following script “**mkphones0.led.txt**” is created

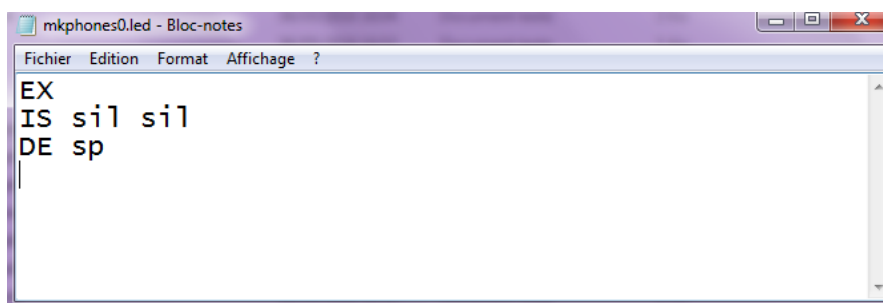


Fig 4.5 mkphones0.led.txt

- The phone level script called “**phones (speaker number).txt**” is created by **HLEd**

```
>HLEd -I '*' -d dict.txt -i phones01.txt mkphones0.led.txt
words2.txt
```

The script file created is shown in **Fig 4.4**.

#### Idea 4.1

The meanings of EX,IS,DE

**EX** is used to replace each word in words1.txt by the its pronunciation in the dictionary **dict.txt**.

**IS** meansto insert a silence modelsil at the start and the end of each utterance.

**DE** is used to delete short pause labels in the script **phones01.txt** in the incoming step

## 4.5 Coding the data

- To code the data configuration files **config.txt** and **config2.txt** are created, the content of each of them is shown below.

### “config2.txt” content:

```
# Coding parameters
SOURCEKIND = WAVEFORM
SOURCEFORMAT = WAV
SOURCERATE = 226.5
TARGETKIND = MFCC_0_D_A
TARGETRATE = 400000
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 1000000
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
ENORMALISE = T
```

### “config.txt” content:

```
# Coding parameters
TARGETKIND = MFCC_0_D_A
TARGETRATE = 400000
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 1000000
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
ENORMALISE = T
```

### Idea 4.2

The values of TARGETRATE and WINDOWSIZE are chosen in this way because they are suitable for the example of isolated word grammar, the default values given in htk book (p31) caused repetitions of words in recognition.

- To extract the feature parameters Hcopy is used, before the command shown below is run, the script file **convert.txt** is created (it contains all the audio files listed in **Table I**), it is shown in **Fig 4.7**.

>HCopy -T 1 -C config.txt -S convert.txt

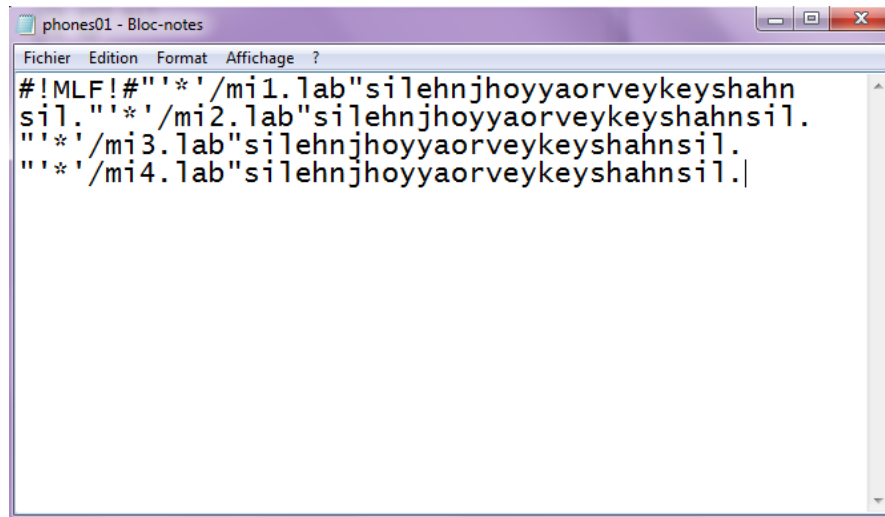


Fig 4.6 phones01.txt

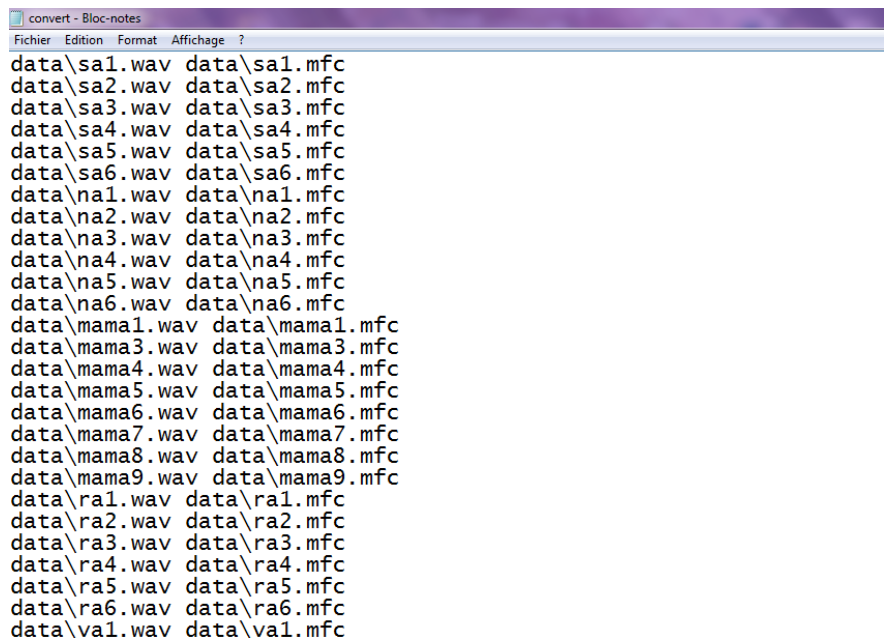


Fig 4.7 convert.txt

#### Idea 4.3

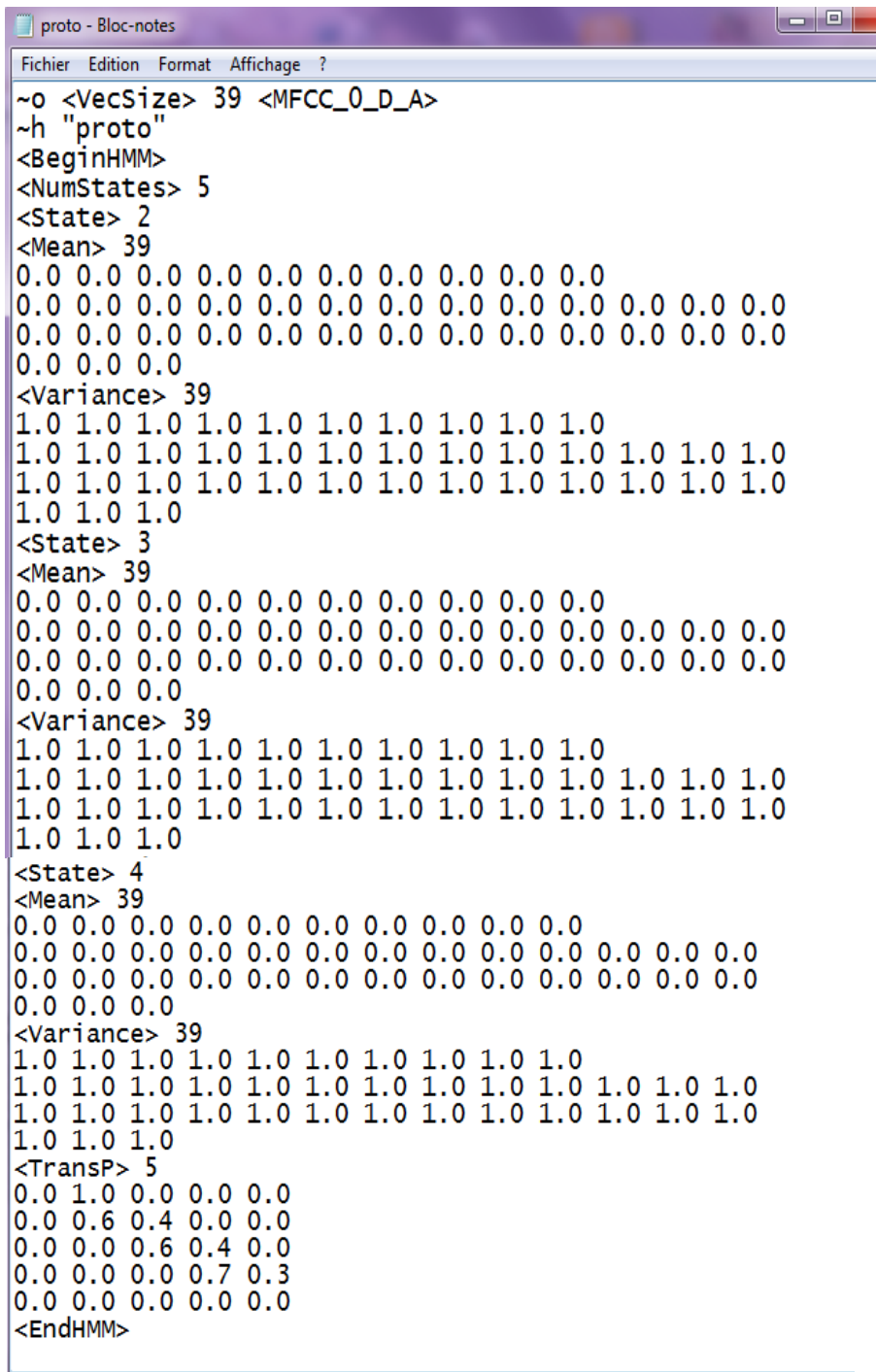
To avoid errors in the command window when executing the command **HERest**, the test file in **Fig 4.6** should be modified in the following way:

“\*” will be \* (“ “ are deleted)

Each line starts by “ and ends by . (Point)

## 4.6 Creating monophone HMMs

- The files **train\_sp(number of speaker).txt** and **proto.txt** are created, **proto.txt** and **train\_sp1.txt** are shown in **Fig 4.8** and **Fig 4.9** respectively.



```

~o <VecSize> 39 <MFCC_0_D_A>
~h "proto"
<BeginHMM>
<NumStates> 5
<State> 2
<Mean> 39
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0
<Variance> 39
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0
<State> 3
<Mean> 39
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0
<Variance> 39
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0
<State> 4
<Mean> 39
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0
<Variance> 39
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0
<TransP> 5
0.0 1.0 0.0 0.0 0.0
0.0 0.6 0.4 0.0 0.0
0.0 0.0 0.6 0.4 0.0
0.0 0.0 0.0 0.7 0.3
0.0 0.0 0.0 0.0 0.0
<EndHMM>

```

Fig 4.8 proto.txt

After that, the command **HCompV** can be executed.

```
>HCompV -C config.txt -f 0.01 -m -S train_sp1.txt -M hmm0_sp1 proto.txt
```

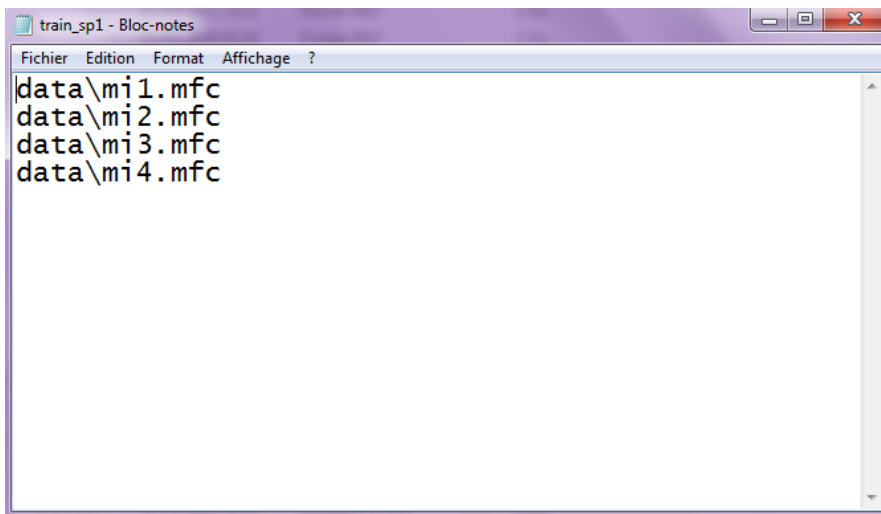


Fig 4.9 train\_sp1.txt

This results in the creation of a new version of **proto.txt** and a script file **vFloors.txt** in the directory **hmm0\_sp1**.

**“proto.txt” (new version) content**

~0

<STREAMINFO> 1 39

<VECSIZE> 39<NULLD><MFCC\_D\_A\_0><DIAGC>

~h "proto"

<BEGINHMM>

<NUMSTATES> 5

<STATE> 2

<MEAN> 39

-6.817831e+000 -1.795901e+001 1.544357e+001 -1.344095e+001

...

#### Idea 4.4

The data used for training are highlighted in **Table I**, in some htk examples 3 samples for training are enough, in this experiment we used 4 sentences for each speaker in order to make the results more accurate.

<ENDHMM>

We can see that the zero means and the unit variances in **Fig 4.9** have been replaced by the global speech means and variances.

Now the 3 lines of **proto.txt** should be cut and pasted into the beginning of **vFloors.txt**, and the term **MFCC\_D\_A\_0** should be replaced by **MFCC\_0\_A\_D**, then the resulting file is saved as **macros.txt**

Now a new script file called **hmmdefs.txt** is created and the content of **proto.txt** (with the 3 lines cut) is copied for each monophone in **monophones0.txt**

#### “macros.txt” content

~0

<STREAMINFO> 1 39

<VECSIZE> 39<NULLD><MFCC\_0\_D\_A><DIAGC>~v varFloor1

<Variance> 39

3.823055e-001 8.135711e-001 4.871610e-001 5.969346e-001 6.232712e-001 7.299178e-001 8.277286e-001 5.443348e-001 7.494858e-001 5.331665e-001 8.273274e-001 2.717662e-001 2.640264e+000 3.069856e-002 6.211357e-002 4.464408e-002 4.441054e-002 5.292542e-002 4.015956e-002 8.592328e-002 5.867302e-002 5.088175e-002 6.099226e-002 5.346099e-002 3.406173e-002 1.073499e-001 4.715789e-003 1.096903e-002 7.957620e-003 7.306530e-003 8.674498e-003 7.634106e-003 1.744311e-002 1.153432e-002 8.500432e-003 1.175649e-002 8.656779e-003 6.373433e-003 8.520158e-003

#### “hmmdefs.txt” content

~h "ah"

<BEGINHMM>

<NUMSTATES> 5

```

<STATE> 2

<MEAN> 39

-6.817831e+000 -1.795901e+001 1.544357e+001 -1.344095e+001

....

<ENDHMM>

~h "ao"

<BEGINHMM>

<NUMSTATES> 5

...

<ENDHMM>

```

- To re-estimate the HMM parameters Baum-welch training is used by the command **HERest**, the models are re-estimated 3 times , each time putting the re-estimated files **hmmdefs.txt** and **macros.txt** in a new directory, in this case **hmm1\_sp1**, **hmm2\_sp1** and **hmm3\_sp1**, **macros.txt** is put also in these files but the values in it doesn't change.

The following command is typed

```

>HERest -C config.txt -I phones01.txt -t 250.0 150.0 1000.0 -S train_sp1.txt -
H hmm0_sp1/macros.txt -H hmm0_sp1/hmmdefs.txt -M hmm1_sp1
monophones0.txt

```

```

>HERest -C config.txt -I phones01.txt -t 250.0 150.0 1000.0 -S train_sp1.txt
-H hmm1_sp1/macros.txt -H hmm1_sp1/hmmdefs.txt -M hmm2_sp1
monophones0.txt

```

```

>HERest -C config.txt -I phones01.txt -t 250.0 150.0 1000.0 -S train_sp1.txt
-H hmm2_sp1/macros.txt -H hmm2_sp1/hmmdefs.txt -M hmm3_sp1
monophones0.txt

```



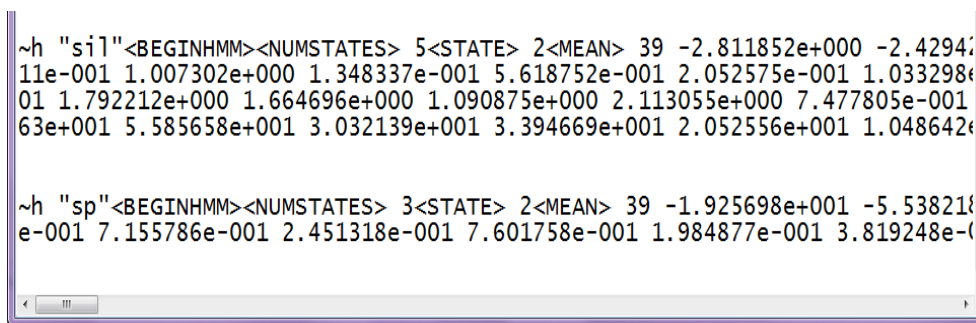
We see that the values of means and variances change each time and the values in the transition matrix too.

#### 4.7 Fixing the silence models

- Now the silence models should be fixed.

Short pause (**sp**) is a one emitting state model, so that state 3 of **sil** is state 2 of **sp**, so **sp** model is created by opening a new directory **hmm4**, then copying the two files **hmmdefs.txt** and **macro.txt** from **hmm3** then the hmm of **sil** is copied then pasted, after that the name is changed to **sp** and state 3 of **sil** becomes state2of **sp**, the remaining parameters are modified to obtain a 1 emitting state **sp** model.

A part of **hmm4\_sp1/hmmdefs.txt** is shown below in **fig 4.10**

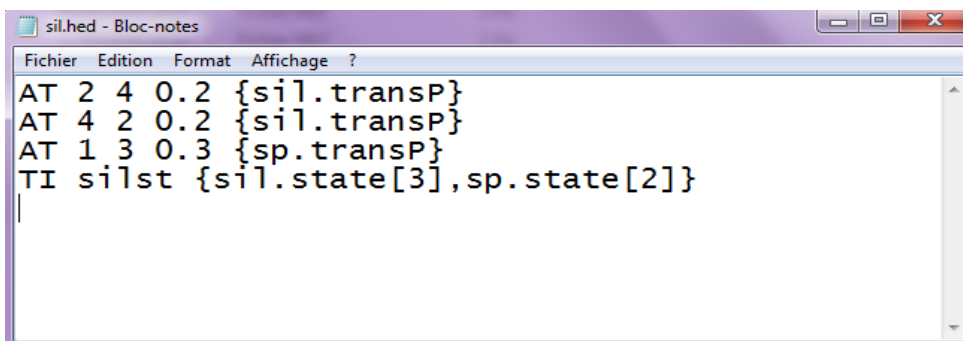


```
~h "sil"<BEGINHMM><NUMSTATES> 5<STATE> 2<MEAN> 39 -2.811852e+000 -2.42942e-001 1.007302e+000 1.348337e-001 5.618752e-001 2.052575e-001 1.033298e-001 1.792212e+000 1.664696e+000 1.090875e+000 2.113055e+000 7.477805e-001 6.3e+001 5.585658e+001 3.032139e+001 3.394669e+001 2.052556e+001 1.048642e-001

~h "sp"<BEGINHMM><NUMSTATES> 3<STATE> 2<MEAN> 39 -1.925698e+001 -5.538218e-001 7.155786e-001 2.451318e-001 7.601758e-001 1.984877e-001 3.819248e-001
```

Fig 4.10 a part of **hmm4\_sp1/hmmdefs.txt**

- We create a new text file called **sil.hed.txt**



```
Fichier Edition Format Affichage ?
AT 2 4 0.2 {sil.transP}
AT 4 2 0.2 {sil.transP}
AT 1 3 0.3 {sp.transP}
TI silst {sil.state[3],sp.state[2]}
```

Fig 4.11 **sil.hed.txt**

The **AT** command adds transitions to the transition matrices and **TI** creates a tied state called **silst**.

The following command is executed.

```
>HHed -H hmm4_sp1/macros.txt -H hmm4_sp1/hmmdefs.txt -M
hmm5_sp1 sil.hed.txt monophones1.txt
```

We notice in **hmm5\_sp1** that the values in **sil.hed.txt** are added to the transition matrices of **sil** and **sp**.

- The re-estimation is applied again to generate **hmmdefs.txt** and **macro.txt** in 2 other directories **hmm6\_sp1** and **hmm7\_sp1**, in this case **sp** is added to the phones in **monophones0.txt** then it is saved as **monophones1.txt**, the following commands are run.

```
>HERest -C config.txt -I phones01.txt -t 250.0 150.0 1000.0 -S train_sp1.txt
-H hmm5_sp1/macros.txt -H hmm5_sp1/hmmdefs.txt -M hmm6_sp1
monophones1.txt
```

```
>HERest -C config.txt -I phones01.txt -t 250.0 150.0 1000.0 -S train_sp1.txt
-H hmm6_sp1/macros.txt -H hmm6_sp1/hmmdefs.txt -M hmm7_sp1
monophones1.txt
```

Before getting into the recognition command, audio files from several speakers are entered to test the recognition process.

The files are in **test.txt**, the content of each of them is mentioned in **Table I**, the files occurring in **test.txt** are all files in **Table I**, except the ones used for training (highlighted in table I) this is for all speakers except for speaker 7 (**MAMA**), for this speaker we tested the files already trained to see the difference.

The content of **test.txt**

data\sa5.mfc

data\sa6.mfc

data\na5.mfc

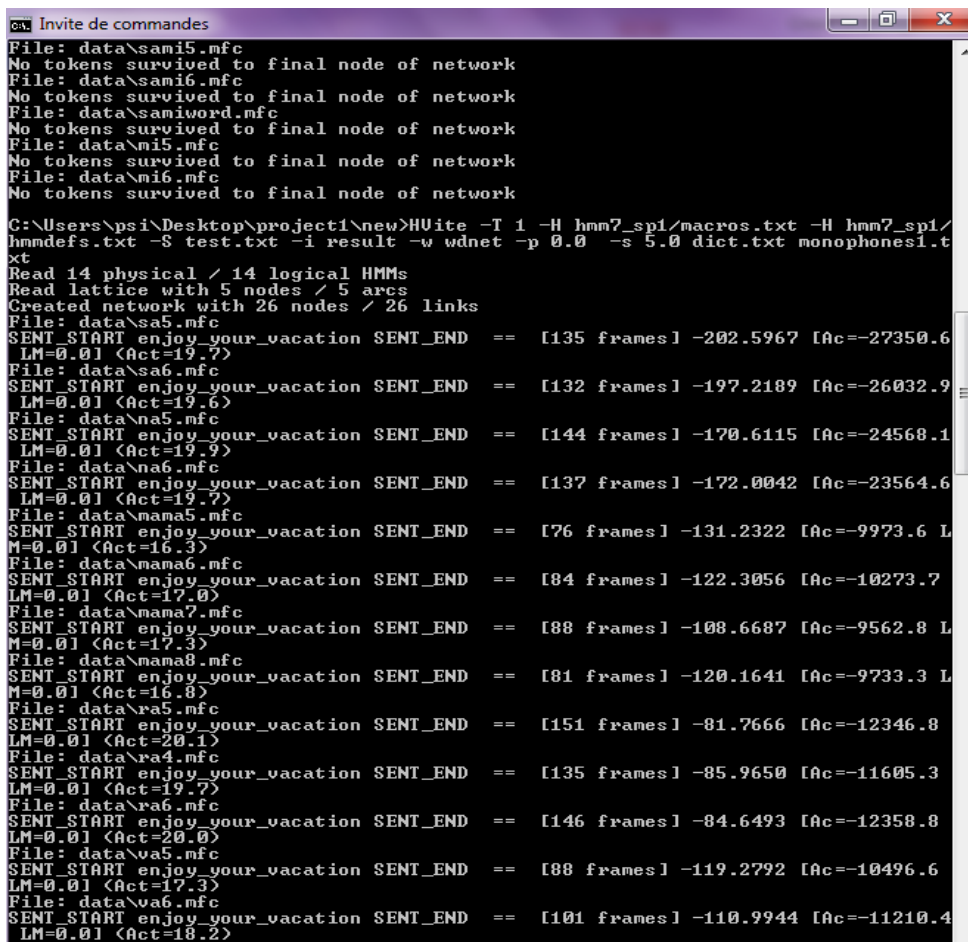
data\na6.mfc...

## 4.8 Recognizer Evaluation

HVite is run for the speaker 1 model and the final recognition results are stored in **result.txt**

```
>HVite -T 1 -H hmm7_sp1/macros.txt -H hmm7_sp1/hmmdefs.txt -S test.txt -i
results.txt -w wdnnet -p 0.0 -s 5.0 dict.txt monophones1.txt
```

The following results appear on the command window:



```

C:\Users\psi\Desktop\project1\new>HVite -T 1 -H hmm7_sp1/macros.txt -H hmm7_sp1/
hmmdefs.txt -S test.txt -i result -w wdnnet -p 0.0 -s 5.0 dict.txt monophones1.t
xt
File: data\sami5.mfc
No tokens survived to final node of network
File: data\sami6.mfc
No tokens survived to final node of network
File: data\samiword.mfc
No tokens survived to final node of network
File: data\mi5.mfc
No tokens survived to final node of network
File: data\mi6.mfc
No tokens survived to final node of network
C:\Users\psi\Desktop\project1\new>HVite -T 1 -H hmm7_sp1/macros.txt -H hmm7_sp1/
hmmdefs.txt -S test.txt -i result -w wdnnet -p 0.0 -s 5.0 dict.txt monophones1.t
xt
Read 14 physical / 14 logical HMMs
Read lattice with 5 nodes / 5 arcs
Created network with 26 nodes / 26 links
File: data\sa5.mfc
SENT_START enjoy_your_vacation SENT_END == [135 frames] -202.5967 [Ac=-27350.6
LM=0.0] (Act=19.7)
File: data\sa6.mfc
SENT_START enjoy_your_vacation SENT_END == [132 frames] -197.2189 [Ac=-26032.9
LM=0.0] (Act=19.6)
File: data\na5.mfc
SENT_START enjoy_your_vacation SENT_END == [144 frames] -170.6115 [Ac=-24568.1
LM=0.0] (Act=19.9)
File: data\na6.mfc
SENT_START enjoy_your_vacation SENT_END == [137 frames] -172.0042 [Ac=-23564.6
LM=0.0] (Act=19.7)
File: data\ma5.mfc
SENT_START enjoy_your_vacation SENT_END == [76 frames] -131.2322 [Ac=-9973.6 L
M=0.0] (Act=16.3)
File: data\ma6.mfc
SENT_START enjoy_your_vacation SENT_END == [84 frames] -122.3056 [Ac=-10273.7
LM=0.0] (Act=17.0)
File: data\ma7.mfc
SENT_START enjoy_your_vacation SENT_END == [88 frames] -108.6687 [Ac=-9562.8 L
M=0.0] (Act=17.3)
File: data\ma8.mfc
SENT_START enjoy_your_vacation SENT_END == [81 frames] -120.1641 [Ac=-9733.3 L
M=0.0] (Act=16.8)
File: data\ra5.mfc
SENT_START enjoy_your_vacation SENT_END == [151 frames] -81.7666 [Ac=-12346.8
LM=0.0] (Act=20.1)
File: data\ra4.mfc
SENT_START enjoy_your_vacation SENT_END == [135 frames] -85.9650 [Ac=-11605.3
LM=0.0] (Act=19.7)
File: data\ra6.mfc
SENT_START enjoy_your_vacation SENT_END == [146 frames] -84.6493 [Ac=-12358.8
LM=0.0] (Act=20.0)
File: data\va5.mfc
SENT_START enjoy_your_vacation SENT_END == [88 frames] -119.2792 [Ac=-10496.6
LM=0.0] (Act=17.3)
File: data\va6.mfc
SENT_START enjoy_your_vacation SENT_END == [101 frames] -110.9944 [Ac=-11210.4
LM=0.0] (Act=18.2)

```

(a)

```

File: data\va7.mfc
SENT_START enjoy_your_vacation SENT_END == [84 frames] -129.9647 [Ac=-10917.0
LM=0.0] (Act=17.0)
File: data\mondayvava.mfc
No tokens survived to final node of network
File: data\mondayvava.mfc
SENT_START enjoy_your_vacation SENT_END == [79 frames] -164.5225 [Ac=-12997.3
LM=0.0] (Act=16.6)
File: data\sunvava.mfc
SENT_START enjoy_your_vacation SENT_END == [80 frames] -135.0412 [Ac=-10803.3
LM=0.0] (Act=16.7)
File: data\8.mfc
SENT_START enjoy_your_vacation SENT_END == [93 frames] -225.7209 [Ac=-20992.0
LM=0.0] (Act=17.7)
File: data\mon2r.mfc
SENT_START enjoy_your_vacation SENT_END == [71 frames] -322.3123 [Ac=-22884.2
LM=0.0] (Act=15.7)
File: data\mon3r.mfc
SENT_START enjoy_your_vacation SENT_END == [88 frames] -218.3849 [Ac=-19217.9
LM=0.0] (Act=17.3)
File: data\WARS.mfc
SENT_START enjoy_your_vacation SENT_END == [98 frames] -105.8984 [Ac=-10378.0
LM=0.0] (Act=18.0)
File: data\WAR6.mfc
SENT_START enjoy_your_vacation SENT_END == [95 frames] -99.4879 [Ac=-9451.3 LM
=0.0] (Act=17.8)
File: data\wordmama.mfc
SENT_START enjoy_your_vacation SENT_END == [76 frames] -123.3907 [Ac=-9377.7 L
M=0.0] (Act=16.3)
File: data\word2mama.mfc
SENT_START enjoy_your_vacation SENT_END == [76 frames] -134.4368 [Ac=-10217.2
LM=0.0] (Act=16.3)
File: data\word3mama.mfc
SENT_START enjoy_your_vacation SENT_END == [57 frames] -165.6243 [Ac=-9440.6 L
M=0.0] (Act=13.7)
File: data\MOTWARD.mfc
SENT_START enjoy_your_vacation SENT_END == [56 frames] -161.7766 [Ac=-9059.5 L
M=0.0] (Act=13.5)
File: data\ouss5.mfc
SENT_START enjoy_your_vacation SENT_END == [91 frames] -121.6560 [Ac=-11070.7
LM=0.0] (Act=17.5)
File: data\ouss6.mfc
SENT_START enjoy_your_vacation SENT_END == [101 frames] -112.8180 [Ac=-11394.6
LM=0.0] (Act=18.2)
File: data\ouss7.mfc
SENT_START enjoy_your_vacation SENT_END == [115 frames] -101.8125 [Ac=-11708.4
LM=0.0] (Act=18.9)
File: data\oussword.mfc
SENT_START enjoy_your_vacation SENT_END == [87 frames] -131.9990 [Ac=-11483.9
LM=0.0] (Act=17.3)
File: data\oussword2.mfc
SENT_START enjoy_your_vacation SENT_END == [63 frames] -156.6242 [Ac=-9867.3 L
M=0.0] (Act=14.7)
File: data\ach5.mfc
SENT_START enjoy_your_vacation SENT_END == [80 frames] -129.1230 [Ac=-10329.8
LM=0.0] (Act=16.7)

```

(b)

```

File: data\ach6.mfc
SENT_START enjoy_your_vacation SENT_END == [112 frames] -119.9161 [Ac=-13430.6
LM=0.0] (Act=18.8)
File: data\achword.mfc
SENT_START enjoy_your_vacation SENT_END == [79 frames] -119.3657 [Ac=-9429.9 L
M=0.0] (Act=16.6)
File: data\sani5.mfc
SENT_START enjoy_your_vacation SENT_END == [87 frames] -109.6668 [Ac=-9541.0 L
M=0.0] (Act=17.3)
File: data\sani6.mfc
SENT_START enjoy_your_vacation SENT_END == [108 frames] -96.8371 [Ac=-10458.4
LM=0.0] (Act=18.6)
File: data\saniword.mfc
SENT_START enjoy_your_vacation SENT_END == [77 frames] -156.8192 [Ac=-12075.1
LM=0.0] (Act=16.4)
File: data\mi5.mfc
SENT_START enjoy_your_vacation SENT_END == [131 frames] -65.5138 [Ac=-8582.3 L
M=0.0] (Act=19.5)
File: data\mi6.mfc
SENT_START enjoy_your_vacation SENT_END == [164 frames] -61.1875 [Ac=-10034.8
LM=0.0] (Act=20.4)
C:\Users\psi\Desktop\project1\new>_

```

(c)

Fig 4.12 (a)-(b)-(c) Recognition results for speaker 1

The information listed for each utterance are:

**The recognized string:** in fact whatever the speaker say, HTK toolkit recognizes only what is in its grammar, since we have written one single sentence in grammar, it recognizes nothing but “Enjoy your vacation”.

**The total number of frames in the utterance:** this gives an idea of the length of the sentence.

**The average log probability per frame:** is the total acoustic likelihood divided over the number of frames (see idea 4.5).

**The total acoustic likelihood:** the total log probability.

**The total language model likelihood:** since the recognizer test the sentence as a whole there is no word –end transitions so the language model likelihood is set to 0.

**The average number of active models**

## 4.9 Discussion

The speaker of the trained sentence ‘Enjoy your vacation’ can be identified by ordering the average log probability per frame in ascending order, the utterance with the higher average Log probability per frame (the closest to zero), corresponds to the speaker in interest, for **speaker1**, from **Table I** the utterances corresponding to the speaker “ RYM” are **mima5.wav** and **mima6.wav**, from **Fig 4.12** , the audio file having the highest average log probability is **mi6.mfc** with a value **-61.1875** then comes **mi5.mfc** with **-61.5138**, both correspond to the speaker **Rym**.

In order to identify the speaker only, without caring for other people, thresholding can be applied, in this case the parameters used for thresholds are the **beam searching** using **-t** and **word end pruning** using **-v** choosing a value of 200 for the beam searching and 200 for the word end pruning means that any model or any observation whose log probability token falls more than 200 below the maximum for all models is deactivated

### Idea 4.5

For an acoustic model of T frames, every frame from the start node to the exit node passes through T emitting HMM states, the log probability is computed by summing the individual transitions in the path and the probability of each emitting state generating the particular observation, the decoder finds the path through the network having the highest log probability. [7]

or killed ,furthermore, tokens from word end nodes that falls more than 200 below the maximum word end likelihood are killed.

The following command is executed.

```
>HVite -T 1 -H hmm7_sp1/macros.txt -H hmm7_sp1/hmmdefs.txt -S test.txt -i  
results1.txt -w wdnnet -p 0.0 -t 200 -v 200 -s 5.0 dict.txt monophones1.txt
```

The results in the command window are shown in **Fig 4.13**, using thresholding caused 4 audio files to be recognized, from **Table I**, it is remarked that the files **mi5.mfc** and **mi6.mfc** belongs to the speaker which means it is a good result, but two other files **ra4.mfc** and **ra5.mfc** belong to speaker4 that is **Rayane** , in fact the values of average likelihood per frame for the two speakers are close, to detect only the audio file corresponding to the speaker the threshold in **-t** option can be decreased, let's use **-t 100**.

```
>HVite -T 1 -H hmm7_sp1/macros.txt -H hmm7_sp1/hmmdefs.txt -S test.txt -i  
results1.txt -w wdnnet -p 0.0 -t 100 -v 200 -s 5.0 dict.txt monophones1.txt
```

```

C:\Users\psi\Desktop\project1\new>HUnit -T 1 -H hmm7_spl/macros.txt -H hmm7_spl/
hmmdefs.txt -S test.txt -i result.txt -w wnet -p 0.0 -t 200 -u 200 -s 5.0 dict.
Read 14 physical / 14 logical HMMs
Read lattice with 5 nodes / 5 arcs
Created network with 26 nodes / 26 links
File: data\sa5.mfc
No tokens survived to final node of network
File: data\sa6.mfc
No tokens survived to final node of network
File: data\na5.mfc
No tokens survived to final node of network
File: data\na6.mfc
No tokens survived to final node of network
File: data\mana5.mfc
No tokens survived to final node of network
File: data\mana6.mfc
No tokens survived to final node of network
File: data\mana7.mfc
No tokens survived to final node of network
File: data\mana8.mfc
No tokens survived to final node of network
File: data\ra5.mfc
SENT_START enjoy_your_vacation SENT_END == [151 frames] -81.7666 [Ac=-12346.8
LM=0.0] (Act=3.9)
File: data\ra6.mfc
SENT_START enjoy_your_vacation SENT_END == [135 frames] -85.9650 [Ac=-11605.3
LM=0.0] (Act=3.9)
File: data\ra6.mfc
No tokens survived to final node of network
File: data\va5.mfc
No tokens survived to final node of network
File: data\va6.mfc
No tokens survived to final node of network
File: data\va7.mfc
No tokens survived to final node of network
File: data\mondayvava.mfc
No tokens survived to final node of network
File: data\mondvava.mfc
No tokens survived to final node of network
File: data\sunvava.mfc
No tokens survived to final node of network
File: data\8.mfc
No tokens survived to final node of network
File: data\mon2r.mfc
No tokens survived to final node of network
File: data\mon3r.mfc
No tokens survived to final node of network
File: data\WAR5.mfc
No tokens survived to final node of network
File: data\WAR6.mfc
No tokens survived to final node of network
File: data\wordmama.mfc
No tokens survived to final node of network
File: data\word2mama.mfc
No tokens survived to final node of network

```

(a)

```

C:\Users\psi\Desktop\project1\new>
File: data\word3mama.mfc
No tokens survived to final node of network
File: data\MOTWARD.mfc
No tokens survived to final node of network
File: data\ouss5.mfc
No tokens survived to final node of network
File: data\ouss6.mfc
No tokens survived to final node of network
File: data\ouss7.mfc
No tokens survived to final node of network
File: data\oussword.mfc
No tokens survived to final node of network
File: data\oussword2.mfc
No tokens survived to final node of network
File: data\ach5.mfc
No tokens survived to final node of network
File: data\ach6.mfc
No tokens survived to final node of network
File: data\achword.mfc
No tokens survived to final node of network
File: data\sami5.mfc
No tokens survived to final node of network
File: data\sami6.mfc
No tokens survived to final node of network
File: data\samiword.mfc
No tokens survived to final node of network
File: data\mi5.mfc
SENT_START enjoy_your_vacation SENT_END == [131 frames] -65.5138 [Ac=-8582.3 L
M=0.0] (Act=3.1)
File: data\mi6.mfc
SENT_START enjoy_your_vacation SENT_END == [164 frames] -61.1875 [Ac=-10034.8
LM=0.0] (Act=3.4)
C:\Users\psi\Desktop\project1\new>

```

(b)

Fig 4.13 (a)-(b) Recognition results for speaker 1 using thresholding

```

C:\Users\psi\Desktop\project1\new>HUnit -T 1 -H hmm7_sp1/macros.txt -H hmm7_sp1/
hmmdefs.txt -S test.txt -i result.txt -w wdnet -p 0.0 -t 100 -v 200 -s 5.0 dict.
txt monophones1.txt
Read 14 physical / 14 logical HMMs
Read lattice with 5 nodes / 5 arcs
Created network with 26 nodes / 26 links
File: data\sa5.mfc
No tokens survived to final node of network
File: data\sa6.mfc
No tokens survived to final node of network
File: data\na5.mfc
No tokens survived to final node of network
File: data\na6.mfc
No tokens survived to final node of network
File: data\ma5.mfc
No tokens survived to final node of network
File: data\ma6.mfc
No tokens survived to final node of network
File: data\ma7.mfc
No tokens survived to final node of network
File: data\ma8.mfc
No tokens survived to final node of network
File: data\ra5.mfc
No tokens survived to final node of network
File: data\ra4.mfc
No tokens survived to final node of network
File: data\ra6.mfc
No tokens survived to final node of network
File: data\va5.mfc
No tokens survived to final node of network
File: data\va6.mfc
No tokens survived to final node of network
File: data\va7.mfc
No tokens survived to final node of network
File: data\mondavava.mfc
No tokens survived to final node of network
File: data\mondvava.mfc
No tokens survived to final node of network
File: data\suvava.mfc
No tokens survived to final node of network
File: data\8.mfc
No tokens survived to final node of network
File: data\mon2r.mfc
No tokens survived to final node of network
File: data\mon3r.mfc
No tokens survived to final node of network
File: data\WAR5.mfc
No tokens survived to final node of network
File: data\WAR6.mfc
No tokens survived to final node of network
File: data\wordmama.mfc
No tokens survived to final node of network
File: data\word2mama.mfc
No tokens survived to final node of network

```

(a)

```

File: data\word3mama.mfc
No tokens survived to final node of network
File: data\MOTWARD.mfc
No tokens survived to final node of network
File: data\ouss5.mfc
No tokens survived to final node of network
File: data\ouss6.mfc
No tokens survived to final node of network
File: data\ouss7.mfc
No tokens survived to final node of network
File: data\oussword.mfc
No tokens survived to final node of network
File: data\oussword2.mfc
No tokens survived to final node of network
File: data\ach5.mfc
No tokens survived to final node of network
File: data\ach6.mfc
No tokens survived to final node of network
File: data\achword.mfc
No tokens survived to final node of network
File: data\sami5.mfc
No tokens survived to final node of network
File: data\sami6.mfc
No tokens survived to final node of network
File: data\samiword.mfc
No tokens survived to final node of network
File: data\mi5.mfc
No tokens survived to final node of network
File: data\mi6.mfc
SENT_START enjoy_your_vacation SENT_END == [164 frames] -61.1875 [Ac=-10034.8
LM=0.0] (Act=2.9)
C:\Users\psi\Desktop\project1\new>

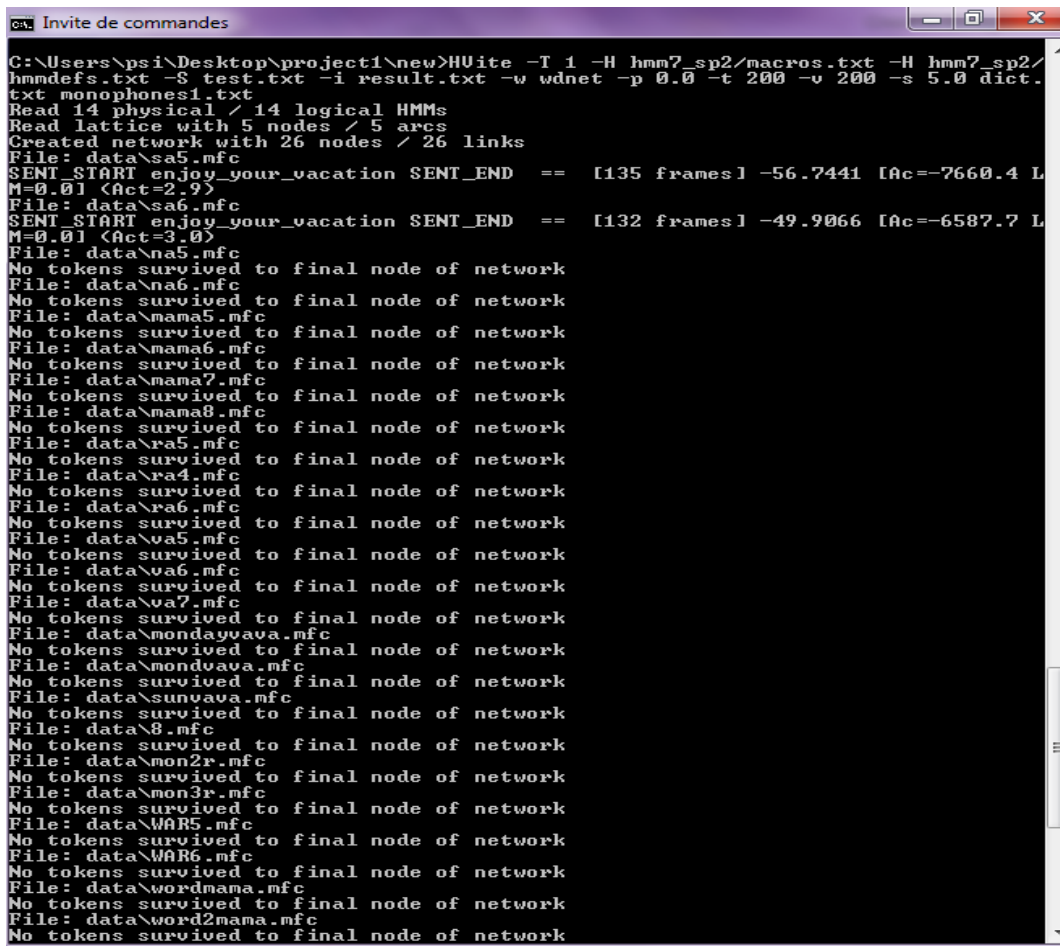
```

(b)

Fig 4.13 (a)-(b) Recognition results for speaker 1 using thresholding 2



Let's apply the same procedure to speaker2 (Sarah), the results are shown in Fig 4.14.

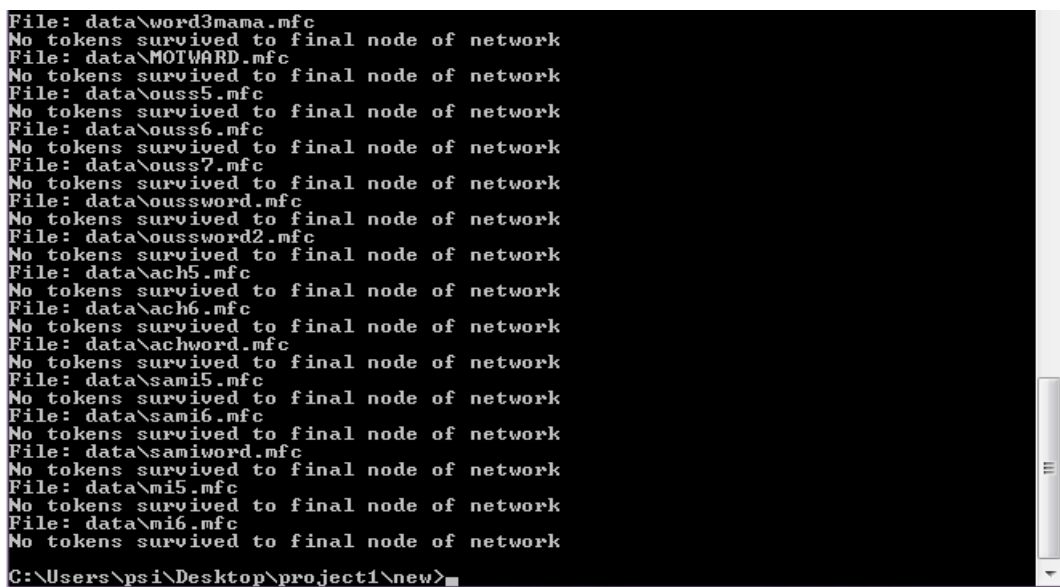


```

C:\Users\psi\Desktop\project1\new>HUnit -T 1 -H hmm7_sp2/macros.txt -H hmm7_sp2/
hmmdefs.txt -S test.txt -i result.txt -w wdnet -p 0.0 -t 200 -u 200 -s 5.0 dict.
txt monophones1.txt
Read 14 physical / 14 logical HMMs
Read lattice with 5 nodes / 5 arcs
Created network with 26 nodes / 26 links
File: data\sa5.mfc
SENT_START enjoy_your_vacation SENT_END == [135 frames] -56.7441 [Ac=-7660.4 L
M=0.0] (Act=2.9)
File: data\sa6.mfc
SENT_START enjoy_your_vacation SENT_END == [132 frames] -49.9066 [Ac=-6587.7 L
M=0.0] (Act=3.0)
File: data\na5.mfc
No tokens survived to final node of network
File: data\na6.mfc
No tokens survived to final node of network
File: data\mama5.mfc
No tokens survived to final node of network
File: data\mama6.mfc
No tokens survived to final node of network
File: data\mama7.mfc
No tokens survived to final node of network
File: data\mama8.mfc
No tokens survived to final node of network
File: data\ra5.mfc
No tokens survived to final node of network
File: data\ra4.mfc
No tokens survived to final node of network
File: data\ra6.mfc
No tokens survived to final node of network
File: data\va5.mfc
No tokens survived to final node of network
File: data\va6.mfc
No tokens survived to final node of network
File: data\va7.mfc
No tokens survived to final node of network
File: data\mondavava.mfc
No tokens survived to final node of network
File: data\mondvava.mfc
No tokens survived to final node of network
File: data\sunvava.mfc
No tokens survived to final node of network
File: data\8.mfc
No tokens survived to final node of network
File: data\mon2r.mfc
No tokens survived to final node of network
File: data\mon3r.mfc
No tokens survived to final node of network
File: data\WAR5.mfc
No tokens survived to final node of network
File: data\WAR6.mfc
No tokens survived to final node of network
File: data\wordmama.mfc
No tokens survived to final node of network
File: data\word2mama.mfc
No tokens survived to final node of network

```

(a)



```

File: data\word3mama.mfc
No tokens survived to final node of network
File: data\MOTWARD.mfc
No tokens survived to final node of network
File: data\ouss5.mfc
No tokens survived to final node of network
File: data\ouss6.mfc
No tokens survived to final node of network
File: data\ouss7.mfc
No tokens survived to final node of network
File: data\oussword.mfc
No tokens survived to final node of network
File: data\oussword2.mfc
No tokens survived to final node of network
File: data\ach5.mfc
No tokens survived to final node of network
File: data\ach6.mfc
No tokens survived to final node of network
File: data\achword.mfc
No tokens survived to final node of network
File: data\sami5.mfc
No tokens survived to final node of network
File: data\sami6.mfc
No tokens survived to final node of network
File: data\samiword.mfc
No tokens survived to final node of network
File: data\ni5.mfc
No tokens survived to final node of network
File: data\ni6.mfc
No tokens survived to final node of network
C:\Users\psi\Desktop\project1\new>

```

(b)

Fig 4.14 (a)-(b) Recognition results for speaker 2 using thresholding

In speaker2 , 2 audio files **sa5.mfc** and **sa6.mfc** are detected, both correspond to speaker2, but one may not know whether they both belong to the speaker in interest, that is why the threshold must be decreased again to obtain only 1 audio file detected.

```

C:\> Invite de commandes
Read 14 physical / 14 logical HMMs
Read lattice with 5 nodes / 5 arcs
Created network with 26 nodes / 26 links
File: data\sa5.mfc
No tokens survived to final node of network
File: data\sa6.mfc
SENT_START enjoy_your_vacation SENT_END == [132 frames] -49.9066 [Ac=-6587.7 L
M=0.0] (Act=2.2)
File: data\sa5.mfc
No tokens survived to final node of network
File: data\sa6.mfc
No tokens survived to final node of network
File: data\mama5.mfc
No tokens survived to final node of network
File: data\mama6.mfc
No tokens survived to final node of network
File: data\mama7.mfc
No tokens survived to final node of network
File: data\mama8.mfc
No tokens survived to final node of network
File: data\ra5.mfc
No tokens survived to final node of network
File: data\ra6.mfc
No tokens survived to final node of network
File: data\ra6.mfc
No tokens survived to final node of network
File: data\va5.mfc
No tokens survived to final node of network
File: data\va6.mfc
No tokens survived to final node of network
File: data\va7.mfc
No tokens survived to final node of network
File: data\mondavva.mfc
No tokens survived to final node of network
File: data\mondvava.mfc
No tokens survived to final node of network
File: data\sunvava.mfc
No tokens survived to final node of network
File: data\8.mfc
No tokens survived to final node of network
File: data\mon2r.mfc
No tokens survived to final node of network
File: data\mon3r.mfc
No tokens survived to final node of network
File: data\WAR5.mfc
No tokens survived to final node of network
File: data\WAR6.mfc
No tokens survived to final node of network
File: data\wordmama.mfc
No tokens survived to final node of network
File: data\word2mama.mfc
No tokens survived to final node of network
File: data\word3mama.mfc
No tokens survived to final node of network
File: data\MOTWARD.mfc
No tokens survived to final node of network

```

(a)

```

File: data\ouss5.mfc
No tokens survived to final node of network
File: data\ouss6.mfc
No tokens survived to final node of network
File: data\ouss7.mfc
No tokens survived to final node of network
File: data\oussword.mfc
No tokens survived to final node of network
File: data\oussword2.mfc
No tokens survived to final node of network
File: data\ach5.mfc
No tokens survived to final node of network
File: data\ach6.mfc
No tokens survived to final node of network
File: data\achword.mfc
No tokens survived to final node of network
File: data\sami5.mfc
No tokens survived to final node of network
File: data\sami6.mfc
No tokens survived to final node of network
File: data\samiword.mfc
No tokens survived to final node of network
File: data\mi5.mfc
No tokens survived to final node of network
File: data\mi6.mfc
No tokens survived to final node of network
C:\Users\psi\Desktop\project1\new>

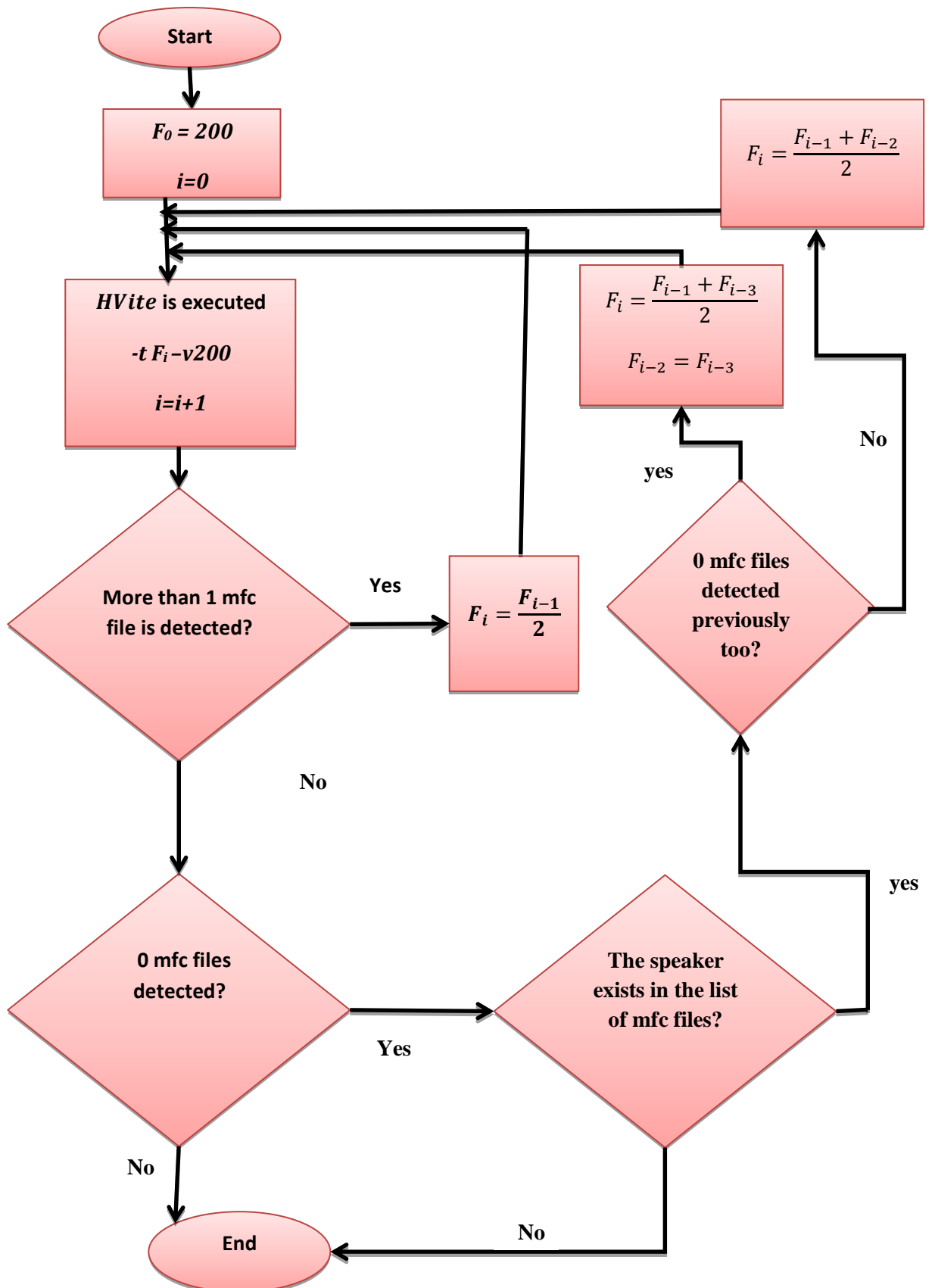
```

(b)

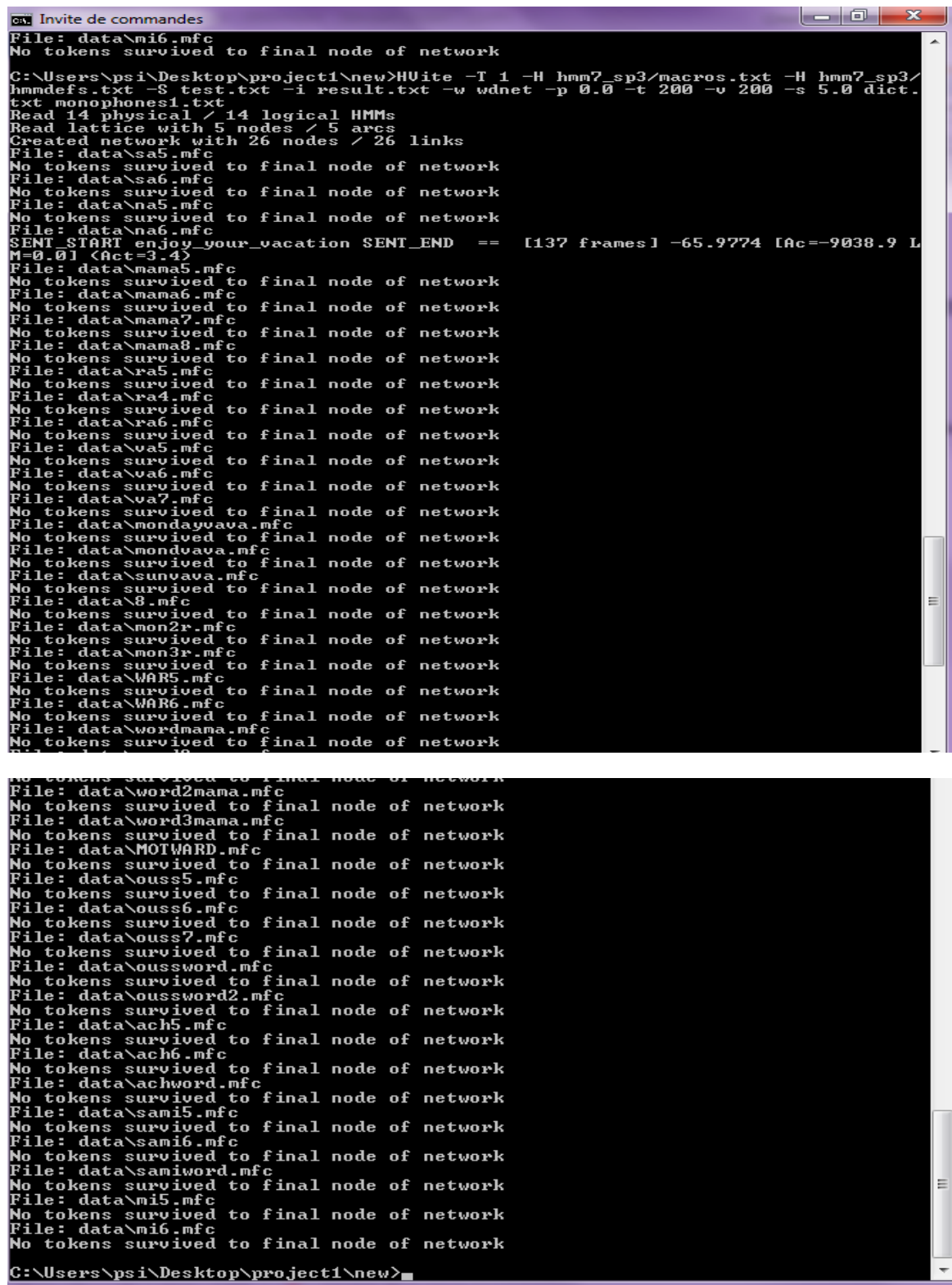
Fig 4.14 (a)-(b) Recognition results for speaker 2 using thresholding 2

Then the process of Thresholding can be summarized in the following chart

Chart I: Method to use Beam searching in recognizing MFC files



The results for speaker 3 is shown in Fig 4.15 below



```

C:\Users\psi\Desktop\project1\new>HUnit -T 1 -H hmm7_sp3/macros.txt -H hmm7_sp3/
hmmdefs.txt -S test.txt -i result.txt -w wdnnet -p 0.0 -t 200 -v 200 -s 5.0 dict.
txt monophones1.txt
Read 14 physical / 14 logical HMMs
Read lattice with 5 nodes / 5 arcs
Created network with 26 nodes / 26 links
File: data\sa5.mfc
No tokens survived to final node of network
File: data\sa6.mfc
No tokens survived to final node of network
File: data\sa5.mfc
No tokens survived to final node of network
File: data\sa6.mfc
SENT_START enjoy_your_vacation SENT_END == [137 frames] -65.9774 [Ac=-9038.9 L
M=0.01 (Act=3.4)
File: data\mama5.mfc
No tokens survived to final node of network
File: data\mama6.mfc
No tokens survived to final node of network
File: data\mama7.mfc
No tokens survived to final node of network
File: data\mama8.mfc
No tokens survived to final node of network
File: data\ra5.mfc
No tokens survived to final node of network
File: data\ra4.mfc
No tokens survived to final node of network
File: data\ra6.mfc
No tokens survived to final node of network
File: data\va5.mfc
No tokens survived to final node of network
File: data\va6.mfc
No tokens survived to final node of network
File: data\va7.mfc
No tokens survived to final node of network
File: data\mondavava.mfc
No tokens survived to final node of network
File: data\mondvava.mfc
No tokens survived to final node of network
File: data\suvava.mfc
No tokens survived to final node of network
File: data\8.mfc
No tokens survived to final node of network
File: data\mon2r.mfc
No tokens survived to final node of network
File: data\mon3r.mfc
No tokens survived to final node of network
File: data\WAR5.mfc
No tokens survived to final node of network
File: data\WAR6.mfc
No tokens survived to final node of network
File: data\wordmama.mfc
No tokens survived to final node of network
File: data\word2mama.mfc
No tokens survived to final node of network
File: data\word3mama.mfc
No tokens survived to final node of network
File: data\MOTWARD.mfc
No tokens survived to final node of network
File: data\ouss5.mfc
No tokens survived to final node of network
File: data\ouss6.mfc
No tokens survived to final node of network
File: data\ouss7.mfc
No tokens survived to final node of network
File: data\oussword.mfc
No tokens survived to final node of network
File: data\oussword2.mfc
No tokens survived to final node of network
File: data\ach5.mfc
No tokens survived to final node of network
File: data\ach6.mfc
No tokens survived to final node of network
File: data\achword.mfc
No tokens survived to final node of network
File: data\sami5.mfc
No tokens survived to final node of network
File: data\sami6.mfc
No tokens survived to final node of network
File: data\samiword.mfc
No tokens survived to final node of network
File: data\mi5.mfc
No tokens survived to final node of network
File: data\mi6.mfc
No tokens survived to final node of network
C:\Users\psi\Desktop\project1\new>

```

Fig 4.15 (a)-(b) Recognition results for speaker 3 using thresholding

**Table II: Recognition results in the command window using Tresholding: several iterations are performed according to Chart I till recognition is performed successfully (green color), “S “means the sentence “ENJOY YOUR VACATION”**

Speaker number	Speaker name	F value in -t option	MFC files detected	Average Log probability per frame $LogP_{av}$	Content of MFC file	Recognition rate?
1	Rym	200	mi5.mfc	-65.5138	S	50%
			mi6.mfc	-61.1875	S	
		100	mi6.mfc	-61.1875	S	100%
2	Sarah	200	sa5.mfc	-56.7441	S	50%
			sa6.mfc	-49.9066	S	
		100	sa5.mfc	-56.7441	S	50%
			sa6.mfc	-49.9066	S	
		50	sa6.mfc	-49.9066	S	100%
3	Nacera	200	na6.mfc	-65.9774	S	100%
4	Rayane	200	ra4.mfc	-45.4212	S	50%
			ra5.mfc	-59.8998	S	
		100	0	0	0	0%
		150	ra4.mfc	-45.4212	S	100%
5	Aziz	200	va5.mfc	-68.7027	S	50%
			va6.mfc	-68.0087	S	
		100	va5.mfc	-68.7027	S	50%
			va6.mfc	-68.0087	S	
		50	va6.mfc	-68.0087	S	100%
6	Wardiya	200	WARD5.mfc	-57.8368	S	50%
			WARD6.mfc	-59.6924	S	
		100	WARD5.mfc	-57.8368	S	50%
			WARD6.mfc	-59.6924	S	
		50	WARD5.mfc	-57.8368	S	100%
7	Mama	200	mama5.mfc	-64.9699	S	16.66%
			mama6.mfc	-50.0685	S	
			mama7.mfc	-46.8897	S	
			mama8.mfc	-46.5117	S	
			ra5.mfc	-100.6124	S	
			ra4.mfc	-92.3274	S	
		100	mama5.mfc	-64.9699	S	25%
			mama6.mfc	-50.0685	S	
			mama7.mfc	-46.8897	S	
			mama8.mfc	-46.5117	S	
		50	mama6.mfc	-50.0685	S	33.33%
			mama7.mfc	-46.8897	S	
			mama8.mfc	-46.5117	S	
		25	mama6.mfc	-50.0685	S	33.33%
			mama7.mfc	-46.8897	S	
		12.5	mama6.mfc	-50.0685	S	33.33%
			mama8.mfc	-46.5117	S	

			mama7.mfc	-46.8897	S	
			mama8.mfc	-46.5117	S	
		6.25	mama6.mfc	-50.0685	S	33.33%
			mama7.mfc	-46.8897	S	
			mama8.mfc	-46.5117	S	
		3.125	mama6.mfc	-50.0685	S	33.33%
			mama7.mfc	-46.8897	S	
			mama8.mfc	-46.5117	S	
		1.5625	0	0	0	0%
		2.34375	mama6.mfc	-50.0685	S	50%
			mama8.mfc	-46.5117	S	
		1.953125	0	0	0	0%
		2.1484375	mama6.mfc	-50.0685	S	50%
			mama8.mfc	-46.5117	S	
		0	0	0	0	0%
			0	0	0	0%
		1.8798828 13	0	0	0	0%
		2.0141598 15	mama6.mfc	-50.0685	S	50%
			mama8.mfc	-46.5117	S	
		mama6.wav and mama8.wav belong to the same speaker <b>Mama</b> and they are too close in average log probability, and also they are detected together for very small values of thresholds, this is because they are the trained data for speaker 8, as mentioned before.				

8	Oussama	200	ra5.mfc	-91.0644	S	16.66%
			ra4.mfc	-89.9052	S	
			va6.mfc	-85.8467	S	
			ouss5.mfc	-63.1234	S	
			ouss6.mfc	-60.6256	S	
			ouss7.mfc	-58.9640	S	
		100	ouss5.mfc	-63.1234	S	33.33%
			ouss6.mfc	-60.6256	S	
			ouss7.mfc	-58.9640	S	
		50	ouss5.mfc	-63.1234	S	33.33%
			ouss6.mfc	-60.6256	S	
			ouss7.mfc	-58.9640	S	
		25	ouss5.mfc	-63.1234	S	33.33%
			ouss6.mfc	-60.6256	S	
			ouss7.mfc	-58.9640	S	
		12.5	ouss5.mfc	-63.1234	S	33.33%
			ouss6.mfc	-60.6256	S	
			ouss7.mfc	-58.9640	S	
		6.25	ouss5.mfc	-63.1234	S	50%
			ouss6.mfc	-60.6256	S	
3.125	ouss6.mfc	-60.6256	S	100%		

9	Achour	200	ach5.mfc	-81.1658	S	100%
10	Samira	200	WAR5.mfc	-133.8591	S	33.33%
			Sami5.mfc	-60.6665	S	
			sami6.mfc	-57.5264	S	
		100	sami5.mfc	-60.6665	S	50%
			sami6.mfc	-57.5264	S	
		50	sami5.mfc	-62.9390	S	100%

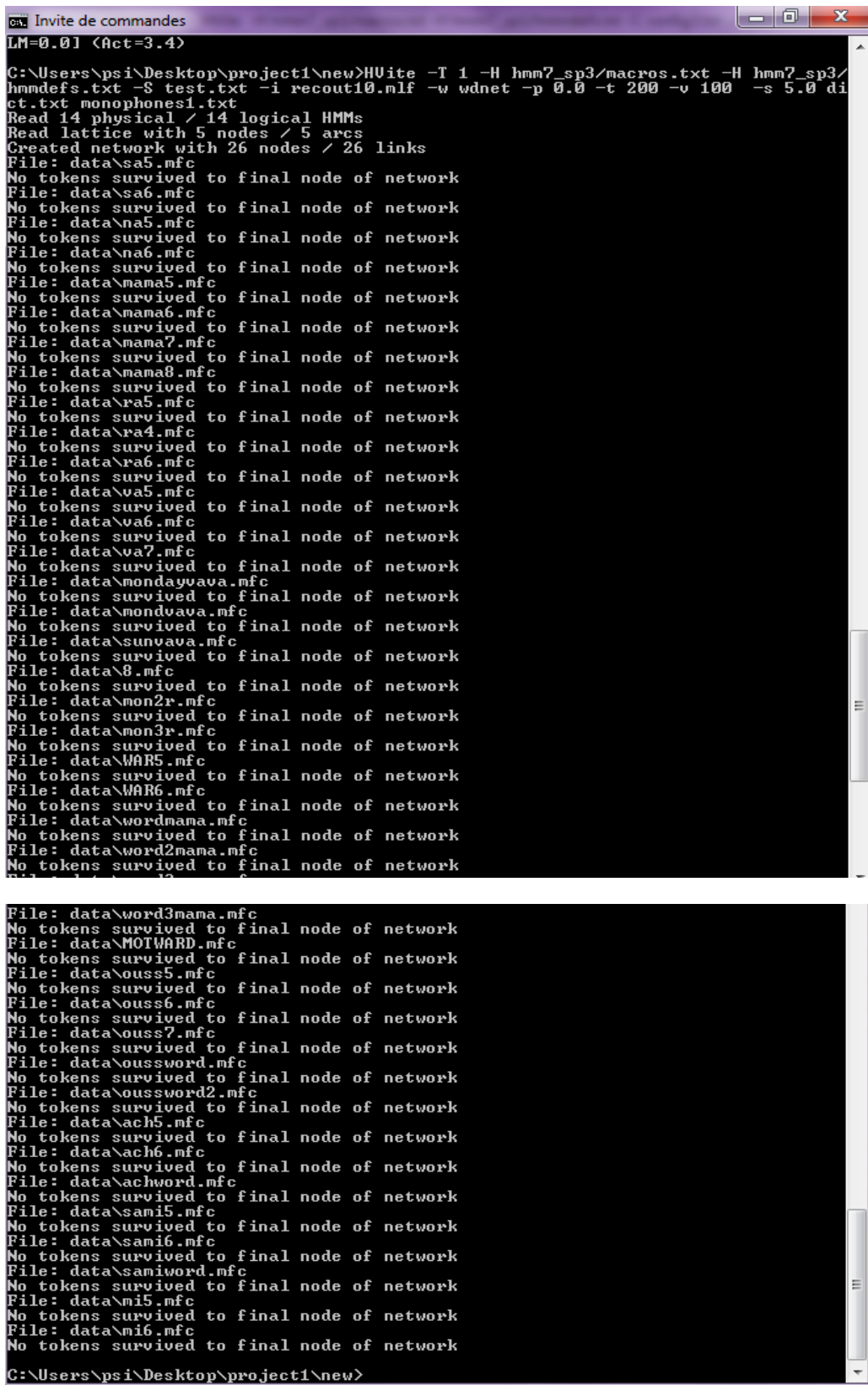
From the above table it is remaked that recognition was perfomed successfully for all the 10 speakers .

As seen in theory, each HMM is composed of a sequence of phone models, each phone model contains subphone states.

To solve the problem of decoding , in the speech recognition task, the best sequence of words is the one that maximizes the two factors,the maximum lnguage model prior  $P(W)$  and the acoustic likelihood  $P(O|W)$ . [10]

$$\tilde{W} = \underset{W \in L}{\operatorname{argmax}} P(O|W)P(W) \quad (4.1)$$

This fact was used for Speaker identification, the thresholding of the language model likelihood enables the detection of the intended sentence “ENJOY YOUR VACATION”, and the maximum Log probability thresholding allows the speaker identifcation, because the more the testing voice is closer to the voice used for training, the Maximum log probability and the average Log acoustic likelihood become higher, the use of the value **200** as an initial value in the **-t** option was set arbitrarily, because it suits all speakers, in the **-v** option (the word end puning), the value **200**, was set by default as the optimum case because withsmaller values, the model risks of not detecting any **mfc** file (see Fig 4.16)



```

C:\Users\psi\Desktop\project1\new>HUnit -T 1 -H hmm7_sp3/macros.txt -H hmm7_sp3/
hmmdefs.txt -S test.txt -i recout10.mlf -w wdnet -p 0.0 -t 200 -v 100 -s 5.0 di
ct.txt monophones1.txt
Read 14 physical / 14 logical HMMs
Read lattice with 5 nodes / 5 arcs
Created network with 26 nodes / 26 links
File: data\sa5.mfc
No tokens survived to final node of network
File: data\sa6.mfc
No tokens survived to final node of network
File: data\sa5.mfc
No tokens survived to final node of network
File: data\sa6.mfc
No tokens survived to final node of network
File: data\ma5.mfc
No tokens survived to final node of network
File: data\ma6.mfc
No tokens survived to final node of network
File: data\ma7.mfc
No tokens survived to final node of network
File: data\ma8.mfc
No tokens survived to final node of network
File: data\ra5.mfc
No tokens survived to final node of network
File: data\ra6.mfc
No tokens survived to final node of network
File: data\va5.mfc
No tokens survived to final node of network
File: data\va6.mfc
No tokens survived to final node of network
File: data\va7.mfc
No tokens survived to final node of network
File: data\mondayvava.mfc
No tokens survived to final node of network
File: data\mondvava.mfc
No tokens survived to final node of network
File: data\sunvava.mfc
No tokens survived to final node of network
File: data\8.mfc
No tokens survived to final node of network
File: data\mon2r.mfc
No tokens survived to final node of network
File: data\mon3r.mfc
No tokens survived to final node of network
File: data\WAR5.mfc
No tokens survived to final node of network
File: data\WAR6.mfc
No tokens survived to final node of network
File: data\wordmama.mfc
No tokens survived to final node of network
File: data\word2mama.mfc
No tokens survived to final node of network
File: data\word3mama.mfc
No tokens survived to final node of network
File: data\MOTWARD.mfc
No tokens survived to final node of network
File: data\ouss5.mfc
No tokens survived to final node of network
File: data\ouss6.mfc
No tokens survived to final node of network
File: data\ouss7.mfc
No tokens survived to final node of network
File: data\oussword.mfc
No tokens survived to final node of network
File: data\oussword2.mfc
No tokens survived to final node of network
File: data\ach5.mfc
No tokens survived to final node of network
File: data\ach6.mfc
No tokens survived to final node of network
File: data\achword.mfc
No tokens survived to final node of network
File: data\sami5.mfc
No tokens survived to final node of network
File: data\sami6.mfc
No tokens survived to final node of network
File: data\samiword.mfc
No tokens survived to final node of network
File: data\mi5.mfc
No tokens survived to final node of network
File: data\mi6.mfc
No tokens survived to final node of network
C:\Users\psi\Desktop\project1\new>

```

Fig 4.16 (a)-(b) Recognition results when setting the word end pruning threshold to 100



Finally, the following facts can be deduced from this model.

- This model is a **text dependent** , because when using thresholds the sentences said by speakers that are other than “ENJOY YOUR VACATION”, like **MOTWOARD.mfc**, **wordmama.mfc** and **oussword.mfc** (see **TABLE I**), are **deactivated** even when using initial values of thresholds.
- This method is a fast method , for most of speakers a small number of iterations (even when more than 1 file corresponding to the text and to the speaker exists), is enough to recognize one sentence allowing the user to say: **it belongs to the speech (“ENJOY YOUR VACATION”) and it is uttered by the speaker in interest.**
- This model can be generalized for a high number of speakers, because it is trained with 10 speakers from different ages and genders.

# CONCLUSION

The concept of constructing text dependent speaker recognition has been presented in this project. Since the speaker recognition is a new idea in the field of signal processing, and speech recognition was developed earlier, the mathematical models that construct the speech recognizer were used to identify a speaker uttering some training data.

To achieve the results, a lot of experiments have been done respecting the HTK toolkit procedure. Furthermore, the results were presented in HTK command window instead of a text file.

By using the concept of thresholding, only the intended audio file corresponding to the speaker in interest appears, and the audio files of other speakers in addition to the sentences other than the trained one, are deactivated even if they belong to the speaker in interest.

As a further, some recommendations are suggested for researchers in the field:

- The recognizer live in HTK needs to be developed to function in a faster rate.
- Constructing a big grammar containing all sentences in English, in order to recognize anything the speaker can say.
- The development of HTK as software having an appropriate user interface to avoid dealing with the traditional command window.

# REFERENCES

- [1] Rana, D. S. (2012). The process of speech recognition, perception, speech signals and speech production in human beings. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 1(9), pp-156.
- [2] Harish ChanderMahendru,(January 2014). "Quick Review of Human Speech Production Mechanism ",Department of Electronics & Communication Engg., Haryana Engineering College, *International Journal of Engineering Research and Development*, Volume 9, Issue 10 .
- [3] <http://www.barcode.ro/tutorials/biometrics/voice.html> (03/2016)
- [4] CHOUGUI & HACIANE(2015). "speaker recognition system using htk", Master final report, Institute of Electrical and Electronics Engineering, .University of Boumerdes.
- [5] Tiwari, V. (2010). MFCC and its applications in speaker recognition. *International Journal on Emerging Technologies*, 1(1), 19-22.
- [6] Rajsekhar, A. (2008). Real time Speaker Recognition using MFCC and VQ. *M. Tech Thesis, National Institute of Technology, Rourkela, India.*
- [7] Steve, Y., Gunnar, E., & MARK, A. (2009). The HTK book (for HTK Version 3.4). *Steve Young, Gunnar Evermann, Dan Kershaw, Gareth Moore, Julian Odell, Dave Ollason, Valtcho Valtchev, Phil Woodland.*
- [8] [https://en.wikipedia.org/wiki/Manner\\_of\\_articulation](https://en.wikipedia.org/wiki/Manner_of_articulation) (05/2016)
- [9] <http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/> (03/2016)
- [10] Martin, J. H., & Jurafsky, D. (2000). Speech and language processing. *International Edition.*
- [11] Makhoul, J., & Schwartz, R. (1995). State of the art in continuous speech recognition. *Proceedings of the National Academy of Sciences*, 92(22), 9956-9963.
- [12] Al, R. T., Spraakherkenning, A., Wiggers, I. P., & Rothkrantz, L. J. M. (2003). AUTOMATIC SPEECH RECOGNITION USING HIDDEN MARKOV MODELS.
- [13] Kotwal, P., & Dixit, M. R. Learning Assistant in Educational Field Using Automatic Speech Recognition.
- [14] Vaseghi, S. V. (2008). Advanced digital signal processing and noise reduction. John Wiley & Sons.
- [15] Jackson, P. (2011). HMM tutorial. Centre for Vision Speech Signal Processing, University of Surrey. Web.
- [16] Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257-286.

# APPENDIX

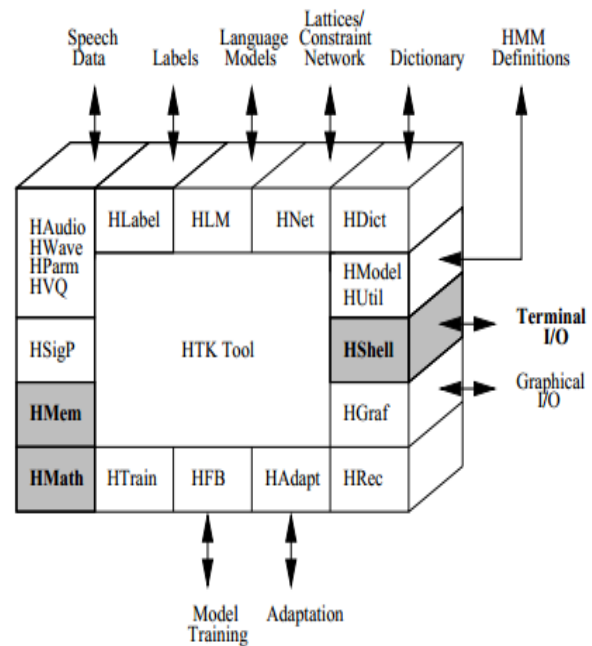
## A

# AN OVERVIEW OF THE HIDDEN MARKOV MODEL TOOLKIT

### A.1 HTK software

### A.2 Available HTK Tools

### A.3 HTK files



The subject of this appendix is Hidden Markov Toolkit HTK, a collection of software libraries and tools for manipulating Hidden Markov models are presented. The architecture of the toolkit will be described and an overview of the tools used in our speaker recognizer will be given.

### A.1 HTK software

The Hidden Markov Model Toolkit (HTK) is a software toolkit for building and manipulating systems that use hidden Markov models. It has been developed at the Machine Intelligence Laboratory of the Cambridge University Engineering Department.

The first version of HTK was developed in 1989 and since then many versions have been developed. In this project HTK version 3.4.1 is used.

HTK is primarily designed for building HMM based speech recognizers although it has been used for numerous other applications including research into speech synthesis and speaker recognition was also used.

HTK consists of a set of library modules and tools available in C source format. These tools provide sophisticated facilities for speech analysis, HMM training, testing and result analysis.

The software supports HMMs using both continuous density mixture Gaussians and discrete distributions and can be used to build complex HMM systems.

HTK tools are designed to run with traditional command line style interface, each tool has a large number of required and optional arguments and most tools require one or more script files.

### A.2 Available HTK Tools

**Fig A.1** gives an overview of the HTK tools subdivided into four groups according to the processing stages in which they are used: Data preparation, Training, Testing and analysis tools.

In this section, only the tools needed for the implementation of speaker recognizer are presented.

The interested reader on this topic is advised to refer to HTK book for more details.

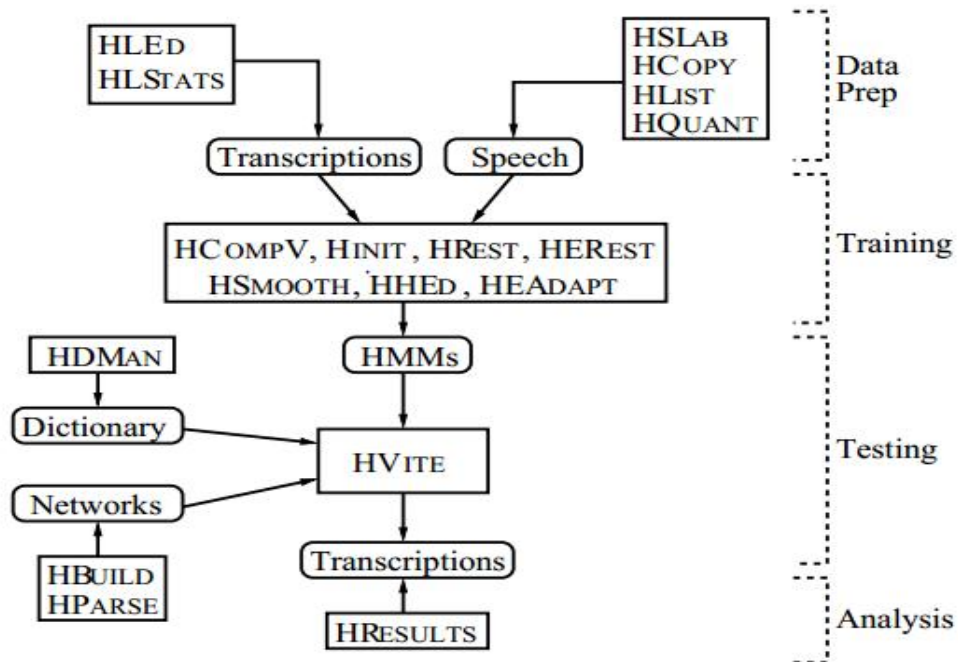


Fig A.1 HTK tools

### A.2.1 Data preparation tools

Before any training or recognition process can be done with HTK, the required data are set up in a format that suits HTK.

The software offers tools to prepare the data for further processing.

Some of them are presented in this appendix with their corresponding function:

#### HCOPY:

This function is used to Copy one or more source files to an output file, and convert the input files to a parametric form, where the behavior is controlled by the configuration file.

#### HList

This tool is used to check the contents of any speech file and to visualize a file with feature vectors.

**HLEd**

It is an editor for manipulating script files, and used to make transformations to label files and to output files to single Master Label File MLF.

The detailed operation of **HLEd** tool is controlled by the following command line options:

Option	Utility
<b>-d s</b>	Read a dictionary from file <b>s</b> and use this for expanding labels when the EX command is used
<b>-i mlf</b>	This specifies that the output transcriptions are written to the master label file <b>mlf</b>
<b>-l s</b>	Directory to store output label files. When the output is directed to an MLF, this option can be used to add a path to each output file name. In particular, setting the option <b>-l '*'</b> will cause a label file named xxx to be prefixed by the pattern "*/xxx" in the output MLF file. This is useful for generating MLFs which are independent of the location of the corresponding data files.

**A.2.2 Training tools**

HTK allows HMMs to be built with any desired topology. The purpose of the prototype definition is only to specify the overall characteristics and topology; the actual parameters will be computed later using the appropriate training tools.

The following gives some of these training tools and their function:

**HCompV**

This function is used to initialize the parameters of the emission probability distribution function (PDF) in HMM states to the global values for a given phoneme, these global parameters are calculated from a set of training data using this tool.

The detailed operation of **HCompV** tool is controlled by the following command line options:

Option	Utility
<b>-f f</b>	Create variance floor macros with values equal <b>f</b> times the global variance and the output is stored in a file called vFloors.
<b>-m</b>	Updates all the HMM component means with the sample mean computed from the training files.

### **HERest**

This function is used to perform a single re-estimation of the parameters of a set of HMMs using an embedded training version of the Baum-Welch algorithm.

The detailed operation of **HERest** tool is controlled by the following command line options:

Option	Utility
<b>-t f</b>	Set pruning threshold to <b>f</b> . during the backward probability calculation, at each time all $t$ all $(\log)\beta$ values falling more than <b>f</b> below the maximum $\beta$ value at that time are ignored. During the subsequent forward pass $(\log)\alpha$ values are only calculated if there are corresponding valid $\beta$ values.

### **HEEd**

It works as a script editor but its input and output are HMM definition files. Its basic operation is to load a set of HMMs then apply a sequence of editing operations and output the transformed set.

### **A.2.3 Testing tools**

Once the training is done. The models are tested against a set of testing data. HTK provides a single recognition tool called **HVite** which uses the algorithm like the one described in the previous chapter to perform Viterbi-based speech recognition.



**HVite**

This is Viterbi decoder or recognizer. Given an unknown word, it computes the Viterbi probability of all models and finds the maximum. The model that emitted the given word with the maximum probability wins.

HVite takes as input a network describing the allowed word sequences, a dictionary defining the pronunciation of each word and a set of HMMs. It operates by converting the word network to phone network and then attaching the appropriate HMM definition to each phone instance. Recognition can be performed on either a list of stored speech files or on direct input.

In this stage HTK provides a tool **HParse** for creating a word network.

The detailed operation of **HVite** tool is controlled by the following command line options:

Option	Utility
<b>-w</b>	Perform recognition from word level networks
<b>-p f</b>	Is a word insertion penalty to <b>f</b> . and it is a fixed value added to each token when it transits from the end of one word to the start of the next
<b>-t</b>	Used to deactivate models whose maximum log probability token falls more than <b>f</b> below the maximum for all models.

**A.2.4 Analysis tools**

Once the HMM-based recognizer has been built. It is necessary to evaluate its performance. And this can be done using tool **HResults**

**HResults**

Is an HTK performance analysis tool, it reads from the output of the recognition tool, and compares them with the corresponding reference transcription files.

### A.2.5 Standard HTK tool options:

Some standard options are common to every HTK tool. In this project some of them are used:

Option	Utility
<b>-T 1</b>	displays some information about the algorithm actions.
<b>-C</b>	used to specify a configuration file name
<b>-S</b>	used to specify a script File name
<b>-I mlf</b>	This loads the master label file <b>mlf</b> , this option may be repeated to load several MLFs
<b>-M dir</b>	Store output HMM macros files in the directory <b>dir</b> , otherwise the new HMM definition will overwrite the existing one.
<b>-H mmf</b>	Load HMM macros model file <b>mmf</b> , this option may be repeated to load multiple MMFs

## A.3 HTK files

### A.3.1 Master Label File (MLF)

This is a file which contains a number of transcriptions stored sequentially within it. Each such transcription is preceded by a pattern and terminated by a period. MLF can also be used to redirect the search for a transcription to another directory thereby allowing an HTK tool to access label files dispersed throughout a large database.

### A.3.2 Configuration File

The configuration file is a text file for setting the parameters of acoustical coefficient extraction. It consists of a list of parameters-values pairs along with an optional prefix which limits the scope of the parameter to a specific module or tool. If no prefix then any module or tool can read it.

[Module:] parameter = value

Only one parameter definition written per line and square brackets indicate that the module name is optional. Parameter definitions are not case sensitive but by convention they are written in upper case. A # character indicates that the rest of the line is a comment.

An example of configuration file is given:

```
# Coding parameters
SOURCEKIND = WAVEFORM
SOURCEFORMAT = WAV
SOURCERATE =226.75
TARGETKIND = MFCC_0_D_A
TARGETRATE =400000
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE =1000000
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
ENORMALISE = T
```

With such a configuration file, an MFCC (Mel Frequency cepstral coefficient) analysis is performed (prefix “MFCC” in the TARGETKIND identifier). For each signal frame the following coefficient are extracted:

- The 12 first MFCC coefficients [c1,..., c12] (since NUMCEPS = 12).
- The “null” MFCC coefficient c0, which is proportional to the total energy in the frame (suffix “\_0” in TARGETKIND).
- 13 “Delta coefficients”, estimating the first order derivative of [c0, c1,...,c12] (suffix“\_D” in TARGETKIND).
- 13 “Acceleration coefficients”, estimating the second order derivative of [c0, c1,..., c12] (suffix “\_A” in TARGETKIND).

And specify that the frame period is 40msec, the output should be saved in uncompressed format. The FFT should use a Hamming window and the signal should have first order pre-emphasis applied using a coefficient of 0.97. The filter-bank should has 26 channels. The variable ENORMALISE is by default true and performs energy normalization on recorded audio files.

### A.3.3 Script files

Scripts files in HTK refer to a sequence of commands or a set of data to be processed.

These files are specified in HTK using the standard -S option and their contents are read simply as extensions to the command line. Thus, they avoid the need for command lines with several thousand arguments,

### A.3.4 HMM definition files

The principle function of HTK is to manipulate sets of hidden Markov models (HMMs). The definition of a HMM specifies the model topology, the transition parameters and the output distribution parameters.

HMM in HTK consists of an arbitrary number of states  $N$ , where the entry state and the exit state  $N$  are non-emitting states. Each state has an associated observation probability distribution  $b_j(O_t)$ , which determines the probability of generating observation  $O_t$  at time  $t$  and each pair of states  $i$  and  $j$  has an associated transition probability  $a_{ij}$ .

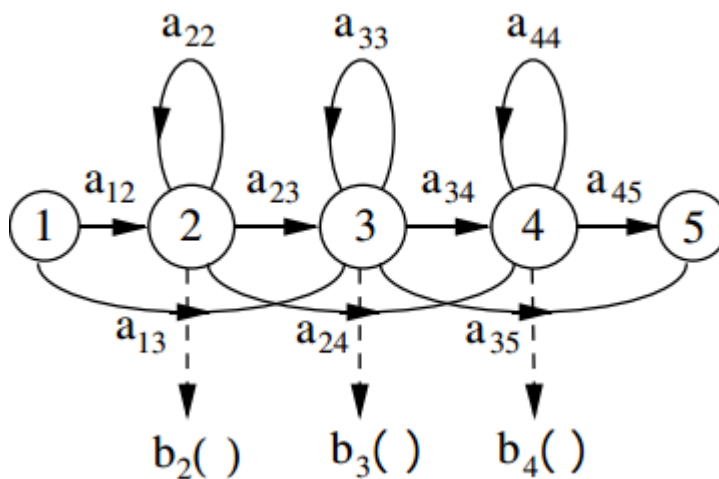


Fig A.2 Simple left-right HMM model

**Fig A.2** shows a simple left to right HMM model with five states in total, three of these are emitting states and have output probability distributions associated with them.

The transition matrix will have 5 rows and 5 columns. Each row will sum to one except for the final row which is always all zero since no transitions are allowed out of the final state.

HMM definitions are stored in text file using a simple formal language whose structure shown in **Fig A.3**.

Keywords are indicated by surrounding angle brackets and the whole definition is bracketed by the keywords <BeginHMM> and <EndHMM>. The notation is meant to be easily readable by humans

The general structure is that global parameters are given first followed by the parameters for each state. Thus, in the example, the HMM has 5 states (<NumStates>), a vector size of 4 (<VecSize>), diagonal covariance (<DIAGC>), no duration parameters (<NULLD>) and data parameterization consisting of Mel-Frequency Cepstral Coefficients (<MFCC>). The data for each state consists of a 4 dimensional mean vector followed by a 4 dimensional variance vector.

Finally, the keyword (<TransP>) introduces the 5x5 transition matrix.

```

~h "hmm1"
<BeginHMM>
  <VecSize> 4 <MFCC>
  <NumStates> 5
  <State> 2
    <Mean> 4
      0.2 0.1 0.1 0.9
    <Variance> 4
      1.0 1.0 1.0 1.0
  <State> 3
    <Mean> 4
      0.4 0.9 0.2 0.1
    <Variance> 4
      1.0 2.0 2.0 0.5
  <State> 4
    <Mean> 4
      1.2 3.1 0.5 0.9
    <Variance> 4
      5.0 5.0 5.0 5.0
  <TransP> 5
    0.0 0.5 0.5 0.0 0.0
    0.0 0.4 0.4 0.2 0.0
    0.0 0.0 0.6 0.4 0.0
    0.0 0.0 0.0 0.7 0.3
    0.0 0.0 0.0 0.0 0.0
<EndHMM>

```

Fig A.3 Definition of left-right HMM model

This Appendix was written from material found on the HTK book, to help the reader of this paper get a quick idea of how the Toolkit works. The interested reader in further details and techniques about HTK is strongly advised to refer to the HTK book.