BENGALI SEGMENTED AUTOMATED SPEECH RECOGNITION

A K M Mahmudul Hoque
Student Id # 02101035

Department of Computer Science and Engineering
May 2006

BRAC University, Dhaka, Bangladesh

DECLARATION

I hereby declare that this thesis is based on the building of **Bengali segmented Automatic Speech Recognizer**. It is a limited word recognizer for Bengali language. The recognizer can not be used by any user at this moment. The recognizer can recognize any word from its dictionary.

Materials and third party utility that I used mention in reference. This thesis, neither in whole nor in part, has been previously submitted for any Degree of BRAC University.

Signature of Supervisor	Signature of Author
----------	----------
Dr. Mumit Khan	A K M Mahmudul Hoque

ACKNOWLEDGMENTS

My heartiest thank goes to my thesis supervisor who always encouraged me to do such a thesis on Bengali Speech Recognition. When I choose such a dynamic topic like speech recognition everybody was discouraging me, but when I went to my thesis supervisor he gave me the courage needed to move forward with this topic. He helped me in all aspects when I got stuck, and he pushed me front always. When everybody told that it is a huge topic for thesis, he gave me all kind of support through out my thesis period. He provided me all kind resources I need for my thesis. I got all kind inspiration and courage to do thesis on Speech recognition from him.

My specials thank goes to Cambridge University Engineering Department. I used their HTK toolkit to build my Recognizer for my thesis. With their toolkit I might not finish my thesis. And again my thanks go to Abul Hasnat who helped me with the toolkit at scratch.

Abstract

Bengali Segmented Automated Speech Recognition

Speech recognition and understanding of spontaneous speech have been an elusive goal of research since 1970. For understanding speech human not only consider for information passed to the ears but also judge the information by the context of the information. That's why human can easily understand the spoken language convey to them even in noisy environment. Recognizing speech by machine is so difficult for the dynamic characteristics of spoken languages. People used different approaches for automated speech recognition system. For recognizing speech people always prefer English as most of the research and implemented for them. So I am intended to have my research on Speech Recognition system but preferably in our mother tongue –Bengali. It is an area where a lot to contribute for our language to establish in computer field.

The contribution of this thesis is to show how to build a speech recognizer using HTK toolkit which can recognize Bengali words. Bengali speech recognizer is built by training the HTK toolkit and can recognize any word in the dictionary. After acoustic analysis of speech signal waves the words are recognized. Technically this thesis presents training the toolkit and builds a segmented speech recognizer of Bengali. Finally the thesis contains the training procedure of the toolkit, how people can build a recognizer with the HTK toolkit.

TABLE OF CONTENTS

List of figures

LIST OF TABLES

1 Introduction

1.1 Motivation

Speech recognition is widely researched topic around the world. Many scientists and researchers are busy with doing works on speech recognition. Worldwide speech recognition is mostly done in different languages mostly English. Most of the languages in the world have speech recognizers of its own. But our mother tongue Bengali is not enriched with a speech recognizers. No one has done significant works on Bengali speech recognizer. This thesis is little try to build a Bengali speech recognizer to enrich our language. Through out the thesis I build a train Bengali limited word speech recognizer.

1.2 Goals and Accomplishments

Speech recognizers implementing for Bengali is our main goal throughout the thesis work. But implementing a recognizer is huge task for a single within a short span of time. So the thesis work then narrowed down to a limited word speech recognizer. This thesis is about building a recognizer for Bengali using an existing tool. First train the tools for building the recognizer. After training the system is built. Then it is tested with the trained words to test. This thesis is about theoretical work on speech recognition and then builds a segmented recognizer for Bengali language. The recognizer can recognize on word at a time and give 75% of success rate.

1.3 Outline

The thesis is mainly divided into two sections: Theoretical section and building recognizer section. The theoretical section consists of chapter 2,3 and 4. Chapter 2 is all about Speech recognition approach of human and machine. Chapter 3 describes features of different phonemes. Chapter 4 gives the description of Hidden Markov Model. And in chapter 5 how the recognizer is build with HTK tool it is described. Result and evaluation is in chapter 6.

2. Speech Recognition

2.1 Speech Recognition By Human

Human brains are too fast to recognize a speech signal and interpreting its meaning. Acoustic waveform is percept by Basilar membrane motion of ear. Spectrum analysis of the signal is done in basilar membrane motion. Then with the neural transduction of the signal is extracted in human brain. And the language sense is already in human mind, so with that language sense, stock of phonemes language translation of the signal is done in brain. And as consequence brain get the correct meaning of the signal and get the message. The message extraction process is done too fast in human brain, millions of are neurons are busy doing speech recognition for human [3].

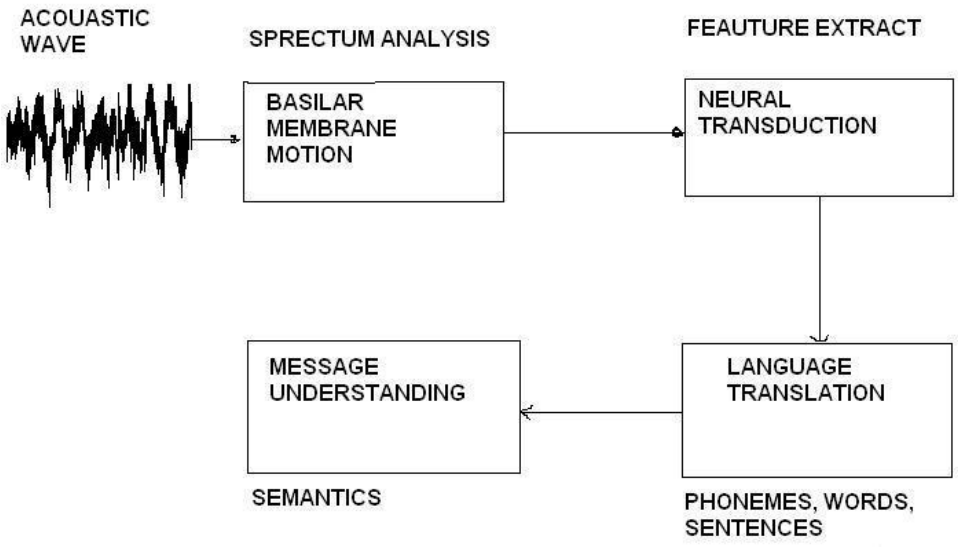

Fig 2.1: Speech recognition: Human approach [1]

2.2 Speech recognition By machine

2.2.1 Training of machine

The approach of machine is almost like human approach accept for everything machine has devices or algorithm to decode speech signal. Machine imitates what human do for recognition of speech. Machine takes speech signal by audio input devices. After the input signal wave is segmented and labeled. Then a series of acoustic analysis is done to extract feature extraction. And with the extracted features are used to train the machine for pattern. With the trained patterns a model/pattern classifier is built to classify any pattern next time. The patterns might be the phonemes or words. For this thesis patterns are words that the recognizer will recognize [1].

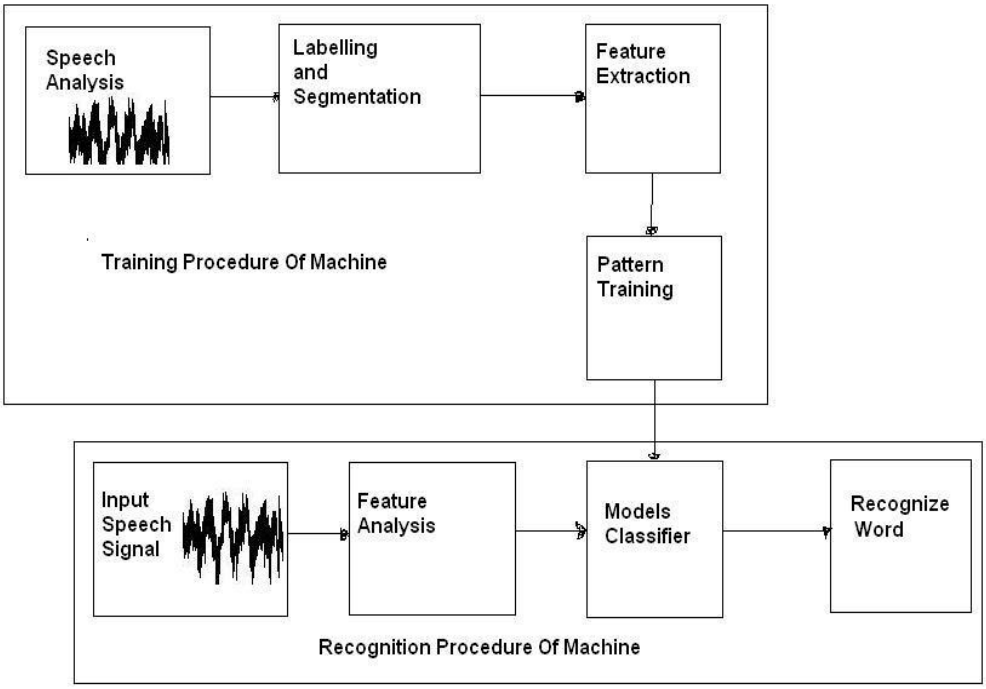

Fig:2.2.1- Speech Recognition system for Machine [1]

2.2.2 Recognition by machine

For recognizing speech signal first is taken input from the devices. Then series of acoustic analysis is done to extract features from the signal. After feature analysis it is passed to model classifier to recognize the correct pattern. After matching the pattern model classifier gives the recognized word as output.

3. Sounds Features and Phonemes

3.1 Speech in time and frequency domains

Speech signal is a time varying signal over a short period of time. But it is considered for a long period of time speech signal is stationary for a speaker. But speech signal is dynamic from person to person. The signal characteristics change to reflect different sounds are produced. The slowly time varying signal can be observed in speech signal. If we label one speech signal we see *unvoiced silence* preceding the main *sound signal* and followed by *voiced stop* [1]. In figure we can see the word "AWG" by a male speaker. First portion is unvoiced silence then the sound for word "AWG" followed by voiced/unvoiced stop.

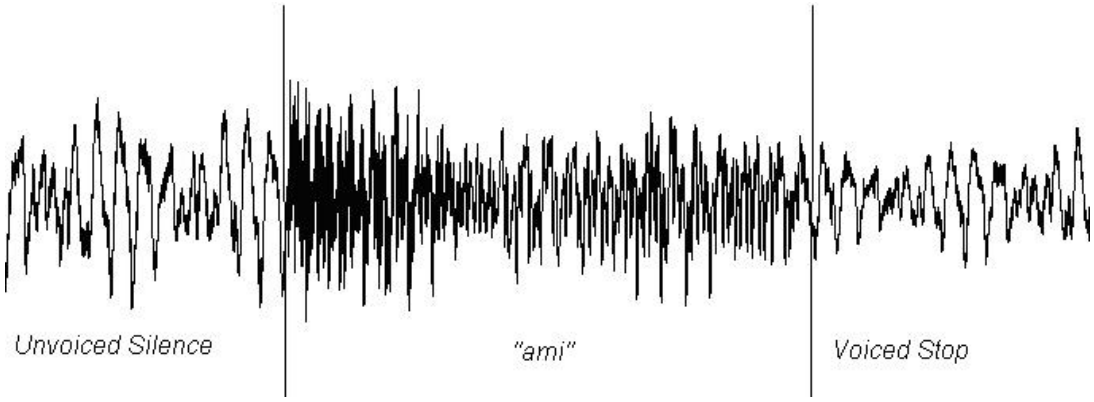

Fig 3.1: Voiced stop and Non-voiced silence with " AWG"

3.2 Vowel phonemes

A vowel is defined as a voiced sound, I forming which the air issues in continuous stream through the pharynx and mouth, there being no obstruction and no narrowing such as would cause audible friction [5]. And vowel sounds are most important in any language for recognition. Successful recognition of vowel sound is obvious to build a recognizer. The performance of a recognizer depends of the correct recognition of vowel sounds.

Vowel sounds are generally long in duration and spectrally well defined. But vowel sounds varies depending on the placement of it in the word. If it is in the starting the sounds are normal, but after some consonants sounds get longer and again in the end of a word vowels sounds is shorter. For example the vowel "Av" has three different lengths in three words ÓAvwgÓ(fig-3.2.a), Ócvcó (fig 3.2.b) and ÓAvkvÓ (fig 3.2.c). In Bengali there are eight cardinal vowels. The vowels list is given in the table.

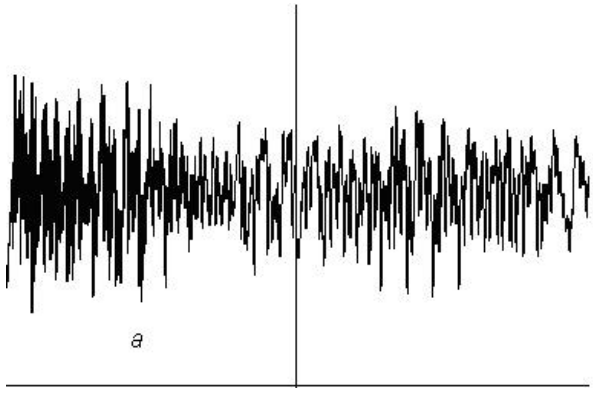

Fig: 3.2.a: "Av "in word " Avwg" by male speaker

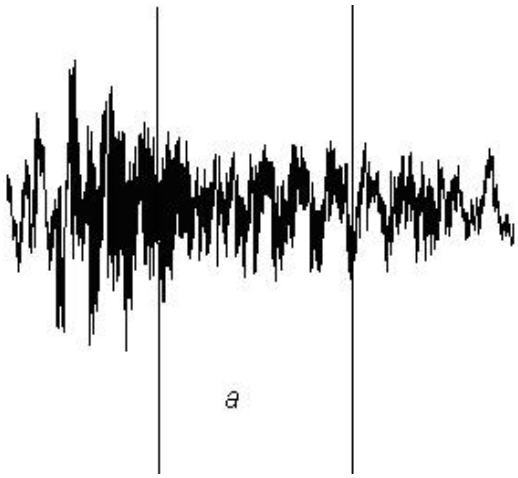

Fig: 3.2.b: "Av" in word "cvc" by male speaker

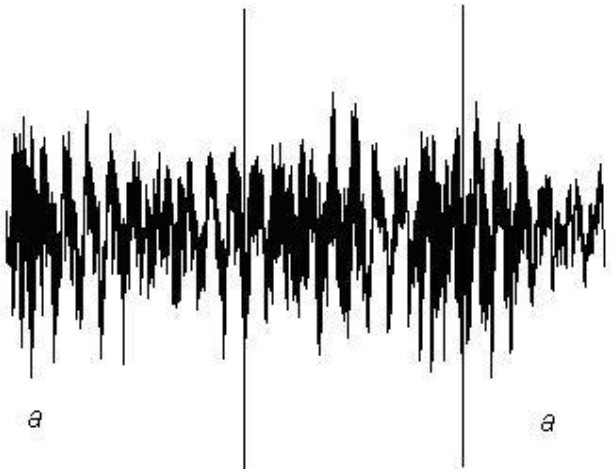

Fig 3.2.c: "Av" in word "Avkv" by male speaker

Table-3.2 List of vowel phonemes

A/O /	Aw/ a /	B-C/ i /	D-E/ u /	F/ ri /
G/e /	H/ oi /	I/ o /	J/ ou /	

3.3 Consonants phonemes

Consonant's sounds are irregular and short in duration. They vary through out the pronunciation. Consonants are result of being obstructed by mouth and narrowing the sounds by dental, alveolar of human when producing the sound [2]. Consonant phonemes are classified by the place of pronunciation or sounds produced when pronounced. Fig shows the utterance of consonants ÓKÓ (fig-3.3.a), ÓUÓ (fig-3.3.b). Bengali has following consonants listed in the table.

Fig 3.3.a: consonants "K" by male speaker

Fig 3.3.b: Consonants "U" by male speaker

Table-3.3: List of consonants phonemes

Manner of articulation		**Place of articulation**										
		Velar		**Palato-alveolar**		**Alveolar**		**Dental**		**Bilabial**		**Glottal**
		In-aspirate -Aíc\Y	Aspirate -gnwc\Y	In-aspirate –Aíc\Y	Aspirate -gnwc\Y	In-aspirate –Aíc\Y	Aspirate -gnwc\Y	In-aspirate -Aíc\Y	Aspirate -gnwc\Y	In-aspirate –Aíc\Y	Aspirate -gnwc\Y	
Plossive/ Stop	Voiceless	K/ k	L/ k^h	P/ c	Q/ c^h	U/ τ	V/ τ□	Z/ τ∉	_/ τ∉ □	c/ π	d/ π□	
	Voiced	M/ g	N/ g^h	R/ □	S/ □	W/ δ	X/ δ□	`/ δ∉	a/ δ∉ □	e/ β	f/ β□	
Fricative	Voiceless			k,l /Σ		m/ σ						t
	Voiced											n /η
Nasal	Voiced	O s/ N		T/ ϑ		b/ ν, Y/ ν				g/μ		
Liquid/ Lateral	Voiced			h/ □		j/ λ						
Trill	Voiced					i/ ρ						
Flapped	Voiced					o/ 4	p/ 4					
Glide	Voiceless			q/ φ								
	Voiced									e/ ω		

3.4 Vowels phoneme clusters/diphthongs

Vowel phoneme clusters are gliding monosyllabic sounds those start at or near the articulatory position for one vowel and moves to the position of another vowel. In Bengali language there are 19 regular and 12 irregular vowel clusters. Vowel clusters are just combination two vowel phonemes. And they are longer than a vowel. But separately vowels are longer than what they are in vowel clusters. Following figure shows the vowel cluster ÓGBÓ and ÓGÓ (fig-3.4.a), ÓBÓ (fig-3.4.b) differently. The phoneme cluster ÓGBÓ (fig-3.4.c) is formed from ÓGÓ, and ÓBÓ.

For recognition of speech the phoneme clusters may be considered. But most recent recognizers do not have separated pattern for phoneme clusters. Single vowel is considered and recognized to recognize the entire words. Table shows the vowel phonemes clusters [4].

Fig-3.4.a: “G” wave signal by male speaker

Fig-3.4.b: “B” by a male speaker

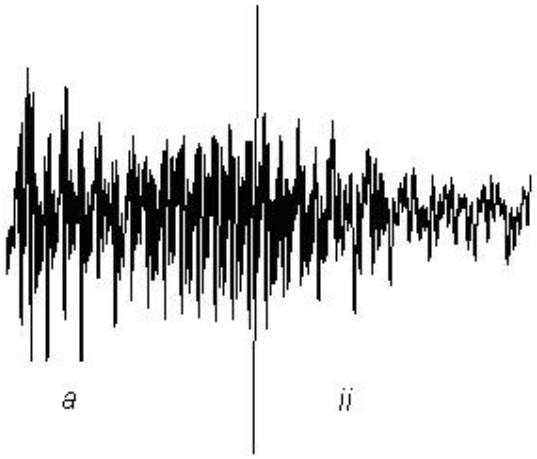

Fig-3.4.c: “G” and “B” in word “G B” by same male speaker

3.5 Consonants phoneme clusters

In Bengali there are 36 consonant phoneme clusters. Without some phoneme clusters most of them are used to place themselves in the middle of a word. So there is no option to be longer or shorter in sounds in end of a word. But consonants sounds in the middle sometimes pronounced as separate consonant phoneme. Most of the time, they are pronounced as phoneme cluster [2].

For recognition of phoneme clusters same strategy is taken as vowel phonemes clusters. Separately each phoneme is recognized, and then combines into cluster. Table shows the list of consonants phoneme clusters.

3.6 Voiced and Non-voiced stops

Voice stops sounds are transient in nature, no continuant sounds produced by building up pressure in the oral contract and suddenly releasing the pressure. Most of the time small amount of low frequency energy is generated. All the stops are dynamic in nature and their properties are highly influenced by the vowels that follow the stop [1].

Unvoiced stops are similar as voiced counterpart without that when the pressure is released, vocal cord does not vibrate. And has low energy than voiced stops [1].

Figure shows the voiced stops and Non-voiced stop. Voiced stops followed by the words ÓAvwgÓ and ÓAvRÓ and non-voiced stops followed by the word ÓAvmeÓ in the sentence ÓAvwg AvR AvmeÓ.

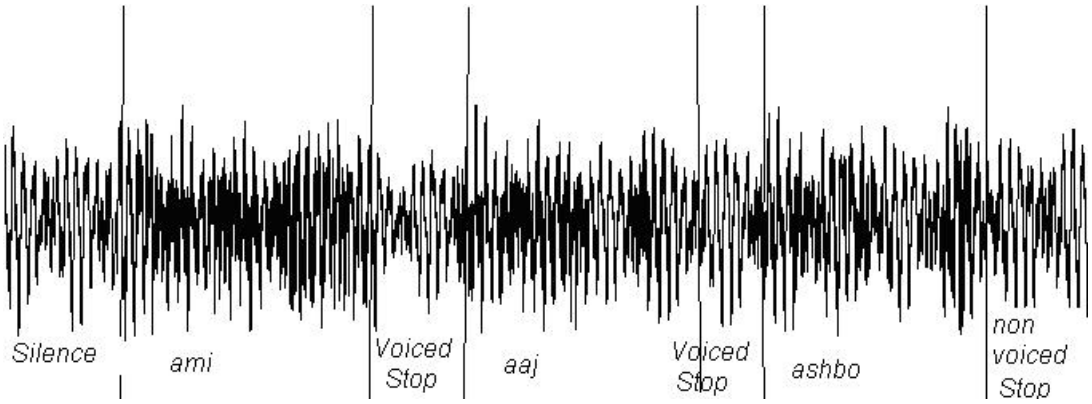

Fig:3.6 Voiced and Non-voiced stops in the sentence-“Avwg AvR Avme”

4. Hidden Markov Model-HMM

4.1 Hidden Markov Model

Hidden Markov Model is defined as triple **λ=(π,A,B)** where the symbol used have following meanings.

λ= model

π= an initial probability distribution over states **π,** such that π_i is the probability that HMM will start at the state i. Obviously for some i, $\pi_i=0$ because they can never be a start state in the model **λ.**

A= a set of probabilities $A= a_{01}, a_{02}\ldots a_{nm}$. Each a_{ij} represents the probability of transition from *i* state to state *j*. The set of these is ***transition probability matrix***.

B= confusion matrix is the set of observation sequence. A set of observation likelihood $B=b_i(o_i)$, each expressing the probability of o_i is generated from a state ***i***.

4.2 Three Basics Problems of HMM models

For the HMM model to be useful in the real world, three basic problems of it must be solved. The problems are following

4.2.1 Evaluation Problem

Given a observation sequence ***O=(o,o,…,o)*** and a model **λ=(π,A,B)** how to evaluate **P(O/ λ)** efficiently, the probability of the observation sequence, given the model [6].

4.2.2 Decoding problem

How do we choose the optimal state sequences ***q=(****q,q,q,….,q)* given the observation sequence ***O=(o,o,..,o)*** and the model **λ.** [1]

4.2.3 Learning problem

To maximize the ***P(O/ λ)*** how do we adjust the model parameters **λ=(π,A,B).** [6]

4.3 Solution to the problems

4.3.1 Solution of evaluation problem

Given a model and observation sequence how to calculate the probability of the observation sequence to get the correct result. For correctly evaluating the observation sequence ***Forward algorithm*** is used [6].

We consider a forward variable αt(i)=***P(o1,o2,….., qt=i| λ)------ (4.1)***
αt(i) is the probability of the partial observation sequence ***o1,o2,….ot*** until the time *t,* given the model **λ.** Then the problem is solved by the following way

Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \le i \le N. \qquad \text{---------- (4.3.1.1)}$$

Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), \quad 1 \le t \le T-1, \quad 1 \le j \le N. \qquad \text{--- (4.3.1.2)}$$

Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \qquad \text{---------- (4.3.1.3)}$$

4.3.2 Solution of decoding problem

To find the optimal state sequences ***q=(****q,q,q,….,q)* given the observation sequence ***O=(o,o,..,o)*** and the model **λ** ***Viterbi Algorithm is used.*** Viterbi algorithm gives the single best state sequences ***q=(****q,q,q,….,q) for the* given observation sequence ***O=(o,o,..,o)*** and the model **λ.** [1] We need to find the best probability along a single path at a time. The quantity is defined by

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P[q_1\, q_2 \cdots q_t = i,\, O_1\, O_2 \cdots O_t | \lambda] \quad \text{---------- (4.3.2.1)}$$

δt(i) is the best score at time t. By induction can be get

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] \cdot b_j(O_{t+1}). \quad \text{---------- (4.3.2.2)}$$

Viterbi algorithm works by following steps

Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), \quad 1 \le i \le N$$
$$\psi_1(i) = 0. \quad \text{---------- (4.3.2.3)}$$

Recursion:

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), \quad 2 \le t \le T, \quad 1 \le j \le N$$
$$\psi_t(j) = \operatorname*{argmax}_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}], \quad 2 \le t \le T, \quad 1 \le j \le N. \quad \text{---------- (4.3.2.4)}$$

Termination:

$$P^* = \max_{1 \le i \le N} [\delta_T(i)]$$
$$q_T^* = \operatorname*{argmax}_{1 \le i \le N} [\delta_T(i)]. \quad \text{---------- (4.3.2.5)}$$

State sequence Backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \cdots, 1. \quad \text{---------- (4.3.2.6)}$$

4.3.3 Solution of learning problem

It is most important to adjust the model parameters **λ = (π,A,B).** Re-estimation procedure is used to adjust the model parameters. The model parameters are chosen such that the probability of ***P (O/ λ)*** increases. Iterative process is used to estimate the model parameters and then re-estimate again until certain convergence criterion is met.

5. Building Recognizer with HTK

5.1. Bengali segmented speech recognition system:

This is limited word speech recognition system for Bengali. The working steps of the recognition system with HTK toolkit is described here. One can easily work out with HTK if he follows the following steps:

5.1.1 Construction steps

The main construction steps are the following:

1. Creation of a training database: each element of the vocabulary is recorded several times, and labeled with the corresponding word.
2. Acoustical analysis: the training waveforms are converted into some series of coefficient vectors.
3. Definition of the models: a prototype of Hidden Markov Model (HMM) is defined for each element of the task vocabulary.
4. Training of the models: each HMM is initialized and trained with the training data.
5. Definition of the task: the grammar of the recognizer (what can be recognized) is defined.
6. Recognition of an unknown input signal.

5.1.2 Work space organization

It is recommended to create a directory structure such as the following:

- data/ : to store training and test data (speech signals, labels, etc.), with 2 sub-directories
- data/train/ and data/test/ to separate the data used to train the recognizer from the ones used for performance evaluation.
- analysis/ : to store files that concern the acoustical analysis step.
- training/ : to store files that concern the initialization and training steps.
- model/ : to store the recognizer's models (HMMs).
- def/ : to store files that concern the definition of the task.
- test/ : to store files that concern the test.

5.2 Creation of training data set:

First we have record the speech signals for every word models I have in my dictionary for recognizing. With these speech signals the recognizer will be trained. Each signal has to be labeled that's associated with a text (a label) describing its content. Recording and labeling is done with the HSLab tool of HTK. We have to follow procedure 2.1 and 2.2 from the command line for each signal to be recorded.

To create and label a speech file use the following command in the command-prompt

HSLab ami.sig

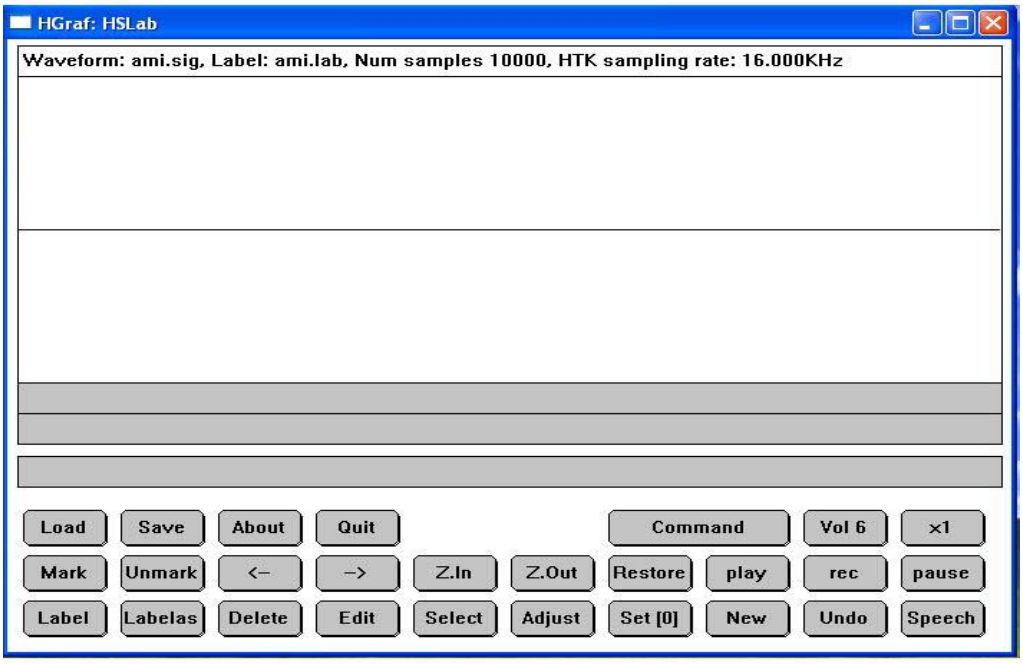

Fig-5.2: HSlab interface

A graphical interface will appear like above after the command. Now we can record speech signal file (*.sig) and create associated label file (*.lab).

5.2.1 Record the Speech Signal

Press "*Rec*" button to start recording the signal, then press "*Stop*" when the word is uttered. A buffer file called ami_0.sig is automatically created in the current directory.

- The signal files (.sig) are here saved in a specific HTK format. It is however possible to use other audio format (.wav, etc.):
- The default sampling rate is 16 *kHz*.

5.2.2 Label the Signal

To label the speech waveform, first press "*Mark*", then select the region you want to label. Then one has to select the region two points A and B. When the region is marked, press "*Labelas*", type the name of the label, then press *Enter*. For every word speech signal there may be couple of "sil" label preceding the main word like in the example "ami" and one or more "sil" label after the label "ami". These regions cannot overlap with each other (but no matter if there is a little gap between them).When the labels have been written, press "*Save*": a label file called ami_0.lab is created. At this point you can press "*Quit*".

***Remark*:**

The ami.lab file is a simple text file. It contains for each label a line of the type:

4171250 9229375 sil

9229375 15043750 ami

15043750 20430625 sil

Where numbers indicate the start and end sample time of each label. Such a file can be modified manually.

The signal files should be stored in a data/train/sig/ directory (the training corpus), the labels in a data/train/lab/ directory (the training label set).

The total process can be represented by following diagram:

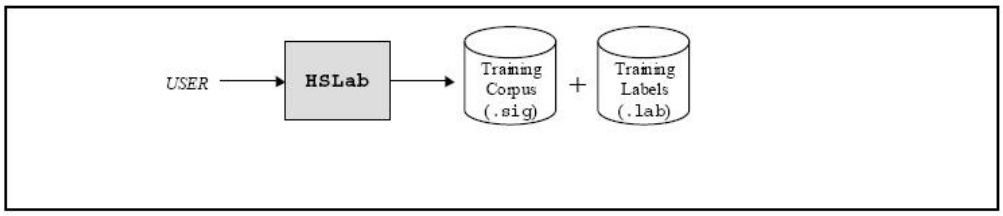

Fig-5.2.2: Recording and labeling training data [4].

5.3 Acoustical Analysis of Speech Signal

The speech recognition tools cannot process directly on speech waveforms. These have to be represented in a more compact and efficient way. This step is called “acoustical analysis”:

- The speech signal is segmented in successive frames (whose length is chosen between 20ms and 40ms, typically), overlapping with each other.
- Each frame is multiplied by a windowing function (e.g. Hamming function).
- A vector of acoustical coefficients (giving a compact representation of the spectral properties of the frame) is extracted from each windowed frame.

The conversion from the original waveform to a series of acoustical vectors is done with the HCopy HTK tool: The following command is needed to extract acoustic vectors.

HCopy -A -D -C analysis.conf -S targetlist.txt

analysis.conf is a configuration file setting the parameters of the acoustical coefficient extraction.

targetlist.txt specifies the name and location of each waveform to process, along with the name and location of the target coefficient files.

5.3.1 Configuration Parameters

The configuration file is a text file named **analysis.conf**. # sign is used before any sentence is known comments. The configuration file used has the following configuration

```
#
# Example of an acoustical analysis configuration file
#
SOURCEFORMAT = HTK        # Gives the format of the speech files
TARGETKIND = MFCC_0_D_A # Identifier of the coefficients to use
                          # Unit = 0.1 micro-second :
WINDOWSIZE = 250000.0     # = 25 ms = length of a time frame
TARGETRATE = 100000.0     # = 10 ms = frame periodicity
NUMCEPS = 12              # Number of MFCC coefficient (here from
c1 to c12)
USEHAMMING = T            # Use of Hamming function for windowing
frames
PREEMCOEF = 0.97          # Pre-emphasis coefficient
NUMCHANS = 26             # Number of filter bank channels
CEPLIFTER = 22            # Length of Cepstral filtering
# The End
```

analysis.conf

With the configuration file, an MFCC (Mel Frequency Cepstral Coefficient) analysis is performed (prefix “MFCC” in the TARGETKIND identifier). For each signal frame, the following coefficients 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)

Altogether, a 39 coefficient vector is extracted from each signal frame.

5.3.2 Source / Target Specification- targetlist.txt

One or more “source file / target file” pairs (i.e. “original waveform / coefficient file”) can be directly specified in the command line of HCopy. When many data are to be processed, the –S option is used instead. It allows specifying a script file of the form: The file is done manually.

targetlist.txt

```
data/train/sig/ami_0.sig data/train/mfcc/ami_0.mfcc
data/train/sig/ami_1.sig data/train/mfcc/ami_1.mfcc
---
data/train/sig/laav_4.sig data/train/mfcc/laav_4.mfcc
data/train/sig/lavv_5.sig data/train/mfcc/laav_5.mfcc
```

The new training corpus (*.mfcc files) is stored in the data/train/mfcc/ directory. The total acoustical analysis process can be shown by the diagram.

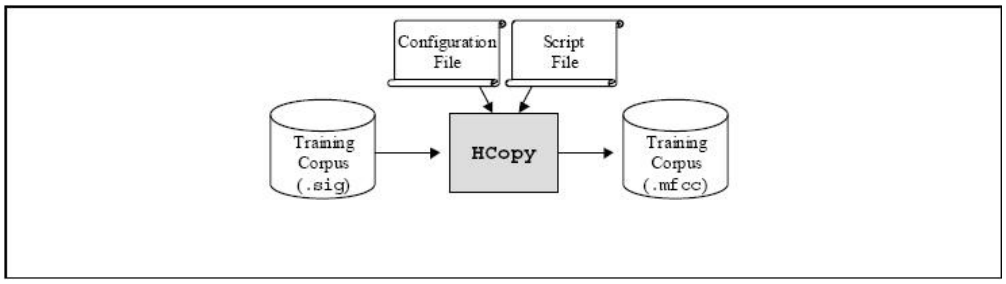

Fig-5.3.2: Conversion of the training data [4].

5.4 Defining HMM for word model

For this recognizer 102 (101 words + sil) acoustical events have to be modeled with a Hidden Markov Model (HMM): “ami”, “laav” and “*sil*”. For each one we will design a HMM.

The first step is to choose *a priori* a topology for each HMM:

- number of states
- form of the observation functions (associated with each state)

- disposition of transitions between states

Such a definition is not straightforward. There is actually no fixed rule for it. Here, we will simply choose the same topology for each of the 102 HMMs (Fig-5.4):

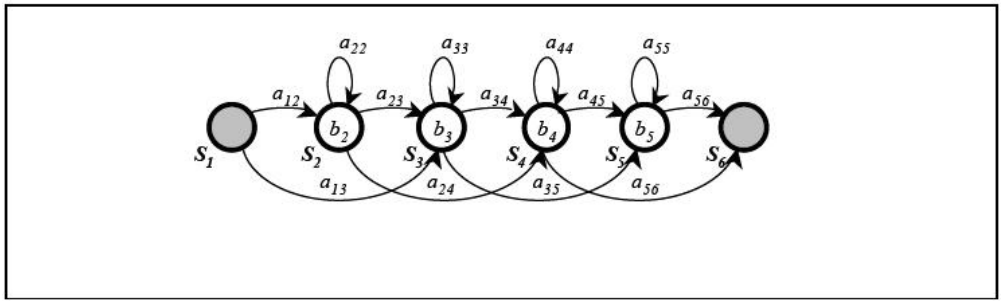

Fig-5.4: Basic topology of HMM [4].

The models consist actually of 4 “active” states {*S2*, *S3*, *S4*, *S5*}: the first and last states (here *S1* and *S6*), are “non emitting” states (no observation function), only used by HTK for some implementation facilities reasons. The observation functions *bi* is single Gaussian distributions with diagonal matrices. The transition probabilities are quoted *aij*.

In HTK, a HMM is described in a text description file. The description file for the HMM depicted on Fig.**3** is of the form:

ami.txt

```
~o <VecSize> 39 <MFCC_0_D_A>
~h "ami"
<BeginHMM>
<NumStates> 6
<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
<State> 5
<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> 6
0.0 0.5 0.5 0.0 0.0 0.0
0.0 0.4 0.3 0.3 0.0 0.0
0.0 0.0 0.4 0.3 0.3 0.0
0.0 0.0 0.0 0.4 0.3 0.3
0.0 0.0 0.0 0.0 0.5 0.5
0.0 0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

HMM description file (prototype)

Description of HMM:

~o <VecSize> 39 <MFCC_0_D_A>

is the header of the file, giving the coefficient vector size (39 coefficients here), and the type of coefficient (MFCC_0_D_A here).

~h "ami" <BeginHMM> (...)<EndHMM>

encloses the description of a HMM called “ami”.

<NumStates> 6

gives the total number of states in the HMM, including the 2 non-emitting states 1 and 6.

<State> 2

introduces the description of the observation function of state 2. Here we have chosen to use single-gaussian observation functions, with diagonal matrices. Such a function is entirely described by a mean vector and a variance vector (the diagonal elements of the autocorrelation matrix). States 1 and 6 are not described, since they have no observation function.

<Mean> 39

0.0 0.0 (...) 0.0 (x 39)

gives the mean vector (in a 39 dimension observation space) of the current observation function. Every element is arbitrary initialized to 0: the file only gives the “prototype” of the HMM (its global topology). These coefficients will be trained later.

<Variance> 39

1.0 1.0 (...) 1.0 (x 39)

gives the variance vector of the current observation function. Every element is arbitrary initialized to 1.

<TransP> 6

gives the 6x6 transition matrix of the HMM, that is: where *aij* is the probability of transition from state *i* to state *j*. Null values indicate that the corresponding transitions are not allowed. The other values are arbitrary

initialized (but each line of the matrix must sum to 1): they will be later modified, during the training process.

Such a prototype has to be generated for each event to model. In our case, we have to write a prototype for 102 HMMs that we will call “ami”, “laav”, and “*sil*” and so on (with headers ~h "ami", ~h "laav" and ~h "sil" in the 102 description files).

These 102 files could be named ami.txt, laav.txt, sil.txt and be stored in a directory called- **model/proto/**

<TransP>6

a11 a12 a13 a14 a15 a16

a21 a22 a23 a24 a25 a26

a31 a32 a33 a34 a35 a36

a41 a42 a43 a44 a45 a46

a51 a52 a53 a54 a55 a56

a61 a62 a63 a64 a65 a66

5.5 HMM training

First we have to initialize each HMM with the tool **HInit** and then train them with train set with the **HRest** tool. The entire training procedure is described by the following diagram.

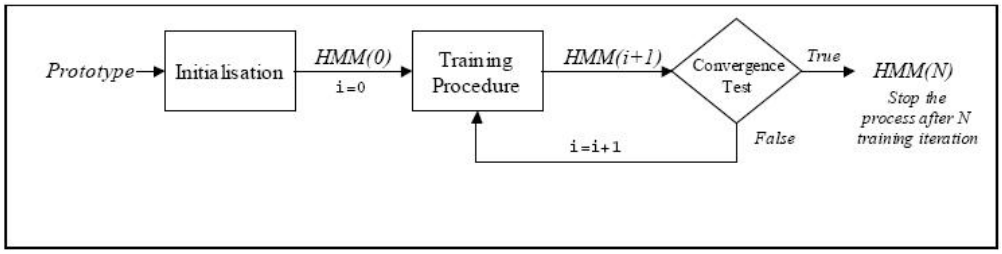

Fig-5.5: Complete training procedure [4].

5.5.1 Initialization

For a fast and precise convergence of the training algorithm the HMM parameters must be initialized with the training data corpus. We are doing this initialization with the tool **HInit**.

The following command line initializes the HMM by time-alignment of the training data with a Viterbi algorithm:

HInit -A -D –T 1 -S trainlist.txt -M model/hmm0 –H model/proto/protohmm -l label -L label_dir hmm_name

hmm_name is the name of the HMM to initialize (here: ami, laav, sil and so on).
protohmm is a description file containing the prototype of the HMM called protohmm (here: proto/ami, proto/laav, proto/sil and so on).
trainlist.txt gives the complete list of the .mfcc files forming the training corpus (stored in directory data/train/mfcc/).
label_dir is the directory (data/train/lab/) where the label files (.lab) corresponding to the training corpus.
label indicates which labeled segment must be used within the training corpus (here: ami, laav, or sil because have used the same names for the labels and the HMMs, but this is not mandatory…)
model/hmm0 is the name of the directory (must be created before) where the resulting initialized HMM description will be output.
This procedure has to be repeated for each model (ami, laav, sil, and so on).

***Remark*:**
The HMM file output by **HInit** has the same name as the input prototype.

The initialization process by **HInit** tool is can be showed by following diagram:

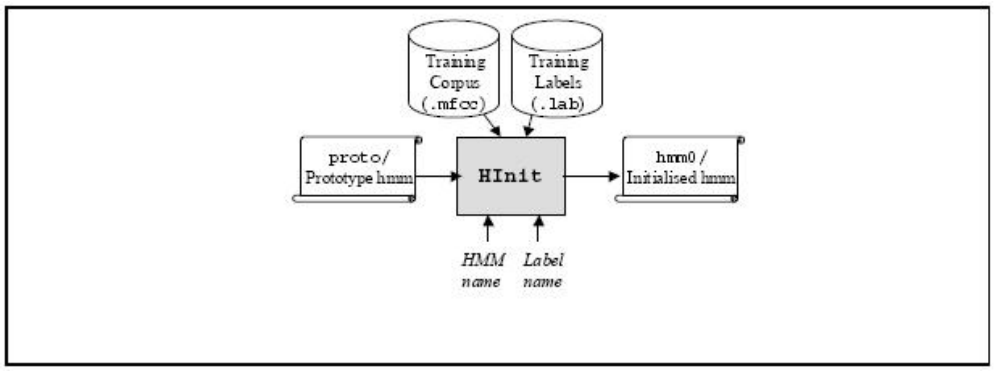

Fig-5.5.1: Initialization from a prototype [4].

5.5.2 Training

Training procedure is done with HTK tool **HRest.** The following command line performs one re-estimation iteration with HTK tool HRest, estimating the optimal values for the HMM parameters (transition probabilities, plus mean and variance vectors of each observation function):

HRest -A -D -T 1 -S trainlist.txt -M model/hmm*i*
-H model/hmm*i-1*/hmmfile -l label -L label_dir hmm_name

hmm_name is the name of the HMM to train (here: yes, no, or sil).
hmmfile is the description file of the HMM called hmm_name. It is stored in a directory whose name indicates the index of the last iteration (here model/hmm*i-1*/ for example).
trainlist.txt gives the complete list of the .mfcc files forming the training corpus (stored in directory data/train/mfcc/).

label_dir is the directory where the label files (.lab) corresponding to the training corpus (here: data/train/lab/).
label indicates the label to use within the training data (yes, no, or sil)
model/hmm*i* , the output directory, indicates the index of the current iteration ***i***.

Diagram of training with **HRest**

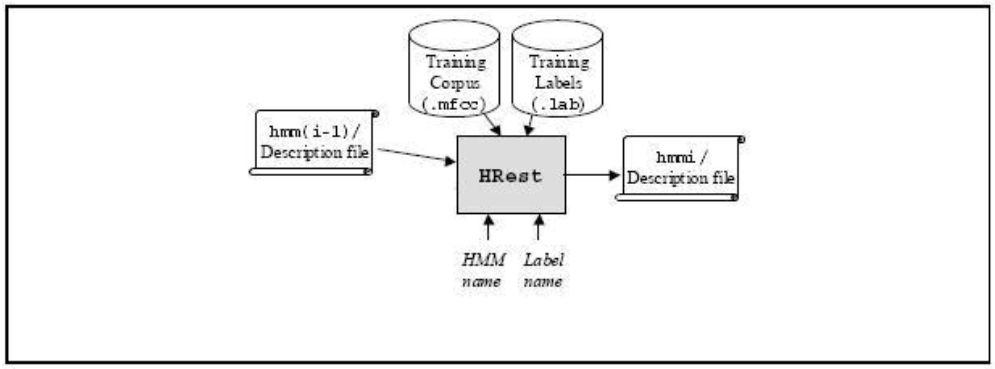

Fig-5.5.2: A Re-estimation process [4].

This procedure has to be repeated several times for each of the HMM to train. Each time, the HRest iterations (i.e. iterations within the current re-estimation iteration…) are displayed on screen, indicating the convergence through the change measure. As soon as this measure do not decrease (in absolute value) from one HRest iteration to another, it's time to stop the process. In our recognizer we did it for 5 times.

The final word HMMs are then: hmm5/ami.txt, hmm5/laav.txt, hmm5/sil.txt.

5.6 Recognizer Task

Definition of the recognition task contains the words list, the recognition grammar and word network created from the task grammar.

5.6.1 Task Grammar

Task grammar is defined with the words list and written in a text file. For segmented speech recognition the grammar is very simple.

```
$WORD = AMI | TUMI | KI | KORA | PORA | BOLA | KORI | PORI | BOLI |
JAUA | JABE | KHAUA | KHABE | CHHARA | KOTHA | SHE | AMRA |
AMAKE | TOMAKE | TOMRA | ASHO | ESHO | GHOR | BOSHA | BOSHI |
VALO | SOMOY | SHUKH | KOSHTO | CHOLO | CHOLA | CHOLCHHE |
DIN | KAL | KHARAP | KHABAR | PAGOL | PAUA | MANUSH | JONNO |
MANUSHER | JIBON | VALOBASHA | GHURA | GHURE | MATHA |
JOKHON | TOKHON | AI | SHEI | AAJ | KALKE | PROTIDIN | BHOY |
SAHOS | AGAMI | MAAR | MARA | UPAR | NICH | UTHA | NAMA | KOTO
| HOYECHHE | SHURU | VASHA | GOTHON | TOIRI | PHOL | HOTE |
PARE | HOLE | NAA | HAA | MOTO | PURBO | MUKH | TAAR | TOBU |
HOY | MOJA | JANAM | DAAT | THAKE | SHOBDO | MULLO | JAY |
SHONA | NIYOM | AMADER | SOTTI | PROTHOM | VITOR | DEKHI |
MEYE | CHHELE | KHAI | PAAP | ASHA | EK | LAAV | GAAN;

( { START_SIL } $WORD  { END_SIL } )
```

Banglagrammar.txt

The WORD variable can be replaced by any of the words given there.

The brackets {} around START_SIL and END_SIL denotes zero or more repetitions (a long silence segment, or no silence at all before or after the word are then allowed).

The WORD variable must be present once.

There are some extra rules also like:

- [$VAR] means zero or one instance of VAR
- <$VAR> means one or more instance of VAR

5.6.2 Task Dictionary

The Recognition system must of course know to which HMM corresponds each of the grammar variables AMI, TUMI, START_SIL and END_SIL …. This information is stored in a text file called the *task dictionary*. In such a simple task, the correspondence is straightforward, and the task dictionary simply encloses the 103 entries:

```
START_SIL   []          sil
END_SIL     []          sil
AMI         [AMI]       ami
TUMI        [TUMI]tumi
KI          [KI]        ki
KORA        [KORA]      kora
PORA        [PORA]      pora
BOLA        [BOLA]      bola
…….
…….
…….
…….
etc….
…….
…….
…….
KORI        [KORI]      kori
PORI        [PORI]              pori
BOLI        [BOLI]      boli
JAUA        [JAUA]      jaua
JABE        [JABE]      jabe
KHAUA       [KHAUA]     khaua
KHABE       [KHABE]     khabe
CHHARA      [CHHARA]    chhara
KOTHA       [KOTHA]     kotha
SHE         [SHE]       she
AMRA [AMRA]       amra
```

Bangladict.txt

The left elements refer to the names of the task grammar variables. The right elements refer to the names of the HMMs (introduced by ~h in the HMM definition files). The bracketed elements in the middle are

optional; they indicate the symbols that will be output by the recognizer: (by default, the names of the grammar's variables would have been used.)

***Remark*:**

Don't forget the new line at the end of the file (if not, the last entry is ignored).

5.6.3 Network

The recognizer's task grammar (described in file **banglagrammar.txt**) has to be compiled with tool HParse, to obtain the *task network* (written in **banglanet.slf**):

HParse -A -D -T 1 banlagrammar.txt banglanet.slf

At this stage, our speech recognition task (Fig.7), completely defined by its **network**, its **dictionary**, and its **HMM set** (the 103 models stored in model/hmm5/), is ready for use.

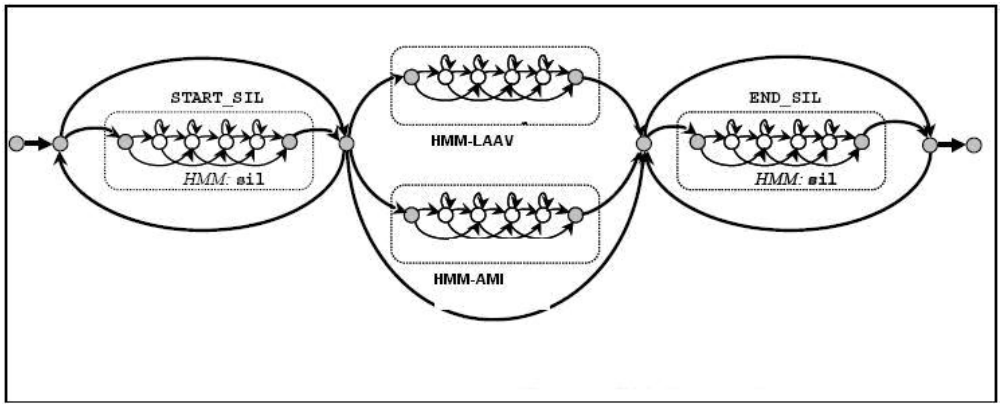

Fig-5.6.3: Recognizer= Network + Dictionary + HMM [4].

5.7 Recognition

The diagram follow describe the recognition procedure:

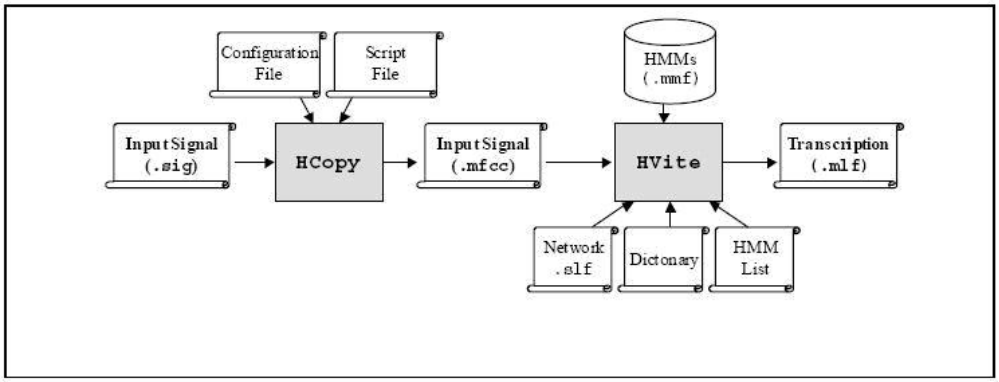

Fig-5.7: Recognition procedure of unknown signal [4].

Recognition procedure:

- An input speech signal input.sig is first transformed into a series of “acoustical vectors” (here MFCCs) with tool HCopy, in the same way as what was done with the training data (Acoustical Analysis step). The result is stored in an input.mfcc file (often called the *acoustical observation*).
- The input observation is then process by a Viterbi algorithm, which matches it against the recognizer’s Markov models. This is done by tool HVite: (one file at a time)

HVite -A -D -T 1 -H hmms.txt -i reco.mlf -w banglanet.slf bangladict.txt hmmlist.txt input.mfcc

input.mfcc is the input data to be recognised.

hmmlist.txt lists the names of the models to use (yes, no, and sil). Each element is separated by a new line character. Don’t forget to insert a new line after the last element.

bangladict.txt is the task dictionary.

banglanet.slf is the task network.

reco.mlf is the output recognition transcription file.

hmms.txt contains the definition of the HMMs. Gather every definitions in a single file. Such a file is simply obtained by copying each definition after the other in a single file, without repeating the header information.

```
~o <VecSize> 39 <MFCC_0_D_A>
~h "ami"
<BeginHMM>
        (definition...)
<EndHMM>

~h "laav"
<BeginHMM>
        (definition...)
<EndHMM>

~h "sil"
<BeginHMM>
        (definition...)
<EndHMM>
```

hmms.txt

We can recognize a list of file by following commands:

HVite -A -D -T 1 -S testlist.txt -H hmms.txt -i reco.mlf -w banglanet.slf bangladict.txt hmmlist.txt

testlist.txt – this file contains the list of file to test.

```
Data/test/ami.mfcc
data/test/tumi.mfcc
data/test/ki.mfcc
data/test/kora.mfcc
data/test/pora.mfcc
data/test/bola.mfcc
data/test/kori.mfcc
data/test/pori.mfcc
data/test/boli.mfcc
```

testlist.txt

The contents of reco.mlf is like follows

```
#!MLF!#
"data/test/ami.rec"
4500000 10200000 AMI -2895.303711
.
"data/test/tumi.rec"
3700000 10300000 TUMI -3155.571289
.
"data/test/ki.rec"
3800000 8900000 KI -2637.285156
.
"data/test/kora.rec"
6300000 12000000 KORA -2786.583252
.
"data/test/bola.rec"
2400000 8000000 BOLA -2850.089600
.
"data/test/kori.rec"
4700000 10700000 KORI -3075.503418
.
"data/test/pori.rec"
5800000 12600000 PORI -3458.541504
.
"data/test/boli.rec"
3000000 8300000 BOLI -2959.312988
.
```

reco.mlf

6 Results and Evaluation

The recognizer can recognize any word within its dictionary. For training set data it gives 100% success rate. Speech is so dynamic in nature, so result of the recognizers varies in nature of the signal environment. The recognizer is trained with 5 sample data for every model HMM. It will work better for more training data set. And for the test data recorded at the same time when training data sets are recorded gives 80% success rate. But for test signal recorded in noisy environment it gives only 50% success rate.

The model classifier gets confused with some words. They are like confused couple sets. Classifier recognizes one instead of another. Confusion created in "Awg" and "Zwg", "Avgv‡K" and "Avgv‡`i", "fq" and "nq". This is because of the probability for word models get confused when frame length is different in different context.

Test Data Set	Environment	Success Rate
Training Data Set	Quiet	100 %
Recorded with training Data Set Recorded	Quiet	80 %
Recorded Anytime	Quiet	74%
Recorded Later	Noisy	50 %

Table-Result of test data

Reference:

1. Fundamentals of Speech Recognition. By Lawrence Rabiner and Biing Hwang Juang.
2. Dhvani Vijnan O Bangla Dhvani Tattwa. By Muhammad Abdul Hai
3. Speech and Language Processing. By Daniel Jurafsky and James H. Martin.
4. The HTK Book. By Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev and Phil Woodland
5. An Outline of English Phonetics- Heffer
6. A Tutorial on Hidden Markov Model and Selected Applications of Speech Recognition- By Lawrence Rabiner.