Project Report

St. Xavier's College (Autonomous), Kolkata

SPEECH RECOGNITION USING MFCC AND HMM/VQ

By

SANTANU MONDAL (ROLL 549)

KALYAN MAJUMDAR (ROLL 553)

SRIJITA BISWAS (ROLL 575)

Under the guidance of

PROFESSOR JAYATI GHOSH DASTIDAR

B.SC. COMPUTER SCIENCE (HONS.)

SEMESTER – VI

DISSERTATION PAPER

CERTIFICATE OF AUTHENTICITY

This is to certify that the project report entitled **'SPEECH RECOGNITION USING MFCC AND HMM/VQ'** submitted to Department of Computer Science, **ST. XAVIER'S COLLEGE [AUTONOMOUS], KOLKATA**, in partial fulfilment of the requirement for the award of the degree of **BACHELOR OF SCIENCE (B.SC.)** is an original work carried out by Mr. Santanu Mondal, Registration no: A01-1112-0949-15, Mr. Kalyan Majumdar, Registration no: A01-1112-0951-15, and Ms. **Srijita Biswas**, Registration no: A01-2112-0967-15, under my guidance. The matter embodied in this project is authentic and is genuine work done by the student and has not been submitted whether to this College or to any other Institute for the fulfilment of the requirement of any course of study.

_____

Signature of Project Supervisor

Date:

ROLES AND RESPONSIBILITIES

Name of Project: Speech Recognition using MFCC and HMM/VQ

Date: 03/04/2018

Sl. No.	Roll No	Name of Team Member	Task and Responsibilities
1	549	Santanu Mondal	GUI Designing (Front-end)
2	553	Kalyan Majumdar	MFCC, HMM, VQ Algorithms (Back-end)
3	575	Srijita Biswas	Application Testing and Debugging

Name and Signature of Project Team Members:

NAME	SIGNATURE
SANTANU MONDAL	
KALYAN MAJUMDAR	
SRIJITA BISWAS	

_____

Signature of the Professor

Date:

ACKNOWLEDGEMENT

I would like to express my gratitude to all those who have helped me and guided me through the project.

I am thankful to Rev. Fr. Dr. J. Felix Raj, Vice Chancellor, St. Xavier's University, Kolkata and erstwhile Principal of St. Xavier's College (Autonomous), Kolkata, Rev. Fr. Dominic Savio, Principal of the Institution, Prof. Shalabh Agarwal, Head of the Department of Computer Science, and my mentor Prof. Jayati Ghosh Dastidar, without whose help my project would have remained incomplete.

We thank the Head of the Department of Computer Science, Prof. Shalabh Agarwal, for giving us this opportunity to work on this project, and thereby, exposing us to a real-life application of Speech Processing and Audio Technologies.

We thank our mentor, Prof. Jayati Ghosh Dastidar, for guiding us through the process of background study, analyses, coding and implementation of this project, and for providing us interactive lectures on the same.

I would like to immensely thank all the teaching and non-teaching staff of the Department of Computer Science for their constant support and help without whom the project could not have been completed.

_____ _____ _____

(SANTANU MONDAL) (KALYAN MAJUMDAR) (SRIJITA BISWAS)

Date: 03/04/2018

ABSTRACT

The two main principles involved in the implementation of a Sound Recognition System that identifies voices, music or, as in our case, speech, are –

(i) Audio Content, and
(ii) Identity of the Object (in our case, isolated words).

The code developed takes an audio input, applies MFCC algorithm on it, compares them with those of the contents of a database containing sound samples of various isolated words in English Language, and presents as output the word in text which matches the audio input.

The input from testing was gathered and meaningful spectral coefficients were extracted. These coefficients are stored in a database for later comparison with future audio samples. Afterwards, the system captures the coefficients of the external sample and matches it with those present in the database.

MFCC algorithm is used to extract the spectral coefficients which are good and can be used for feature matching purpose discarding any static and background noise, if any.

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER - 1

INTRODUCTION

1.1 BACKGROUND

As a subfield of computational linguistics, the field of voice recognition has undergone tremendous research through the ages. It was first implemented in the single-speaker digit recognition systems developed by Bell Labs. The two main aspects of the work involve audio content and speaker/instrument identity.

This project takes audio inputs which are strictly English words, and converts the speech taken as input to text.

1.2 OBJECTIVES

The objective is to construct a code that will input an audio, recognise the patterns in the same, classify them and present as output in the form of text, the word that has been spoken.

1.3 PURPOSE

The purpose of this project is to create an application that will make the recognition of certain isolated words fast and easy. Since this is an implementation of speech recognition, it can find its uses in various fields.

1.4 SCOPE

The algorithm developed in this project can be used in various fields. This can be used by people not so familiar with the English tongue, to recognise words with a complicated pronunciation or fast spoken words. This can

also be used by differently abled people or people with special needs to issue voice commands. More advanced and refined versions of this application can be used to implement automated subtitle generation for video or audio files or taking down dictations.

1.5 ACHIEVEMENTS

The achievements of this project can be highlighted as follows:

i. It was successful in identifying the words spoken to it and return valid text outputs.

ii. It pointed out the limitations of using these algorithms in a noisy environment.

iii. It could be trained dynamically to accept new words and add them to its current vocabulary set.

CHAPTER – 2

SURVEY OF TECHNOLOGIES

Voice recognition has been growing as an application since very early in the twentieth century. The two major branches of the application include speech recognition and speaker recognition. Such disciplines have had their foundation in various technologies and have evolved into various advanced forms in modern technology.

The first single-speaker digit recognition system was first built by Bell Labs in 1932. This paved the way for technologies in the 1950s, where single-speaker systems came into use.

In 1960, the source-filter model of speech production came into being. From the ten-word vocabulary system in the 1950s systems, the dynamic time warping algorithm, developed by Soviet researchers in the 1960s, created a recognizer capable of operating on a vocabulary of two hundred words.

In the late 1960s, the mathematics of Markov chains came into existence, which further gave rise to the Hidden Markov Model for speech recognition.

By the 1990s, due to the swift growth of technology, speech recognition systems began to process vocabularies larger than that of the average human being.

In May 2013 Barclays Wealth was to use passive speaker recognition to verify the identity of telephone customers within 30 seconds of normal conversation. A verified voiceprint was to be used to identify callers to the system and the system would in the future be rolled out across the company.

We now have the speech recognition as something that is commercially successful, and has found its use in our everyday lives, the biggest example being Google's digital assistant, Google Assistant and Samsung's digital assistant, Bixby.

2.1 METHODS AND ALGORITHMS USED FOR FEATURE EXTRACTION AND SOUND CLASSIFICATION

a. Mel-frequency Cepstral Coefficients (MFCC)

The mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a non-linear mel scale of frequency.

Mel-frequency cepstral coefficients (MFCC) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip. The difference between cepstrum and the mel-frequency cepstrum is, that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum.

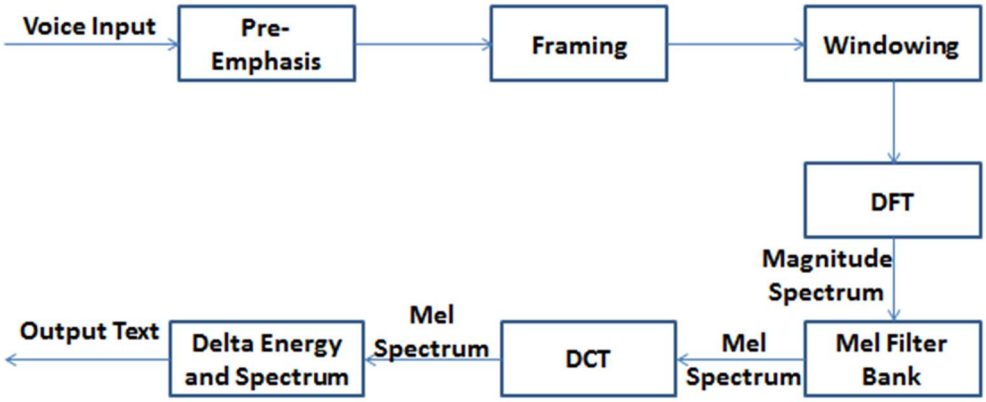

Figure 1: Block Diagram of MFCC

b. Linear Prediction Cepstral Coefficients (LPCC)

The theme behind linear prediction cepstral coefficients (LPCC) is that one speech sample at the current time can

be predicted as a linear combination of past speech samples.

Since the energy contained within a speech signal is distributed more in the lower frequencies than in the higher frequencies, the input signal is first pre-emphasized using a first order high pass filter.

One of the most efficient methods used for estimating the linear prediction cepstral coefficients and the filter gain is the autocorrelation method. Autocorrelation technique is almost an exclusively used method to find the correlation between the signal and itself by auto-correlating each frame of the windowed signal using
following equation

$$R[i] = \sum_{n=1}^{N_w-1} s_w(n) \, . \, s_w(n-1), 0 \leq i < p$$

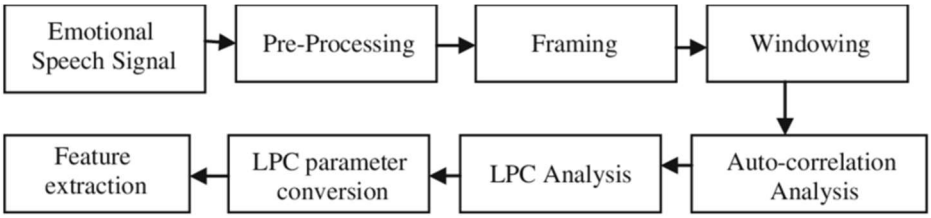

Figure 2: Block Diagram of LPCC

c. Perceptual Linear Predictive (PLP) Analysis

In the PLP technique, several well-known properties of hearing are simulated by practical engineering approximations, and the resulting auditory like spectrum of speech is approximated by an autoregressive all-pole model.

The flow of the process is given as follows:

SPEECH
|
CRITICAL BAND ANALYSIS
|
EQUAL LOUDNESS; PRE-EMPHASIS
|
INTENSITY-LOUDNESS CONVERSION
|
INVERSE DISCRETE I FOURIER TRANSFORM
|
SOLUTION FOR AUTO-REGRESSIVE COEFFICIENTS
|
ALL-POLE MODEL

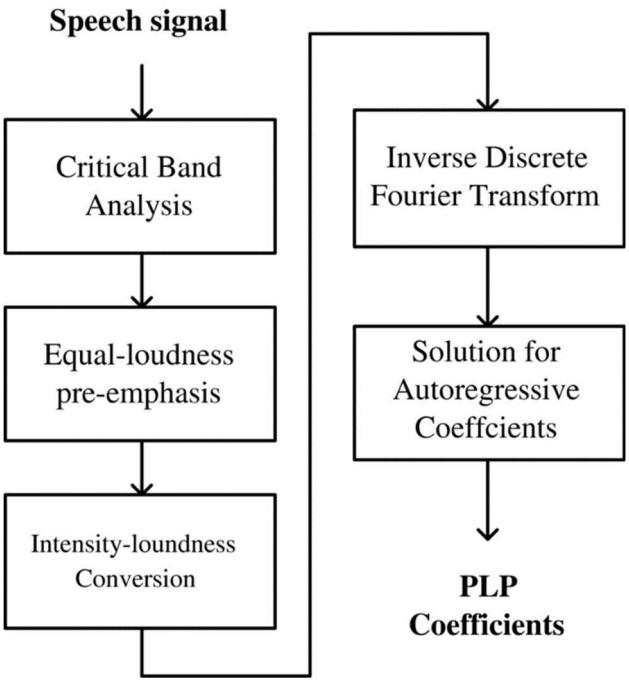

Figure 3: Block Diagram of PLP

d. Hidden Markov Model (HMM)

Hidden Markov Model (HMM) is a statistical Markov model in which the system being modelled is assumed to be a Markov process with unobserved (i.e. hidden) states.

The HMM can be represented as the simplest dynamic Bayesian network. The mathematics behind the HMM were developed by L. E. Baum and co-workers. HMM is closely related to an earlier work on the optimal non-linear filtering problem by Ruslan L. Stratonovich, who was the first to describe the forward-backward procedure.

In simpler Markov models (like a Markov chain), the state is directly visible to the observer and therefore the state transition probabilities are the only parameters, while in the HMM, the state is not directly visible, but the output (in the form of data or "token" in the following), dependant on the state, is visible. Each state has a probability distribution over the possible output tokens generated by an HMM gives some information about the sequence of states; this is also known as pattern theory, a topic of grammar induction.

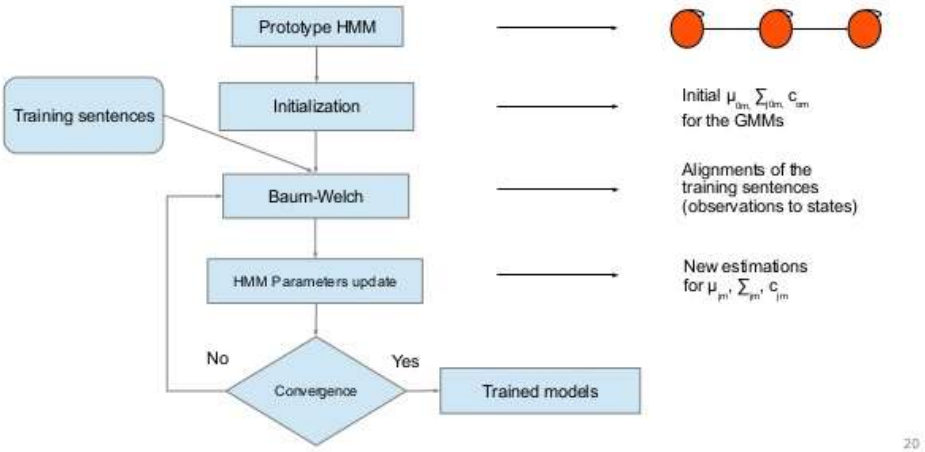

Figure 4: Block Diagram of HMM

e. Gaussian Mixture Model (GMM)

A Gaussian Mixture Model (GMM) is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with

unknown parameters. One can think of mixture models as generalising k-means clustering to incorporate information about the covariance structure of the data as well as the centres of the latent Gaussians.

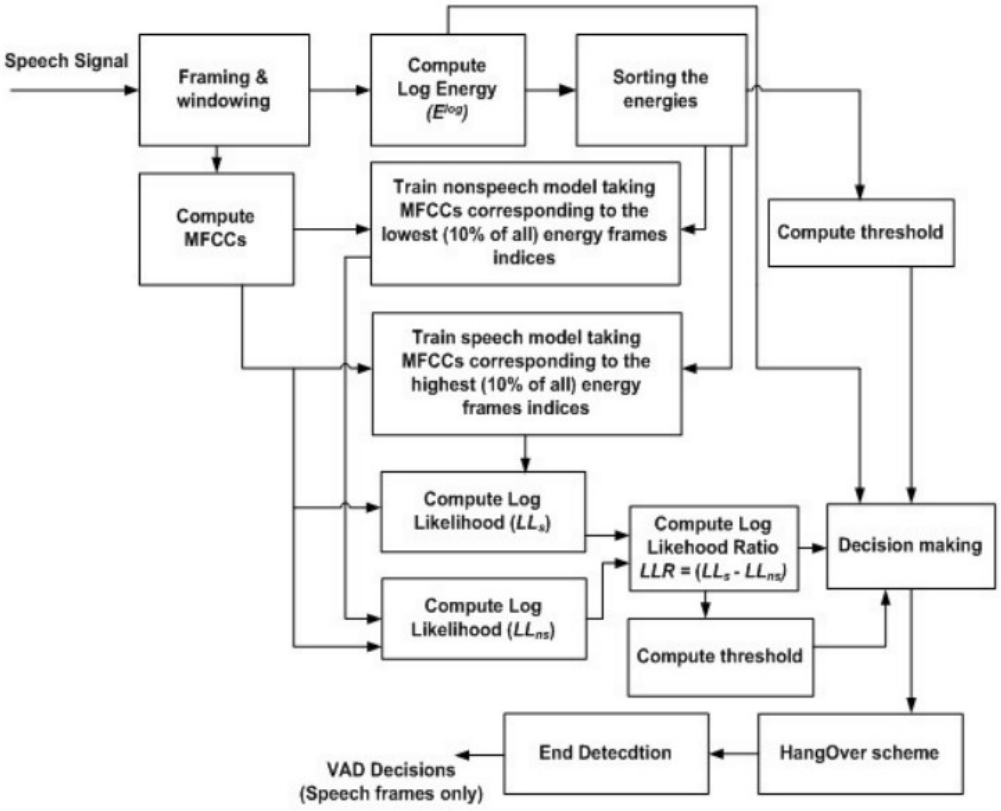

Figure 5: Block Diagram of GMM

CHAPTER – 3
REQUIREMENTS AND ANALYSIS

3.1 PROBLEM DEFINITION

The two main principles involved in the implementation of a voice recognition system are: (i) speech content, and (ii) speaker identity.

The code developed takes an audio input, applies MFCC algorithm on it, compares them with those of the contents of a database containing speech samples pre-recorded by different people, and presents as output the text which matches the audio input.

Thus, we have to design a code which can take external sample and after analyzing that sound, it should produce what word it is or which word matches closest to those present in the pre-recorded training set.

3.2 REQUIREMENT SPECIFICATION

This project matches the features of an external audio sample, with those which are present in the training set. In order to do so, MFCC algorithm has been used to extract unique features of the sample. To compare and match the features, Hidden Markov Model (HMM) has been used and the closest word matching is being shown.

- **Software Requirements**
  - Java JRE 1.5 and onwards
  - Windows XP and onwards/Linux/Mac OS
- **Hardware Requirements**
  - Laptop or Desktop

- Processor Clock Speed: 1.2GHz and above
- RAM: 2GB and above
- Microphone

3.3 PRELIMINARY PRODUCT DESCRIPTION

The final product is a software or an application or an interface to identify a particular isolated English word by extracting the unique features using MFCC and Hidden Markov Model (HMM)/Vector Quantisation (VQ).

For designing the above described product, we have studied a few programming languages like Java, MATLAB, C, C++, Python, Kotlin, etc. All of these languages could have been implemented but we settled down for Java because it is one of the most comprehensive and customisable languages and is robust and platform-independent.

3.4 WHY JAVA?

Java is a high-level programming language and computing platform developed by Sun Microsystems in 1995. Since then, the language has been regularly updated with Java SE 9.0.4 version being the latest version, released in January 2018.

Based on the advantages of Java, it gained wide popularity and multiple configurations have been built to suit various types of platforms including Java SE for Macintosh, Windows and UNIX, Java ME for Mobile Applications and Java EE for Enterprise Applications.

With the growing importance of web based and mobile based applications, Java today is the foundation for most networked applications and is considered to be useful for scripting, web-based content, enterprise software, games and mobile applications.

Java is a good choice for building big software systems. There are many reasons for that. It is fast, comparable in its efficiency to C++ and, at the same time, has a much lower cost of development. Java is tailored for designing various enterprise features, and Sun Microsystems, who developed Java, and later Oracle, counted on this, on server technologies and related things.

- **Performance:**
  Modern JVMs have excellent Just-In-Time (JIT) compilation which allows Java byte code compiled on any platform to be converted to native binaries for the machine running the code. This provides performance that is quite comparable to code directly compiled for a given native system (e.g., C/C++).

- **Flexibility:**
  Code written in Java may be implemented, compiled and deployed on nearly every modern platform. For example, code can be written by a mixed team on Max OS X and Windows, and then deployed to a cluster of servers running Linux. This is a huge advantage over platform-specific languages like C# or Objective-C.

- **Interoperability:**
  Java code can easily mingle at the byte-code level with many other languages that target the JVM such as Scala, Clojure, Groovy, Jython, JRuby, Kotlin etc. Certain components may lend themselves more towards one language or another.

- **Ecosystem:**
  Java's (relatively) long history and popularity has produced a rich ecosystem of people and libraries. As a result, a very good implementation exists for nearly every modern pattern or computing concept. The environment also tends to produce free and open source solutions for most common problems.

- **Refactoring:**
  Dynamic vs. static languages always have trade-offs, but the strictly static nature of Java means very powerful refactoring capabilities. While the code itself is more rigid, it can be restructured in very methodical ways.

3.5 CONCEPTUAL MODELLING

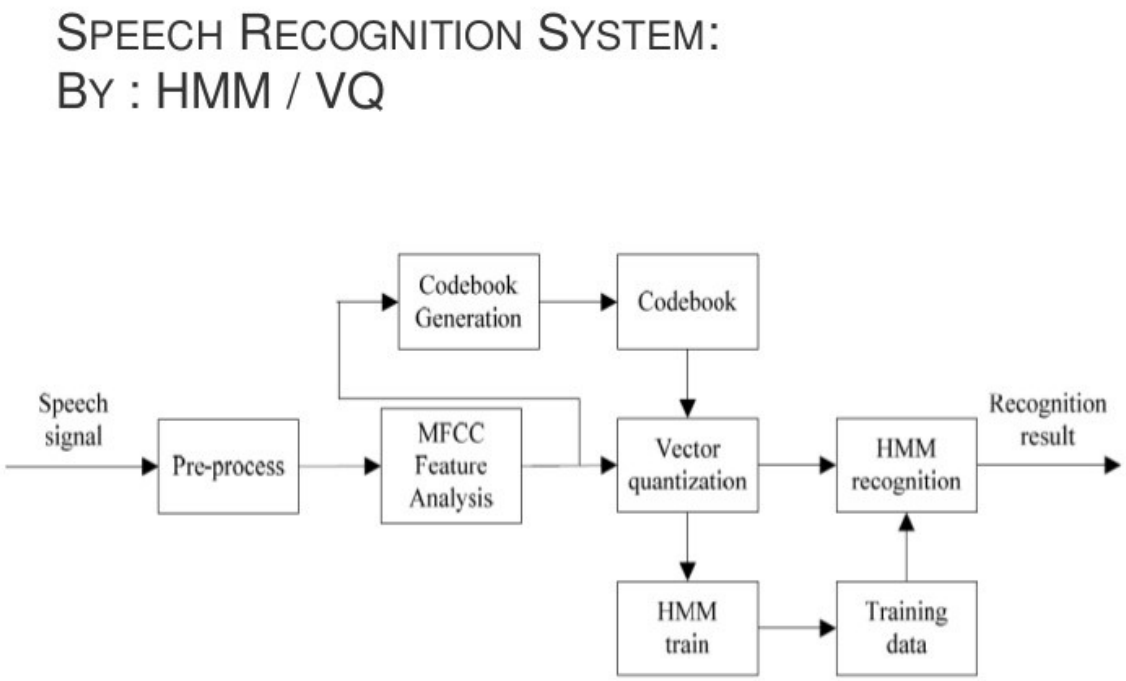

Figure 6: System Flow Chart

CHAPTER – 4

SYSTEM DESIGN

4.1 BASIC MODULES

The entire project has the following modules:

- Mel-Frequency Cepstral Coefficient (MFCC)
- Hidden Markov Model (HMM)
- Vector Quantisation (VQ)

4.1.1 MFCC

The most commonly used feature extraction method in automatic speech recognition (ASR) is Mel-Frequency Cepstral Coefficients (MFCC). This feature extraction method was first mentioned by Bridle and Brown in 1974 and further developed by Mermelstein in 1976 and is based on experiments of the human misconception of words.

To extract a feature vector containing all information about the linguistic message, MFCC mimics some parts of the human speech production and speech perception. MFCC mimics the logarithmic perception of loudness and pitch of human auditory system and tries to eliminate speaker dependent characteristics by excluding the fundamental frequency and their harmonics. To represent the dynamic nature of speech the MFCC also includes the change of the feature vector over time as part of the feature vector.

The standard implementation of computing the Mel-Frequency Cepstral Coefficients is shown in Figure 1.

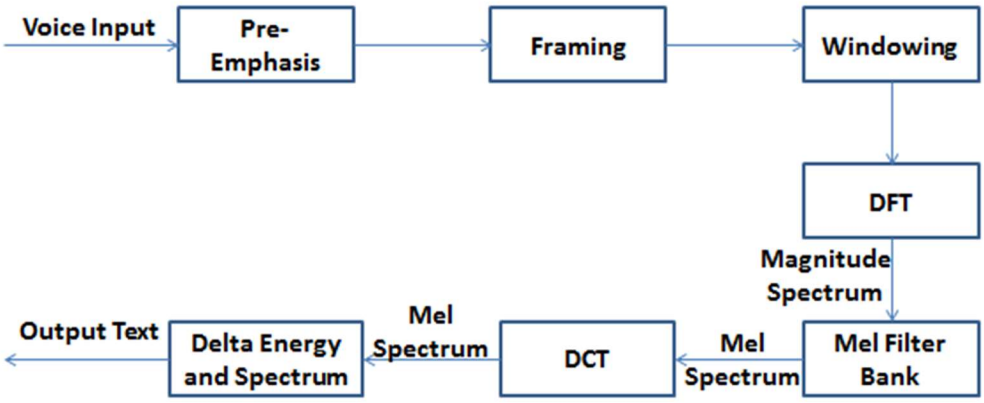

Figure 7: Block Diagram of MFCC

4.1.2 Block Framing

The most common approaches in speech signal processing are based on short-time analysis. The pre-emphasised signal is blocked into frames of N samples. Frame duration typically ranges between 10 - 30 msec. Values in this range represent a trade-off between the rate of change of spectrum and system complexity. The proper frame duration is ultimately dependent on the velocity of the articulators in the speech production system. Some sounds (e.g., stop consonants) exhibit sharp spectral transition which can result in spectral peaks shifting as much as 80 Hz/msec. The amount of overlap to some extent controls how quickly parameters can change from frame to frame.

In this part of the code, the continuous 1D signal are blocked into several frames of N samples, with the next frames separated by M samples where M<N. Therefore, the adjacent frames overlap by N-M frames.

The standard value N=256 and M=100 are taken with a purpose of dividing the 1D signal into small frames, thus, having sufficient samples to extract enough

information. However, if frames of smaller size are taken then the number of samples in the frames will not be enough to extract reliable information and with large size frames. With large size frames, frequent changes are bound to occur inside the frame. The process of breaking down the entire signal into small frames will continue until the whole 1D array is broken down into small frames.

4.1.3 Windowing

A signal observed for a finite interval of time may have distorted spectral information in the Fourier transform due to the ringing of the *sin(f)/f* spectral peaks of the rectangular window. To avoid or minimize this distortion, a signal is multiplied by a window-weighting function before parameter extraction is performed. Window choice is crucial for separation of spectral components which are near one another in frequency or where one component is much smaller than another. There are many types of windows including rectangular, Hamming, Hanning, Blackman, Bartlett and Kaiser. In speech recognition, the Hamming window is almost exclusively used. The Hamming window is a special case of the Hanning window. A generalised Hanning window is defined as

$$w(n) = \frac{\alpha - (1-\alpha)\cos(\frac{2\pi n}{N})}{\beta}$$

For n = 1, ……., N

And $w(n)=0$ elsewhere. α is defined as a window constant in the range [0,1] and N is the window duration in samples. To implement a Hamming window, the window constant is set to $\alpha = 0.54$. β is

defined as a normalisation constant so that the root mean square value of the window is unity.

$$\beta = \sqrt{\frac{1}{N}\sum_{n=1}^{N} w^2(n)}$$

4.1.4 Fast Fourier Transform (FFT)

This module is basically used for conversion from the spatial domain to the frequency domain. All the frames having N samples are converted into frequency domain. Fourier transformation is an algorithm to apply Discrete Fourier Transform (DFT).

DFT is basically as described below:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\pi kn}{N}} \qquad k = 0, \dots, N-1$$

Here N represents the total data points. Also, X_k represents the DFT coefficients in the original domain and x_n denotes the audio signal. The definition of both DFT and FFT are almost the same or the output for the transformation is the same. But they differ only in their computational complexity. In case of DFT, each of the frames which has N-M samples shall be directly used for Fourier transformation. On the other hand, the frame for FFT is divided into smaller DFT. Thus, computation is overall done on the small and divided DFT hence making the entire process faster and easier.

Thus, in recent digital processing instead of using DFT, FFT is implemented for applying DFT. After calculating DFT we obtain the magnitude spectrum.

4.1.5 Mel Scale

This phase is basically the continuation of the previous phase. Here the spectrum which was calculated above is just mapped on Mel scale. With this we can easily get to know about the existing energy at all spots. This can be found out with the assistance of Triangular overlapping window which is also known as the triangular filter bank. These filter banks are just a set of band pass filters having spacing along with bandwidth which is decided by the Mel frequency time. Mel scale gives us an idea regarding how to space the filter and how much wider it should be since, if the frequency gets higher, the spacing becomes much wider. In this case, the mapping needs to be done among the given real frequency scale which is in Hertz (Hz) and the perceived frequency scale or the Mel-Scale. When the mel scale is mapped, the scaling remains linear is the frequency value is till 100Hz, but when the frequency grows above 1000Hz, at that point the curve becomes logarithmic. The formula to convert normal frequency to Mel *mf* is given by the equation below:

$$m_f = 2595 \log_{10}\left(\frac{f}{100} + 1\right)$$

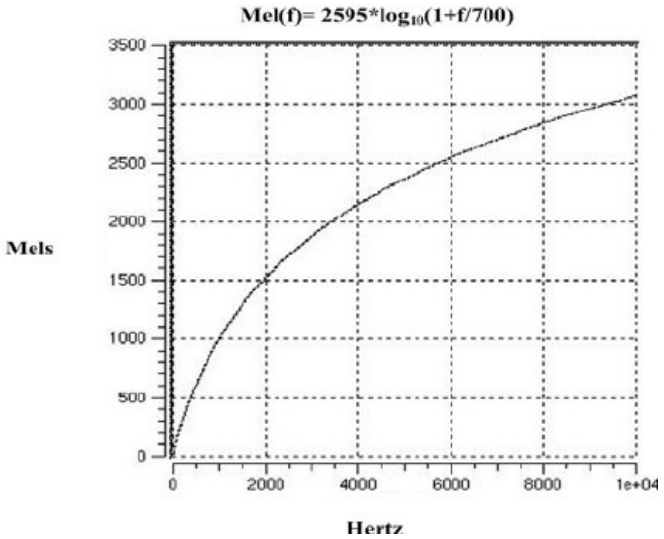

Figure 8: Mel Scale

Thus, with the help of mel filter bank proper spacing is done and it becomes a lot easier to estimate the energies present at each spot. The log of these energies is known as Mel spectrum which can further be used for calculating the coefficients using DCT. Increase in the number of coefficients in general means faster changes in the estimated energies and therefore, less information can be used for classifying the signals present.

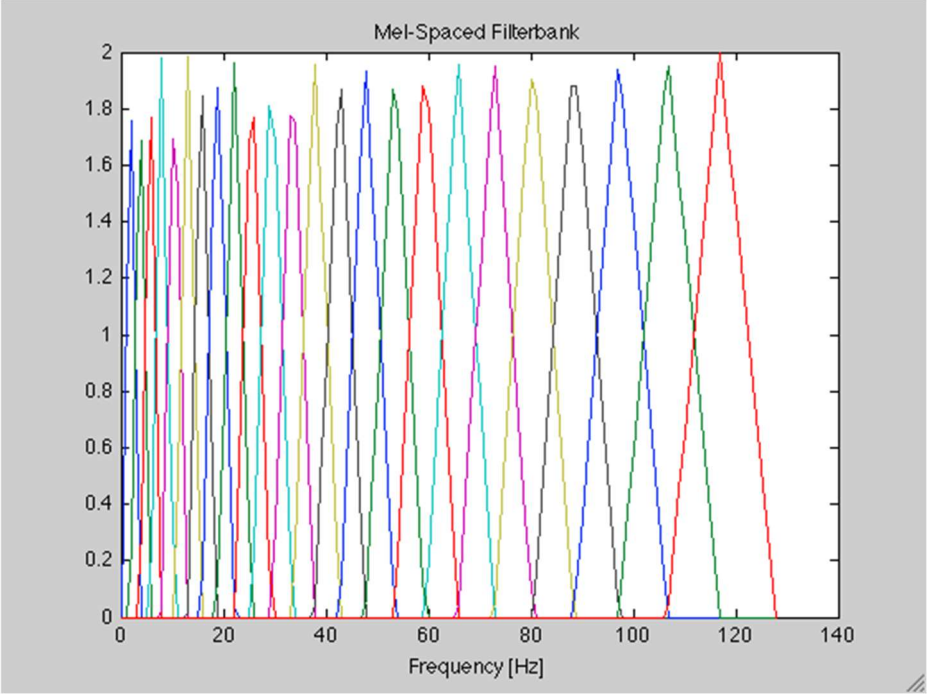

Figure 9: Mel Spaced Filter Bank

4.1.6 Inverse FFT and Discrete Cosine Transform (DCT)

During this phase, the previously calculated Mel spectrum is plotted back into the spatial domain. In this domain both the DCT as well as DFT can be used. However, DFT is primarily used to divide a sequence of finite data into discrete vector and is majorly used for spectral analysis. DCT is mainly used for data compression as the signals generated from DCT have more information which is being concentrated into small number of coefficients.

Hence, it is highly desirable to use DCT instead of DFT because the outputs contain important amounts of energy. The output generated after applying DCT is what is known as MFCC.

$$C_m = \sum_{k-1}^{N} \cos[m * (k - 0.5) * \pi/N] * E_k \qquad m = 1,2, \dots, L$$

C_m represents the MFCC coefficients and N is the number of triangular bandpass filters, L is the number of mel scale cepstral coefficients.

In a nutshell, all the steps which are carried out for signal processing using the MFCC algorithm can be viewed as follows:

- The input audio signal is at first converted from a 2-dimensional array to a 1D array.
- This array is broken into frames containing N samples each.
- In order to prevent any sort of loss of information, the frames are separated by M where M is strictly less than N.
- Thus, initially the first frame will be having Nm samples and then onwards all of them will begin from M samples after previous samples. Therefore, these frames overlap with N-M in order to prevent loss of transformation.
- All the value for N is taken as 256 and all the values of M is taken as 100.
- Each frame now consists of N samples which are now transformed into the frequency domain by using FFT. FFT is basically a high-speed algorithm that is used to implement the

conversion to frequency domain from time domain.

- After this, the Discrete Fourier Transform is applied to find out the magnitude spectrum and further along, it is again transformed into the Mel frequency.

- The last step is to take the logarithm of the spectrum in order to get the cepstral coefficients by performing DCT or Discrete Cosine transform.

4.1.7 Intermediate Steps

After all the features is extracted using MFCC algorithm, a single frame of the audio signal is chosen and considered to be the mel coefficients. This is placed in a newly created Feature Vector.

All available samples of each word in the vocabulary is grouped together using Hidden Markov Model and the MFCC vector of all these samples of a single word is kept in a 2D array.

The same procedure is continued for other words as well. Hence, there is just one single 2D array of each for each separate word. Now, an external sample is taken for matching. This unknown sample goes through same procedure of MFCC and the unique features are extracted and stored in a separate Feature Vector.

4.1.8 Static Removal

Here, the entire audio signal has been divided into several frames. In this case, the frame length is considered to be 0.05 times the sampling rate.

Accordingly, total number of frames is also calculated depending on the frame length. Successively, each frame was analysed in a loop and the maximum amplitude of that frame was determined. If the max amplitude of the frame was less than 0.03, then that particular frame is considered static and hence is discarded from the final audio signal on which further sound processing is done.

4.1.9 Feature Matching with Hidden Markov Model

The HMM is a sequence model. A sequence model or sequence classifier is a model whose job is to assign a label or class to each unit in a sequence, thus mapping a sequence of observations to a sequence of labels. An HMM is a probabilistic sequence model; given a sequence of units (words, letters, morphemes, sentences, whatever), they compute a probability distribution over possible sequences of labels and choose the best label sequence.

Hidden Markov Models (HMMs) provide a simple and effective framework for modelling time-varying spectral vector sequences. As a consequence, almost all present day large vocabulary continuous speech recognition (LVCSR) systems are based on HMMs.

The core of all speech recognition systems consists of a set of statistical models representing the various sounds of the language to be recognised. Since speech has temporal structure and can be encoded as a sequence of spectral vectors spanning the audio frequency range, the Hidden Markov Model (HMM) provides a natural framework for constructing such models.

The hidden state space is assumed to consist of one of N possible values, modelled as a categorical distribution. This means that for each of the N possible states that a hidden variable at time t can be in, there is a transition probability from this state to each of the N possible states of the hidden variable at time $t+1$, for a total of N^2 transition probabilities. Note that the set of transition probabilities for transitions from any given state must sum to 1. Thus, the $N \times N$ matrix of transition probabilities is a Markov matrix. Because any one transition probability can be determined once the others are known, there are a total of $N(N-1)$ transition parameters.

In addition, for each of the N possible states, there is a set of emission probabilities governing the distribution of the observed variable at a particular time given the state of the hidden variable at that time. The size of this set depends on the nature of the observed variable. For example, if the observed variable is discrete with M possible values, governed by a categorical distribution, there will be $M-1$ separate parameters for a total of $N(M-1)$ emission parameters over all hidden states. On the other hand, if the observed variable in an M-dimensional vector distributed according to an arbitrary multivariate Gaussian distribution, there will be M parameters controlling the means and $\frac{M(M+1)}{2}$ parameters controlling the covariance matrix for a total of

$$N\left(M + \frac{M(M+1)}{2}\right) = \frac{NM(M+3)}{2} = O(NM^2)$$

emission parameters. (In such a case, unless the value of M is small, it may be more practical to restrict the nature of the covariances between individual elements of the observation vector, e.g. by assuming that the elements are independent of each other, or less restrictively, are independent of all but a fixed number of adjacent elements.)

Probability of an observed sequence:

The task is to compute in a best way, given the parameters of the model, the probability of a particular output sequence. This requires summation over all possible state sequences:

The probability of observing a sequence

$$Y = y(0), y(1), \dots, y(L-1)$$

of length L is given by

$$P(Y) = \sum_{X} P(Y|X)P(X)$$

where the sum runs over all possible hidden-node sequences

$$X = x(0), x(1), \dots, x(L-1)$$

4.2 DATA DESIGN

For this project, Object IO capabilities of Java have been exploited to the most. The two major models used are:

i) Codebook, and
ii) Hidden Markov Model

The Codebook <*codebook.cbk*> is generated using the Linde-Buzo-Gray (LBG) algorithm. The LBG algorithm is a vector quantisation algorithm for deriving a good codebook. At each iteration, each vector is split into two new vectors.

- A; initial state: centroid of the training sequence;
- B; initial estimation #1: codebook of size 2;
- C; final estimation after LGA (Llyod's algorithm): Optimal codebook with 2 vectors;
- D; initial estimation #2: codebook of size 4;
- E; final estimation after LGA: Optimal codebook with 4 vectors;

The model reads all available words and generates their respective Feature Vectors using MFCC algorithm. Then,

the coordinates are stored in k-dimensional spaces. Finally, Centroids are generated up to length 1282. Then, these centroids are stored as Java Objects after vector quantisation.

Next, the HMM is trained using this generated Codebook. The HMM loads back all the Feature Vectors and uses the quantisation to find the closest Centroid to a Point. Then finally the model is trained using the Forward-Backward algorithm, Baum-Welch algorithm, Scaling, Viterbi, etc. The generated training set is finally stored as the <word name>.hmm and is the trained model for the word.

This model and Codebook are referenced at the point of feature matching. The values of the given sample/recording/recorded voice are matched with those already stored in the model and then HMM tries to predict the word closest to the spoken word/recording statistically.

CLASSES AND FUNCTIONS TO STORE THE DATA

The Class `Operations.java` in the `edu.sxccal.cmsa.mediator` package is the main class responsible for handling the firing of different events and hence generation of Codebook and training HMMs.

The entire `edu.sxccal.cmsa.db package` is responsible for the different Object IO handling. The `ObjectIO.java` class is responsible for the actual reading and writing of models. The `ObjectIODataBase.java` class acts as a link between the software and the models on secondary memory.

The package `edu.sxccal.cmsa.classify.speech` and its sub-package `edu.sxccal.cmsa.classify.speech.vq` contains the required classes for Codebook generation and HMM and Vector Quantisation.

The Screenshot below is the file codebook.cbk (an example):

Figure 10: Example of codebook.cbk

Figure 11: Example of HMM for word “Hello” as hello.hmm

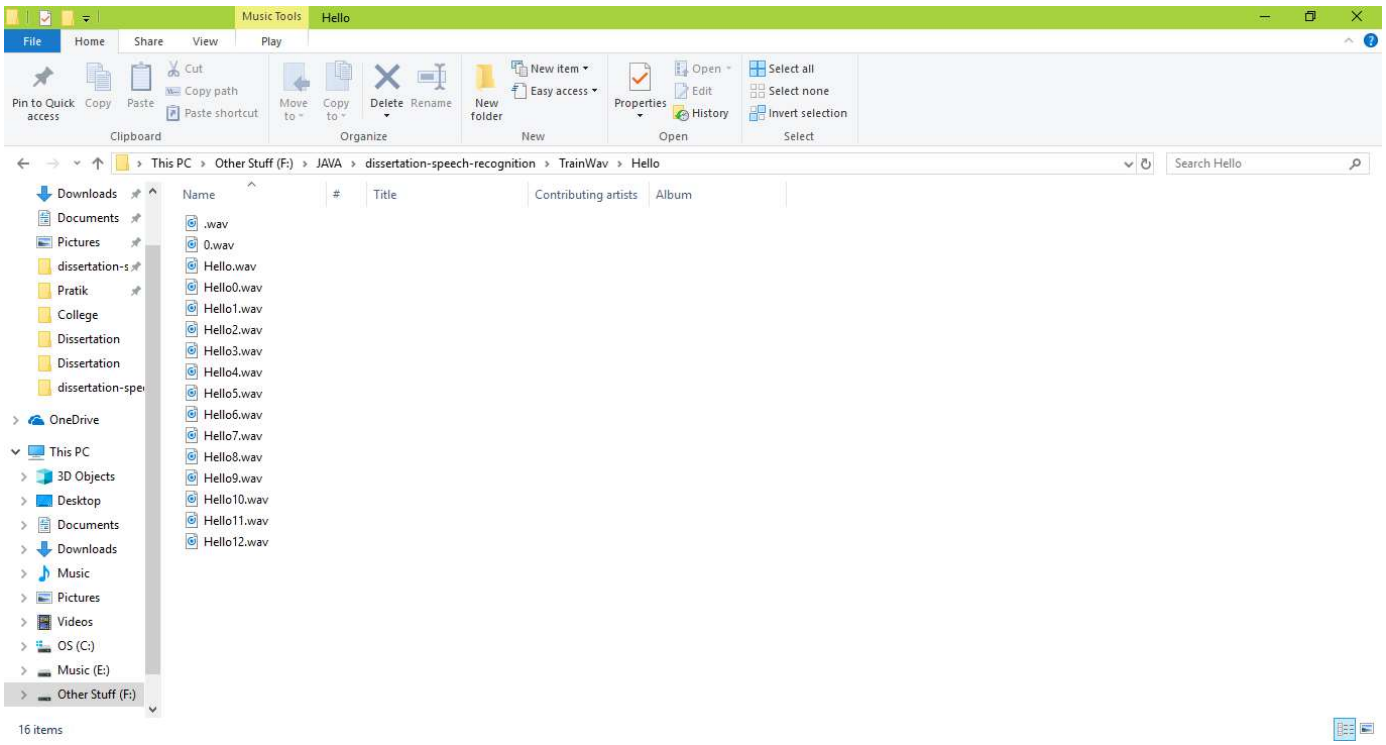

Figure 12: Directory listing for word "Hello" - All Training Files.

4.3 ALGORITHM DESIGN

The algorithm of the main modules are discussed as follows:

A) Windowing and Block Framing

The primary purpose of this part of code is to partition or cut the signal into frames. In this function we are just taking s, fs, m and n as the input parameters to this function. Here, s contains the signal which is to be analysed, fs is the sampling rate of frequency and m is the distance between the neighbouring frames. The final output is stored in matrix M3 which contains all the frames.

Step 1: Take input s, fs, m, n

Step 2: l <- length(s). Calculating the length of the audio signal

Step 3: Calculating the number of frames or nbFrame as:
nbFrame <- floor((1-n)/m)+1

Step 4: Loop i=1 to n

Step 5: Loop j=1 to nbFrame

Step 6: M(i,j) <- s(((j-1)*m)+i)

Step 7: Go to Step 4

Step 8: Go to Step 3

Step 9: Calculate the hamming window h

Step 10: Calculate M2=diag(h)*M

Step 11: Loop i=1 to nbFrame

Step 12: M3(i^{th} column) = fft(M2(i^{th} column))

Step 13: Go to Step 11

Step 14: End

B) MFCC

This is the primary function in this project. The output of this function is the MFCC vector of the audio signal or the transformed signal. This function takes the framed audio signal as input.

Step 1: Find the magnitude spectrum using FFT

Step 2: Pre-emphasise on the signal. Emphasise high frequency signals.

Step 3: Prepare filter for melFilter

Step 4: Process Mel Filter bank using magnitude spectrum.

Step 5: Non-linear transformation

Step 6: Find Cepstral Coefficients by DCT

Step 7: End

C) Make MFCC Vector

This function is used to create a feature vector of mfcc coefficients.

Step 1: Calculate MFCC for all framed Signals

Step 2: Perform Cepstral Mean Normalisation

Step 3: Find delta for MFCC

Step 4: Find delta delta for MFCC

Step 5: Calculate energy

Step 6: Find Energy delta

Step 7: Find Energy delta delta

Step 8: Generate Feature Vector

Step 9: End

D) Generate Codebook

Step 1: Get list of all files present

Step 2: Create Feature Vector for each word and each file

Step 3: Add Feature Vector to All Features List

Step 4: Generate Points to store coordinates in k-dimensional space

Step 5: Generate Codebook using LBG algorithm

- Step 1: Add all training points to single cell
- Step 2: Set initial code vector to average of cell points
- Step 3: Repeat splitting step and K-means until required number of codewords is reached

Step 6: Save Model to file

Step 7: End

E) HMM Train

Step 1: Read available wave files

Step 2: Extract feature from files

Step 3: Create Feature Vector

Step 4: Get Points from Feature Vector

Step 5: Quantize Vectors

Step 6: Build HMM

- Step 1: Build Database
- Step 2: Generate Random probabilities for transition, output
- Step 3: Set Training Sequence for Re-estimation
- Step 4: Re-estimate 20 times

Step 7: Save HMM to file

Step 8: End

4.4 GRAPHICAL USER INTERFACE (GUI)

The graphical user interface makes it easier for the user to get hold of the end product. This aims to make the software more user friendly for the one who is using it. Through the use of push buttons, labels, text box, combo boxes, etc. we

aim to make the Speech Recognition software more flexible and understandable for the user.

Here are a few screenshots of what the software looks like:

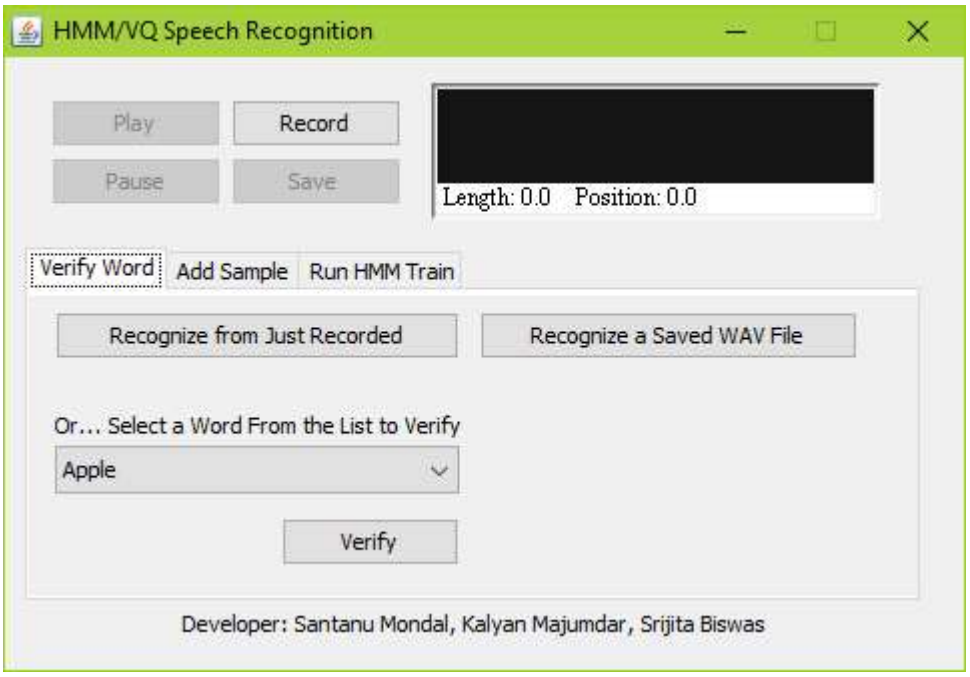

Figure 13: Verify Tab in GUI

Play – To play recorded audio

Record – To Record audio from microphone

Pause – To Pause playback of recorded audio

Save – To Save recorded audio to a particular path

Recognize from Just Recorded – HMM Recognise using recorded audio

Recognize a Saved WAV File – HMM Recognise using loaded WAV file

Verify – Verify if the spoken word is the word selected from the Combo Box

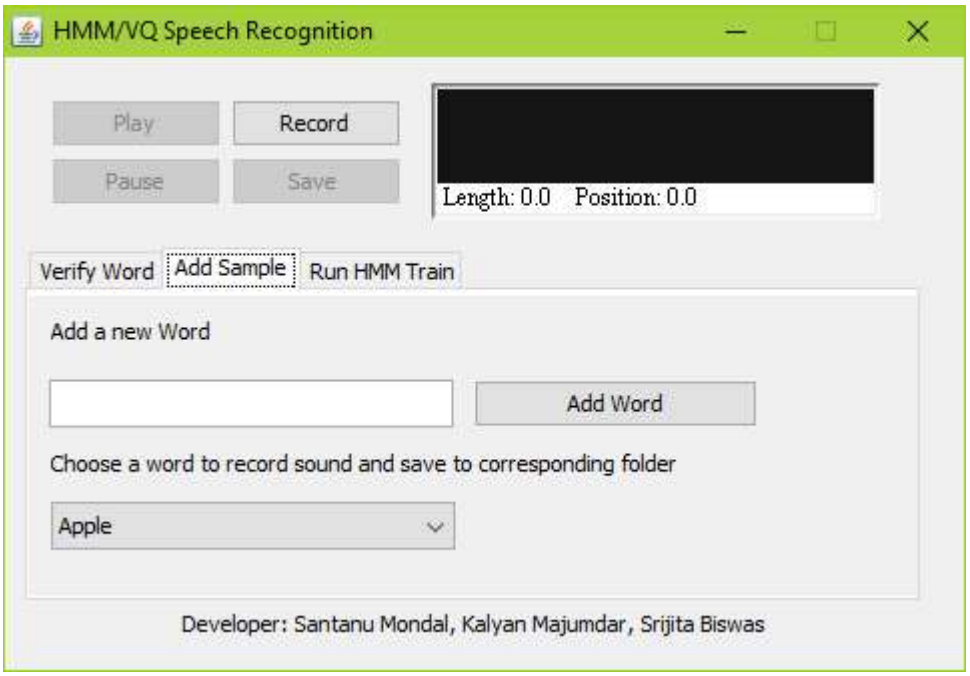

Figure 14: Add Sample tab in GUI

Add Word – Add a new word to the vocabulary

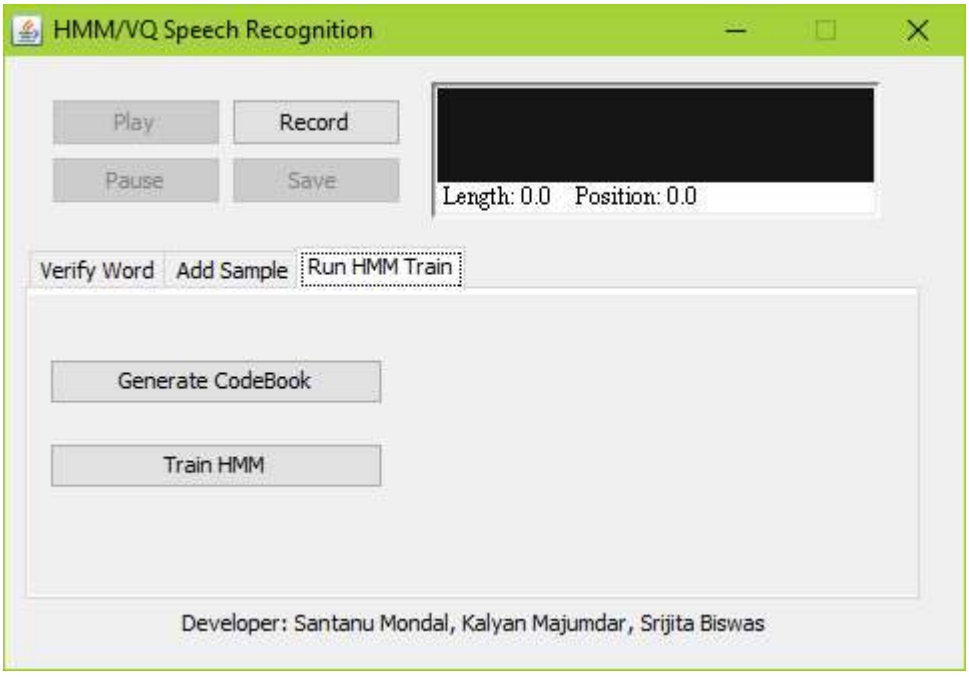

Figure 15: Run HMM Train tab in GUI

Generate CodeBook – Generate Vector Quantised CodeBook

Train HMM – Train HMM for prediction

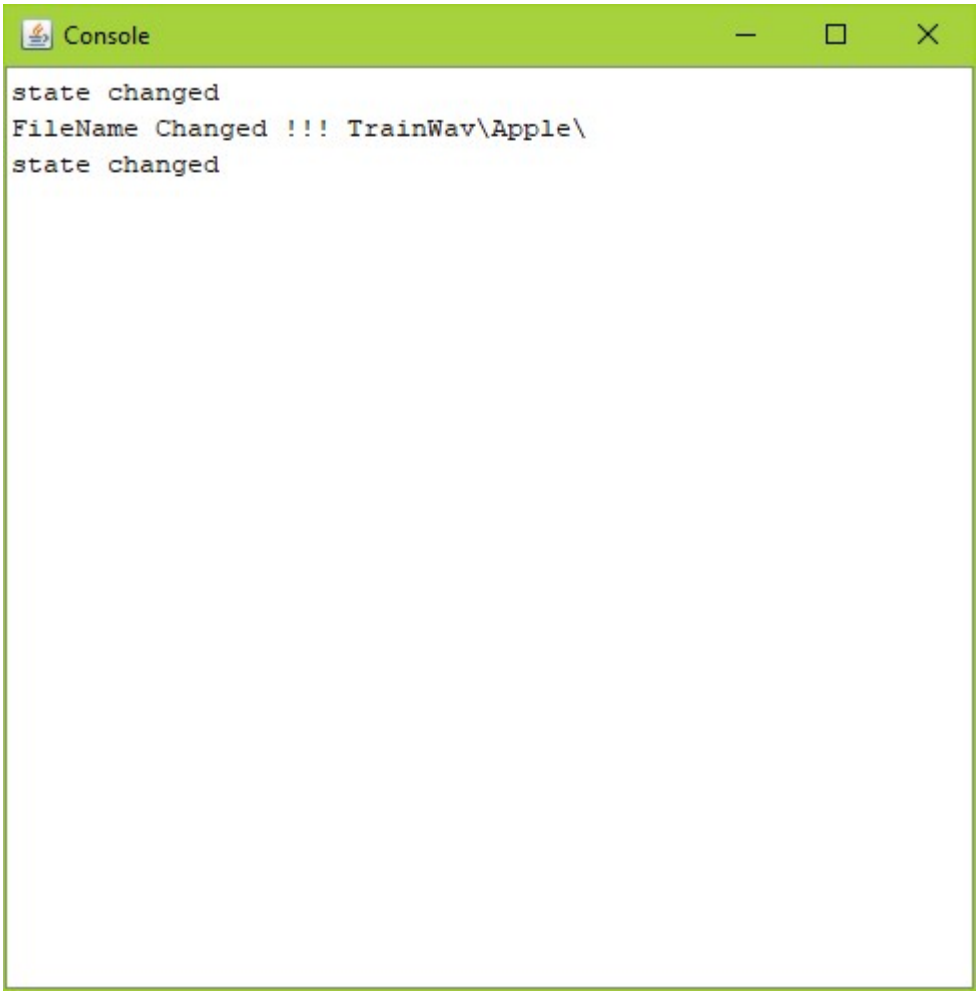

Figure 16: Modified Console Window

Modified Console window for logging the algorithms.

CHAPTER – 5

IMPLEMENTATION AND TESTING

5.1 CODE EFFICIENCY

After finishing up with the back-end coding, several English words were added during unit testing. In order to do so, the FRR or the False Reject Ratio and FAR or False Accept Ratio was calculated. This ratio gives us an idea about the efficiency of the system.

FRR takes into account all those words which are falsely rejected versus total number of samples tested.

Therefore, the formula is:
FRR = (Words which are falsely rejected)/ (Total number of samples tested)

Testing of various audio samples was done for certain word and the following results were found.

Word	Number of Samples Rejected	Total Number of Samples Tested	FRR of each Word
Apple	4	20	0.2
Developer	5	20	0.25
Hello	2	20	0.1
Ship	6	20	0.3
Test	5	20	0.25
One	5	20	0.25
Zebra	8	20	0.4

Total Number of Samples Tested = 140
Total Number of Genuine Samples Rejected = 35
FRR = 35/140 = 0.25

FAR takes into account all those words which are falsely accepted versus total number of samples tested.

Therefore, the formula is:
FAR = (Words which are falsely accepted)/ (Total number of samples tested)

Testing of various audio samples was done for certain word and the following results were found.

Word	Number of False Samples Accepted
Apple	4
Developer	3
Hello	6
Ship	4
Test	3
One	5
Two	6

Total Number of Samples Tested = 140
Total Number of False Samples Accepted = 31
FAR = 31/140 = 0.2214285714285714 ~ 0.2214

5.2 TESTING APPROACH

The testing was done in two separate parts. First is the unit testing which was done before the GUI was written. Second was the integrated testing which was done after integrating the back-end coding part with the front-end GUI part. Screenshots of each of these testing phases are given in the next section.

5.3 UNIT TESTING

Here, the various audio samples were tested separately before the final GUI was made for the user. The entire software is written and just the GUI are not implemented during this phase. The MFCC vectors matrix is generated for each sample separately.

After MFCC Calculation, Codebook generation and HMM Training is also done.

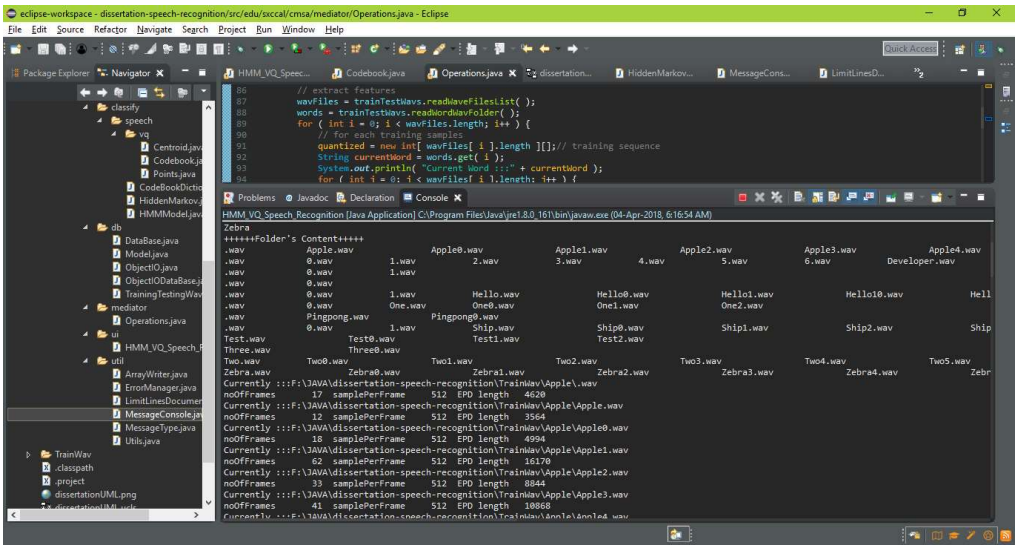

Figure 17: Codebook Generation - 1

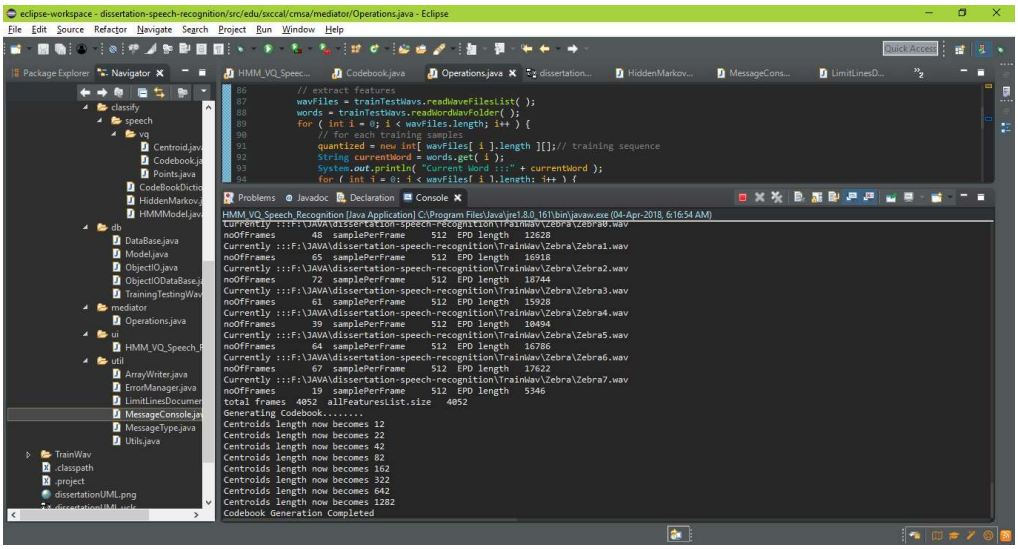

Figure 18: Codebook Generation – 2

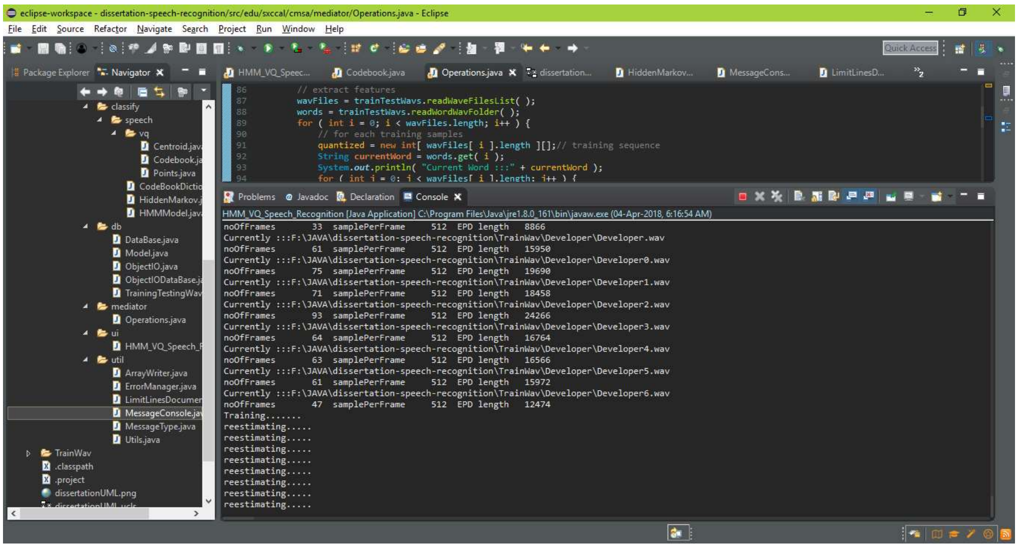

Figure 19: HMM Training – 1

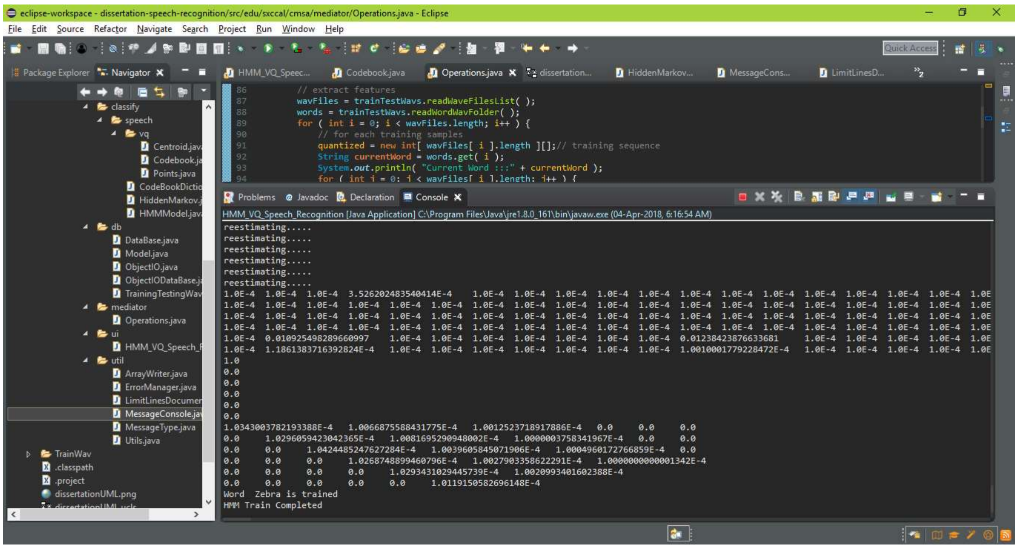

Figure 20: HMM Training – 2

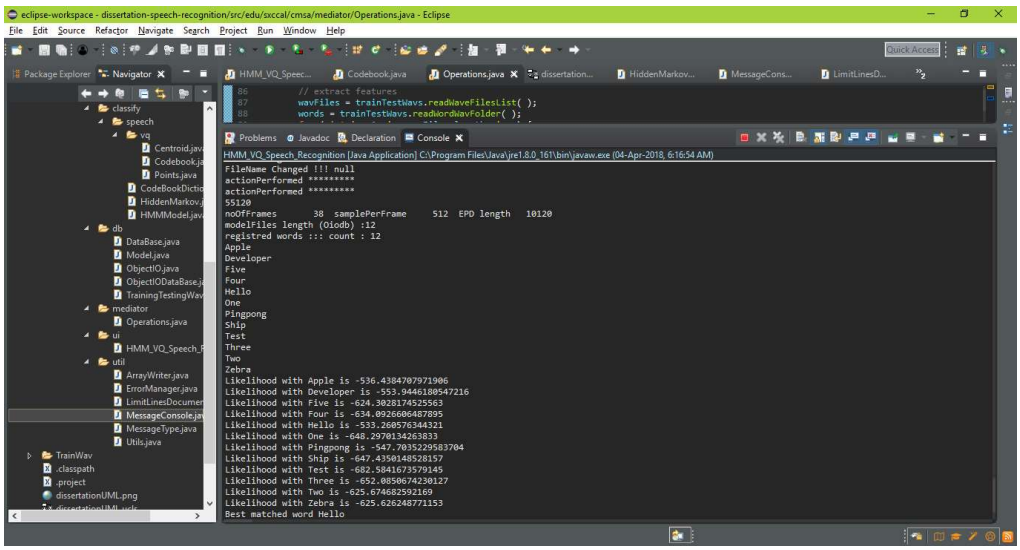

Figure 21: Recognising Word "Hello" using HMM

5.4 INTEGRATED TESTING

Separate GUI files were created, all the modules were integrated. Thus, the separate functions which include the MFCC vector calculations are all integrated together and the final product was made. Here the test cases are not the ones used to make the initial standard database.

Screenshots of the final software along with test cases input and output are given as follows:

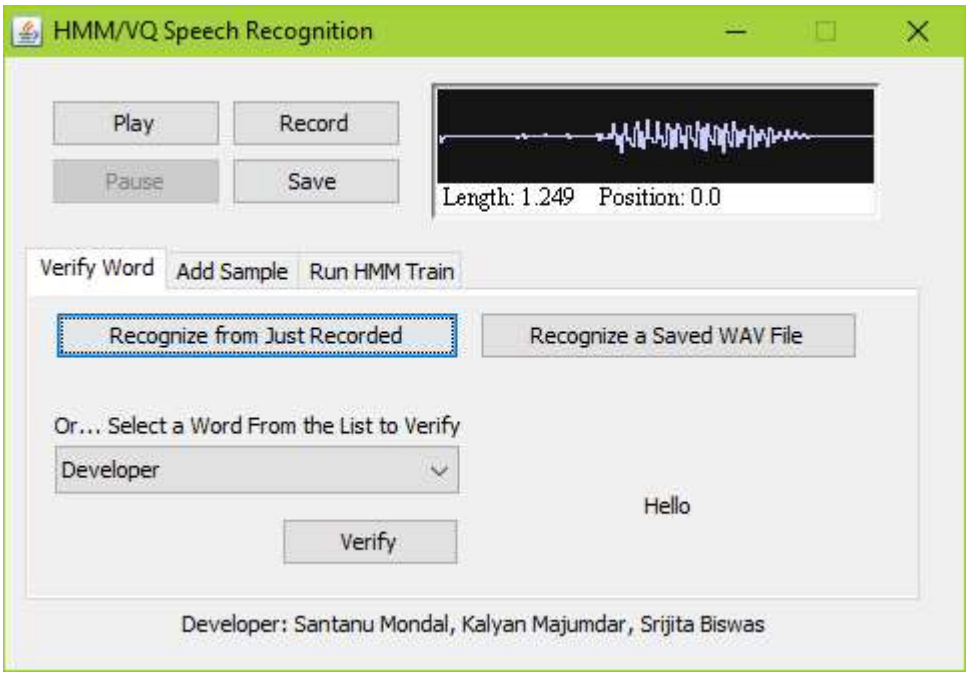

Figure 22: Correct Output as "Hello"

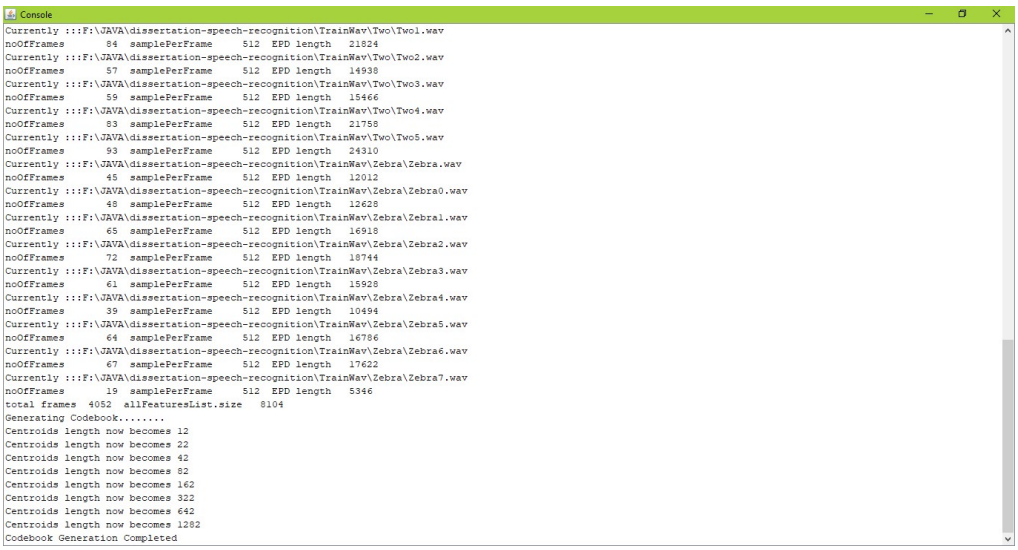

Figure 23: Codebook Generation complete

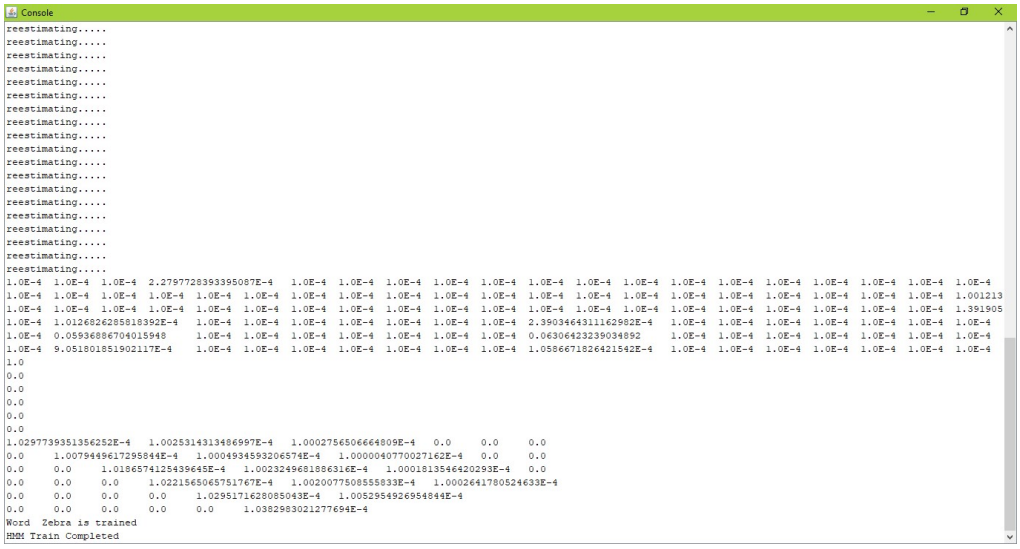

Figure 24: HMM Training Complete

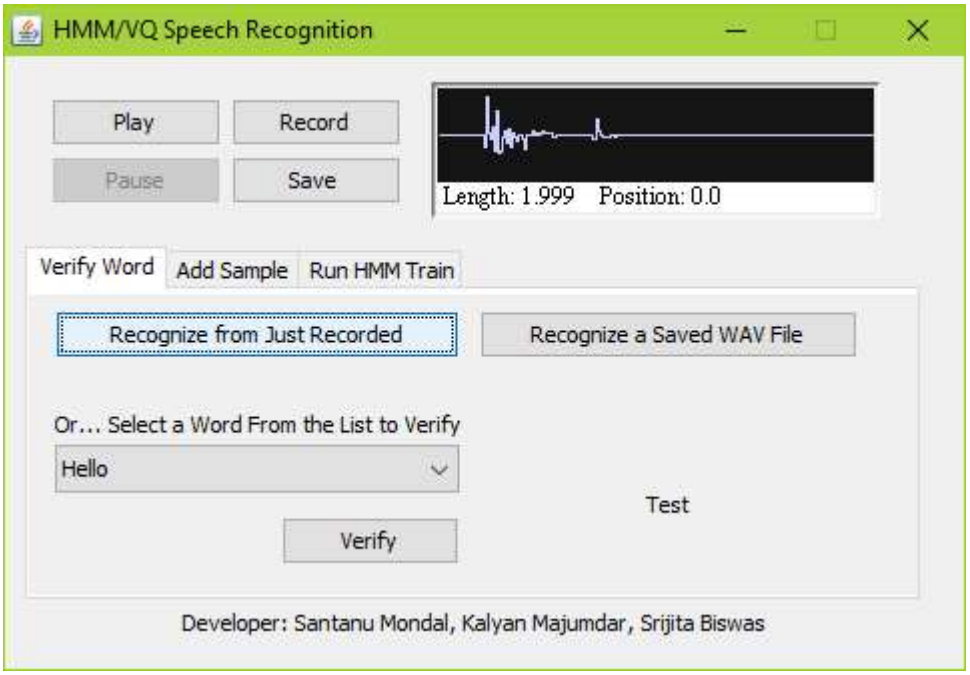

Figure 25: Recognising "Test" Correctly

5.5 IMPLEMENTATION IN JAVA

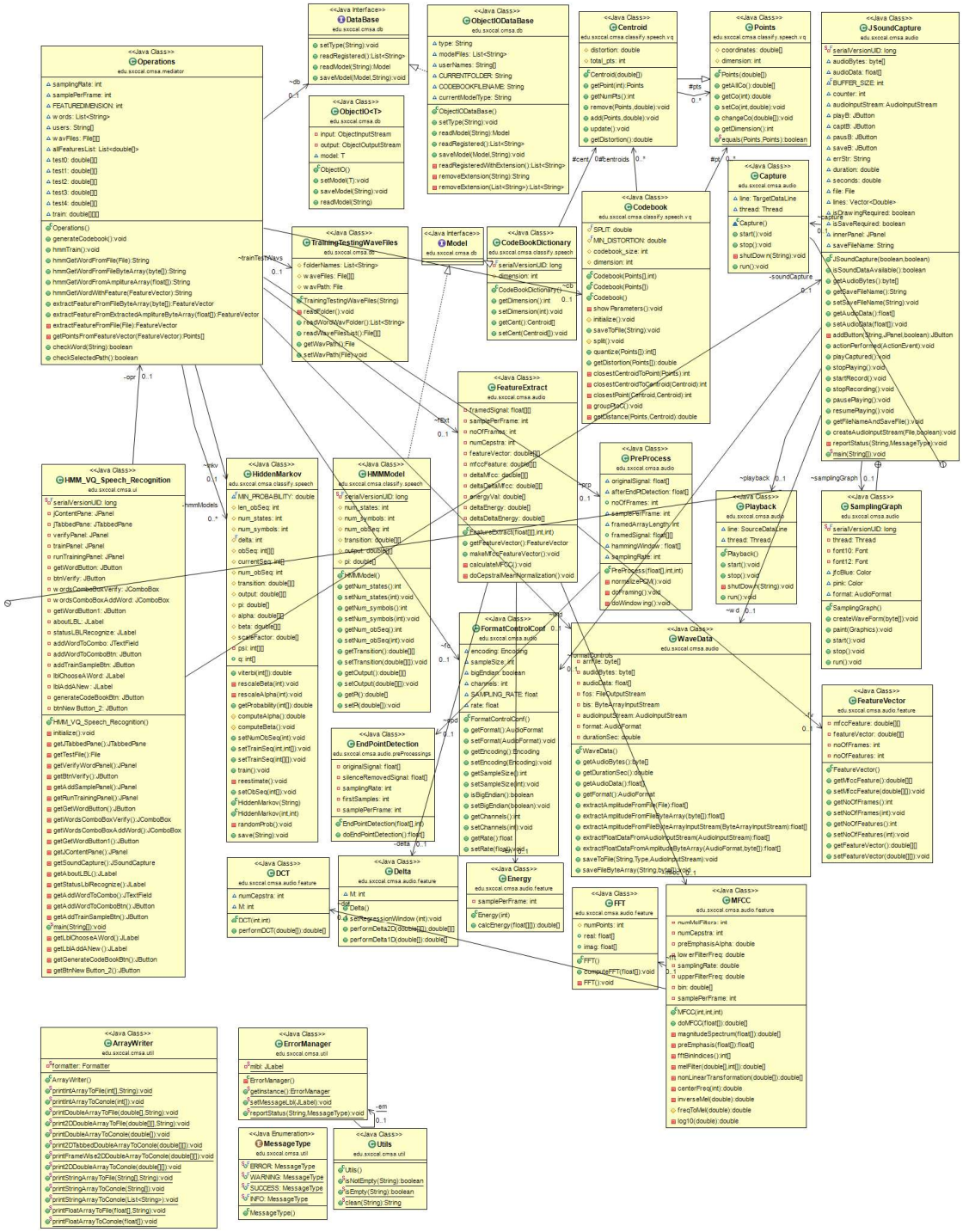

Figure 26: UML Diagram of Java Implementation

CODE ATTACHED AS APPENDIX 'A' TO END OF PROJECT

CHAPTER – 6

6.1 CONCLUSION

The field of voice recognition has crossed various milestones in the field of technology. It has evolved into a technology that can be expanded and implemented as various applications which we can use in our daily lives. Finer and more specialized principles of voice recognition have come into use, the biggest example being Google's well-known Assistant Google Assistant and Samsung's Bixby.

Voice recognition has, in the recent past, been modified and applied to various technologies exploring the principles of speech recognition, the recognition of specific words, and the recognition of regional and foreign languages beyond English. We have used the principles in our project to classify audio inputs and recognise isolated English words.

From a collection of words (vocabulary) added to a codebook, we implemented our code on audio inputs to find out which English word among those in the database the input sample is similar to.

Our algorithm can be expanded to include more words. It can be improved so as to be applicable to multi-track audio samples. This project proves to be relevant for future use and for future modification and improvement. It shall find its use in the technology-for-differently-abled industry, in research and in machine learning.

6.2 APPLICATION

After multiple decades of research, the process of audio recognition has witnessed major evolution. It has found its usage in various fields and is now a part of our daily lives.

With the growth of voice recognition applications, requirements for finer implementation of the software have increased. Hence, applications for recognizing words in specific languages like Hindi, French, Urdu are being developed.

Our project on sound classification and matching with isolated words has various fine uses.

The various fields in which our project on sound classification and the matching with isolated words can find wide usage are:

i. People-with-disability/Accessibility
ii. Education
iii. Security
iv. Customer Support
v. Machine Learning

6.3 LIMITATION

i. This algorithm has a very low efficiency for multi-track audio samples.

ii. It is preferred that all audio samples have a sampling frequency high enough.

iii. The database is limited. The efficiency of the algorithm may reduce if the database is expanded.

iv. Since no threshold value of features has been considered, an input audio sample which may not belong to any of the musical instruments defined in the database may be accepted into the database of one of the instruments.

6.4 SCOPE FOR FUTURE ASPECT

Our algorithm for sound classification can be improved as follows:

1. Our algorithm can be further improved to include more words. This would lead to a bigger database containing more audio samples, and our application would hence be wider. (Though the option is available to dynamically add words to the vocabulary)

2. Our algorithm can be further applied on multi-track audio samples. Suppose an audio sample contains the voices of multiple people. They can be separated and classified into their respective words.

3. Our algorithm can be further applied to recognise continuous speech. The words can be separated and classified.

6.5 REFERENCES

[1] Vibha Tiwari, *"MFCC and its applications in speaker recognition"*, Dept. of Electronics Engg., Gyan Ganga Institute of Technology and Management, Bhopal (MP), India, 10th February, 2010

[2] Beth Logan, *"Mel Frequency Cepstral Coefficients for Music Modelling"*, Cambridge Research Laboratory, Compaq Computer Corporation, One Cambridge Center, Cambridge MA 02142

[3] Aldebaro Klautau, *"The MFCC"*, 22nd November, 2005

[4] Milan Sigmund, *"Voice Recognition by Computer"*, Tactum Verlag, 2003

[5] Mark Gales, Steve Young, *"The Application of Hidden Markov Models in Speech Recognition"*, Cambridge University Engineering Department, 2008

[6] Beth Logan, *"Mel Frequency Cepstral Coefficients for Music Modeling"*, Cambridge Research Laboratory, Compaq Computer Corporation