KONKANI SWAR (VOWELS) ASR SYSTEM

Submitted by

1614	Jolwina Fernandes
1615	Joshua Fernandes
1629	Alisha Lotlekar

Guide : Prof. Baskar Sundarrajan

in partial fulfillment for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

in

COMPUTER SCIENCE

Department of Computer Science and Technology

GOA UNIVERSITY

JUNE 2018

Declaration

We hereby declare that this project is done by us to the best of our knowledge that we acquired during this course.

Jolwina Fernandes	**Joshua Fernandes**	**Alisha Lotlekar**

Acknowledgement

First of all, we would like to take this opportunity to thank our teacher and guide, Prof. Baskar sundarrajan for his valuable time and guidance, constant encouragement and helpful feedbacks at every step.

We would also like to thank our phd student, Swapnil Phadte for imparting us with all the knowledge which was of immense help in our project.

A big thanks to the university for providing us with the facilities to finish this project on time.

We would also like to thank all my friends and family members for their support throughout this project.

Last but not the least, we would like to thank all our classmates for their help and support.

Table of Contents

Introduction

Background Research

Speech recognition is the inter-disciplinary sub-field of computational linguistics that develops methodologies and technologies that enables the recognition and translation of spoken language into text by computers. It is also known as automatic speech recognition (ASR).

What is ASR?

Automatic Speech Recognition System(ASR), accurately translates spoken utterances into its text equivalent. Examples will be, youtube's closed captioning, Dictation Systems, Siri, Cortana, Google Assistant, etc.

Figure 1: Examples of Speech Recognizers.

Cortana **Siri** **Alexa** **Google Assistant**

Speech Recognition is one of the fastest developing field in Machine Learning. Speech Recognition is a field of Computer Science and Electronics wherein we deal with Signals and Systems, Signal Processing, Signal Enhancement etc. In Speech Recognition, user speaks using microphone, which is taken as an input for the machine (Speech Waveform). Processing of this speech waveform is then done by framing speech signal, converting signal from time domain to frequency domain, speech enhancement etc. Resulting output is then given in the text format to the machine which has to identify the word which the user has spoken.

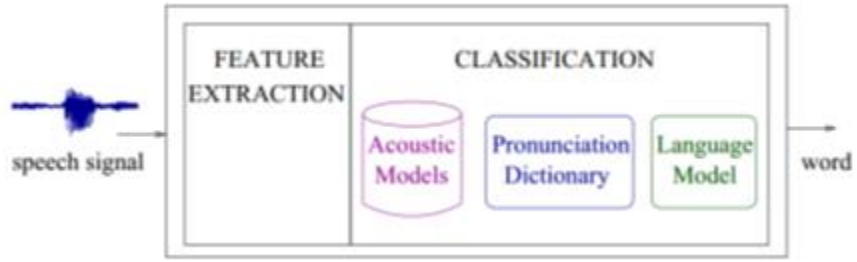


Figure 2: Basic ASR Model

Speech Technology is a developing technology with variety of tools and methods for better and fast implementation.

To discuss speech recognition technology, we use following terminology in this report.

• **Utterance: -** Utterance is the vocalization of a word. (In our case a single Vowel).

• **Vowels: -** Vowels in our report that are considered according to Dr. Madhavi Sardessai's interpretation. (A Table depicting the same is shown below).

• **Training: -** Training is the process of learning the characteristics of sound unit. Developing the model using the training data to learns the parameters of the models.

• **Hidden Markov Model: -** Hidden Markov Models (HMM) is used to produces the sequence of observable output in order to model unknown processes at discrete time interval. Hidden Markov Model is a finite set of states, each of which is associated with probability distribution. Transition probability gives transition among the states. For a particular state, outcome generated depend on the probability distribution but these outcomes are not visible for external observer therefore states are hidden.

Brief History of Konkani Language

Konkani is among one of the most under resourced Language, so we decided to put our machine learning efforts to develop a small ASR that could identify vowels as a contribution towards our mother tongue.

The tenacity and resilience of the Konkani language us proven by its ability to survive and transcend various obstacles like the horrors of the inquisition, inhuman and cruel killings, torture in the time of the tyrannical Portuguese rule in Goa. As a consequence, people from Goa migrated towards various parts of India, this severed the link to the original dwellers and Konkani started taking more forms.

The Birth of Dialects: Goa is a small place in terms of population and geographical spread, but have a rich heritage besides food and music but also oral literature. Dr. S.B. Kulkarni has stated in his book that there are over 25 different dialects of Konkani, if you consider the variations in nasal sounding words to distinctly identify Konkani Language.

In Goa the way konkani is spoken differs from region to region and from person to person. Within the state of Goa, the language has various dialects such as Antruz, Malvani, Pednekari, Bardezkari, Kukna, Chitpavani, Mangalorean, Daldi, saxtti etc

In 1961 after Goa was Liberated. After that in the opinion poll of 1967 Goa was established as a union territory. On 26th February 1975 the Central Sahitya Academy recognised Konkani as a separate and independent language. On 4th February 1987 Konkani was declared as the official Language of Goa. And finally after Goa getting its statehood on 30th May 1987, Konkani was included in the 8th Schedule of the Indian Constitution on 20th August 1992.

Objective:

To Build a Simple ASR system that can recognize isolated Konkani Vowels(Swar). In our study, we propose to use HMM model which is trained using audio data. Our main focus is on minimizing the information loss while extracting features from audio signal (using the MFCC feature extraction technique as will be explained later in the report). Appropriate techniques for efficient feature extraction will be investigated. However, our study is restriction only towards Konkani Vowels.

Domain

The purpose of this project is to recognize isolated Konkani vowels (Swar). Hidden Markovian Models have been successful when it comes to NLP. So supervised Machine learning HMM will be one of the models that will be considered to train our Data. Along with HMM we will also try and test our data with Multilayer Perceptron and cross check the error rate/ accuracy.

Data will be the recorded voice samples and this would be the input for our HMM and also MFCC features will be extracted to be fed into the MLP.

Domain of this project can be explained in two major steps ie. Konkani Language and Speech Recognition.

1a. Konkani Language.

Konkani is an Indo-Aryan language from the family of Indo-European languages and is spoken on the western coast of India. It is one among the twenty-two scheduled languages mentioned within the eighth schedule of the Indian Constitution and the official language of the Indian state of Goa.
It is a minority language in Karnataka (Karwar, Mangalore), Maharashtra, Kerala and Daman and Diu.

1b. Vowels and Consonants.

Literature Survey on Vowels sounds in Konkani: Vowels and consonant are fundamental part of any language, we can have a vowel without consonant, but cannot have consonant without a vowel. Some languages like Japanese, duration of vowel sounds matters and changes in duration leads to a different meaning of the word. Compared to consonant, duration of vowel sounds in speech signal is longer and it also has higher energy. This property of vowel makes it interesting to attempt them first if any language is studied for the first time.
According to Matthew Almeida, konkani has 11 vowels (4 front vowel, 4 back vowel and 3 middle vowels). Konkani is a highly nasalized language and each oral vowel sound can also be nasalized.
Dr. Madhavi Sardesai another renowned linguist has a different view about vowels she classified 9 vowels in 3 group (3 front, 3 back, and 3 middle) and unlike Almeida she has not classified short and long vowels as 2 different vowels. She has also included one extra vowel sound which is not present in the Almeida's classification.

For Our Research we will consider Dr. Madhavi Sardessai's take on konkani vowels. We choose Ma'am's take on because Matthew Almeida als considered similar sounding vowels as a separate vowel, but it won't help us distinctly identify them, plus Dr. Madhavi Sardessai is closer to major konkani speaking locality.

Below is the Table with the vowels and its corresponding(assumed) IPA notations According to Dr. Madhavi Sardessai.

Nagari Grapheme- Related with little variance.

Sr No	Assumed IPA Notation	Vowel Symbol	Nagari Grapheme	Examples
1	i	इ	इ and ई	वीस(Twenty), कवी(Poet)
2	e	ए	ए	पेर(Tree), खेळ(Game), मेज(Count)
3	ɛ	अॅ / एॅ		पॅर(Guava), खॅळ(Play), मॅज(Table)
4	a	आ	आ	आडसर(Tender Coconut), राजा(King)
5	u	उ	उ and ऊ	उस(Sugarcane), पूल(Bridge)
6	o	ओ	ओ	चोर(Thief), दोन(Two)
7	ɔ	आॅ		चॉर(Thieves), बॉल(Ball)
8	ɨ	अं	अ	कंर(To do), बंस(sit), पंड(basket)
9	ə	अ		कर(Tax) बस(bus),पड(fall)

Konkani Speech Sounds According to Matthew Almeida.

Modified Nagari	**Phonetic Symbol**	**Nagari Graph**	**Roman Graph**	**Example**
अ	**ɵ**	अ	**o**	अट
आ	**a**	आ	**a**	आज
इ	**i**	इ	**i**	इनाम

(ई)	**i:**	ई	**i**	रीत
उ	**u**	उ	**u**	उजवो
(ऊ)	**u:**	ऊ	**u**	सूत
ए	**e**	ए	**e**	पेड
ऍ	**ɛ**	ए	**e**	पेड
ओ	**o**	ओ	**o**	बोर

Methodology

Since our focus is limited to Speech to text of Konkani vowels using HMM. We will build a speaker independent speech recognizer. For this we will need to extract features from the speech signal, known
as acoustic Information of the Speech. Further, this information is given to the HMM model and processing is done, the output given by the model is the hypothesis transcription of the isolated word.

To develop speech-to-text recognition system has 4 main stages.

1. Data Preparation
2. Training
3. Recognition on testing data
4. Analysis

To develop the system, the following are sub-processes :

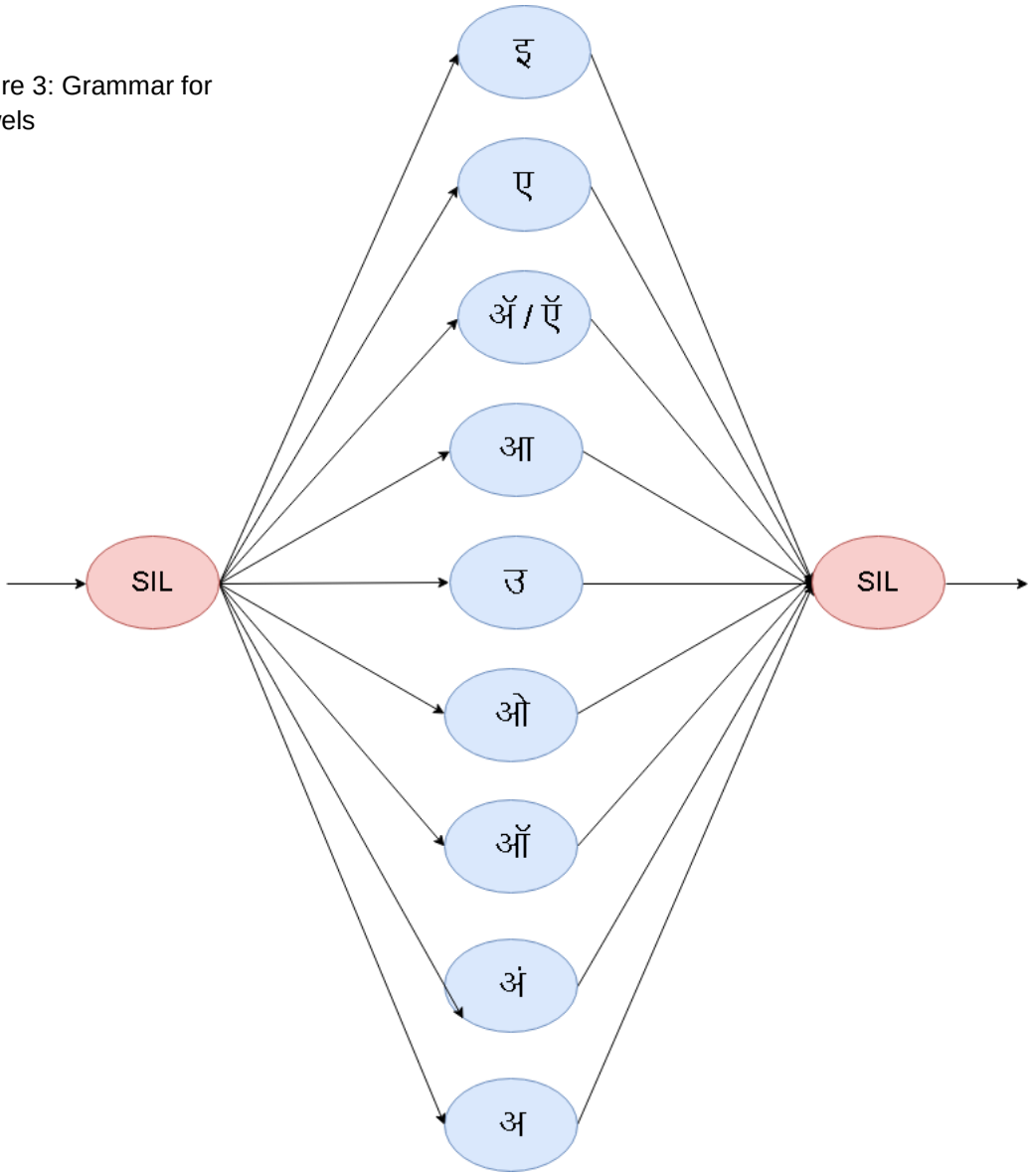
i. Building the task grammar.
ii. Recording the data.
iii. Creating transcription files for training data
iv. Extracting features from training data
v. (Re)training the acoustic models
vi. Evaluating the recognisers against the test data
vii. Analysis on recognition results

Task Grammer

For any Language there should be proper a grammar defined and for speech recogniser, we need to define task grammar for Konkani language.

The defined grammar was in BN form which starts with $sil defines the starting silence of the speech signal than $syl defines the word and again $sil which defines the end silence of the speech signal, where | stands for logical or so out of 9 vowels only one vowel which has the highest probability will be selected and given as the output.

Figure 3: Grammar for Vowels



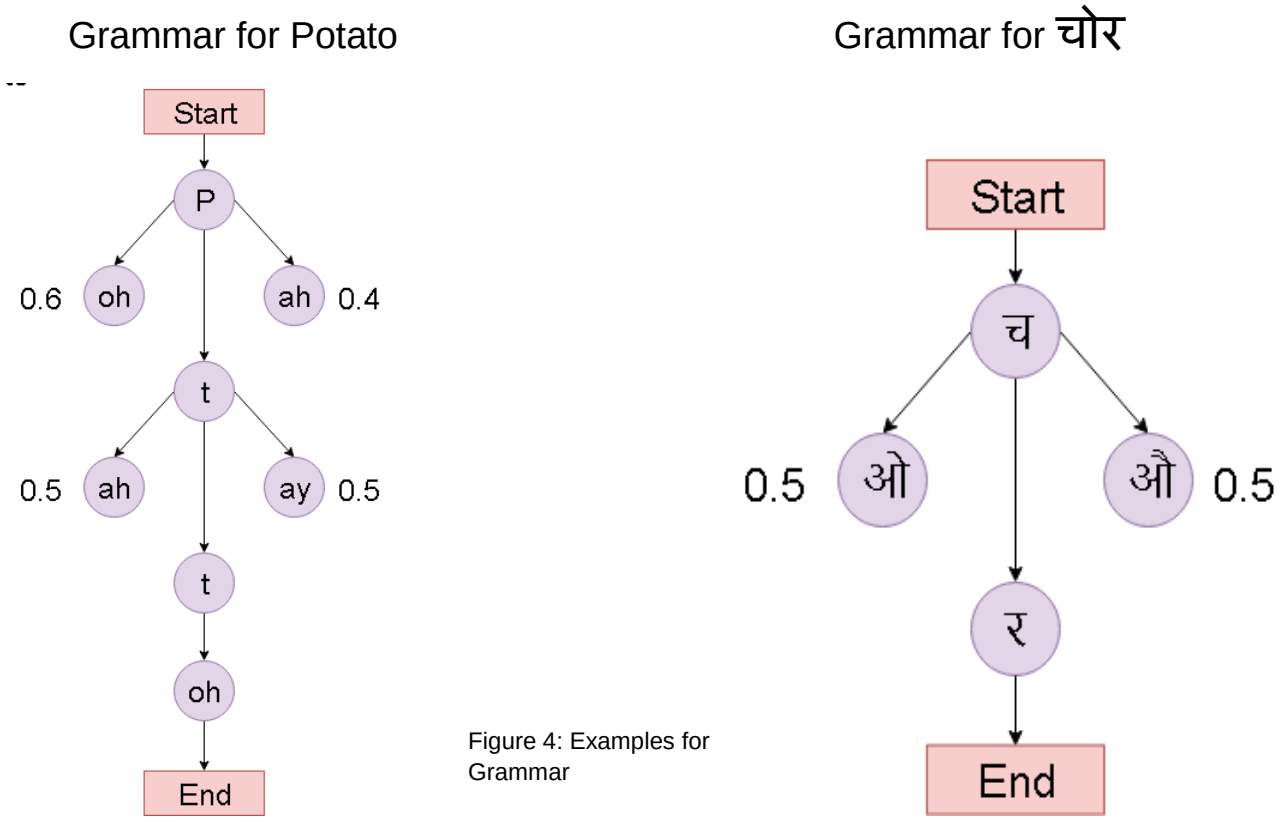


Figure 4: Examples for Grammar

Classification of Sound Units

The sound units of konkani language is broadly classified into 2 categories, vowels and consonants. The vowel sound units are further classified into 3 categories short vowels, long vowels, and diphthongs. Between short and long vowels, for long vowels the duration of production will be longer, typically nearly double that of short vowels. In case of diphthongs, 2 vowel sounds are produced in succession without any pause.

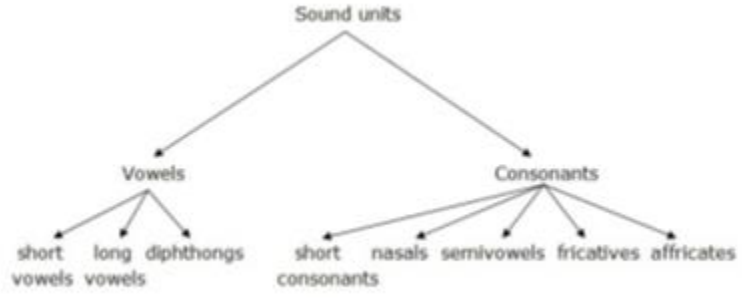


Figure 5: Sound Classification

Short Vowel	/a/(अ)	/i/(इ)	/u/(उ)	/e/(ए)	/o/(ओ)
Long Vowel	/a:/(आ)	/i:/(ई)	/u:/(ऊ)	/e:/(ए)	/o:/(ओ)
Diphthongs	/ai/(ऐ)	/au/(औ)			

Figure 6: Character Vowel Sound

Speech Recognition Techniques

Template Based approach: Since there are many variations in speech with respect to time, there will be many templates i.e. collection of recorded words in the training set for each word. An unknown word will be compared to these templates in the training set.

Knowledge Based/ Rule Based approach: A knowledge base technology requires an expert in speech Processing whose expertise generate and store very complex structured and unstructured data used by speech recognition system**.** But to get such an expert knowledge in speech recognition is very difficult**.** This method was impractical hence, they moved towards automatic learning, wherein system learn by itself based on the training**.**

Statistical Based approach: This is the most widely used speech recognition technique. Speech signals are the hidden states and features extracted from the speech signals are the visible states**.** In this approach we need to find the probability of the hidden states using the visible states. These new HMM based statistical modeling approaches are state**-**of**-**art approaches for speech recognition

Learning Based approach: These are neural networks or genetic programming. In this approach knowledge is acquired automatically through evolutionary process**.**

Artificial Intelligence: This is a hybrid approach of pattern recognition and acoustic phonetic approaches**.** It utilizes the concepts and ideas of pattern recognition and acoustic phonetic methods**.** It makes an attempt to recognize speech in similar way as humans, such as visualizing, analyzing and making decision based on acoustic features**.**

* Our Approach will be Statistical Based. (Below Schematic of statistical speech recognition.)

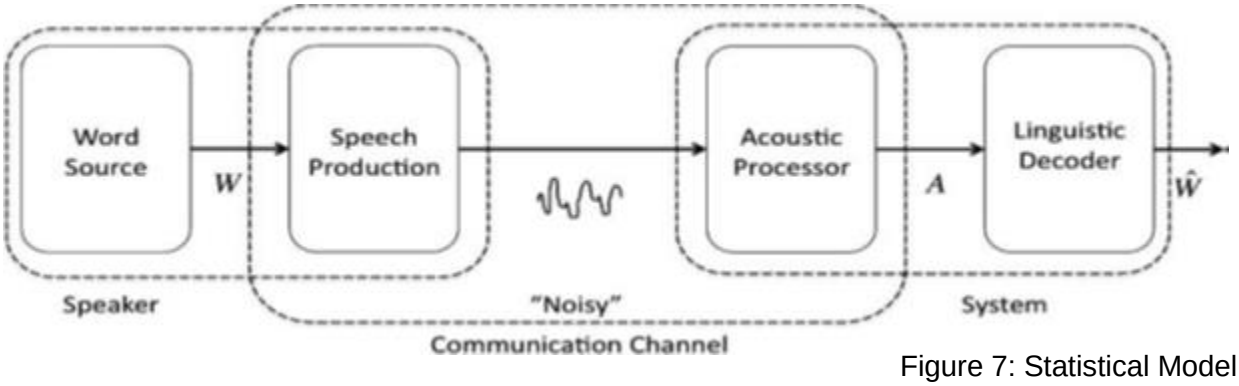


Figure 7: Statistical Model

Speech Recognition in Statistical approach

Some of the most known and used strategies in speech recognition are the statistical methods. The ASR aims to convert voice signal into text and this process can be formulated statistically as follows. Given a set of acoustic observations $O=(o_1,o_2,\ldots,0_n)$ (sequence of speech vectors, where oi is the speech vector observed at time i), which is the sequence of words $W=(w_1,w_2,\ldots..,w_n)$ that has the maximum probability:

$$W = \arg_w \max P(W|O) = \arg_w \max \frac{P(W)P(O|W)}{P(O)} \text{ ----------(1)}$$

Equation (1) specifies the most probable word sequence using Bayes rule, and P(O) - the probability of the speech utterance - can be ignored, because it is independent of the sequence W. Thus, (1) becomes:

$$W = \arg_w \max P(W)\, P(O|W) \text{ ----------(2)}$$

Equation (2) contains two factors which can be directly estimated: the prior probability of the word sequence P(W) and the probability of the acoustic data, given the word sequence P(O|W). The first factor P(W) can be estimated using only a language model, and the second factor can be computed on the basis of the acoustic model. The two models can be built independently, but they will be used together to recognize a spoken message.

How do computers process the speech data to recognize what speech it is?

To convert speech to on-screen text or a computer command, a computer has to go through several complex steps. When you speak, you create vibrations in the air. The **analog-to-digital converter** (ADC)translates this analog wave into digital data that the computer can understand. To do this, it **samples, or digitizes**, the sound by taking precise measurements of the wave at frequent intervals. The system filters the digitized sound to remove unwanted noise, and sometimes to separate it into different bands of **frequency** (frequency is the wavelength of the sound waves, heard by humans as differences in pitch).

When we speak, our voices generate little sound packets called **phones** (which correspond to the sounds of letters or groups of letters in words); so speaking the word cat produces phones that correspond to the sounds "c," "a," and "t." Although you've probably never heard of these kinds of phones before, you might well be familiar with the related concept of **phonemes**: any of the perceptually distinct units of sound in a specified language that distinguish one word from another. Although the difference between phones and phonemes is complex and can be very confusing, one way to remember it: phones are *actual* bits of sound that we speak (real, concrete things), whereas phonemes are *ideal* bits of sound we store (in some sense) in our minds (abstract, theoretical sound fragments that are never actually spoken).

Basically can be summed up in the following order

- **Digitization:** Converting Voice data from Analog to Digital.

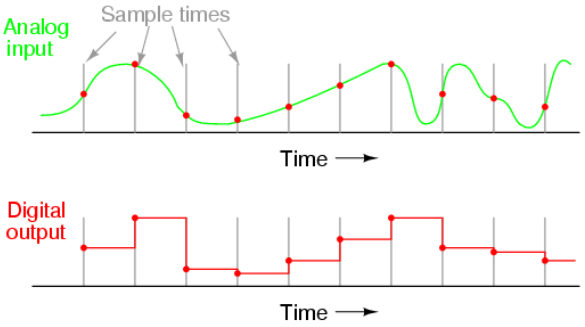


Figure 8: Digitization

- **Sampling:** is the conversion of a soundwave (a continuous signal) to a sequence of samples (a discrete-time signal) captured at discrete intervals of time
  - **Sampling Frequency:** The sampling frequency (or sample rate) is the number of samples per second in a Sound.

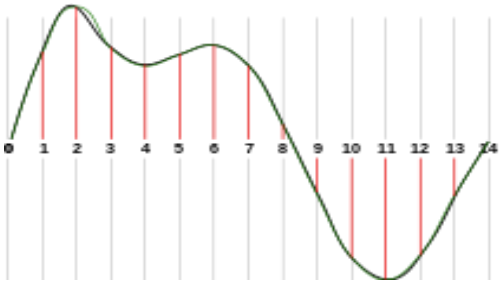


Figure 9: Sampling

Quantization: Process of converting a continuous range of values into a finite range of discrete values.

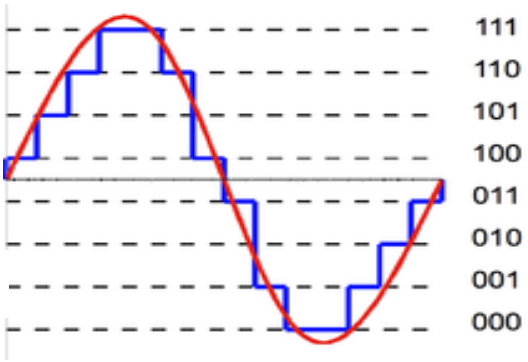
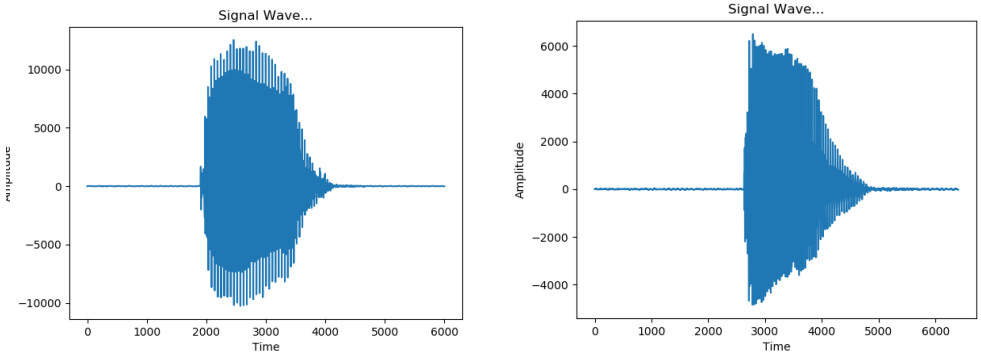


Figure 10: Quantization

Pattern Matching: Matching the features(MFCC features) of different sound samples and giving output based on the best match.

Below if the Frequency domain graph for two samples of "ɔ (ऑ)"



Speaker 1

Figure 11: Frequency Graphs

Speaker 2

Removing Noise: 1. Can be done if you record samples in a noise free environment like a studio.

2. By Using **Fourier transform**.

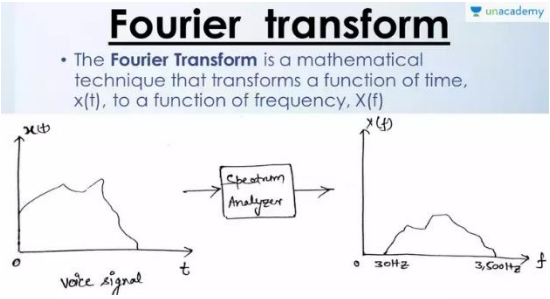


Figure 12: Fourier Transform

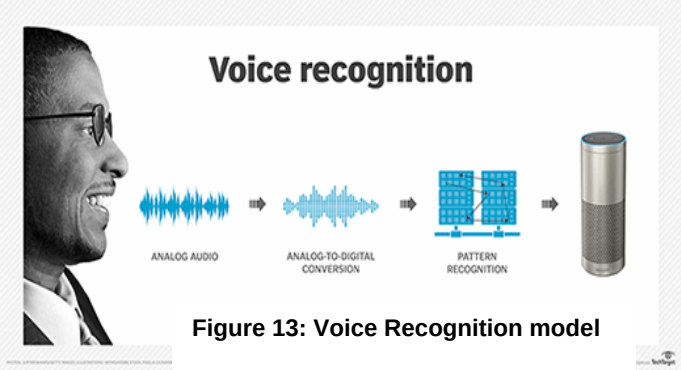


Figure 13: Voice Recognition model

Pattern Matching:

Feature Extraction using Mel-frequency cepstrum.

The first stage of speech recognition is to compress a speech signal into streams of acoustic feature vectors, referred to as speech feature vectors. The extracted vectors are assumed to have sufficient information and to be compact enough for efficient recognition The first task during Speech Recognition is to extract these features.

Mel Frequency Cepstral Coefficients (MFCCs) are a feature widely used in automatic speech and speaker recognition. They were introduced by Davis and Mermelstein in the 1980's, and have been state-of-the-art ever since.

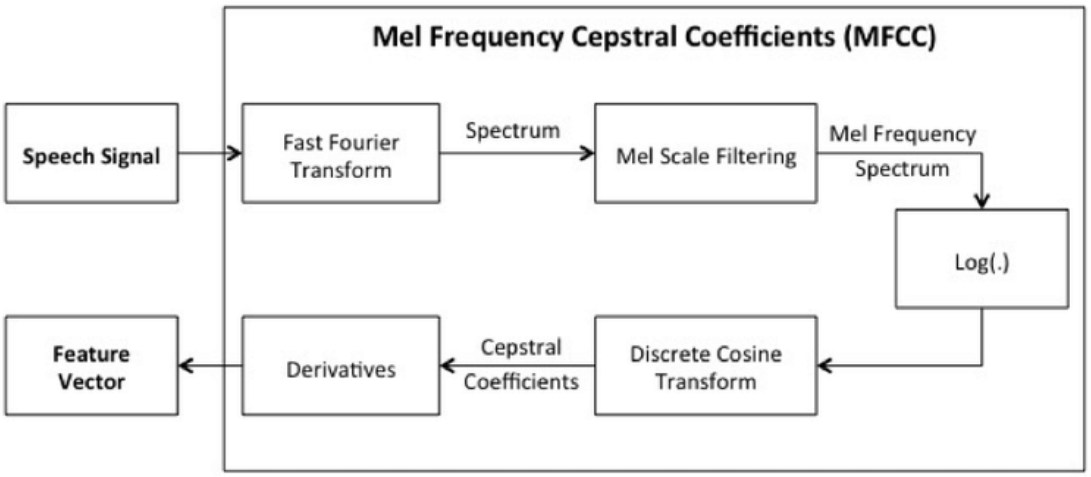


Figure 14: MFCC block diagram

1) Advantage

As the frequency bands are positioned logarithmically in MFCC, it approximates the human system response more closely than any other system.

2) Disadvantage

MFCC values are not very robust in the presence of additive noise, and so it is common to normalize their values in speech recognition systems to lessen the influence of noise MFCC Algorithm for Speech Feature Extraction

MFCC Feature Extraction Algorithm for Speech Recognition

Step 1: Framing and Windowing of signal

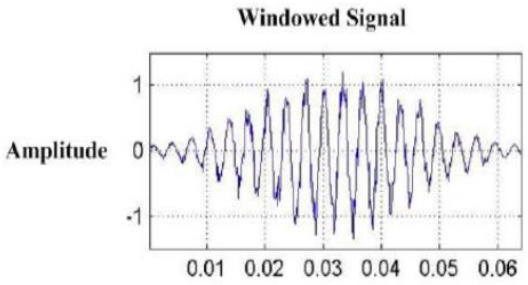


Figure 15: Windowing

Framing is the process of blocking the speech signal into frames of n samples in the time domain. After framing step, each individual frame is windowed using window function.

The window function is a mathematically function that is used to minimize signal discontinuities at the beginning and at the end of each frame by taking the block of next frame in consider and integrates all closet frequency lines. This step makes the end of each frame connects smoothly with the beginning of the next.

Step 2: Taking fourier transform of a signal

In this step each tone with actual frequency that is measured by hertz is converted into subjective pitch that is measured by scale called "mel"
scale. The main purpose from the process of converting is to mimic the behaviour of ear, human ear act as filter, it concentrates on only certain frequency.

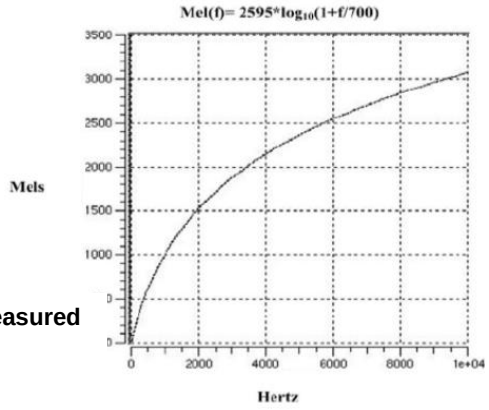


Figure 16: The formula that is used to convert from actual frequency that is measured by Hz into subjective pitch that is measured by mel scale

Step 3: Mapping of powers of spectrum onto mel

In digital signal processing, signals are studied in three domains frequency domain, time domain and wavelet domain.

Time domain is a term that is used to describe the domain for analysis of signals with respect to time rather than frequency, when an audio signal is examined in the time domain, the X-axis is time, so the value of the Y-axis depends on the changing of the signal with respect to time .

Frequency domain is a term that is used to describe the domain for analysis of signals with respect to frequency, rather than time. When an audio signal is examined in the frequency domain, the X-axis is frequency, so the value of the Y-axis depends on the changing of the signal with respect to frequency .

The **Fourier Transform** is used to convert each frame from the time domain into the frequency domain.

Step 4: Logs of powers at each of the mel frequencies

The logarithm of powers is taken at each of the Mel frequencies

Step 5: Discrete cosine transform of the list of mel log powers

This step is to convert the log mel spectrum into the time domain using discrete cosine transform. The MFCC feature is the amplitudes of the resulting spectrum, this representation of speech spectrum provides a good representation of the local spectral properties. The results are produced from this step is called mel frequency cepstrum coefficients

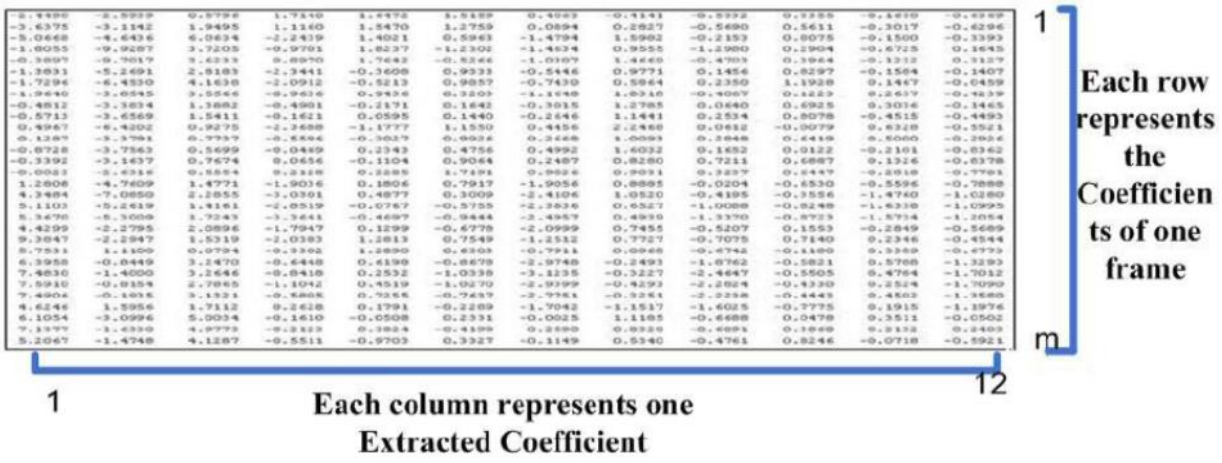


Figure 17: The numeric representation of MFCC matrix

Technologies Used: MFCC, HMM

Model Used: Hidden Markovian Model

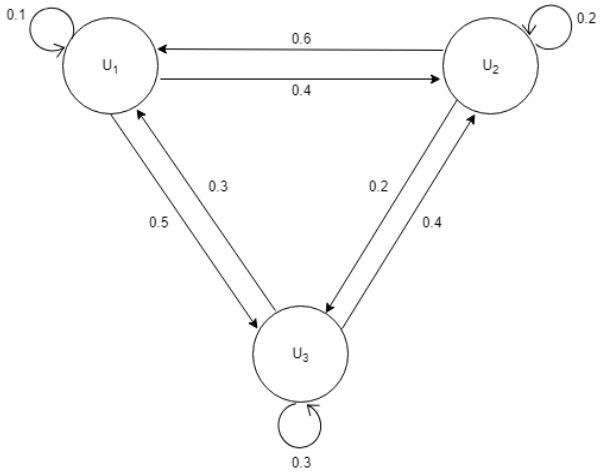
Hidden Markovian Model(HMM)

The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multidimensional) probability distribution []. Transitions among the states are governed by a set of probabilities called transition probabilities. The **hidden Markov model** can be represented as the simplest dynamic Bayesian network.

Markov Process

A Markov process is a random process in which the future is independent of the past, given the present. Thus, Markov processes are the natural stochastic analogs of the deterministic processes described by differential and difference equations. They form one of the most important classes of random processes.

Figure 18: Example of HMM:



Transition Probability

	U1	U2	U3
U1	0.1	0.4	0.5
U2	0.6	0.2	0.2
U3	0.3	0.4	0.3

Observation Probability

	R	G	B
U1	0.3	0.5	0.2
U2	0.1	0.4	0.5
U3	0.6	0.1	0.3

P(x1**)=**0.2 **P(**x2**)=**0.1 **P(**x3**)=**0

Formula: t(U1|R) = [((P(x1)*P(U1|U1)) + (P(x2)*P(U2|U1)) + (P(x3)*P(U3|U1)) * P(U1|R)]

Trellis Diagram

t->	0	R	R	G
0	0	0	0	0
U1	0.2	0.024	0.009	0.004866
U2	0.1	0.01	0.00404	0.0051808
U3	0	0.072	0.02136	0.0011716

3 problems of HMM:

1. Evaluation problem:

-Model(Θ), consists of hidden states and visible states. Given the model(Θ), Θ=> w, v, aij, bjk. Calculate the probability to get the sequence in given data.

-Forward Algorithm and Backward Algorithm is used for this problem

2. Decoding Problem:

-Given a sequence of symbols and a model, finds out the most likely sequence of states that produced the sequence.

-Algorithm used is, Best State Sequence, even known as Viterbi algorithm

3. Learning Problem:

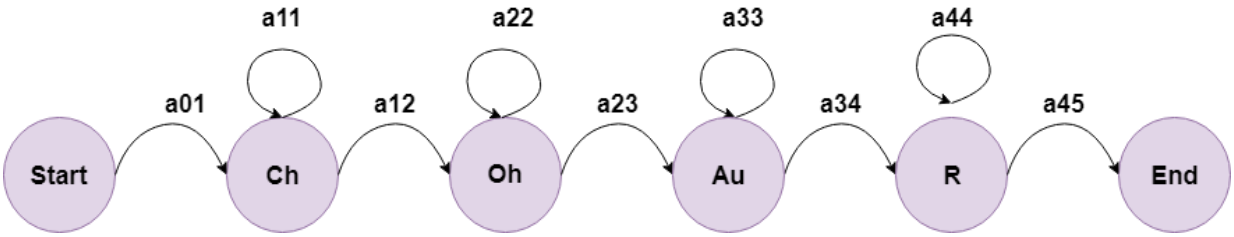
-Given a model structure and a set of sequences, find the model that best fits the data

-Algorithm used

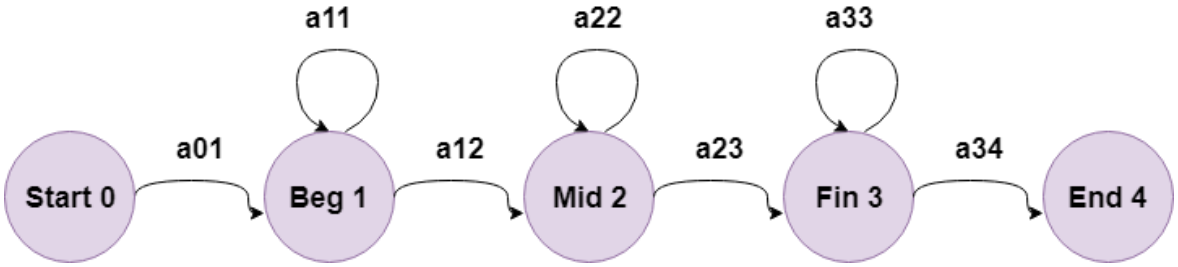
1. Forward-backward algorithm, also known as Baum welch Algorithm.
2. Viterbi training (not the same as decoding)
3. MLE (Maximum Likelihood Estimation)

4.

HMM for the word “CHOR”:



Each Phone has three subphones:



Resulting HMM word model for “Chor” with their subphones:

Data Collection

For the completion of this project data samples were collected from our family members and villagers. Some of them are also from our classmates and friends.

Attribute	Value
Total number of samples collected	450
Total number of data samples used for training	350
Total number of samples used for testing	100
Total number of speakers	50
Format of data samples	32-bit wav
Rate of each wave file	8000 Hz
Channel Type	Mono
Average duration of each session	1-2 secs

The speakers belong from different regions of Goa, having different accents. Each Speaker was made to say konkani vowels and each vowel was stored as a separate wav file.

Implementation Details :

Python Libraries: Numpy, MatPlotLib, python_speech_features, hmmlearn, scipy.

Voice data will be converted into **wav** format and each voice sample for different vowels will be stored into separate folders. This folder will then be fed into our program. The Hmm will train on test data and test it with our test files provided within the program and display the recognized output.

We have 450 Samples of voice data out of which 350 will be used to Train and 100 will be used for Testing.

Numpy: NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

python_speech_features: This library provides common speech features for ASR including MFCCs and filterbank energies.

MatPlotLib: matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.

hmmlearn: Simple algorithms and models to learn HMMs (Hidden Markov Models) in Python,

Follows scikit-learn API as close as possible, but adapted to sequence data, Built on scikit-learn, NumPy, SciPy, and matplotlib,

SciPy: Python library used for scientific computing and technical computing.

All these Libraries work together to build a working ASR for our dataset.

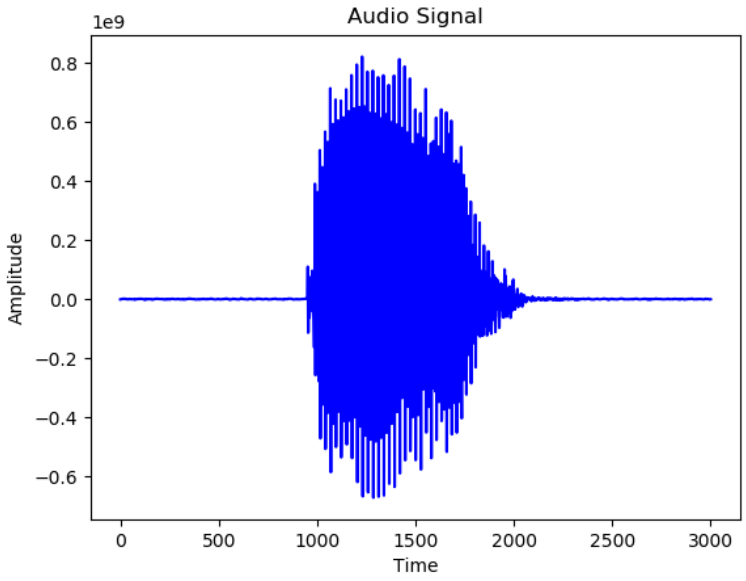
Programs:

Reading Sample and Plotting it on Frequency-Amplitude Graph

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
import wave

#read
sampling_freq, audio = wavfile.read('Speaker_1.wav')
spf = wave.open('Speaker_1.wav','r')

#print values
print('\n Shape:',audio.shape,'\n Datatype:',audio.dtype,
'\n Duration:',round(audio.shape[0]/ float(sampling_freq),3),'seconds',
'\n Sample Width : ',spf.getsampwidth(),
'\n Number of Frames : ',spf.getnframes(),
'\n Rate in Hz : ',spf.getframerate())

channel = spf.getnchannels() # return 1 for mono 2 for sterio
#checking channel
if channel == 1:
    print (' Channel Type : Mono')

signal = spf.readframes(-1)

signal = np.frombuffer(signal, 'Int32')

#plotting the chopped audio signal
plt.figure(1)
plt.plot(signal,color='blue')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Audio Signal')
plt.show()
```

OutPut:

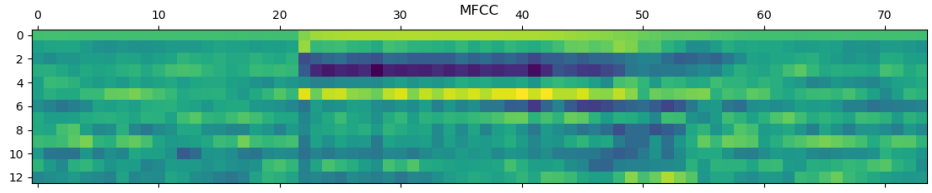
Shape: (6002,)
Datatype: int16
Duration: 0.75 seconds
Sample Width : 2
Number of Frames : 6002
Rate in Hz : 8000
Channel Type : Mono

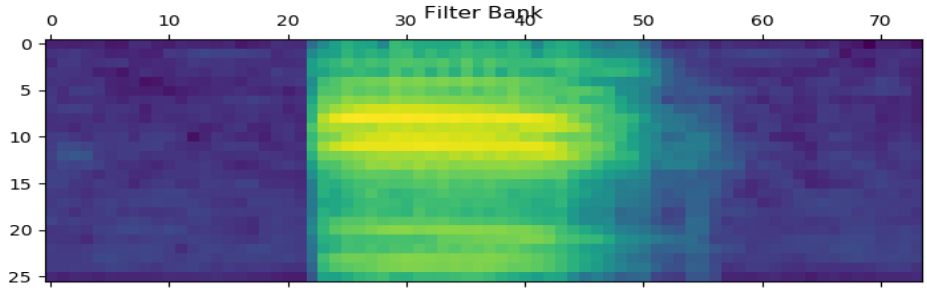


Extracting MFCC features and Plotting it

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
from python_speech_features import mfcc, logfbank

#read input file
sampling_freq, audio=wavfile.read("Speaker_1.wav")

#extract MFCC and filter bank features
mfcc_features=mfcc(audio, sampling_freq)
filterbank_features=logfbank(audio, sampling_freq)

#print parameters
print('Filter Bank Features')
print(filterbank_features[0])
print('\nMFCC Features')
print(mfcc_features[0])
print('\nMFCC:\nNumber of Windows=',mfcc_features.shape[0])
print('Length of each feature=',mfcc_features.shape[1])
print('\nFilter Bank:\nNumber of Windows=',filterbank_features.shape[0])
print('Length of each feature=',filterbank_features.shape[1])

#plot the features
mfcc_features=mfcc_features.T
plt.matshow(mfcc_features)
plt.title('MFCC')

filterbank_features=filterbank_features.T
plt.matshow(filterbank_features)
plt.title('Filter Bank')
plt.show()
```

output:





Filter Bank Features
[2.25562372 3.21349763 2.57533333 3.44338088 3.26456831 3.5670379
3.61429085 3.57201952 3.06998342 3.07303491 3.75600913 3.89337285
4.18744586 3.78194135 4.16313383 4.13157002 3.85859637 3.59620185
3.96188407 4.43441117 3.29417542 3.56152053 3.93266029 3.73106845
3.78551128 2.57422011]

MFCC Features
[6.92852554 -2.86066241 -5.91025616 1.26280841 -2.80199905
-2.67530834 -8.07705821 0.27379353 0.22364916 5.55186987
-4.91452155 3.02552492 -10.38716829]

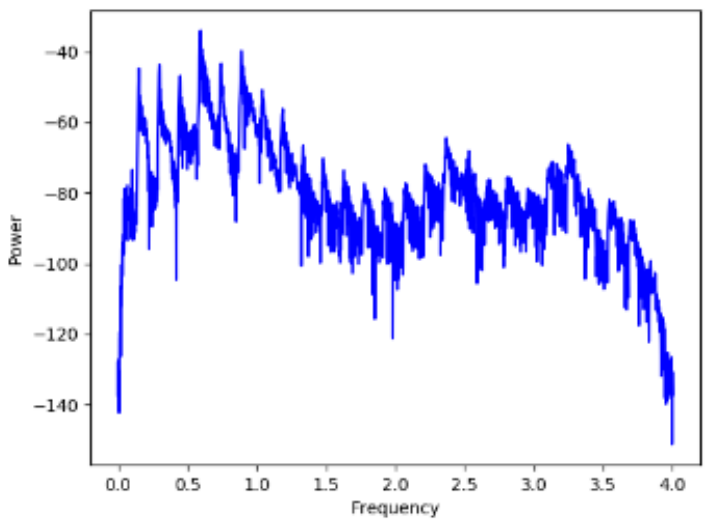
MFCC:
Number of Windows= 74
Length of each feature= 13

Filter Bank:
Number of Windows= 74
Length of each feature= 26

Applying Fourier Transform

```
#fourier transform
import numpy as np
from scipy.io import wavfile
import matplotlib.pyplot as plt

sampling_freq, audio =wavfile.read('Speaker_1.wav')

audio=audio/(2.**15)

len audio=len(audio)#length of audio

#apply fourier transform
transformed_signal=np.fft.fft(audio)
half_length=np.ceil((len_audio+1)/2.0)
transformed_signal=abs(transformed_signal[0:int(half_length)])
transformed_signal/=float(len_audio)
transformed_signal**=2

#extract length of transformed signal
len_ts=len(transformed_signal)

#taking care of even/odd cases
if(len_audio%2):
    transformed_signal[1:len_ts]*=2
else:
    transformed_signal[1:len_ts-1]*=2

#extract power in dB
power=10*np.log10(transformed_signal)

#build the time axis
x_values=np.arange(0,half_length,1)*(sampling_freq/len_audio)/1000.0

#plot the figure
plt.figure()
plt.plot(x_values,power,color='blue')
plt.xlabel('Frequency')
plt.ylabel('Power')
plt.show()
```

OUTPUT



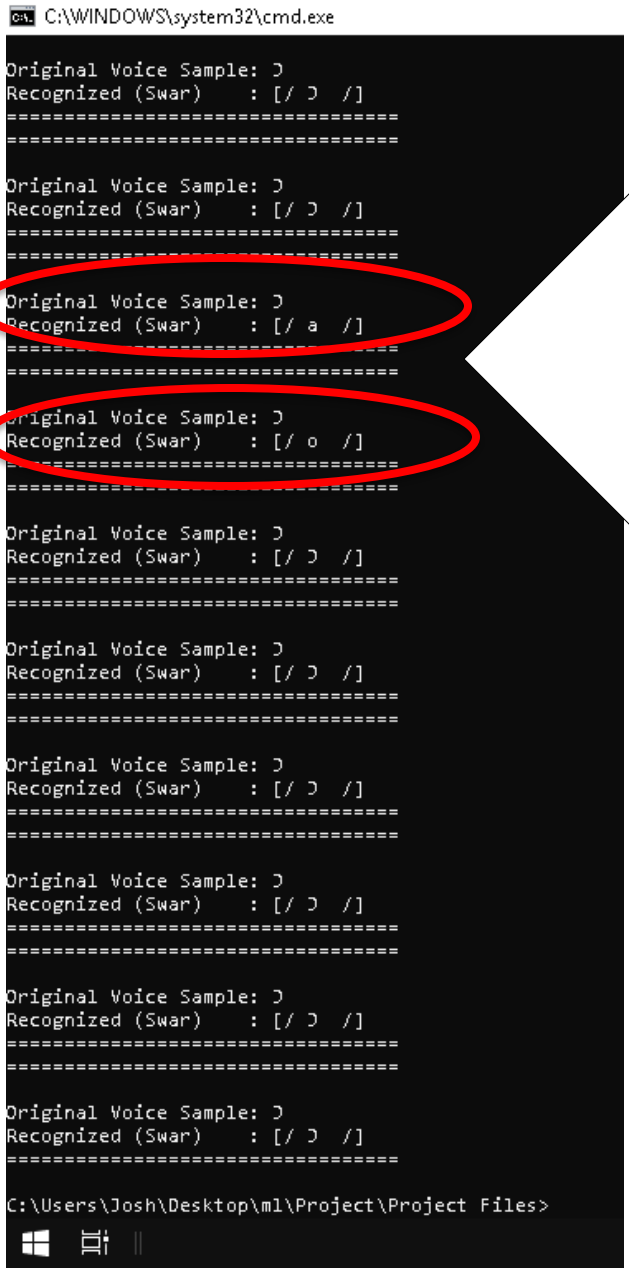
Training the Model and Testing it out with our Test samples

```
import os
import argparse
import warnings

import numpy as np
from scipy.io import wavfile

from hmmlearn import hmm
from python_speech_features import mfcc

# Define a function to parse the input arguments
def build_arg_parser():
    parser = argparse.ArgumentParser(description='Trains the HMM-based speech \
            recognition system')
    parser.add_argument("--input-folder", dest="input_folder", required=True,
            help="Input folder containing the audio files for training")
    return parser

# Define a class to train the HMM
class ModelHMM(object):
    def __init__(self, num_components=4, num_iter=1000):
        self.n_components = num_components
        self.n_iter = num_iter

        self.cov_type = 'diag'
        self.model_name = 'GaussianHMM'

        self.models = []

        self.model = hmm.GaussianHMM(n_components=self.n_components,
                covariance_type=self.cov_type, n_iter=self.n_iter)

    # 'training_data' in a 2D numpy array where each row is 13-dimensional
    def train(self, training_data):
        np.seterr(all='ignore')
        cur_model = self.model.fit(training_data)
        self.models.append(cur_model)

    # Run the HMM model for inference on input data
    def compute_score(self, input_data):
        return self.model.score(input_data)

# Define a function to build a model for each word
def build_models(input_folder):
    # Initialize the variable to store all the models
    speech_models = []

    # Parse the input directory
    for dirname in os.listdir(input_folder):
        # Get the name of the subfolder
        subfolder = os.path.join(input_folder, dirname)

        if not os.path.isdir(subfolder):
            continue
```

```
        # Extract the label
        label = subfolder[subfolder.rfind('/') + 1:]

        # Initialize the variables
        X = np.array([])

        # Create a list of files to be used for training
        # We will leave one file per folder for testing
        training_files = [x for x in os.listdir(subfolder) if x.endswith('.wav')][:-1]

        # Iterate through the training files and build the models
        for filename in training_files:
            # Extract the current filepath
            filepath = os.path.join(subfolder, filename)

            # Read the audio signal from the input file
            sampling_freq, signal = wavfile.read(filepath)

            # Extract the MFCC features
            with warnings.catch_warnings():
                warnings.simplefilter('ignore')
                features_mfcc = mfcc(signal, sampling_freq)

            # Append to the variable X
            if len(X) == 0:
                X = features_mfcc
            else:
                X = np.append(X, features_mfcc, axis=0)

        # Create the HMM model
        model = ModelHMM()

        # Train the HMM
        model.train(X)

        # Save the model for the current word
        speech_models.append((model, label))

        # Reset the variable
        model = None

    return speech_models

# Define a function to run tests on input files
def run_tests(test_files):
    # Classify input data
    for test_file in test_files:
        # Read input file
        sampling_freq, signal = wavfile.read(test_file)

        # Extract MFCC features
        with warnings.catch_warnings():
            warnings.simplefilter('ignore')
            features_mfcc = mfcc(signal, sampling_freq)

        # Define variables
```

```
        max_score = -float('inf')
        output_label =  [float("-inf")]

        # Run the current feature vector through all the HMM
        # models and pick the one with the highest score
        for item in speech_models:
            model, label = item
            score = model.compute_score(features_mfcc)
            if score > max_score:
                max_score = score
                predicted_label = label

        # Print the predicted output
        start_index = test_file.find('/') + 1
        end_index = test_file.rfind('/')
        original_label = test_file[start_index:end_index]
        print('==================================')
        print('\nOriginal Voice Sample:', original_label[:-5])
        print('Recognized (Swar)    : [/', predicted_label[:-5],'/]')
        print('==================================')

if __name__=='__main__':
    args = build_arg_parser().parse_args()
    input_folder = args.input_folder

    # Build an HMM model for each word
    speech_models = build_models(input_folder)

    # Test files -- the last 10 file's in each subfolder
    test_files = [
    'voice/ɔ ( ऑ )/Speaker_41.wav',
    'voice/ɔ ( ऑ )/Speaker_42.wav',
    'voice/ɔ ( ऑ )/Speaker_43.wav',
    'voice/ɔ ( ऑ )/Speaker_44.wav',
    'voice/ɔ ( ऑ )/Speaker_45.wav',
    'voice/ɔ ( ऑ )/Speaker_46.wav',
    'voice/ɔ ( ऑ )/Speaker_47.wav',
    'voice/ɔ ( ऑ )/Speaker_48.wav',
    'voice/ɔ ( ऑ )/Speaker_49.wav',
    'voice/ɔ ( ऑ )/Speaker_50.wav'
    ]
    #for root, dirs, files in os.walk(input_folder):
    #    for filename in (x for x in files if '15' in x):
    #        filepath = os.path.join(root, filename)
    #        test_files.append(filepath)

    run_tests(test_files)
```

Observations for Different Input:



After Training and testing the samples, we found that for testing 10 samples of the vowel (ऑ) , 2 of the samples were incorrectly classified. So that Gives us a probability of 80% accurate.

Similarly..

P(ɔ)=8/10 , P(e)=10/10

P(ɨ)=8/10 , P(a)=8/10

P(ə)=10/10 , P(i)=7/10

P(o)=6/10 , P(u)=10/10

P(ɛ)=9/10

Therefore total performance could be calculated as

(8+10+8+8+10+7+6+10+9/100)∗100=

So its about 76% accurate

Limitations and Future Scope

A Speech Recognition is a challenging task with lots of difficulties in implementation since speech has lots of parameters which need to be accurately captured. Some of the parameter that limits the application are as follows:

i. Number of speakers: More the number of speakers, more will the training data, hence more will the accuracy of the system. To build speaker independent system with more robust and higher accuracy we need to have large speech database as training data.

ii. Utterance: Each utterance should be properly recorded, such as starting silence uttered word and again ending silence. Also, each utterance should be properly labeled with tags.

iii. Speaker: There can be variations in speech in terms of age, sex and accent etc of the speaker.

iv. Environment: Noise pollution negatively affects speech recognition system in terms of distorted signal noise which will decrease the speech recognition system performance.

Future Scope: A Properly developed acoustic model for the language can significantly improve speech recognition accuracy, whether its phoneme level or word level acoustic implementation.

References.

1. The Konkani Language: Nature and Tradition. –Dr. S. B. Kulkarni.
2. https://www.researchgate.net/publication/49587837_A_Review_on_Speech_Recognition_Technique
3. Using Statistical Methods in a Speech Recognition System for Romanian Language- Daniela Schiopu*
4. TSKK, Konkani Basic Course, by Matthew Almeida, S.J.pages (8-9).
5. https://www.researchgate.net/publication/266895811_Arabic_Audio_News_Retrieval_System_Using_Dependent_Speaker_Mode_Mel_Frequency_Cepstral_Coefficient_and_Dynamic_Time_Warping_Techniques
6. Introduction to Machine Learning - Ethem Alpaydin