EE 769 Project Report
Speaker Recognition Application

Mohit Patil (150050017), *CSE, IIT Bombay,* and Huzefa Chasmai(15D170013), *CSE, IIT Bombay,*
Guide: Prof Amit Sethi, *EE dept., IIT Bombay*

***Abstract*—This report contains the discussion and implementation details of our course project of making a base Speaker Recognition Application, easily extendable to make applications on Voice Bio-metric Attendance System as well as Voice Based App-locker. We discuss the basic issues in Speaker Recognition with respect to the two problems of Speaker Verification and Speaker Identification. The report also discusses our approach using Hidden Markov Models to model the speech for a particular person using the MFCCs extracted features from the audio input file. It also discusses the implementation details as well as the framework for the application. Finally we present our results on the test data as well as discuss the shortcomings and improvements of our project along with the references.**

***Index Terms*—Speaker Recognition, Voice Bio-metric Attendance Application, Voice Based AppLocker, Speaker Identification, Speaker Verification**

I. Introduction

The speech signal conveys to a great extent the information of the identity of the speaker to the listener. While speech recognition aims at recognizing the word spoken in speech, the goal of automatic speaker recognition systems is to extract, characterize and recognize the information in the speech signal conveying speaker identity. In our project we try to exploit this information as being a key idea for identification and verification of the speaker in our applications. We have in mind two applications that we are targeting :

- *Voice Based Application Locker:* An App locker which protects other applications and unlocks those applications on Voice Input given by the user
- *Voice Bio-metric Attendance System:* A system for biometric attendance using voice samples as input

The problem of Speaker recognition can be classified into **speaker identification** and **speaker verification**. Speaker identification is the process of determining from which of the registered speakers a given utterance comes. Speaker verification is the process of accepting or rejecting the identity claimed by a speaker. Most of the applications in which voice is used to confirm the identity of a speaker are classified as speaker verification. Our applications are also based on these issues separately. The first application deals with Speaker Verification since the user needs to be verified for unlocking of the application whereas the second application has to deal with Speaker Identification, since it has to identify which student is giving the attendance.

Keeping in mind the applications above and given the time constraints we decided to build a base application that can be easily extended to either of the two applications above. The base application has the following features

- Model Creation Module : For each user, there is a an option to upload audio files to train/improve their corresponding HMM model on the server. (We have added support to store the Audio inputs and given more samples, to consolidate and retrain the model for that speaker on both the new and old inputs.)
- Speaker Identification Module : Feature for uploading new audio samples to the backend and displaying which speaker it matched to with highest probability
- Speaker Verification Module : Feature for uploading new audio samples to the backend and checking for the match of the audio samples with the users model

So with respect to implementing our application we discuss the learning algorithm using HMMs as well other implementation details in the next few sections.

II. Learning Algorithms

We use the Hidden Markov Model as our predictive model for the speakers voice.

A. Hidden Markov Model

Hidden Markov Models are finite state devices with fixed count of states and it is deployed to characterize the spectral assets of voice signal. HMM have two kinds of possibilities. In HMM the output is visible (Known Variables) but the states are not visible directly (Unknown Variables), output visibility is dependent on the states and the probability distribution used to produce the output. Hidden Markov Modelling will be characterized by the following hyper-parameters:

- The number of hidden states of model, represented by n.
- Distinct observation symbols correspond to the physical output is represented by S of the certain model

Our implementation :

1) Plan: Let $S_1, S_2, ...S_n$ be the set of speakers and **O** be an utterance, then we predict the speaker of the utterance **O** as:

$$S_i = argmax_i(Pr(\mathbf{O}/(HMM(S_i)))) \tag{1}$$

2) Building the HMM: Let the utterances spoken by S_i in the training set be

$$O_1, O_2, ..., O_T \tag{2}$$

Sound is produced when air passes through vocal track. Different shapes of vocal tract are possible while a sound is produced. A sound can be produced from any of the shapes of the vocal track and we dont know which shape produced the sound. It has been observed that the order of change of the vocal tract shapes corresponds to the text spoken while the correspondence between the vocal tract shapes and the sounds emitted identifies a speaker. So the shapes of the vocal tract are hidden states. For identifying a speaker, the emission probabilities emitted in these hidden states play an important role. So we keep our transition probabilities constant. That means if we assume **m** states, the transition probability matrix would be an **m x m** matrix where each entry is **1/m**. We assume that the emission sound of a shape of vocal tract follows a Gaussian distribution. So we model the emission probabilities as a vector of shape **d**, then
For each state we have μ_d, Σ_d
for estimating emission vectors. In total for the HMM we have $\Pi = (\Pi_1, \Pi_2, ..., \Pi_m)$ start probabilities and (μ_i, Σ_i) for a state.
i.e for each state we have to estimate (Π_i, μ_i, Σ_i).
Given $O = (O_1, O_2, ..., O_T)$, the acceptance probability by the HMM is equal to

$$P_{acc} = max_i(\Pi_i \times N(\mu_i, \Sigma_i, O_i)) \times \prod_i (max(N(\mu_i, \Sigma_i, O_i)) \tag{3}$$

This acceptance probability resembles that of Gaussian Mixture Model with **m** number of gaussians. We compare the acceptance probabilities across all HMMs and give out the label corresponding to the HMM which gives maximum acceptance probability

B. Feature Extraction and Data Set

1) Data Set: We are using the LibriSpeech ASR Corpus's dev-clean dataset for training and testing our HMM models.

2) Mel-cepstral Frequency Coefficients (MFCCs): MFCC is used to abstract the features and to calculate the coefficient which represents the cepstral frequency. Short term power spectrum of human voice is represented by Mel-cepstral Frequency Coefficients. In this method the frequency bands are separated with equal spacing that estimates the human voice more accurately.

We assume the mfccs, their deltas and their double deltas are sufficient enough to represent an emission sound frame. Each utterance is divided into frames of length 25ms with a stride of 10ms. We extracted 13 values from each frame, Combining these 13 values with their deltas and double deltas gives us a 39 dimensional vector for each frame. We took the number of states in the HMM model as 50. Increasing the number of states overfits model to the training data of the speaker while decreasing the number of states will reduce the speaker specificness of the HMM, We train using the Expectation Maximization algorithm and the algorithm converges in less than 10 iterations for each speaker. Only 10 utterances per speaker are used as they give good enough results while reducing the training time. librosa library is used for extracting mfccs. HMMLearn is used for creating the HMM model

C. Training and Results

Tuning of hyper-parameters : Used 20% of the data for validation to tune the parameters. Hyper-parameters : number of states in hmm, number of speakers in training set, number of audio files per speaker, number of iterations of the hmm training EM algorithm. Th accuracy mentioned is the validation accuracy. The key values of the hyperparameters and their accuracies are listed below.

states	speakers	files/speaker	iterations	accuracy
50	40	40	10	98.99
50	40	10	10	92.67
40	10	10	10	91.41
50	40	20	10	91.92

Based on the validation as well as time taken to train and test we finalized the following parameters : Number of iterations of the EM algorithm : 10; number of states in the HMM : 50; number of files per speaker : 10

D. Verification Procedure

For each of the models we maintain a the worst log likelihood on the training data samples. If the inputs expected log likelihood is greater than the min log likelihood we very the speaker and also send the probability e^{LL} where LL is the log likelihood obtained by training this test sample on the speakers HMM. This is the value we are currently returning in out Speaker verification module

III. Implementation of Application

A. Front End

The front end is build using react-native framework. React Native is a Javascript framework for rendering mobile application in iOS and Android. React is a Javascript library for building user interfaces which targets mobile platforms provided by Facebook. We used the following libraries of react native : react-native-audio, react-native-sound, react-native-simple-toast and react-navigation. The app is build on react native version v-0.55.

B. Back End

The back end is built using **Django** framework in python. Django takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. The data modeling and testing is done using this server. For learning and feature extraction from data input we used these libraries in python: librosa (audio file preprocessing), HMMlearn(for the hmm models) and numpy.

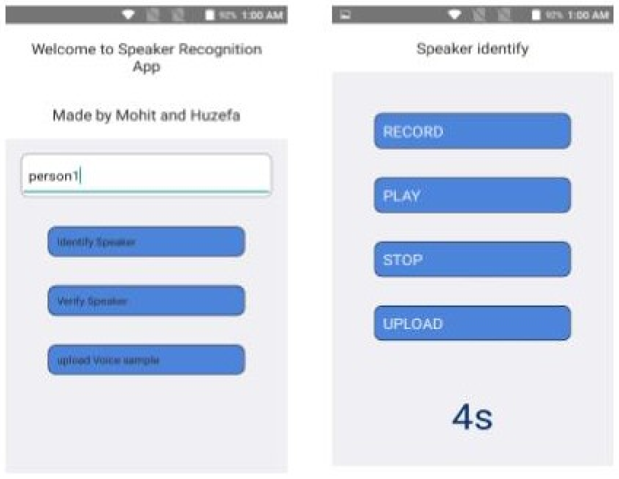

Fig. 1. HomePage and Speaker identification page

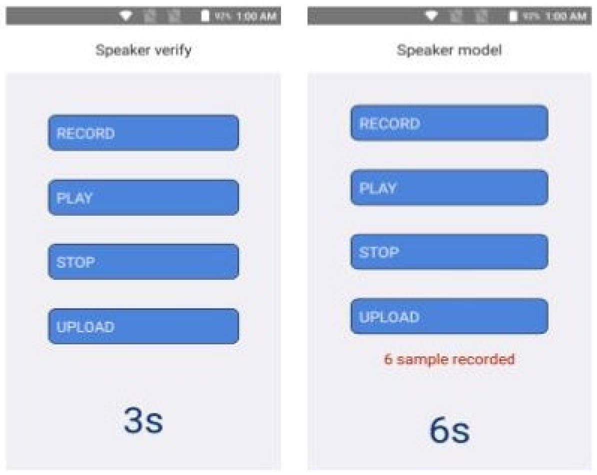

Fig. 2. Speaker verification page and upload sample page

IV. Application Framework

A. Front End Framework

The application starts with the home page[Fig. 1] consisting of text input to enter the name and different button to go to other screens. The application consist of 4 screens (including home screen). The *AudioExample* and *AudioScreen* are used to record, play and upload the audio file to server. Finalpage is the screen we land up after uploading audio file to server. AudioScreen allows user to upload multiple sample files and these files are used for training the HMM for that user. We have three different options on the home page : one for identifying speaker, one for verifying speaker and last one for uploading sample file. Identify speaker page makes use of AudioExample and records a sample and uploads it to the server for identifying speaker, the success of identification leads to the final page screen which shows name of the speaker. Verification page also make use of the AudioExample page, it is used to record a sample and upload it to server which in turn responds with the user with which the sample matches with the speaker's already created model. Success of verification leads to final page which displays the user matching of the audio file, were as failure to verify toast's the error message and ask to retry. The upload sample page makes use of audio screen. on audio page multiple voice samples can be recorded for the speaker and uploaded to server, the server used these samples to train HMM and create model for the speaker. After models are created we land up on the last page showing the message about the status of model creation for the speaker.

B. Back End Framework

The Backend consist of the Django server which has separate handling of the three types of requests : Model Creation, Speaker Identification and Speaker Verification. For each of the users of the application the backend maintains the list of audio files used to create the HMM model for that speaker and also the HMM model is stored in the pickle format. Upon a query for speaker identification, backend loads all the HMMs present, gets the log likelihood of the probability for each model of generating the input and returns to the user the model with the highest log likelihood. One additional feature is that upon adding new train data the model for that speaker is recreated using the new as well as the old data resulting in training on more samples.

V. Conclusion

This simple HMM model was able to produce really good results. But while HMM models gave good results for speaker identification it was very difficult to use it for speaker verification. The ratio of the acceptance probabilities of an utterance O on an HMM model of the actual speaker and the HMM models of the other speakers is small so threshold of acceptance for verification is very difficult to set. We can increase this ratio by overfitting the model (by increasing the number of states) to the utterance, but then our model won't give good results on unseen utterances because of the overfitting. One other Flaw in this approach is that we must know beforehand all the future speakers. To summarize
Pros:

- Very fast training
- Very small training data per speaker to give good results
- Accuracy of around 90%

Cons:

- Cannot identify out of domain speakers
- Verification very tough using the log likelihood of the test
- Storing a HMM for every speaker consumes a lot of memory

References

[1] Database used from LibriSpeech ASR corpus: http://www.openslr.org/12/
[2] React native : https://facebook.github.io/react-native/
[3] React native audio Library : https://github.com/jsierles/react-native-audio
[4] Paper for Voice Bio-metric Application: https://pdfs.semanticscholar.org/6808/58420b06f98e174328f5f63dba919cbf20eb.pdf
[5] Paper implementing similar HMM based Speaker Recognition: http://www.wcsit.org/pub/2012/vol.2.no.6/Text-Independent%20Speaker%20Identification%20Using%20Hidden%20Markov%20Model.pdf