**ABSTRACT**

Audio information plays a rather important role in the increasing digital content that is available today; resulting in a need for methodologies that automatically analyze such content. Speaker Identification is one of the vital field of research based upon Voice Signals. Its other notable fields are: Speech Recognition, Speech-to-Text Conversion, and vice versa, etc. Mel Frequency Cepstral Coefficient (MFCC) is considered a key factor in performing Speaker Identification. But, there are other features lists available as an alternate to MFCC; like- Linear Predictor Coefficient (LPC), Spectrum Sub-band Centroid (SSC), Rhythm, Turbulence, Line Spectral Frequency (LPF), ChromaFactor, etc. Gaussian Mixture Model (GMM) is the most popular model for training on our data. The training task can also be executed on other significant models; viz. Hidden Markov Model (HMM). Recently, most of the model training phase for a speaker identification project is executed using Deep learning; especially, Artificial Neural Networks (ANN). In this project, we are mainly focused on implementing MFCC and GMM in pair to achieve our target.

We have considered MFCC with “tuned parameters” as the primary feature and delta- MFCC as secondary feature. And, we have implemented GMM with some tuned parameters to train our model. We have performed this project on two different kinds of Dataset; viz. “VoxForge” Dataset and a custom dataset which we have prepared by ourselves. We have obtained an outstanding result on both of these Datasets; viz. 100% accuracy on VoxForge Dataset and 95.29 % accuracy on self prepared Dataset. We demonstrate that speaker identification task can be performed using MFCC and GMM together with outstanding accuracy in Identification/ Diarization results.

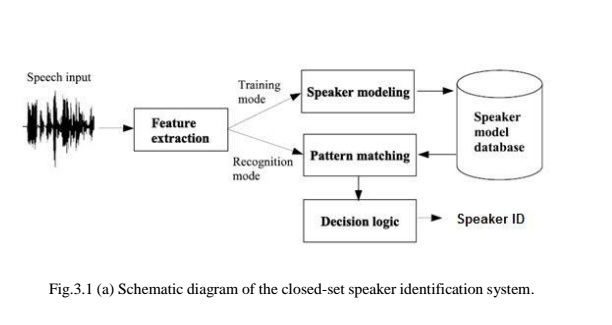
**Keywords:** *Speaker Identification, LPC, LSF, Speaker Diarization, MFCC, HMM, GMM, VoxForge, ANN, SSC, Turbulence, ChromaFactor, etc.*

Chapter: 1

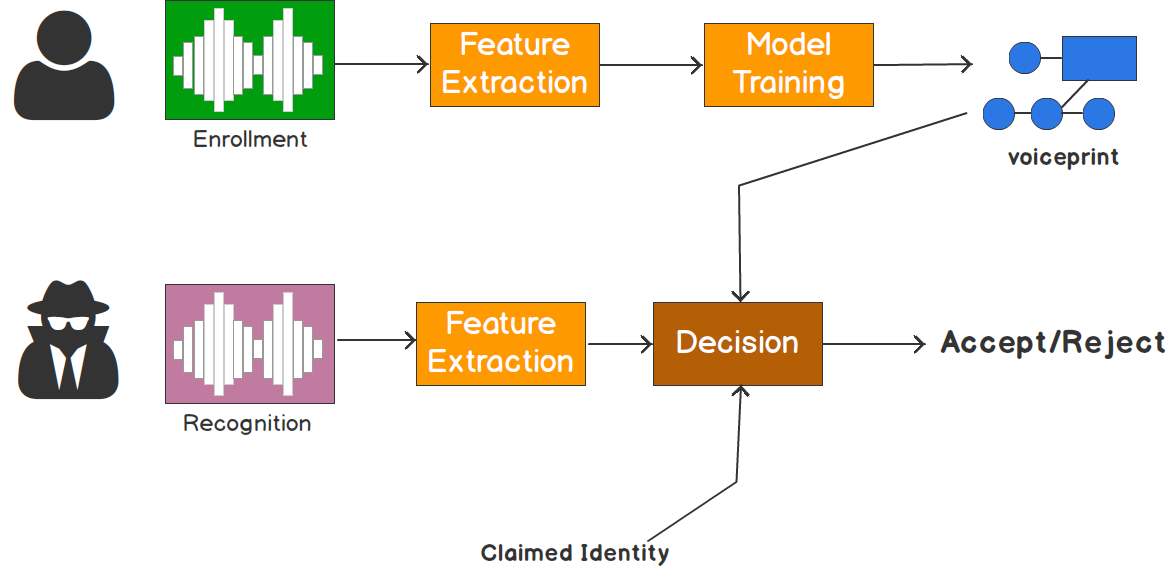
**Introduction**

*Speaker recognition* is the process of automatically recognizing who is speaking on the basis of individual information included in speech signals. It can be divided into *Speaker Identification* and *Speaker Verification*. **Speaker identification** determines which registered speaker provides a given utterance from amongst a set of known speakers. **Speaker verification** accepts or rejects the identity claim of a speaker - is the speaker the person they say they are? Speaker recognition technology makes it possible to a the speaker's voice to control access to restricted services, for example, phone access to banking, database services, shopping or voice mail, and access to secure equipment. Both technologies require users to "enroll" in the system, that is, to give examples of their speech to a system so that it can characterize (or learn) their voice patterns.

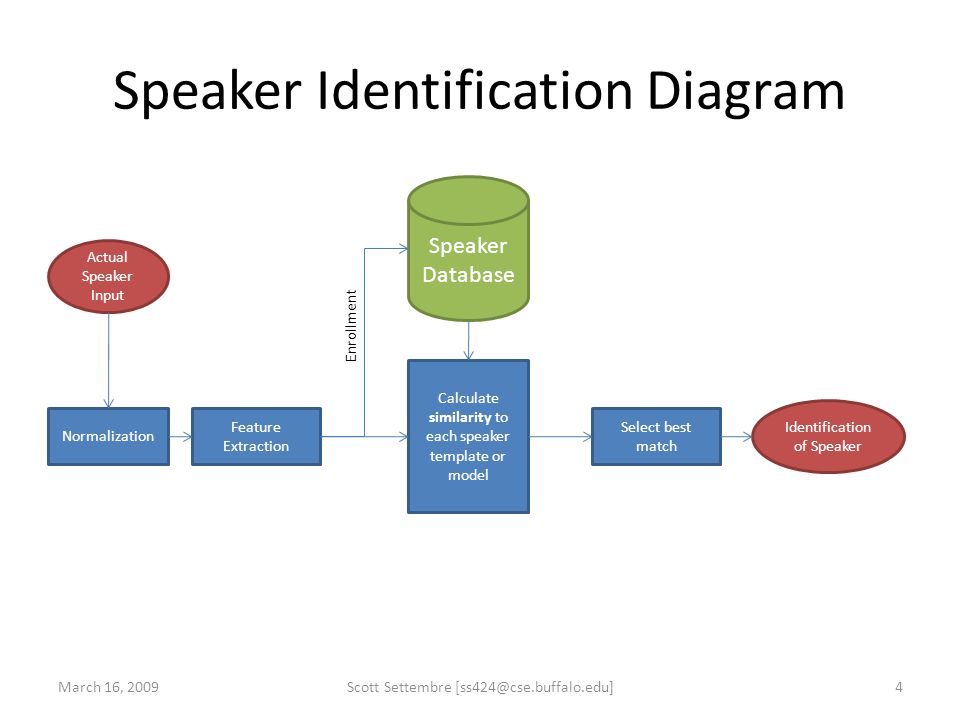
**Speaker Recognition:** The speaker recognition task can be performed in two ways: Text Dependant (Speaker Verification) or Text Independent (Speaker Identification).



**Speaker Verification:** To perform speaker verification, one must use the same data for training and testing purpose. Viz. train data and test data are same. It is basically **1: 1** recognition system; in which identity of 1 speaker is matched and verified with the claimed identity of 1 speaker in audio sample.



**Speaker Identification:** To perform speaker identification, different train and test data is used. It is basically **1: N** recognition system; in which identity of 1 speaker is matched with N speaker’s voice samples stored in the training model.



* 1. **Technology**

Speaker recognition is a “[**pattern recognition**](https://en.wikipedia.org/wiki/Pattern_recognition)” problem. The various technologies used to process and store voice prints include [*frequency estimation*](https://en.wikipedia.org/wiki/Frequency_estimation)*,*[*hidden Markov models*](https://en.wikipedia.org/wiki/Hidden_Markov_model) *(HMM),*[*Gaussian mixture models*](https://en.wikipedia.org/wiki/Gaussian_mixture_model) *(GMM),*[*pattern matching*](https://en.wikipedia.org/wiki/Pattern_matching)*algorithms,*[*neural networks*](https://en.wikipedia.org/wiki/Neural_networks) *(ANN or MLP),*[*matrix representation*](https://en.wikipedia.org/wiki/Matrix_representation)*, Vector Quantization* and [*decision trees*](https://en.wikipedia.org/wiki/Decision_tree_learning). Some systems also use "**anti-speaker**" techniques, such as [*cohort models*](https://en.wikipedia.org/wiki/Cohort_model), and *world models*. Spectral features are predominantly used in representing speaker characteristics.

[*Ambient noise levels*](https://en.wikipedia.org/wiki/Ambient_noise_level) can impede both collections of the initial and subsequent voice samples. **Noise reduction algorithms** can be employed to improve accuracy, but incorrect application can have the opposite effect. Performance degradation can result from changes in behavioral attributes of the voice and from enrollment using one telephone and verification on another telephone ("cross channel"). Integration with [two-factor authentication](https://en.wikipedia.org/wiki/Two-factor_authentication) products is expected to increase. Voice changes due to ageing may impact system performance over time. Some systems adapt the speaker models after each of the successful verification to capture such long-term changes in the voice, though there is debate regarding the overall security impact imposed by automated adaptation.

Capture of the biometric is seen as non-invasive. The technology traditionally uses existing microphones and voice transmission technology allowing recognition over long distances via ordinary telephones (wired or wireless).

* 1. **Applications**

The first international patent was filed in 1983, coming from the telecommunication research in [CSELT](https://en.wikipedia.org/wiki/CSELT) (Italy) by *Michele Cavazza* and [*Alberto Ciaramella*](https://en.wikipedia.org/wiki/Alberto_Ciaramella) as a basis for both future *Telco* services to final customers and to improve the noise-reduction techniques across the network.

In May 2013 it was announced that [*Barclays Wealth*](https://en.wikipedia.org/wiki/Barclays_Wealth) was to use passive speaker recognition to verify the identity of telephone customers within 30 seconds of normal conversation The system used had been developed by voice recognition company [*Nuance*](https://en.wikipedia.org/wiki/Nuance_Communications) (that in 2011 acquired the company [*Loquendo*](https://en.wikipedia.org/wiki/Loquendo), the spin-off from CSELT itself for speech technology), the company behind [*Apple's*](https://en.wikipedia.org/wiki/Apple_Inc.)[*Siri*](https://en.wikipedia.org/wiki/Siri_(software)) technology. 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.

The private banking division of Barclays was the first financial services firm to deploy voice biometrics as the primary means to authenticate customers to their [call centers](https://en.wikipedia.org/wiki/Call_center). 93% of customer users had rated the system at "9 out of 10" for speed, ease of use and security.

Since then, *Nuance Voice Biometrics solutions* have been deployed across several financial institutions, including *Banco Santander, Royal Bank of Canada, Tangerine Bank*, and *Manulife*.

In August 2014 *GoVivace Inc.* deployed a speaker identification system that allowed its telecom industry client to positively search for an individual among millions of speakers by using just a single example recording of their voice.

Speaker recognition may also be used in criminal investigations, such as those of the 2014 executions of, amongst others, [*James Foley*](https://en.wikipedia.org/wiki/James_Foley_(journalist)) and [*Steven Sotloff*](https://en.wikipedia.org/wiki/Steven_Sotloff).

In February 2016 UK high-street bank [HSBC](https://en.wikipedia.org/wiki/HSBC) and its internet-based retail bank [First Direct](https://en.wikipedia.org/wiki/First_Direct) announced that it would offer 15 million customers its biometric banking software to access online and phone accounts using their fingerprint or voice.

***Source:*** *Wikipedia*

* 1. **Concepts**

Imagine a scenario. A criminal is being tried in court. He denies saying something. The prosecution brings a recording, saying they have his confession on tape. As the accused vigorously denies the voice being his, an expert shows just why the voice could be no one else’s.

A fictional scene! Perhaps, but it is a reality that **no two persons** in the world have **exactly the same voices**. Do you know why this is so?

Our vocal cords can be in any of about 170 different positions. If the vocal cords are slack, they may vibrate about 80 times per second and the result is deep tones. If they are tense, they vibrate rapidly, perhaps a 1000 times a second, and produce short sound waves or high tones. The pitch of voice depends upon the length of the vocal cords. Each voice has a certain range of frequencies. It is this range that determines the kind of voice a person has. The movement of the tongue against the palate, the shaping of the lips and arrangement of teeth also bring about changes in the voice. Since the structures and movements of all these organs are different in different persons, the **voices of no two persons** in the world **can be identical**.

This is the basic idea behind this huge application of audio analysis and speaker recognition or speaker identification/ verification. The key features that differentiate two human voices can be extracted from a voice sample through feature extraction. These can be further used for model reference training; and finally, the identification of the speaker.

Chapter: 2

**Literature Review**

Biometric is physical characteristic unique to each individual. Due to the increased number of dialogue system applications, the interest in that field has grown significantly in recent years. Nevertheless, there are many open issues in the field of automatic speaker identification. Among them, the choice of the appropriate speech signal features and machine learning algorithms could be mentioned.

We have also studied, compared and tried to incorporate different approaches and algorithms to find out the most efficient model for speaker recognition. We believe MFCC-GMM model is most appropriate based on parameters like identification accuracy, computation time, false rejection, and false acceptance rate. The proposed system is a version of voice biometric which incorporates text independent speaker identification implemented independently.

While hunting for the best optimal approach to achieve our target, we tried and encountered with several blogs, publications, videos, modules, libraries etc. Here, we are stating few key points of them.

**2.1 Neural Nets for Speaker identification**

In automatic speech recognition (ASR) the projection provided by the pre-squashed outputs from a one hidden layer multi-layer perceptron (MLP) trained to recognize speech sub-units (phonemes) has previously been shown to significantly increase ASR performance. An analogous approach cannot be applied directly to speaker recognition because there is no recognized set of "speaker sub-units" to provide a finite set of MLP target classes, and for many applications it is not practical to train an MLP with one output for each target speaker. The output from the second hidden layer (compression layer) of an MLP with three hidden layers trained to identify a subset of speakers selected at random from a set of training speakers, can provide a significant relative error reduction for common Gaussian mixture model (GMM) based speaker identification [1].

The recent application of deep neural networks (DNN) to speaker identification (SID) has resulted in significant improvements over current state-of-the-art on telephone speech.  In this work, we report the same achievement in DNN-based SID performance on microphone speech.  We consider two approaches to DNN-based SID:  one that uses the DNN to extract features, and another that uses the DNN during feature modeling. Several methods of DNN feature processing are then applied to bring significantly greater robustness to microphone speech.  To direct future research, the DNN-based systems are also evaluated in the context of audio degradations including noise and reverberation [2].



**2.2 Classical Machine Learning Models**

The classical machine learning models to train an audio dataset somehow revolves around Gaussian Mixture Model (GMM) and Hidden Markov Model (HMM). A GMM (Gaussian mixture model) can be thought of as a single state HMM (Hidden Markov Model).  In other words, a state in an HMM can be thought to have a mixture of distributions, with the probability of belonging to a distribution being represented by the emission probability (aka observation probability); each state in the HMM can have a unique set of emission probabilities. Therefore, each state in an HMM can be thought of as a GMM (with emission probabilities representing the probability of association to a distribution). Other methods include computing the out-of- sample log-likelihood upon the addition of each state and the number of states maximizing the log-likelihood is chosen as the reasonable number of states to work with. Though, GMM is preferably used because it is more reliable than HMM, although HMM possess more accuracy. Secondly, GMM yields output much faster than HMM. It consumes optimal resources than compared to HMM. Most of the times either of them are used independently or with a DNN (deep neural network). However, sometimes they both can be used together in combination. Choosing any of the above prescribed approach is Arts-more-than-Science. The Researcher’s analytical abilities help him choose the better model for his project or work.

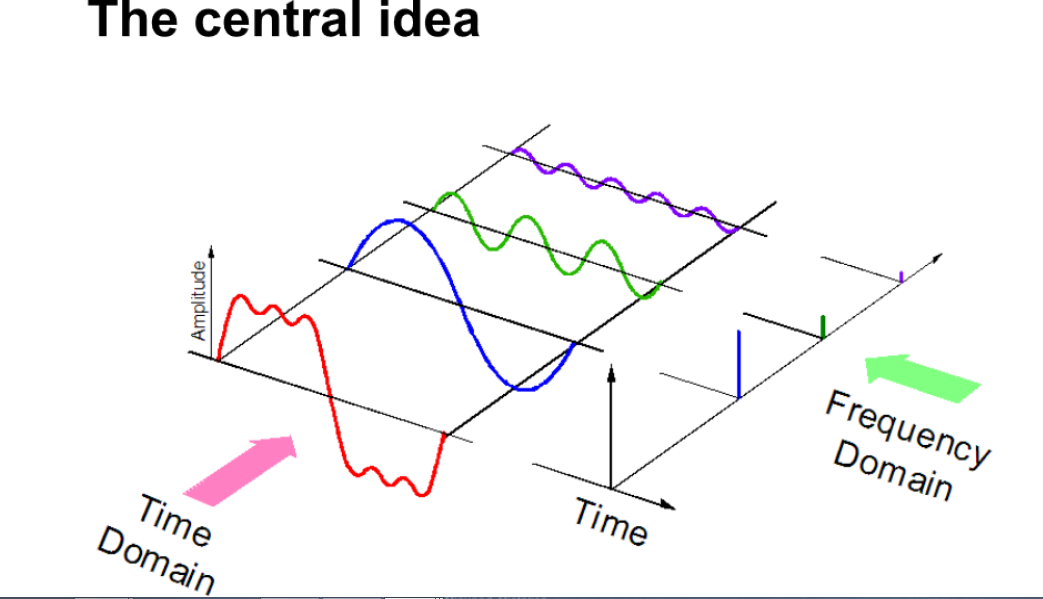
**2.3 Data Handling in Audio domain**

As with all unstructured data formats, audio data has a couple of preprocessing steps which have to be followed before it is presented for analysis. Here we will get an intuition on why this is done.

The first step is to actually load the data into a machine understandable format. For this, we simply take values after every specific time steps. For example; in a 2 second audio file, we extract values at half a second. This is called **sampling of audio data**, and the rate at which it is sampled is called the **sampling rate**.

Another way of representing audio data is by converting it into a different domain of data representation, namely the **frequency domain**. When we sample an audio data, we require much more data points to represent the whole data and also, the sampling rate should be as high as possible.

On the other hand, if we represent audio data in frequency domain, much less computational space is required. To get an intuition, take a look at the image below:

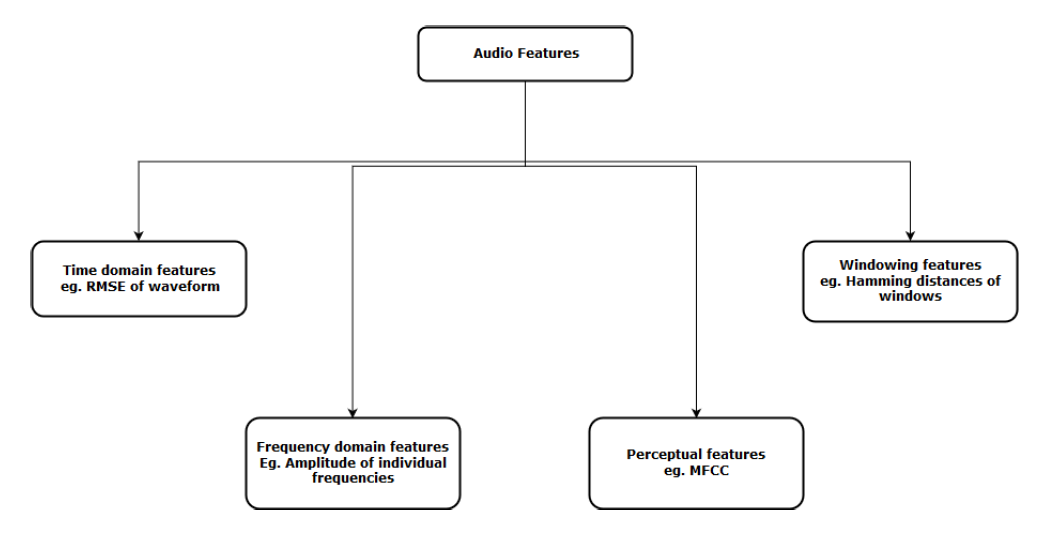


Here, we separate one audio signal into 3 different pure signals, which can now be represented as three unique values in frequency domain.

There are a few more ways in which audio data can be represented, for example using MFCs (Mel Frequency Cepstrum) [3].

**2.4 Feature Extraction Techniques**

The next step is to extract features from these audio representations, so that our algorithm can work on these features and perform the task it is designed for. Here’s a visual representation of the categories of audio features that can be extracted.



After extracting these features, it is then sent to the machine learning model for further analysis [3]. Audio feature extraction underpins a massive proportion of audio processing, music information retrieval, audio effect design and audio synthesis. Design, analysis, synthesis and evaluation often rely on audio features, but there is a large and diverse range of feature extraction tools presented to the community. An evaluation of existing audio feature extraction libraries was undertaken [4]. Almost 7 + libraries were tried to perform the feature extraction task for retrieval of MFCC and delta MFCC. A summary of official documentation about those libraries with their features and application areas is illustrated below.

**2.5 Libraries**

* **pyAudioAnalysis**

PyAudioAnalysis is an open Python library that provides a wide range of audio-related functionalities focusing on feature extraction, classification, segmentation and visualization issues.

It lays 34 small and medium features and several other large features can also be extracted using those 34 features. *Dependencies***:** Below you can find a list of library dependencies.

1. NUMPY
2. MATPLOTLIB
3. SCIPY
4. SKLEARN
5. Hmmlearn
6. Simplejson
7. eyeD3
8. pydub

Due to very large number of dependencies, it is quite tough to implement pyAudioAnalysis.

* **YAFFE**

Yaafe is an audio features extraction toolbox. The Yaafe acronym, it’s just *Yet Another Audio Feature Extractor [5]*.

*Easy to use:*

The user can easily declare the features to extract and their parameters in a text file. Features can be extracted in a batch mode, writing CSV or H5 files. The user can also extract features with Python or Matlab.

*Efficient:*

Yaafe automatically identifies common intermediate representations (spectrum, envelope, autocorrelation.) and computes them only once. Extraction is processed block per block so that arbitrarily long files can be processed, and memory occupation is low.

* **LibROSA**

LibROSA is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. It can extract both spectral as well as rhythmic features. Few examples are:

***Spectral features-***

|  |  |
| --- | --- |
| [chroma\_cens](https://librosa.github.io/librosa/generated/librosa.feature.chroma_cens.html#librosa.feature.chroma_cens)([y, sr, C, hop\_length, fmin, ...]) | Computes the chroma variant “Chroma Energy Normalized” (CENS). |
| [melspectrogram](https://librosa.github.io/librosa/generated/librosa.feature.melspectrogram.html#librosa.feature.melspectrogram)([y, sr, S, n\_fft, ...]) | Compute a mel-scaled spectrogram. |
| [mfcc](https://librosa.github.io/librosa/generated/librosa.feature.mfcc.html#librosa.feature.mfcc)([y, sr, S, n\_mfcc]) | Mel-frequency cepstral coefficients |
| [rmse](https://librosa.github.io/librosa/generated/librosa.feature.rmse.html#librosa.feature.rmse)([y, S, frame\_length, hop\_length, ...]) | Compute root-mean-square (RMS) energy for each frame, either from the audio samples y or from a spectrogram S. |
| [spectral\_centroid](https://librosa.github.io/librosa/generated/librosa.feature.spectral_centroid.html#librosa.feature.spectral_centroid)([y, sr, S, n\_fft, ...]) | Compute the spectral centroid. |

***Rhythm Features-***

|  |  |
| --- | --- |
| [tempogram](https://librosa.github.io/librosa/generated/librosa.feature.tempogram.html#librosa.feature.tempogram)([y, sr, onset\_envelope, ...]) | Compute the tempogram: local autocorrelation of the onset strength envelope. |

LibROSA is computationally the worst efficient python library. That’s why we decided to leave exploring LibROSA any further.

* **Essentia**

Essentia is an open-source C++ library for audio analysis and audio-based music information retrieval. It contains an extensive collection of algorithms including audio input/output functionality, standard digital signal processing blocks, statistical characterization of data, and a large set of spectral, temporal, tonal and high-level music descriptors. Essentia is cross-platform and it is designed with a focus on optimization in terms of robustness, computational speed and low memory usage, which makes it effective for many industrial applications. The library is also wrapped in Python and includes a number of command-line tools and third-party extensions, which facilitate its use for fast prototyping and allow setting up research experiments very rapidly.

#### Specialties:

#### Extensive collection of reusable algorithms

#### Cross-platform

#### Fast prototyping

#### Optimized for computational speed

But, We didn’t go with Essentia because its libraries was written in C++, and we faced very difficulty in making the platform ready for its implementation; since cross language libraries were causing hectic. We could not even set up the environment for the Essentia; it was demanding some Visual Studio C++ build tool 14.0; which caused some error while installation repeatedly.

* **python\_speech\_features**

This library provides common speech features for ASR including MFCCs and filterbank energies. You will need numpy and scipy to run these files.

*Supported features*:

* **python\_speech\_features.mfcc()** - Mel Frequency Cepstral Coefficients
* **python\_speech\_features.fbank()** - Filterbank Energies
* **python\_speech\_features.logfbank()** - Log Filterbank Energies
* **python\_speech\_features.ssc()** - Spectral Subband Centroids

We have used this particular library in our project. It fulfilled all of our requirements in the feature engineering without causing any trouble. We have even achieved a remarkable accuracy in identification task.

* **Scikits.Talkbox**

Talkbox is set of python modules for speech/signal processing. The goal of this toolbox is to be a sandbox for features which may end up in scipy at some point. The following features are planned before version 1.0 release:

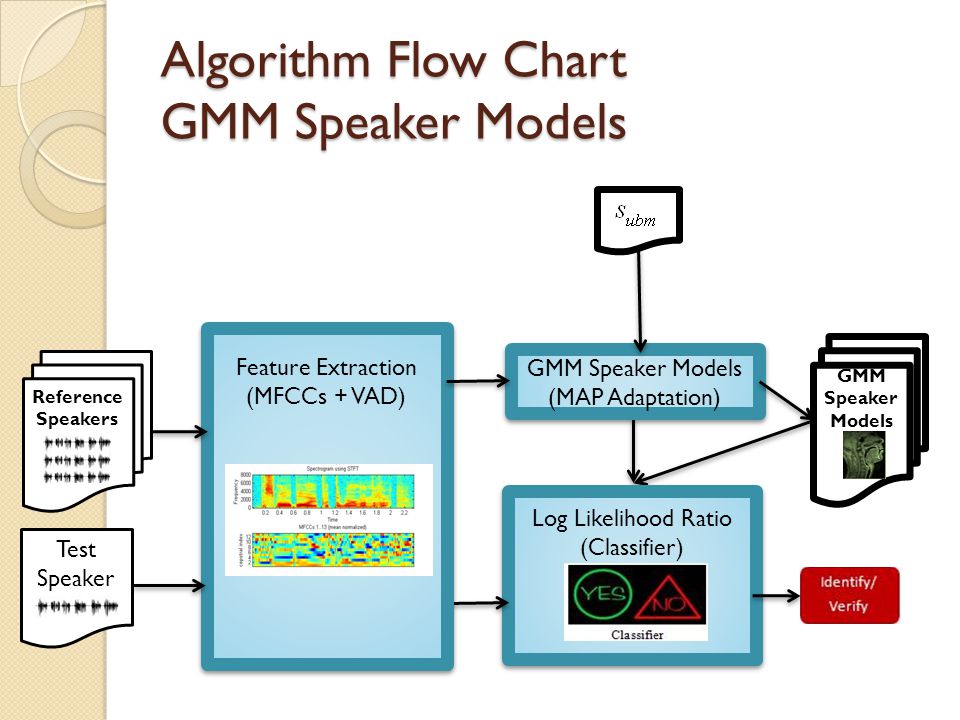
* Spectrum estimation related functions: both parametric (lpc, high  
  resolution methods like music and co), and non-parametric (Welch,  
  periodogram)
* Fourier-like transforms (DCT, DST, MDCT, etc...)
* Basic signal processing tasks such as resampling
* Speech related functionalities: mfcc, Mel spectrum, etc.

Since, this module is still under massive development. So, there are few bugs with its current build. Secondly, we didn’t get proper documentation and any supporting tutorial. That’s why we even planned to drop this library. It can be used as an alternate to LibROSA. They both can never be used together.

Chapter: 3

**Proposed Methodology**

After breaching through various approaches and different libraries, we finally stuck to using **python\_speech\_features** library. Our main methodology includes feature extraction through MFCC and then Model Training by GMM. The picture below illustrates our methodology pictorially.



Delta)

As we can see, this can be summarized in **5 basic Phases**:

* 1. Data Acquisition
  2. Data pre-processing
  3. Feature Extraction
  4. Model Training
  5. Perform Testing (identification)

Here, we are going to explain each phase in little detail. Starting from the very first phase;

* **Data Acquisition:**

Though we tested our model’s working and accuracy on a downloaded dataset; viz. VoxForge DATASET. But, to check its reliability and correctness; we also prepared a dataset by ourselves. It was voice recording of our friends, colleagues, and relatives.

So, we had two datasets in particular:

1. VoxForge Dataset
2. Manual self-made Dataset

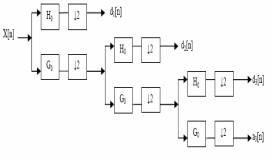
* **Data preprocessing:**

The data must be pre-processed in order to achieve better outputs and prediction results. This is to ensure that the model is trained with minimum errors.

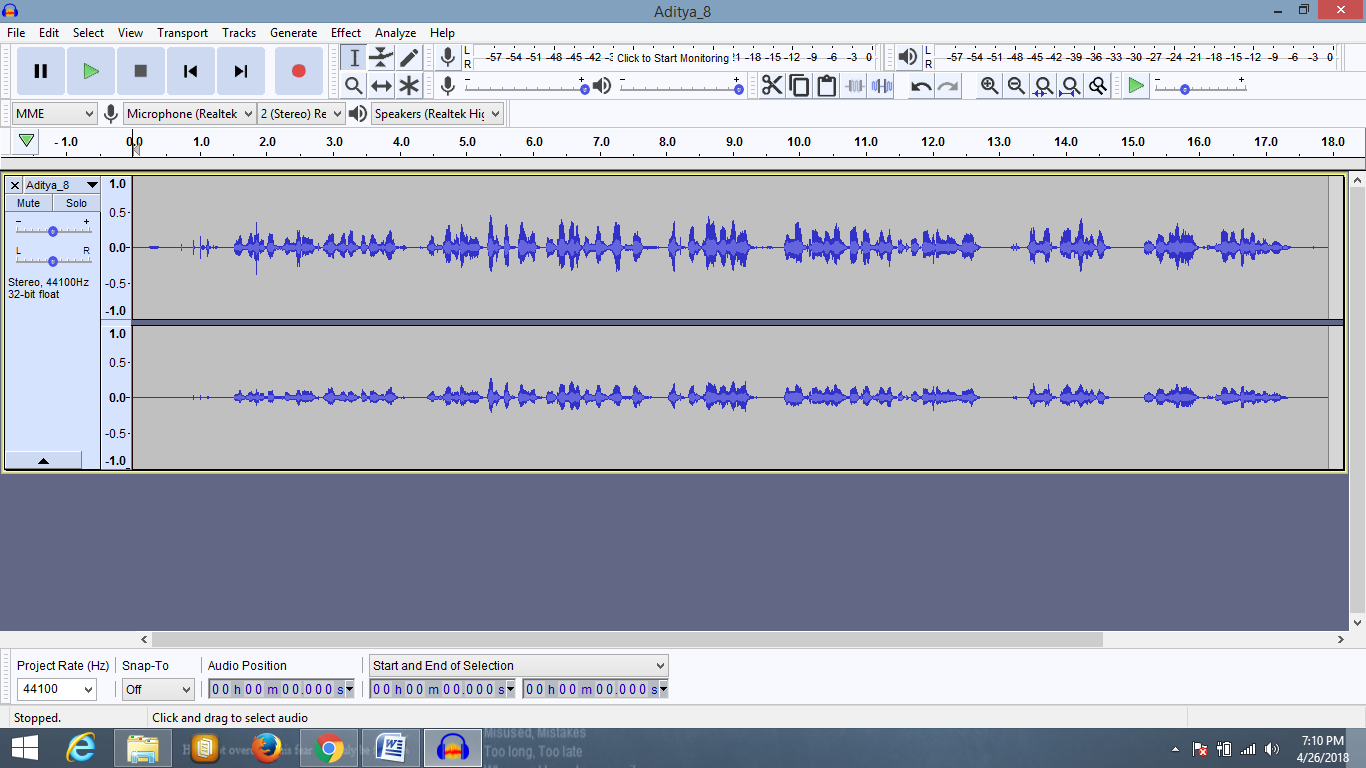
* VoxForge dataset was already clean and noise free. So, we just skipped the pre-processing part in its case.

1. ***Noise Reduction + Silence Removal:***

Noise reduction and Silence removal technique was applied on the audio samples. Noise leads to degrade the performance of speaker identification system .the de-noise process done by wavelet decomposition technique. The de-noising process consists of decomposing the original signal, thresholding the detail coefficients, and reconstructing the signal. The decomposition portion of de-noising is accomplished via the DWT. The Discrete Wavelet Transform (DWT) is commonly employed using dyadic multirate filter banks, which are sets of filters that divide a signal frequency band into sub bands. These filter banks are comprised of low pass, high-pass, or band pass filters. If the filter banks are wavelet filter banks that consist of special low-pass and high-pass wavelet filters, then the outputs of the low-pass filter are the approximation coefficients. Also, the outputs of the high-pass filter are the detail coefficients. The process of obtaining the approximation and detail coefficients is called decomposition. If a threshold operation is applied to the output of the DWT and wavelet coefficients that are below a specified value are removed, then the system will perform a “de-noising” function. There are two different threshold operations. In the first, hard thresholding, coefficients whose absolute values are lower than the threshold are set to zero. Hard thresholding is extended by the second technique, soft thresholding, by shrinking the remaining nonzero coefficients toward zero.



But, since pre-processing technique was not our main concern; we achieved this task completion by using a music/ audio amplification software; named **Audacity.** We performed noise removal and Silence removal manually one by one through a GUI interface of the software.



1. ***Conversion from .mp3 to .wav format:***

We further converted all of the mp3 files into .wav file format. Since we are using ***Scipy.iofile.wav*** module for reading only ***.wav*** files in our project***.***

* **Feature Extraction:**

Here we are focusing on two main features **MFCCs** and their Derivatives, say **Delta-MFCC**. We calculated 20 MFCCs and 20 Delta-MFCCs. So, totally we had **40 features** in hand. Delta MFCC was calculated by a custom defined function under **featureextrction.py** module.

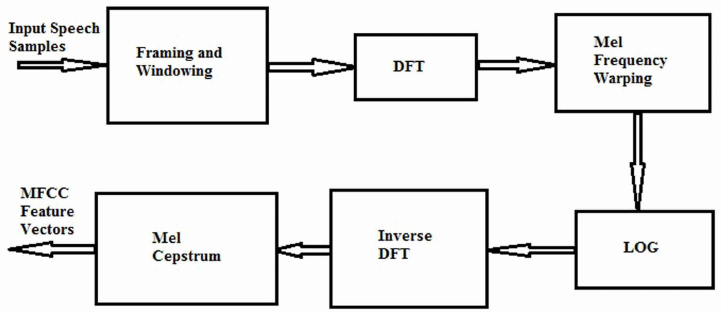
MFCC is one of the important feature extraction techniques for speaker identification technique. The goal of feature extraction is to find a set of properties of an utterance that have acoustic correlations to the speech signal which are parameters that can somehow be computed or estimated through processing of the signal waveform. Such parameters are termed as features.

## *Mel frequency Cepstral coefficient (MFCC) estimation*

After the process of removing background noises from voice signal has finish, theprocess of feature extraction will begin. Feature extraction is a process of obtaining different features of voice signal such as amplitude, pitch and the vocal tract. It is a task of finding parameter set obtained from the input voice signal. The extracted features should have some criteria in dealing with the speech signal such as: Stable over time

* Stable over time
* Should occur frequently and naturally in speech
* Should not be susceptible to mimicry
* Easy to measure extracted speech features
* Shows little fluctuation from one speaking environment to another
* Discriminate between speakers while being tolerant of intra speaker variability’s

Mel Frequency Cepstrum Coefficients (MFCC) to extract features in the voice signal. MFCC focuses on series of calculation that uses Cepstrum with a nonlinear frequency axis following Mel scale. To obtain melcepstrum, the voice signal is windowed first using analysis window and then Discrete Fourier Transform is computed. The main purpose of MFCC is to mimic the behavior of human ears. MFCC estimation includes following process. MFCC process subdivided into six phases or blocks.



*MFCC through python\_speech\_features:*

**python\_speech\_features.base.mfcc (signal, samplerate=16000, winlen=0.025, winstep=0.01, numcep=13, nfilt=26, nfft=512, lowfreq=0, highfreq=none, preemph=0.97, ceplifter=22, appendEnergy=True, winfunc=<function <lambda>>)**

|  |  |
| --- | --- |
| **Parameters:** | * **signal** – the audio signal from which to compute features. Should be an N\*1 array * **samplerate** – the samplerate of the signal we are working with. * **winlen** – the length of the analysis window in seconds. Default is 0.025s (25 milliseconds) * **winstep** – the step between successive windows in seconds. Default is 0.01s (10 milliseconds) * **numcep** – the number of cepstrum to return, default 13 * **nfilt**– the number of filters in the filterbank, default 26. * **nfft**– the FFT size. Default is 512. * **lowfreq** – lowest band edge of mel filters. In Hz, default is 0. * **highfreq** – highest band edge of mel filters. In Hz, default is samplerate/2 * **preemph**– apply preemphasis filter with preemph as coefficient. 0 is no filter. Default is 0.97. * **ceplifter** – apply a lifter to final cepstral coefficients. 0 is no lifter. Default is 22. * **appendEnergy** – if this is true, the zeroth cepstral coefficient is replaced with the log of the total frame energy. * **winfunc** – analysis window to apply to each frame. By default no window is applied. Use numpy window functions here e.g.winfunc=numpy.hamming |
| **Returns:** | A numpy array of size (NUMFRAMES by numcep) containing features. Each row holds 1 feature vector. |

* **Model Training:**

We implemented the GMM approach for model training. The speaker identification system module can be separated into four modules:

* Front-end processing
* Speaker modeling
* Speaker database
* Decision logic

## *Front- end processing*

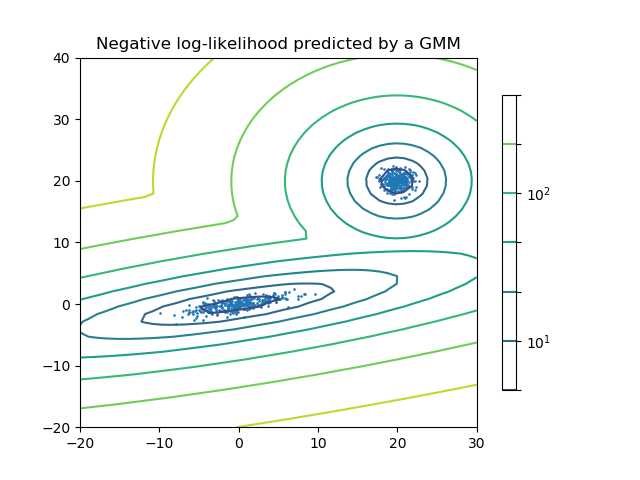
* This step is the first step to create feature vectors. It is the “signal processing” part, which converts the sampled speech signal into set of feature vectors, which characterize the properties of speech that can separate different speakers. Front-end processing is performed both in training and testing phases. The objective in the front-end processing is to modify the speech signal, so that it will be more suitable for feature extraction analysis. The front-end processing operation based on noise cancelling, framing, windowing and pre-emphasis. The goal of feature extraction is to find a set of properties of an utterance that have acoustic correlations to the speech-signal, that is parameters that can somehow be computed or estimated through processing of the signal waveform. Such parameters are termed as features. It includes the process of measuring some important characteristic of the signal such as energy or frequency response, augmenting these measurements with some perceptually meaningful derived measurements and statically conditioning these numbers to form observation vectors.

## *Modeling*

* The objective of modeling technique is to generate models for each speaker using specific feature vector extracted from each speaker. It performs a reduction of feature data by modeling the distributions of the feature vectors. The speaker reorganization is also divided into two parts that means speaker dependant and speaker independent. In the speaker independent mode of the speech reorganization the computer should ignore the speaker specific characteristics of the speech signal and extract the intended message .on the other hand in case of speaker dependent mode speech reorganization machine should extract speaker characteristics in the acoustic signal. The main aim of speaker identification is comparing a speech.

***Gaussian Mixture model (GMM)***

sklearn.mixture is a package which enables one to learn Gaussian Mixture Models (diagonal, spherical, tied and full covariance matrices supported), sample them, and estimate them from data. Facilities to help determine the appropriate number of components are also provided.

**Two-component Gaussian mixture model:** data points, and equi-probability surfaces of the model.

A Gaussian mixture model 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 generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.

Scikit-learn implements different classes to estimate Gaussian mixture models that correspond to different estimation strategies.

## *Speaker database*

* The speaker models are stored here. These models are obtained for each speaker by using feature vector extracted from each speaker. These models are used for identification of unknown speaker during the testing phase.

## *Decision logic*

* It makes the final decision about the identity of the speaker by comparing unknown speaker to all models in the data base and selecting the best matching model.
* **Perform Identification:**

The log-likelihood for each .gmm model of every speaker was calculated in the model training phase. It was stored as a database in a separate folder. This data dictionary is used for matching 1: N speaker’s gmm file. The speaker with the highest score is chosen and identified.

Chapter: 4

**System Analysis**

**4.1 Data Dictionary**

We gathered two different datasets:

* 1. ***VoxForge:***
* **34 speakers** each accompanied 5 voice samples for training data. And 5 voice samples for testing data. So, total of **340 cleaned and pre-processed** voice sample Data. Each voice sample was around **4 sec.** in length. So default value of **nfft = 512** in **mfcc()** just worked fine.
  1. ***Self-Made Dataset:***
* **17 speakers** each accompanied 15 voice samples for training data. And 5 voice samples for testing data. So, total of **340 voice sample** Dataset. Each voice sample was around **30 sec**. to **1 minute 40 sec** in length. So, we had to **tune parameters** in **mfcc()**. We adjusted **nfft** values from **512 to 2000 to 1800 to 1500**, finally.

**4.2 State Transition Diagram**

Feature

Feature Vectors

(MFCC + Delta MFCC)

.wav format audio Files.

Extraction

Model Train

Log-likelihood for each speaker.

Gmm Files for each individual Speaker.

***Scores***

Chapter: 6

**Conclusion and Future Scope**

**Conclusion:**

The performances of speaker identification system using various feature extraction and matching techniques. MFCC algorithm is used in our system because it has least false acceptance ratio. I n order to improve system performance and also to achieve high accuracy GMM model can be used in feature matching technique. The speakers were trained and tested by using MFCC and GMM model. They give better identification rate for speaker features. In future work, this technique integrated the pitch information with MFCC and also to analyze the speaker identification system performance in the presence of noise.

**Future Scope:**

Speaker Recognition field is very interesting and exploring. But, we have got a very limited research in this field as per current scenario. So, there is a lot to achieve in this field. We deal with a huge chunk of audio data daily. Plus, in this advancing world, we could never leave behind synthesizing speeches and recognizing speakers.

My project can be further extended with a **GUI interface** for ease in user experience. If we talk about its applicability, it has enormous future scope. The procedure followed and outputs from my project would be **helpful for other researchers**. They can find and suggest new methods to **improve accuracy and consistency** of the model.

This particular project can be used as a **voice biometric system** to identify and catch criminals. I have a vision of extending this project further by adding Speech Recognition into this project. **Speech Recognition + Speaker Recognition** would serve perfect for my purpose. I would like to make an intelligent system that can start **tapping phone calls** when getting any **vulnerable keywords**, mostly used by terrorists; then the speaker recognition system would **recognize that terrorist** or other criminal. This would help a lot in reducing crime and terror attacks in our country.

**Bibliography/References**

**1.** Dalei Wu, Andrew Morris, Jacques Koreman; MLP Internal Representation as Discriminative Features for Improved Speaker Recognition, 2005.

**2.** [Mitchell McLaren](https://www.sri.com/about/people/mitchell-mclaren); Advances in deep neural network approaches to speaker recognition, **ICASSP** – April, 2015.

**3.** Getting Started with Audio Data Analysis using Deep Learning (with case study) **- AnalyticsVidhya.com**

**4.** David Moffat, David Ronan, Joshua D. Reiss ;AN EVALUATION OF AUDIO FEATURE EXTRACTION TOOLBOXES; Nov 30 - Dec 3, 2015

**5.** B.Mathieu, S.Essid, T.Fillon, J.Prado, G.Richard; YAAFE, an Easy to Use and Efficient Audio Feature Extraction Software, proceedings of the 11th **ISMIR conference, Utrecht,** Netherlands, 2010.