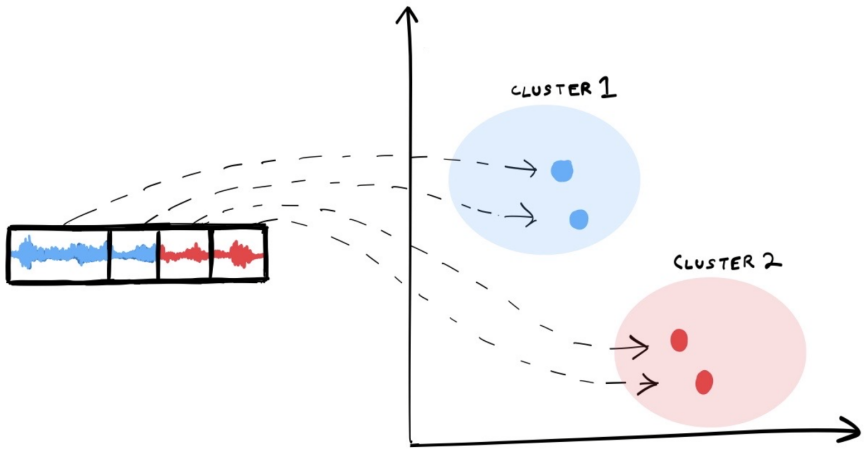
Speaker Diariazation

Anant Sharma - M1

December 2024



Contents

Abstract

This report encapsulates the implementation of the signal project for the course of Signal Processing (UE-3a) for the final project. The report presents a comprehensive study on speaker diariazation, an automatic process that partitions an audio stream according to who is speaking and when. The methodology involved is explained in section 2 i.e. segmenting the audio into smaller chunks, extracting the Mel-frequency cepstrum (MFCCs), grouping segments corresponding to the same speaker, and labeling each unique speaker. As a toy model execution, in Section 3, a 27-second conversation between a professor and a student is used, and the system automatically identifies all points in time when each person speaks without manual intervention within the model's limits.

1 Introduction

Speaker Diariazation is one of the most important and tedious tasks in audio signal processing. It essentially aims to solve the problem of "Who spoke when?" by analyzing acoustic features to separate or tag segments of speech according to the speaker. This process can become surprisingly intricate due to overlapping speech, varying recording conditions, and the need for robust feature extraction.

While the idea behind diariazation can initially seem straightforward ("just label each speaker's turn"), the reality involves significant effort in data preprocessing, feature engineering, and model configuration. Despite these challenges, automating speaker labeling has far-reaching benefits, including improved transcription accuracy, efficient meeting analytics, and enhanced user experiences in voice-driven interfaces. Speech diariazation protocols are still one of the most exciting and advanced research fields of audio signals processing.

2 Methodology

Here in this section, we will in depth go over the needed concepts and then put it all together. Things we need to understand before attempting to tackle speaker diariazation are

- Understanding Speech
- Mel-Scale
- Mel Spectrogram(s)
- Mel Frequency Cepstral Coefficient (MFCCs)

2.1 Understanding Speech

Speech Speech can be understood through the well-known source-filter theory, which says the human voice is the product of two main components: a sound source and a filter. The sound source is the glottal pulse, generated when air from the lungs causes the vocal folds to vibrate. This vibration creates a periodic waveform—often called the "excitation signal"—that provides speech's fundamental frequency (or pitch).

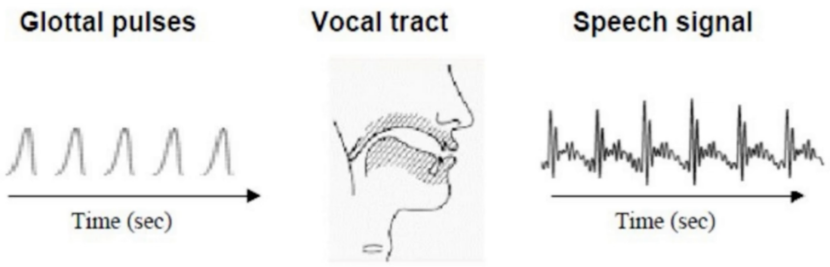


Figure 1: Origin of Speech - source-filter theory

Once the glottal pulse has been generated, it passes through the filter: the vocal tract. This tract includes the throat, oral, and nasal cavities, which can change shape to form resonant frequencies. By altering the positions of the tongue, lips, and jaw, we effectively shift how air resonates within these cavities, shaping the acoustic signal's characteristics that eventually emerge as audible speech. Understanding the glottal pulse and vocal tract filtering sheds light on how speech signals arise, why they vary from speaker to speaker, and what features might be critical for tasks like speaker diariazation. This foundational knowledge underpins the motivation behind extracting certain acoustic features (e.g., MFCCs, pitch) for downstream speech processing tasks.

2.2 Mel-Scale

Mel Scale The **mel scale** is designed to align more closely with the way humans naturally perceive sound, which follows a roughly *logarithmic* progression rather than a strictly linear one. In other words, humans are more sensitive to changes in lower frequencies and less sensitive to higher frequencies. By applying a mel-frequency transformation— using a function such as

$$\mathrm{m} = 2595 * \log\Big(1 + \frac{f}{500}\Big),$$

we map linear frequency f (in Hz) into a scale that better reflects human perception. This results in **mel spectrograms**. Consequently, mel spectrograms place greater emphasis on lower-frequency details while compressing the higher end, making them more representative of how we perceive sound compared to standard spectrograms. One can also do the transformation other way around using the below function.

$$f = 700 * \left(10^{\frac{m}{2595}} - 1\right).$$

Where m is in mels.

2.3 Mel Spectrogram(s)

Mel Spectrograms A **mel spectrogram** represents an audio signal that aligns more closely with human perception than a standard spectrogram. By mapping raw frequencies to the mel scale, we prioritize lower-frequency ranges where our hearing is more sensitive while compressing higher frequencies accordingly. So, Mel spectrogram - is just taking a Short Time Fourier Transforms (STFT) or spectrogram and expressing it in the Mel scale using the formula above.

Obtaining a Mel spectrogram:

1. Extract STFT (same as normal spectrogram procedure)
2. Convert Amplitude to DBs (same as normal spectrogram procedure)
3. Convert Frequencies to Mel Scale (New)

2.3.1 Converting Frequencies to Mel Scale

When converting the Frequencies to Mel scale one needs to do 3 things.

1. Choose Number of Mel Bands - (How Many?)
2. Construction of Mel filter Banks - (How?)
3. Apply Mel-filter banks to spectrogram - (How?)

Questions adjoint 1, 2 and 3 are the questions I had, I try to answer them below so the process makes more sense, because while implementing scripts the library `Librosa` is used and a function call will solve this.

Answering 1, it is problem-dependent there is not one answer for this, and it acts like a parameter. (assumed # in further discussions)

Answering 2 It is 5 step process,

i Convert the lowest and highest frequency to the Mel representation using the formula given 2.2.

ii Create # bands equally spaced points (between the highest and the lowest frequency)

iii Convert these points between highest and lowest back to the hertz scale using the other formula 2.2. (Note: These represent the central frequency of different Mel Bands)

iv Round to the nearest frequency bin.

v Then we create triangular filters (weights between 0-1)

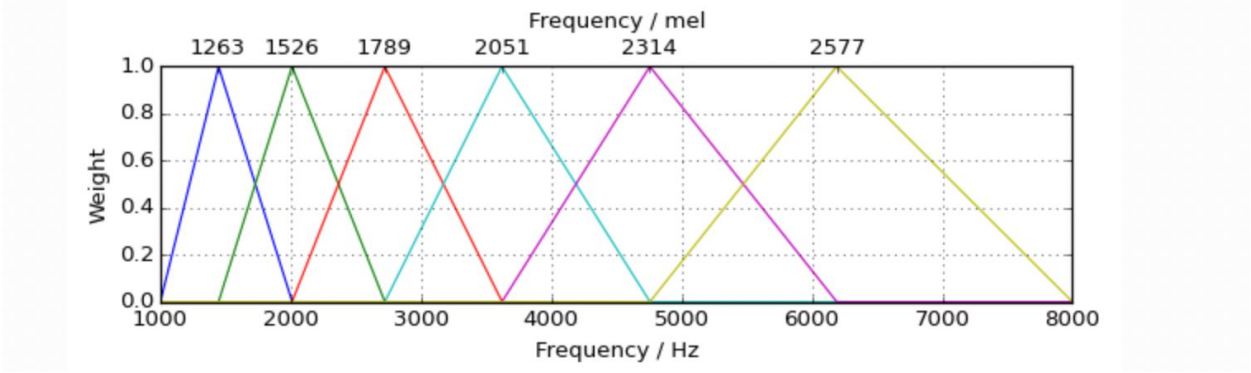


Figure 2: Mel Filter Banks - 6 Mel bands under consideration with central frequencies (in mels): 1263, 1526, 1789, 2501, 2314, and 2577.

Note: Look at the difference between two subsequent Mel band central frequencies, it is always a constant value. The same for the corresponding value in [Hz] is not true. Consider the Mel band 1526, the lower end and the higher end of this band will correspond to the center frequency of the Mel band before and after it. Also, the frequency below and higher than these values have a weight = 0. Next, we connect the lower and higher limits with the corresponding central frequency and have a triangular filter! Doing such a thing for all the central mel band frequencies we obtain a Mel Filter Bank as showcased in Figure 2.

Answering 3 is simple,

All these things mentioned afore can be very easily manipulated once represented using linear algebra. A Mel-filter Bank can be represented using a matrix. Also, the spectrogram can be represented using a matrix. Thus, applying the Mel filter bank to the spectrogram becomes just a problem of multiplying 2 matrices. Shown below they are generally matrix multiplication compatible.

Consider;

1. Size of mel bank filter matrix (M) - (No. of bands, **framesize/2 + 1**) - (rows, columns)
2. Size of the spectrogram matrix (Y) - (**framesize/2 + 1**, No. of frames) - (rows, columns)

Note: Framesize here refers to the Nyquist frequency

Finally, Mel-spectrograms become, $M * Y$ - (No. of bands, No. of frames) - (rows, columns)

When plotting spectrogram and mel spectrogram(s) they are very much alike, only the representation of frequency changes as in the case of mel spectrograms is more psychologically relevant.

2.4 Mel Frequency Cepstral Coefficient (MFCCs)

Here we deep dive into the main section of the report explaining how MFCCs work, how they can be extracted, and what they mean.

- **Mel - Frequency:** Refers to a Mel spectrogram.
- **Cepstral:** [Noun - Cepstrum], Meaning spectrum but the first four letters are the other way around. So the cepstrum is related to the spectrum.
- **Coefficients:** Refers to the fact that these coefficients describe the features of the sound.

We already know how to calculate the Mel spectrogram, now we look into how can we calculate the cepstrum of a time domain signal.

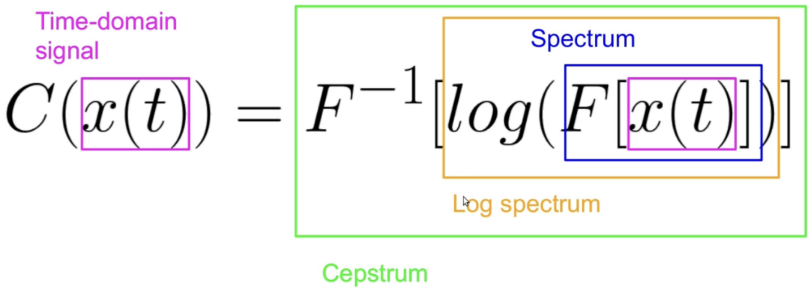


Figure 3: Cepstrum of time domain singal

From this, the cepstrum for a time domain signal $x(t)$ is,

1. Fourier Transform of the signal $F(x(t))$ - Spectrum
2. Log scale representation $log(F(x(t)))$ - Log Spectrum
3. Inverse Fourier Transform $F^{-1}log(F(x(t)))$ - Cepstrum

So essentially, **cepstrum is nothing but a spectrum of a spectrum.** Since it is not exactly a spectrum we name the properties of a cepstrum differently than a spectrum.

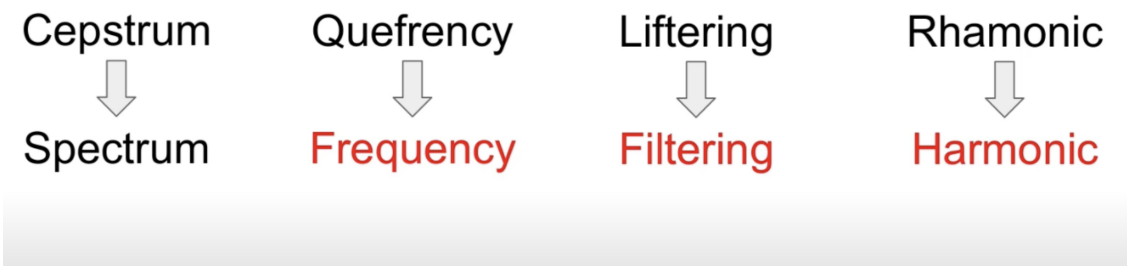


Figure 4: Basic properties

2.5 Speech Audio and Cepstrum

We already understood speech using the source and filter theory, another way to represent speech is by saying that *speech is a convolution of the vocal tract frequency response with a glottal pulse.*

$$\begin{aligned} x(t) &= e(t) \cdot h(t) && \text{Time Domain} \\ X(t) &= E(t) \cdot H(t) && \text{Frequency Domain} \\ log(X(t)) &= log(E(t) \cdot H(t)) \\ log(X(t)) &= log(E(t)) + log(H(t)) \end{aligned}$$

So speech can be thought of as having 2 components, $E(t)$ & $H(t)$ which are the Glottal pulses and Vocal tract frequency response. (Where $E(t)$ & $H(t)$ are already in the frequency domain)

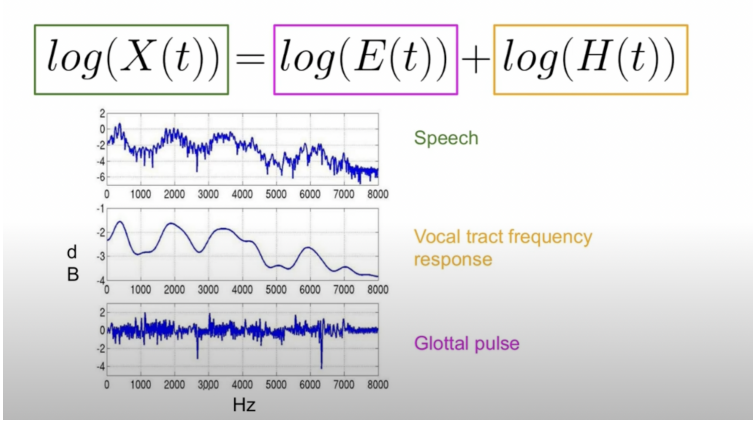


Figure 5: Linking Speech components and signals

But rarely do we have such a separation of audio available to us so, one usually tries to develop a pipeline where such 2 components of audio can be separated and this is where the concept of "Cepstrum" comes in handy!

We need to apply an Inverse Discrete Fourier Transform (IDFT) to this. The speech signal, since we are dealing with a spectrum of a spectrum or a cepstrum we get a "Quefrency" axis as shown below.

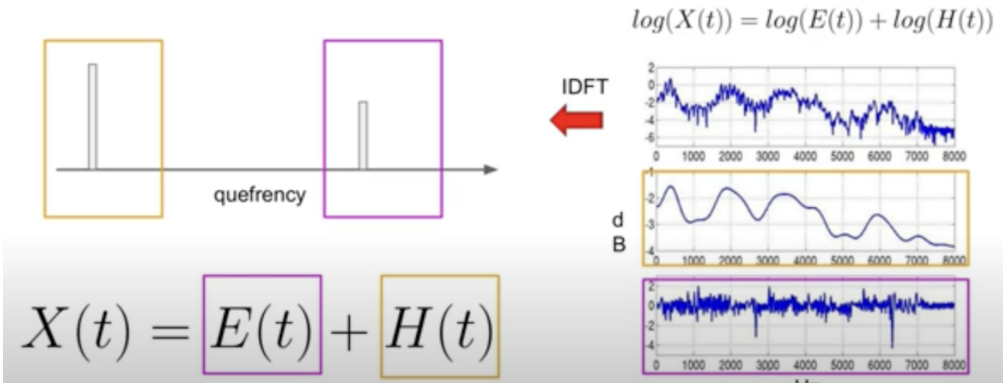


Figure 6: IDFT on Speech Signal

Finally, we can apply a low pass filter to just pass the Quefrencies corresponding to the Vocal tract frequency response which are characteristic of the sounds and are also important to audio processing as shown below,

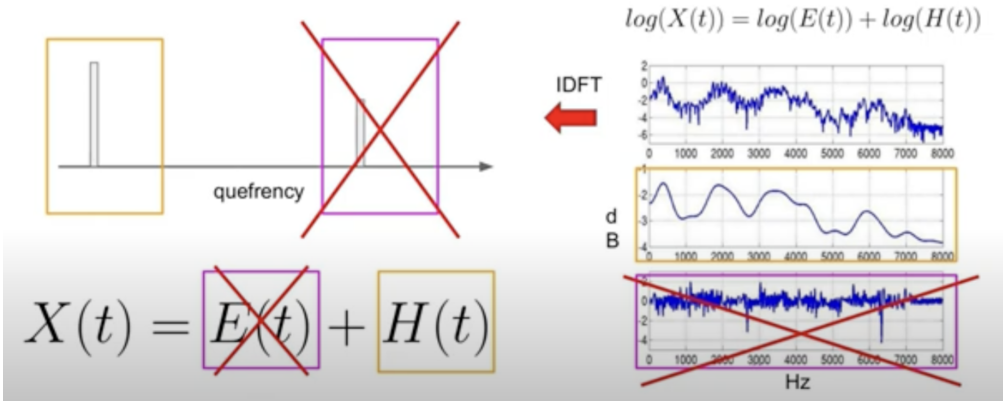


Figure 7: Applying a Low Pass Filter

Finally concluding on steps to extract MFCCs

1. Input Signal
2. Discrete Fourier Transform
3. Log Amplitude of Spectrum
4. Mel Scaling (apply filter banks)
5. Applying IDFT
6. YOU HAVE THE MFCCs!

Note: We use something similar to the Inverse Discrete Fourier Transform, called the Discrete Cosine Transform (DCT) for 2 reasons

1. DCT is a simplified version of the Fourier Transform.
2. Gives real-valued solutions (Fourier Transform gives complex values)

3 Implementation

Now we try and implement scripts trying to do some speaker diariazation. I work with `Python 3.12.8` and `librosa 0.10.2.post1` and other environment settings are presented in the `requirements.txt` of the GitHub [2]. The overall workflow is given below,

1. Use Pitch and MFCCs to characterize the speaker. (Feature Extraction)
2. Train the machine learning model using the training data and allocate labels. (Machine Learning)
3. Test it on the `text.wav` file and present results. (Testing)

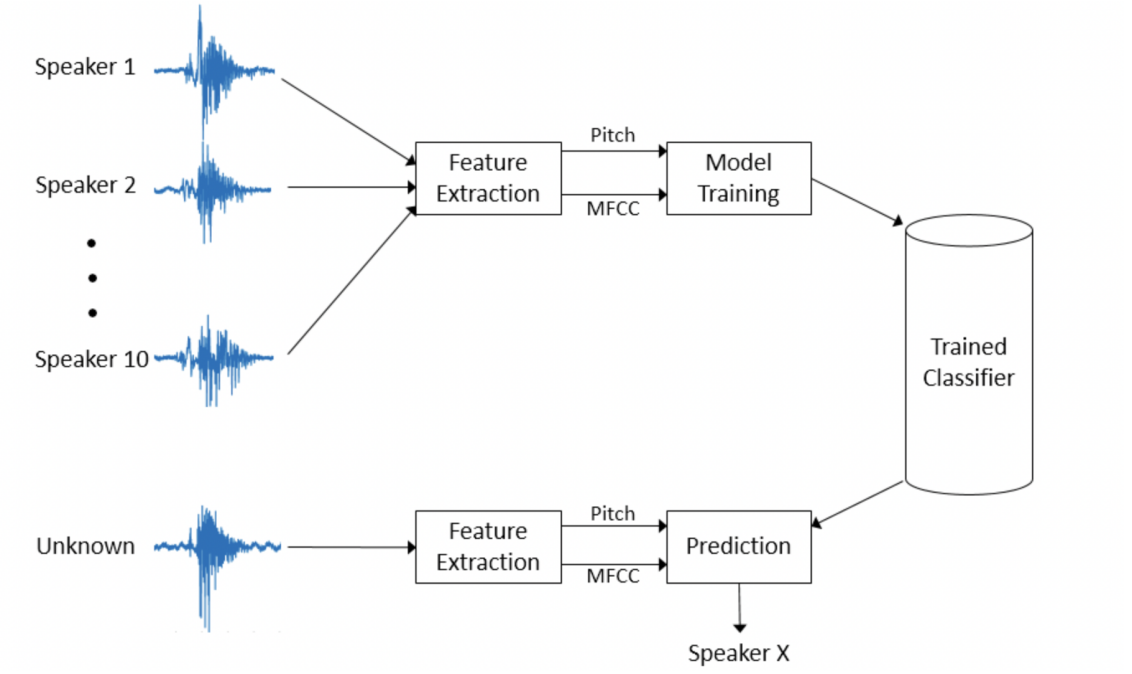


Figure 8: Workflow Implementation [3]

3.1 Data

I used open-source data available [1] for easy replication. It's a basic dataset with a test audio clip of 27 seconds between a professor and a student. The aim is to create a Python script that can differentiate the 2 speakers in the audio clip.

3.2 Script

1. Load Data for testing and training.
2. Extract Features from training data and train the model.
3. Store the ground truth labels (Actual results given for every second of the clip).
4. Test and Conclude

> Note: In the script, you will see the fact I have used a function called **chunker()** which creates 1-second chunks of audio and then uses these chunks individually to train the model. It's called **chunk based training**, If this is not done the model performs quite badly with an accuracy of 55% (This may be because the training data is not enough to process a 27 seconds clip) But using this chunk-based training one can get results with upto 96% accuracy.

The Repository In the GitHub Repository [2], one can find the file called `singals_V2.ipynb` which is the main script used in the implementation. The file is commented with all the functions being used and is straightforward.

Machine Learning Model Using already existing models, I used `MLPClassifer` viz. Multi-layer Perceptron (MLP) classifier which is a neural network that can be used for classification purposes. One can play around with the number of hidden layers and other parameters and check how the results vary.

4 Conclusion

Working on this was fun, I learned a lot throughout the project period. I successfully demonstrated how speaker diariazation works (the basic level at least) with the model performing with an accuracy of 96 %. For more complicated audio files apart from MFCCs and Pitch more features can be used for labeling the speakers. Also, the classification part of the script can involve many more advanced machine learning techniques either as pre-processing or post-processing pipelines for precise and optimized results.

Time (s)	**GROUND TRUTH**	**PREDICTION**
0.00–1.00	Professor	Professor
1.00–2.00	Professor	Professor
2.00–3.00	Professor	Professor
3.00–4.00	Professor	Professor
4.00–5.00	Professor	Student
5.00–6.00	Student	Student
6.00–7.00	Student	Student
7.00–8.00	Student	Student
8.00–9.00	Student	Student
9.00–10.00	Student	Student
10.00–11.00	Student	Student
11.00–12.00	Student	Student
12.00–13.00	Student	Student
13.00–14.00	Student	Student
14.00–15.00	Student	Student
15.00–16.00	Professor	Student
16.00–17.00	Professor	Professor
17.00–18.00	Professor	Professor
18.00–19.00	Professor	Student
19.00–20.00	Student	Student
20.00–21.00	Student	Student
21.00–22.00	Student	Student
22.00–23.00	Student	Professor
23.00–24.00	Student	Student
24.00–25.00	Student	Student
25.00–26.00	Student	Student
26.00–27.00	Student	Student

Table 1: Chunk-level predictions for `test.wav`

Thank you Dr. Coillet for letting me work on this, it was super cool!

5 Acknowledgements

I would like to thank Mr.Nikhil Raghav who is a Ph.D student at the Institute of Advanced Intelligence (IAI), TCG - CREST, Kolkata, India. He helped me direct the project in this direction and guided me in learning the conceptual sections.

References

[1] Dataset of Professor and Student Kaggle Link

[2] GitHub Repository GitHub Repository

[3] Matlab Tutorial Matlab Tutorial

[4] Youtube Tutorial Youtube Tutorial

[5] Article - 1 Link