AUDIO LAB REPORT

# Audio Segmentation by Feature-Space Clustering using LDA and Dynamic Programming

BY
MOHAMMAD MEHDI BALOUCHI

AT
UNIVERSITY OF BONN

SEP 2022

SUPERVISOR: SEBASTIAN URRIGSHARDT

# Abstract

In this lab project, we have implemented an audio segmentation method by feature-space clustering using the LDA method (for dimensionality reduction). Moreover, we have used a dynamic programming algorithm [1] to improve the results.

This implementation is based on study by Godwin et.al. [1,2] and we used some new features such as Chroma-stft [3] and Mel-frequency cepstral coefficients (MFCCs).

We have implemented our code using python programming language and librosa library (audio and music processing in Python) [4].

We have shown the evaluation result of speech, non-speech audio segmentation using the MIREX2015 example training dataset, however, our implementation has the ability to be used for multi label segmentation as well.

# Acknowledgements

I’m grateful to the audio lab group, especially Prof. Dr. Frank Kurth who gave me this opportunity to work and learn on my favorite topic which is the combination of computer science and audio/music.

I dedicate this project to all brave women of my homeland Iran, specially to a 22 year old girl #MahsaAmini who was killed by Islamic regime because of her supposedly inappropriate hijab.

# Contents

# Chapter 1

# Introduction

The goal of this project is to implement a method for audio segmentation using Linear Discrimination Analysis (LDA) and a Dynamic Programming (DP) method to improve the segmentation [1, 2]. We consider the problem of segmenting an audio signal into characteristic regions based on feature-set similarities. We also show the problems of pick peaking and how the dynamic programming method could improve it.

## 1.1 Context

The mentioned approach in the introduction section is a classical approach of audio segmentation without using any neural network. However, we added more robust audio features such as Mel-frequency cepstrum (MFCC) and spectral chroma. Audio segmentation (often called audio classification) is a preprocessing step in audio analysis that separates different types of sound, for example, speech, music, environmental sounds, silence, and combinations of these sounds [21, 74].

We can have several target classes types as follows:

- Speech, non-speech segmentation called automatic speech recognizer (ASR)
- Music, non-music segmentation
- Speech/Music discrimination (SMD)

Several classes segmentation such as speech, music, silence, noise, environment, etc.

## 1.2 Motivations and Applications

Since this is a classical topic in the area of audio processing, you can find many applications which are not practical anymore. For example, one of the most famous early paper's motivations [5] was the automatic monitoring of radio channels' audio content for changing the channel when it's playing talks or commercials on broadcast radio.

Another applications of audio segmentation we can mention:

1. **Low bit-rate coding:** separate codec designs are used to digitally encode speech and music signals; speech coders work better for speeches and audio coders do better on music [7].
2. **Content-based audio retrieval:** This is an emerging multimedia application which will remove the subjectivity inherent in the classification process and finally it will speed up the retrieval process [6].
3. **Broadcast Transcription:** It is important to disable the speech recognizer during the non-speech portion of the audio streams, you can see an example of it in youtube CC functionality.

## 1.3 Tools

For implementing this project we have used Python programming language (v.3.10.4) and also Jupyter Notebook for better result visualization. Moreover, here you can find the most important libraries we used in this project:

- **Librosa:** A python library for audio processing which we used most of the audio features extraction from this library, also, we used framing and windowing and some of the util methods of it [4].
- **Numpy:** It is a python library that contains many fast and robust implementations for numeric and linear computations.
- **Scikit-learn:** A machine learning python library which has implemented the Linear Discrimination Analysis (LDA) and also evaluation report.
- **Pandas:** A Data manipulation and analysis python library which we use for reading the dataset annotations for labeling from the CSV files.
- **Matplotlib:** It is one of the most famous and powerful libraries for plotting in python.
- **Pickle:** It is a Python object serialization built-in library which we use for saving loaded audios and all the labeling and calculated basic features on the hard disk in order to save some memory for the calculations.

## 1.4 Pipeline

The proposed approach of this project which is based on the main papers [1,2], First, we generate the basic features of the audio signals for short periods of time (20 ms) then for

each basic feature we calculate some statistics for a longer period (1 sec). We have used moving mean and moving standard deviation. Since the number of basic features are 78, we have to reduce the dimensionality of the feature space, for this purpose we used Linear Discriminant Analysis (LDA), then we used a non heuristic dynamic programming method in order to classify the LDA output for the audio segmentation.

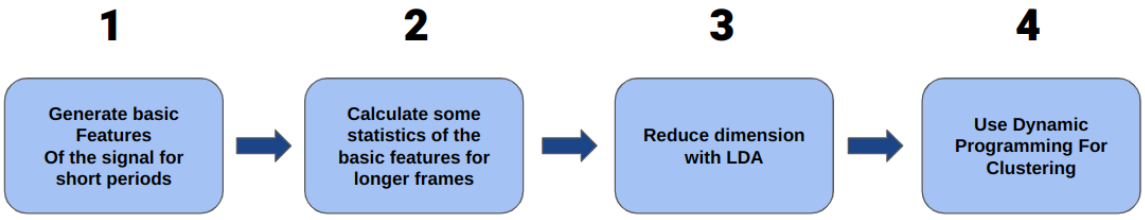


Fig1. Method Pipeline

## 1.5 Basic Configuration

Since in audio segmentation we have to calculate the features based on framing the audio signals, it is important to have a good setup for the size of the windows and also a proprate hop length. So we chose 20 msec window size for the basic features and 1 second window size for the actual features based on previous literatures [8]. Here you can find other basic configurations as well:

**Sample Rate:** 44100 (which is the sample rate of the recordings)

**Window Size:** 20 msec

**Hop Length:** 10 msec (window size/2)

**Mean and STD Window Size:** 1 sec

**Mean and STD Hop Length:** 0.5 sec

# Chapter 2

# Dataset

We used the MIREX2015 example training data set for Music/SPEECH segmentation which contains annotations just for two classes (music and speech). The dataset contains 7 audio files in MP3 format which is from the internet archive.

## 2.1 Annotations

The annotation files contain manual annotations for each of the audio files in CSV format. The format of the files is as follows:

For each row, a file contains the onset (seconds), duration (seconds), and the class (music or speech, represented by lower-case 'm' or 's') of each note event separated by a comma, ordered in terms of onset times. These annotations have been obtained by manual annotation and cross-checking all done by the MuSpeak team at City University London. The annotation guidelines allowed a 500 ms tolerance, and annotations were checked by at least one other person besides the annotator. Below you can find an example of one of the annotation files:

| Start (sec) | Duration (sec) | Label |
|---|---|---|
| 2.229115646 | 3.575873015 | s |
| 6.037188208 | 141 | m |
| 66.22040816 | 82.04693878 | s |
| 148.9 | 240.75 | m |
| 390.5081633 | 48.11020408 | s |
| 439.1183673 | 272.1959184 | m |
| 712.0979592 | 133.2306122 | s |
| 843.2326531 | 265.1346939 | m |
| 1102.506122 | 57.06938776 | s |

Table 1. Dataset Annotations Format

## 2.2 Training And Testing Set

We used around 70% of the dataset as our training data and the rest for the testing data.

here you can see all the audio files:

| Audio Names | Type | Duration |
|---|---|---|
| ConscinciasParalelasN11-OEspelhoEOReflexoFantasiasEPerplexidadesParte413-12-1994.mp3 | Training | 30 min 17 sec |
| eatmycountry1609.mp3 | Training | 58 min 20 sec |
| theconcert16.mp3 | Training | 59 min 6 sec |
| ConscinciasParalelasN7-OsSentidosOSentirEAsNormasParte715-1-1994.mp3 | Training | 28 min 11 sec |
| theconcert2.mp3 | Training | 43 min 50 sec |
| UTMA-26.mp3 | Testing | 64 min 47 sec |
| ConscinciasParalelasN3-OsSentidosOSentirEAsNormasParte318-10-1994.mp3 | Testing | 29 min 46 sec |

## 2.3 Labeling

As we used a supervised method for audio segmentation, we have to have accurate and well-structured labeling. The process of labeling in this project has three main steps:

1. **Parsing dataset annotations:** This is the first step of the labeling process and the task is to read the csv annotation files in the python code.
2. **Sample labeling:** Each of the loaded audio signal are in the numpy array format. and the length of it is related to the chosen sample rate setting. So we assigned a label to each index of audio samples based on the parsed annotations. As there are just music and speech annotations and they have also some overlaps for the sections that is the mixture of the talk and music, we decided to choose these labels for the samples:

   **0** = None annotated periods which are silence parts of the audios

   **1** = Pure music periods

   **2** = Pure speech periods

   **3** = Periods which are the overlap of the music and speech
3. **Frame Labeling:** The final evaluation will be on the frames of the final features and LDA output (which has the same framing format of the final features) so we have to decide for the actual labels of each frame of the audio signal. For example when we are interested in having an automatic speech recognition (ASR), we have to have just speech and non-speech labels and we choose the 1 label for the frames that their majority sample labels are pure speech or overlap of music and

speech. However, just by changing the frame labeling method in our codes you would be able to change the target classes and use it for different applications.

## 2.4 Music And Speech Examples From Dataset

Here you can find one speech and one music example from the dataset with a duration of 10 seconds. All the plots we have in the next sections are related to these two examples.

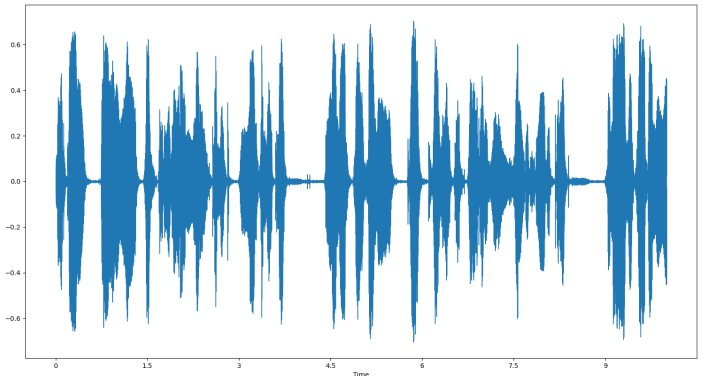

Fig 2. 10 sec speech (UTMA-26.mp3 - 11:53 - 12:03)

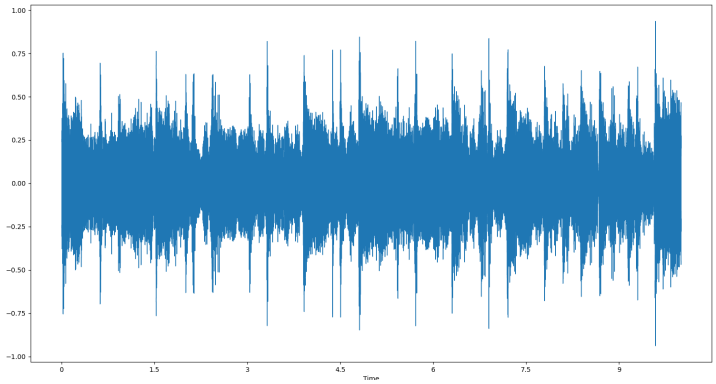

Fig3. 10 sec music (UTMA-26.mp3 - 17:00 - 17:10)

# Chapter 3

# Features

There are many good features in the previous literature [8] such as spectral centroid, spectral rolloff, spectral flux, zero crossing rate and RMS. We also added some new features such as spectral flatness, spectral bandwidth, spectral chroma and spectral MFCC. We will show their performance separately in our dataset as well.

## 3.1 Basic Features

There are two types of audio features in this project:

1. **Temporal:** The features related to time domain
2. **Spectral:** The features related to the fourier transform

Later we will show that in the dimensionality reduction step that the LDA method will eliminate non-effective features, hence, we are not picky about the features to be discriminative on our target class. Here are the list of the basic features:

### 3.1.1 Zero Crossing Rate (ZCR)

The number of times that the signal crosses the zero value in the buffer.

### 3.1.2 Spectral Centroid

An indicator of the “brightness” of a given sound, representing the spectral center of gravity. If you were to take the spectrum, make a wooden block out of it and try to balance it on your finger (across the X axis), the spectral centroid would be the frequency that your finger “touches” when it successfully balances.

### 3.1.3 Spectral Flux

A measure of how quickly the spectrum of a signal is changing. It is calculated by computing the difference between the current spectrum and that of the previous frame.

### 3.1.4 Spectral Rolloff

The frequency below which contains n% of the energy of the spectrum. We used n = 95.

### 3.1.5 Root Mean Square (RMS)

The root mean square of the waveform, which corresponds to its loudness.

### 3.1.6 Spectral Flatness

The flatness of the spectrum. It is computed using the ratio between the geometric and arithmetic means. It is related to how noisy a sound is. For example a pure sine wave will have a flatness that approaches 0.0, and white noise will have a flatness that approaches 1.0

### 3.1.7 Spectral Bandwidth

Spectral bandwidth is calculated as follows [9] :

$$(\sum_{k} S[k,t] * (freq[k,t] - centroid[t])^{p})^{\frac{1}{p}}$$

Where k is the frequency bin index, t is the time index, S [ k , t ] is the STFT magnitude at frequency bin k and time t , f r e q [ k , t ] is the frequency at frequency bin k and time t , centroid is the spectral centroid at time t , and finally p is the power to raise deviation from spectral centroid.

### 3.1.8 Spectral Chroma

Calculates the how much of each chromatic 12 pitch classes (C, C♯, D, D♯, E, F, F♯, G, G♯, A, A♯, B) exists in the signal. We used all the 12 features as our basic features.

### 3.1.9 Spectral MFCC

As humans don't interpret pitch in a linear manner, various scales of frequencies were devised to represent the way humans hear the distances between pitches. The Mel scale is one of them, and it is now widely used for voice-related applications. Here we used 20 Mel-Frequency Cepstral Coefficients.

## 3.2 Feature Comparison Plots

Here we demonstrate some comparison plots for different features:

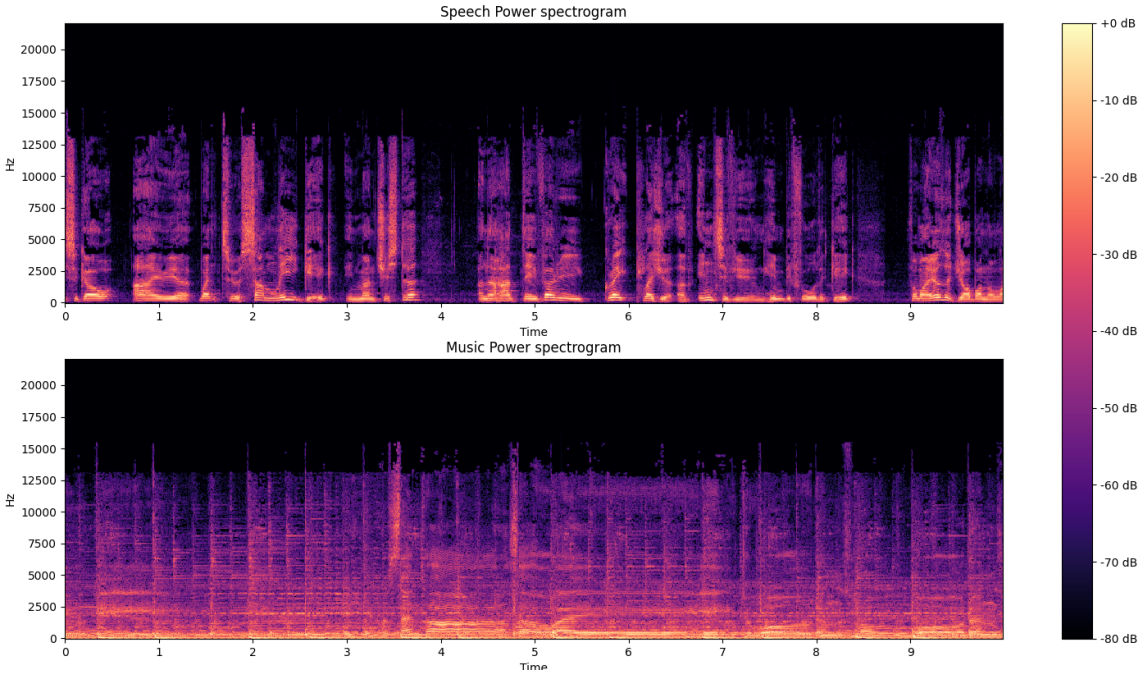


**Fig 4. Music and speech power spectrogram comparison**

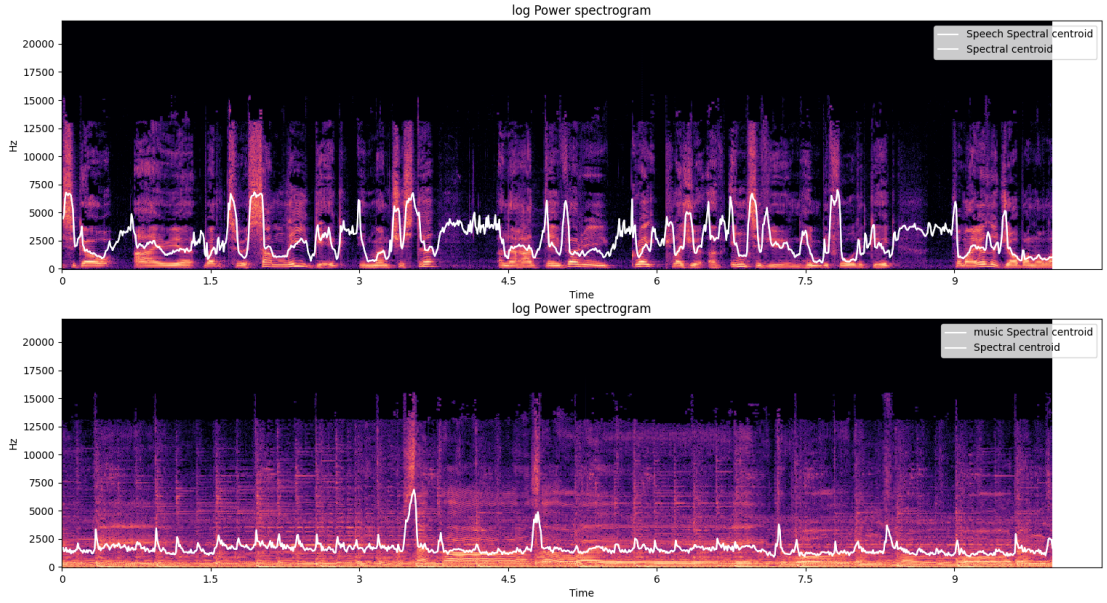


Fig 5. Music and speech spectral centroid comparison

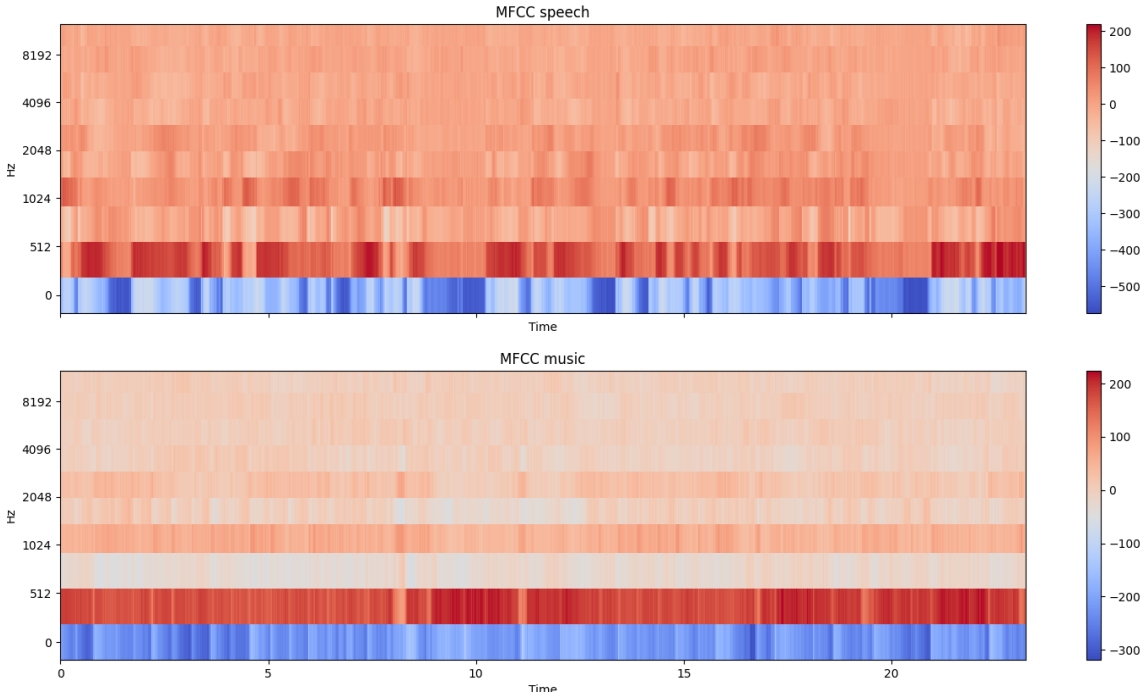


Fig 6. Music and speech MFCC comparison

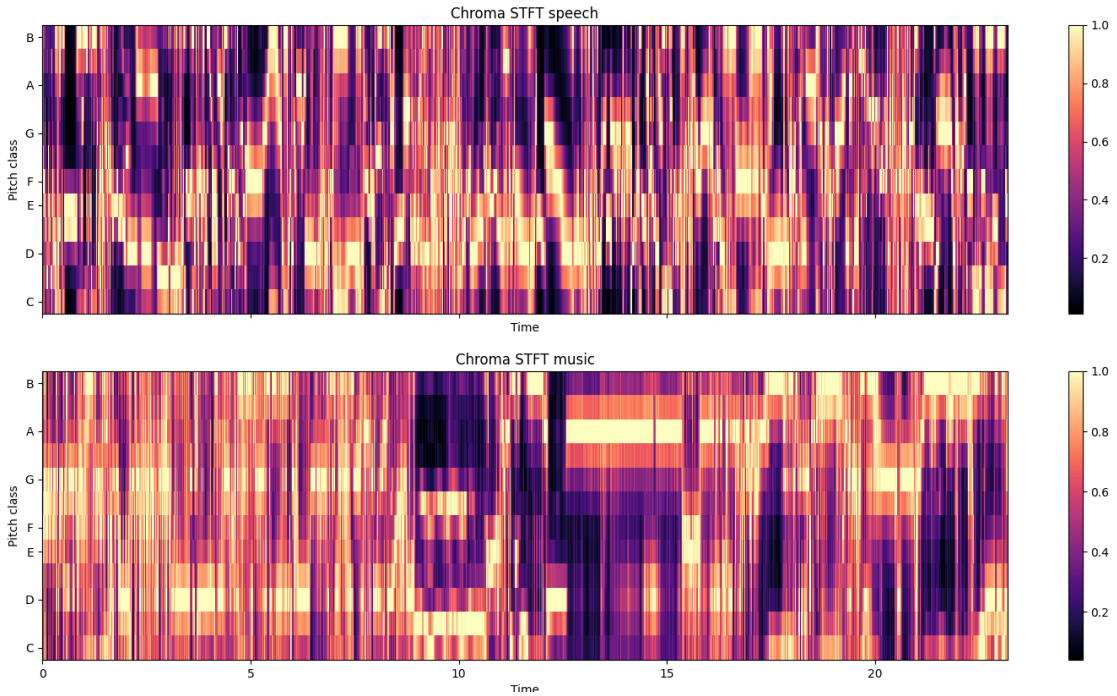


Fig 7. Music and speech spectral chroma comparison

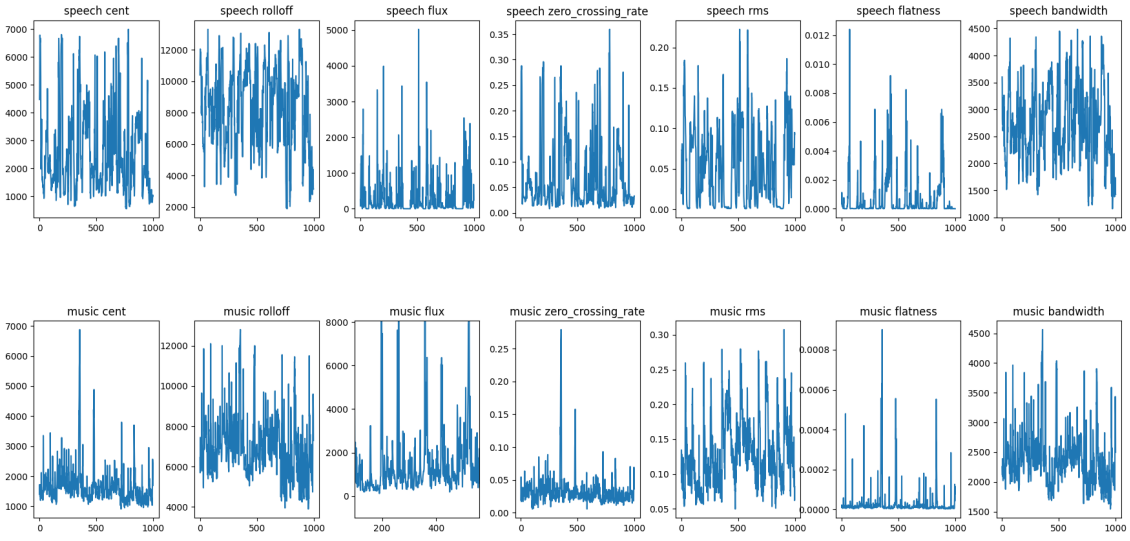


Fig 8. Music and speech other basic features comparison

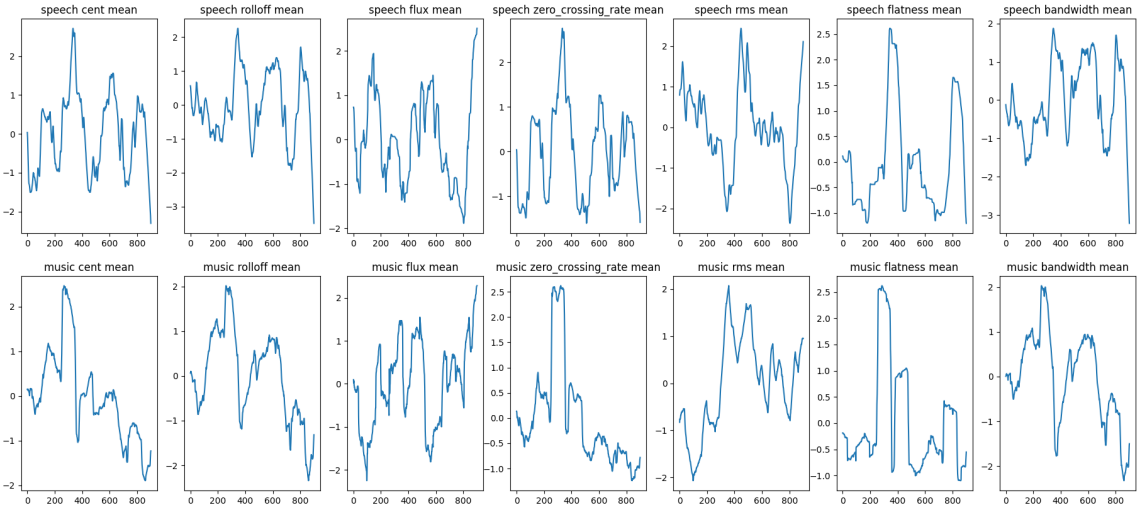


Fig 9. Music and speech features moving mean comparison

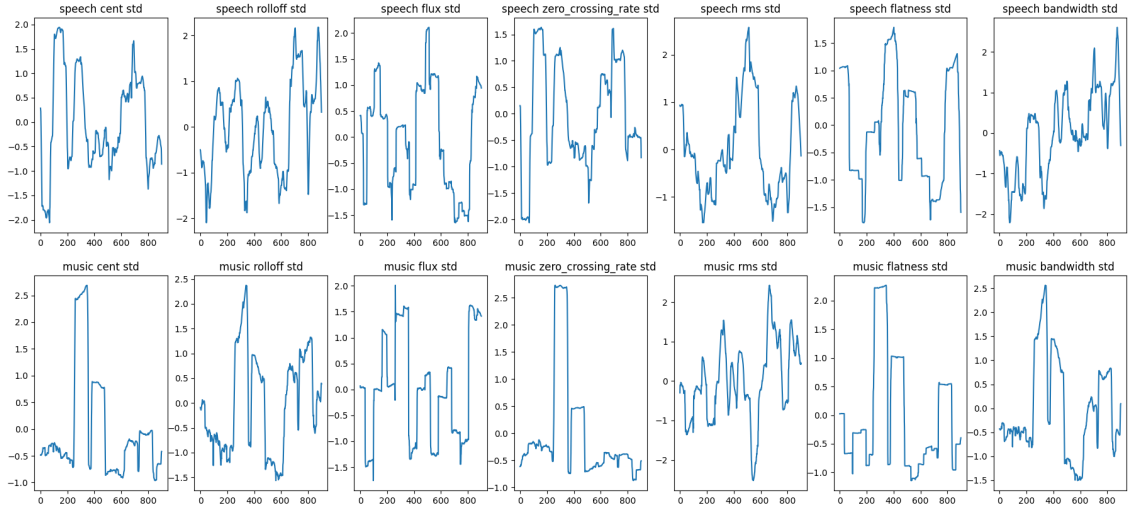


Fig 10. Music and speech features moving STD comparison

## 3.3 Single Feature Results

We calculated the LDA prediction method for each of our basic features separately in order to have a glimpse of their performance.

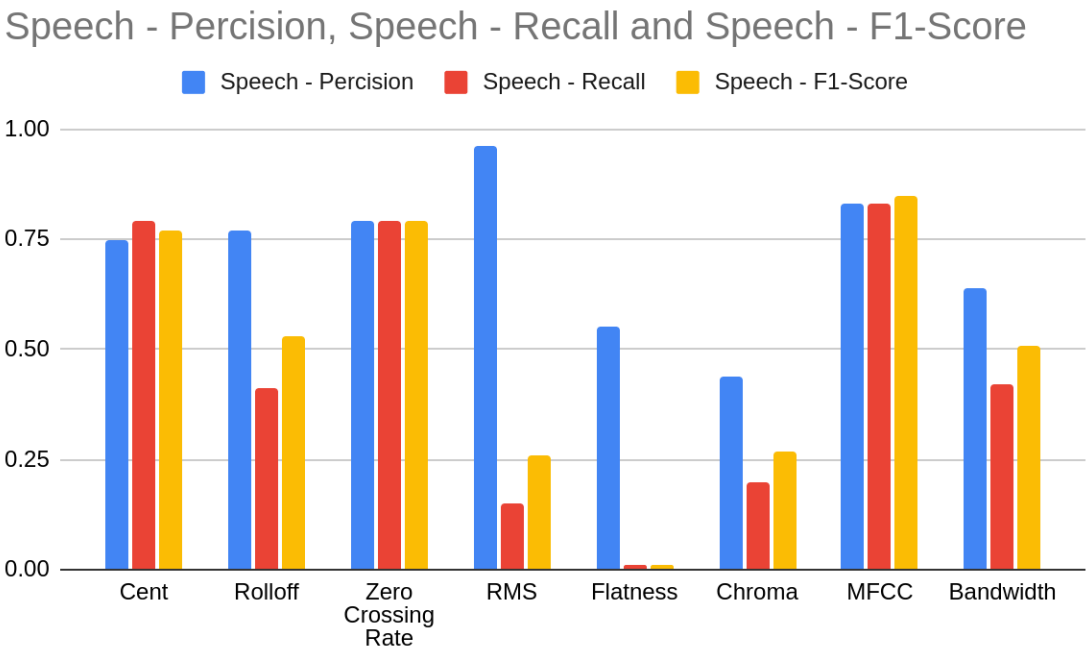


Fig 11. Speech features LDA predict performance comparison

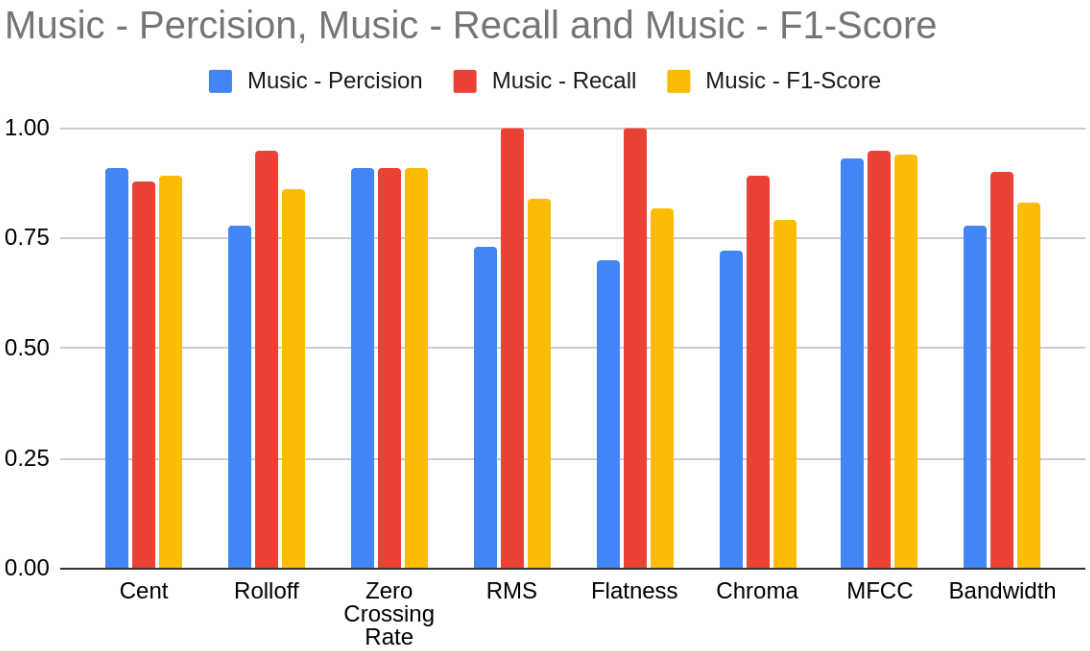


Fig 12. Music features LDA predict performance comparison

## 3.4 Final Features

Our basic features consist of ZCR + RMS + Centroid + Flatness + Bandwidth + Flux + Rolloff + Chroma (it has 12 separate features ) + MFCC ( we used separate coefficient as separate features)

Basic Features Dimension = 39

For each basic feature we calculate the moving mean and std in larger windows and we use these statistics as our final features.

Final Features = 39 (basic features) X 2 (statistics) = 78

# Chapter 4

# Segmentation

Audio segmentation can be defined as a classification task in which the adjacent segments have different class labels. In order to detect the boundaries between the segments, we can calculate the audio features' distance between adjacent frames. In this chapter we will explain the novelty function concept and dimension reduction methods.

## 4.1 Novelty Function

The novelty function is defined as the sequence of differences between successive feature vectors [1]. This function is related to the extent of change in the audio signal between adjacent frames. Peaks in the novelty function indicate the boundaries in the audio signal. However, we have to use a heuristic threshold in order to find the correct peaks and ignore the spurious peaks.

## 4.2 Dimensionality Reduction

The audio segmentation task with predefined classes is like a classification task in which the segments are separate adjacent classes.

There are two popular methods of dimensionality reduction which could be useful in our segmentation task:

**Unsupervised:** Principal Component Analysis (PCA)

**Supervised:** Linear Discriminant Analysis (LDA)

Based on the definitions, PCA would not be a good choice for classification, since it is an unsupervised method, but on the other hand LDA could be a good choice for our task.

### 4.2.1 PCA

The basic idea behind the PCA is to find new component axes that maximize the variance of our data.

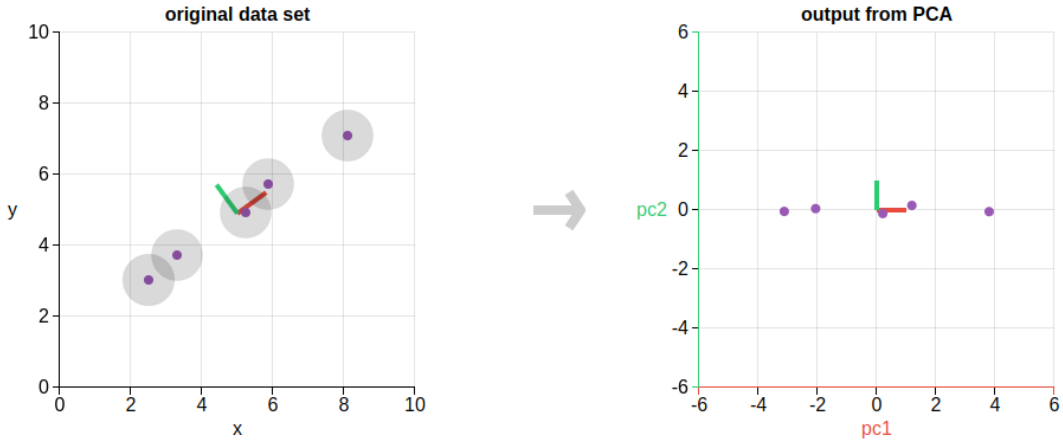


Fig 13. PCA coordination adjust visualization [10]

### 4.2.2 LDA

LDA is so similar to the PCA method except that it tries to maximize the separation between multiple classes while minimizing the inter-class variance.

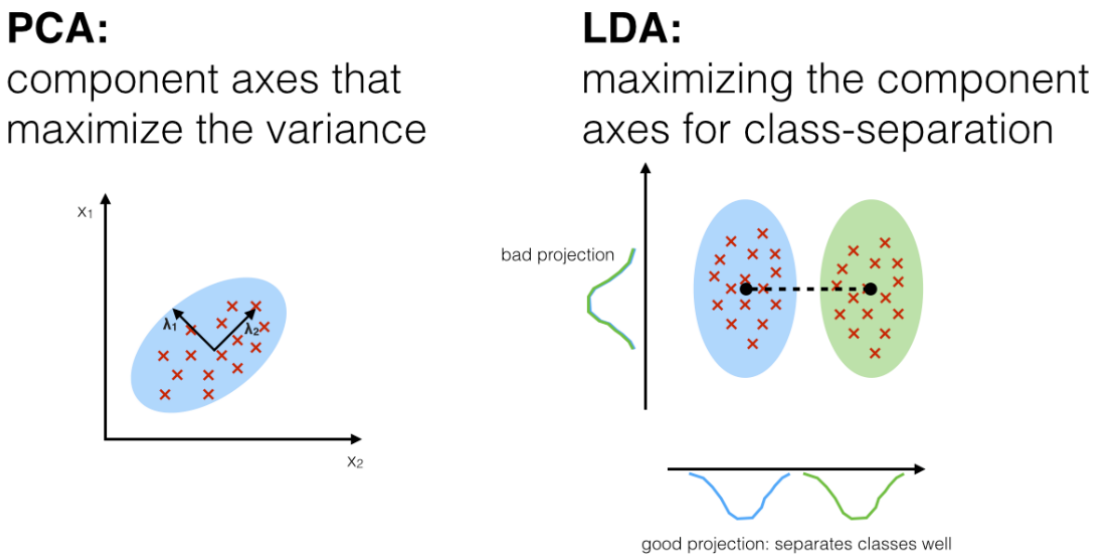


Fig 14. LDA and PCA comparison [11]

In our implementations we have used the scikit-learn library LinearDiscriminantAnalysis method.

### 4.2.3 LDA Final Transform Plot

In general the dimensionality of the transformed feature set is one less than the number of training classes [1].

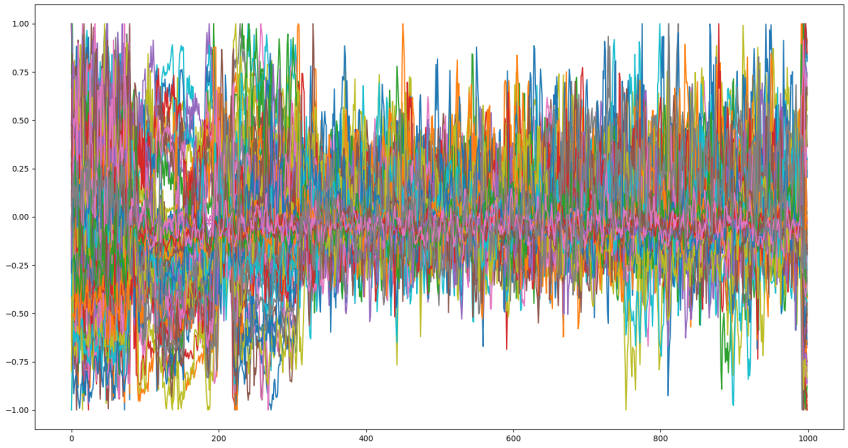


Fig 15. Normalized plots of all 78 final features

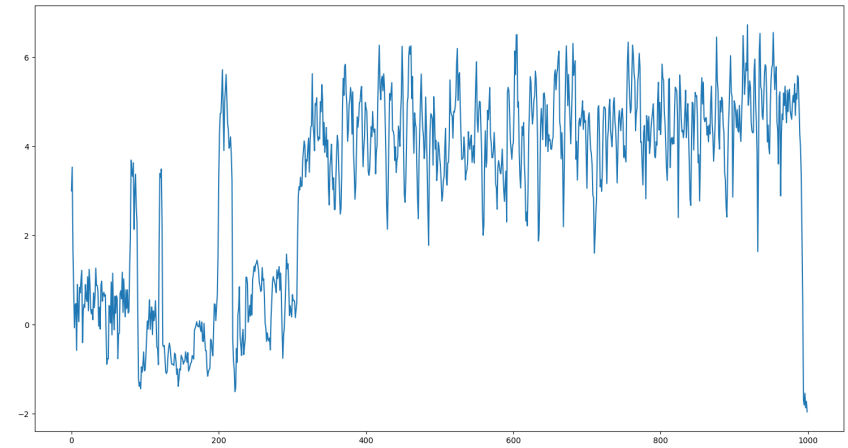


Fig 16. LDA output of the above features

## 4.3 LDA Predict Evaluation

After fitting the LDA model from Scikit-learn, we used the predict method of it, which is using the SVD (singular value decomposition) solver. Below you can find the evaluation of the LDA predict on test data for the speech target class.

| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| **Non-Speech** | **0.91** | **0.96** | **0.93** | **15778** |
| **Speech** | **0.89** | **0.77** | **0.83** | **6911** |

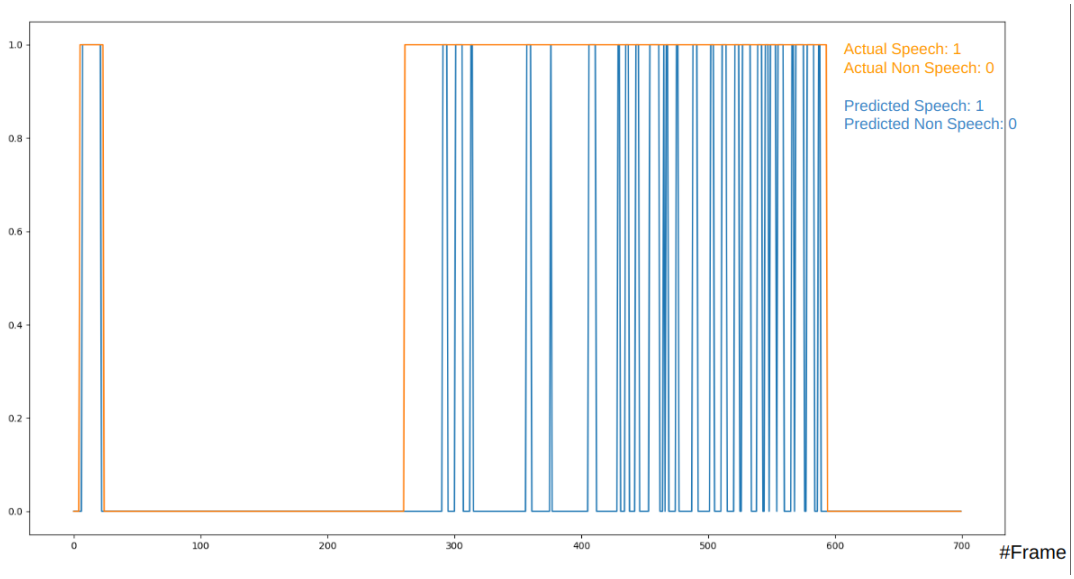


Fig 17. Speech, Non-speech LDA predict result

# Chapter 5

# Dynamic Programming

The main goal of the dynamic programming method is to overcome the shortages of the novelty function and peak picking method. The outcome of using DP method for the classification task are:

- Detecting the global signal trends
- Identifying segmentation boundaries when samples start to aggregate around a new mean

We will achieve these objectives by defining a cost function which is the combination of a local cost and a transition cost.

## 5.1 Novelty Function And Peak Picking Problem

Using novelty function and peak picking for classification has two major drawbacks:

1. It is designed to identify the local changes between successive samples (frames) and it does not consider the global changes between groups of samples which leads to the appearance of several spurious peaks. This case happens when two successive samples within a single cluster are further apart than successive samples in different clusters.

2. We won’t have any peaks in some actual boundaries and this happens when a sequence of samples increases gradually from one cluster to the adjacent cluster.

## 5.2 DP Structure

Suppose we have a feature set with length of N (like LDA transition output for N frames), note that F[j] which is showing the value of the feature set at time frame j can also be a vector of features. DP will create a N X N state machine. For each time frame j, there are N candidate states (which are the indices of the Features). letting i be the vertical state index, search state Sij is associated with choosing the i th Feature candidate for the j th actual feature value. The diagonal path of the state machine is the nominal feature-space trajectory of the signal (at time j, the nominal path is in state j). DP is able to find a path through the state machine which optimizes some specific cost function which can be composed of a local cost for each state as well as costs for transitions between states.

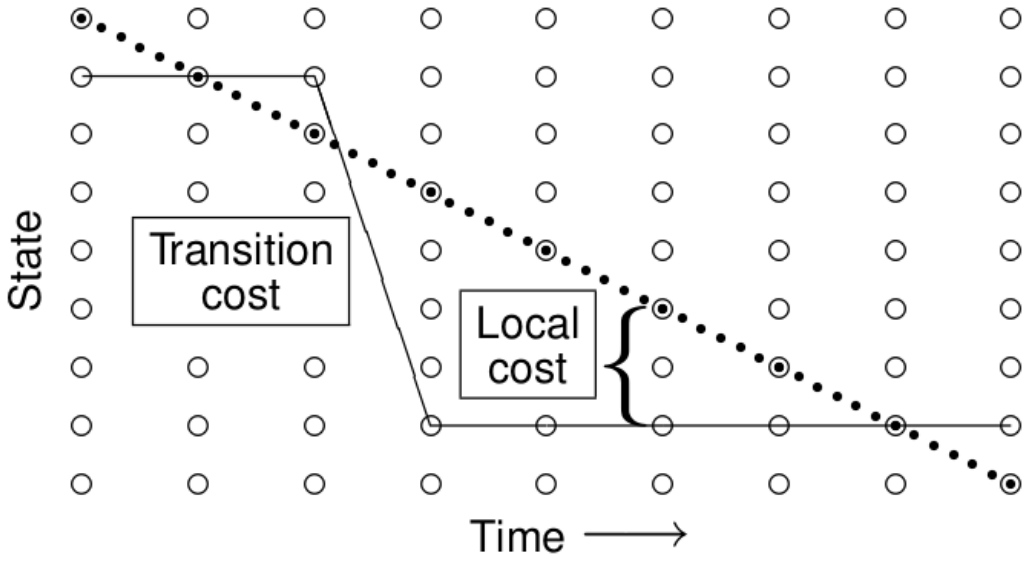


Fig 18. In the dynamic program for feature-space clustering, the diagonal (dotted) is the nominal feature path. A candidate cluster path with one transition is shown; there is a transition cost as well as a local cost for being in any state that is not on the nominal path [1, 2].

## 5.3 Cost Function

Since the goal of DP here is to find a path in the state machine which is a combination of horizontal segments divided by transitions, we define the cost functions in a way that the final optimal path is showing the candidate feature frame in each segment (horizontal path) and when we have a transition it means that we have a new segment.

We define the local cost and transition cost as follows:

**Local cost:** Is the Euclidean distance between Si, j and Si0, j which is how likely to be in state i0 at time frame j. It should be small if the candidate Si0, j is similar to the real feature value which is Sj,j

**Transition Cost:** The cost should be high for transition from state i to state j if they're similar and conversely it should be small for transition between dissimilar states. which a simple and reasonable choice is to use the inverse of Euclidean distance.

$$r[j] = \arg\min_{p[j]} \sum_{j} \alpha L\left(p[j], n[j]\right) \; + \; \beta T\left(p[j], p[j-1]\right)$$

$$= \arg\min_{p[j]} \{\alpha C_L(p[j]) + \beta C_T(p[j])\}$$

This is the aggregate cost of a candidate path in the state machine which is the summation of the local and transition cost with their coefficients for controlling the transition rate

## 5.4 DP Evaluation Report

Here is the evaluation of using dynamic programming on LDA output:

| | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|
| **Non-Speech** | **0.95** | **0.98** | **0.97** | **15778** |
| **Speech** | **0.95** | **0.89** | **0.92** | **6911** |

you can also observe how DP combined the oscillated misclassified frames and found a better classification.

Fig 19. Speech, Non-speech LDA predict and DP  result

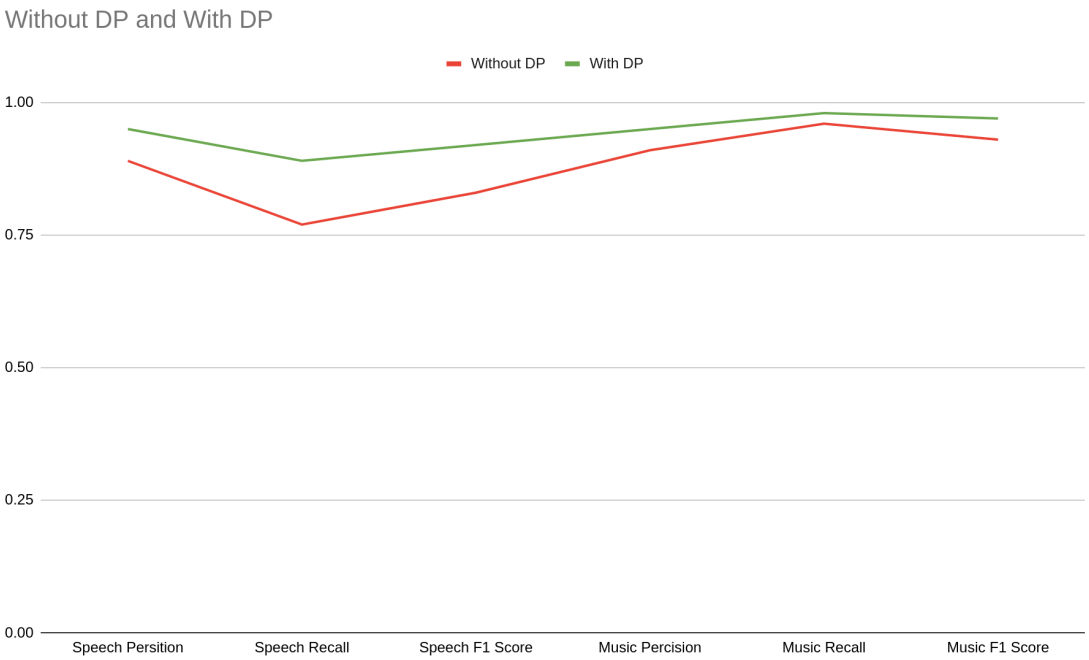


Fig 20. DP performance evaluation improvement

# Chapter 6

# Conclusion

Audio segmentation has many applications in a wide range of topics. You can find lots of publications in this area in which they proposed different kinds of methods for tackling this problem. In this audio lab project we implemented the Godwin papers [1,2] method and we added new features to it such as Chroma and MFCC. Moreover, we compared different features' performance (check out Fig 11, 12). At the end we showed how the dynamic programming method improved the segmentation task (Fig 20).

## 6.1 Summary

We implemented a method for audio segmentation task including these steps:

1. Preprocessing for labeling
2. Basic feature extraction in a short time periods (20 ms)
3. calculating some statistics for each basic feature (Mean and STD) for longer frames
4. Using LDA method for dimensionality reduction
5. Using Dynamic Programming method for segmentation (a non-heuristic method instead of peak-picking on novelty function)
6. showing some evaluation, how DP method improved the segmentation

## 6.2 Further Works

- We implemented the code in a way that it can be used for multi label segmentation as well. In order to use it for that application you have to change the sample and frame labeling methods for it. One of the future works could be testing and running the project for new multi label datasets.
- As we are using the raw features of Chroma and MFCC, we are able to improve these features to better features.
- We can implement a real time (with short time delay) function for audio segmentation.
- Creating a web or mobile application as a change environment detector, however, as we are not aware of the labels in the environment, we have to use an unsupervised method. One of the quickest worth trying approaches is to use PCA instead of LDA in our project

# Bibliography


[1] Goodwin, Michael M., and Jean Laroche. "Audio segmentation by feature-space clustering using linear discriminant analysis and dynamic programming." 2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No. 03TH8684). IEEE, 2003.

[2] Goodwin, Michael M., and Jean Laroche. "A dynamic programming approach to audio segmentation and speech/music discrimination." 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing. Vol. 4. IEEE, 2004.

[3] Ellis, Dan. "Chroma feature analysis and synthesis." Resources of laboratory for the recognition and organization of speech and Audio-LabROSA 5 (2007).

[4] Brian McFee, Alexandros Metsai, Matt McVicar, Stefan Balke, Carl Thomé, Colin Raffel, Frank Zalkow, Ayoub Malek, Dana, Kyungyun Lee, Oriol Nieto, Dan Ellis, Jack Mason, Eric Battenberg, Scott Seyfarth, Ryuichi Yamamoto, viktorandreevichmorozov, Keunwoo Choi, Josh Moore, … Taewoon Kim. (2022). librosa/librosa: 0.9.2 (0.9.2). Zenodo. https://doi.org/10.5281/zenodo.6759664

[5] J. Saunders, “Real-time discrimination of broadcast speech/music,” Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing (Atlanta, GA), pp. 993–996, May 1996.

[6] El-Maleh, Khaled, et al. "Speech/music discrimination for multimedia applications." *2000 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 00CH37100)*. Vol. 4. IEEE, 2000.

[7] S. A. Ramprashad, “A multimode transform predictive coder (MTPC) for speech and audio,” Proc. IEEE Workshop on Speech Coding for Telecom. (Porvoo, Finalnd), pp. 10–12, June 1999.

[8] Tzanetakis, George & Cook, Perry. (2003). Multifeature Audio Segmentation For Browsing And Annotation. 10.1109/ASPAA.1999.810860.

[9] https://blog.paperspace.com/introduction-to-audio-analysis-and-synthesis/

[10] https://setosa.io/ev/principal-component-analysis/#:~:text=By%20Victor%20Powell,easy%20to%20explore%20and%20visualize.

[11] https://sebastianraschka.com/Articles/2014_python_lda.html