Final Report
Speaker Recognition

Digital Signal Processing

by Vũ Vương Quốc Anh, Phạm Minh Long, Hoàng Nhật Tân, Nguyễn Trường Giang, Đỗ Đức Mạnh, Trần Minh Vương, Nguyễn Minh Hoàng, Nguyễn Gia Phong and Nguyễn An Thiết

June 12, 2020

Contents

1 Introduction

1.1 Brief Description

In this project, we have been experimenting with automatic speaker identification, with the main aim of being able to let the computer recognize each member of the team via their voice without depending on the content of the spoken content. This report is going to demonstrate the procedure has been carried out to achieve the goal, along with the understandings, mostly in digital signal processing, backing the results.

For reproduciblity, the code as well as the data are published on GitHub[*]. The software is released under AGPLv3+ with exceptions to some audios derived from all-right-reserved tests, while this report is licensed under a CC BY-SA 4.0 license.

1.2 Authors and Credits

The work has been undertaken by group number 6, whose members are listed in the following table.

Full name	Student ID
Vũ Vương Quốc Anh	BI9-046
Nguyễn Trường Giang	BI9-084
Nguyễn Minh Hoàng	BI9-108
Phạm Minh Long	BI9-146
Đỗ Đức Mạnh	BI9-163
Nguyễn Gia Phong	BI9-184
Hoàng Nhật Tân	BI9-205
Trần Minh Vương	BI9-239
Nguyễn An Thiết	BI8-174

We would like to express our special thanks to Dr. Trần Hoàng Tùng, whose lectures gave us basic understanding on the key principles of digital signal processing. Without his guidance, we might never have a chance to take an in-depth exploration on this topic.

[*] `https://github.com/McSinyx/speakerid`

Why is everything with you so complicated?

Rihanna, *Complicated*

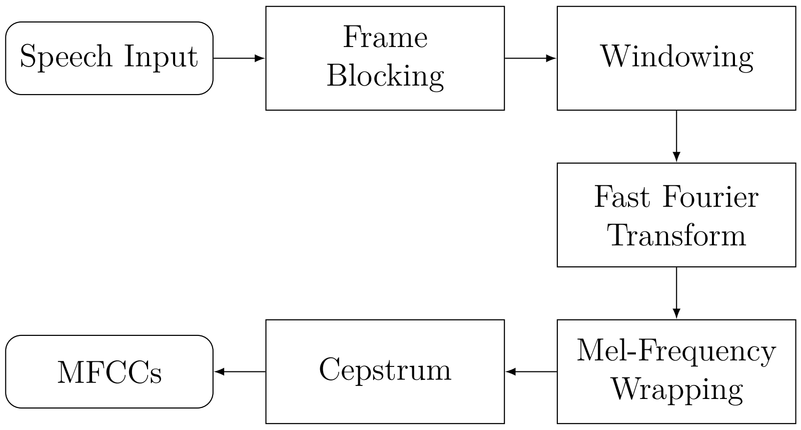
2 Theoretical Background

2.1 Mel Frequency Cepstral

The first step of automatic speaker identification is to extract the features of the voice, or precisely, the components of the audio signal that can identify the linguistic content while discarding all the irrelevant informations such as noise. One way to tackle this problem is to utilize mel-frequency cepstral coefficients (MFCCs).

MFCCs are coefficients that collectively make up an mel-frequency cepstrum, a short-term power spectrum representation of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. As low frequencies are spaced linearly and high frequencies logarithmically, it is believed that the representation is similar to the way human preceive sound [1]. For this reason, MFCCs are widely-used to represent the voice signal, and to extract the features of a signal for speaker recognition [2].

The process of calculating MFCC is presented in the following diagram [3]



2.1.1 Frame Blocking

The continuous input speech signal is blocked into frames of N samples, where the frames overlaps by a value R. This process is finished when all the speech is accounted for at least one frame [6].

2.1.2 Windowing

Instead of applying discrete Fourier transform on the speech signal, we perform such task on smaller frames of about 20 ms to 25 ms, as an average is speaking is about 3 to 4 words per second [4]. The use of framing will cause the suddenly drop of frequency in the border of the frames that might cause noises when applying DFT. In order to smooth such frames before applying DFT we use Hamming window to ensuring the continuity at the first and last points, where N is the length of the frame [5]:

$$w[n] = 0.54 - 0.46 \cos \frac{2\pi n}{N}, \qquad 0 \leq n \leq N$$

2.1.3 Fast Fourier Transform

In this step, we take the Fourier Transform of each frame, which will convert the frames from time domain to frequency domain. The Fast Fourier Transform is a nice algorithm to apply in this step. We will obtain spectrum as a result [6].

2.1.4 Mel-frequency Wrapping

Studies have shown that human perceives the frequency content of sounds for speech signal not in a linear scale. Therefore, the *mel* scale is used to measure a subjective pitch of each tone. The mel-frequency scale is linearly spaced below 1000 Hz and logarithmically spaced above 1000 Hz. A common method to simulate the subjective spectrum is to use mel filter bank [6].

2.1.5 Cepstrum

In the final step, we apply the Discrete Cosine Transform (DCT) to convert the log mel spectrum back to time. DCT is used because the mel spectrum coefficients are real numbers. Also, since the windows overlaps, DCT can helps to decorrelate them. The results of this step is MFCCs [6].

2.2 Gaussian Mixture Model (GMM)

After extracting features from MFCCs, these data will be trained using GMM. Expectation Maximization (EM) algorithm is used to train the extracted features from human voice in system.

2.2.1 GMM Overview

A Gaussian mixture model is a probabilistic clustering model for representing the presence of sub-populations within an overall population. The idea of training a GMM is to approximate the probability distribution of a class by a linear combination of k Gaussian distributions/clusters, also called the components of the GMM [7].

2.2.2 EM Algorithm

Initially, it identifies k clusters in the data by the K-means algorithm and assigns equal weight $w = \frac{1}{k}$ to each cluster. The number of clusters created then equal to the number of Gaussian Distribution. The algorithm proceeds iteratively in 2 steps [8]:

E-step: Assume that π_k, Σ_k and π and are set to the ones from the previous iteration of the algorithm. We have the normalized probability of each each point x_i belonging to one of the K Gaussians weighted by the mixture distribution π_k

$$r_{ik} = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i|\mu_j, \Sigma_j)}$$

M-step: Assume that the responsibilities r_{ik} are fixed. We will maximize our likelihood function across the variables μ_k, Σ_k and π_k using the following equations

$$\begin{aligned}
\pi_k &= \frac{1}{N} \sum_i r_{ik} \\
\mu_k &= \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}} \\
\Sigma_k &= \frac{\sum_i r_{ik}(x_i - \mu_k)(x_i - \mu_k)}{\sum_i r_{ik}}
\end{aligned}$$

Each iteration of EM increases the log-likelihood of the model. In practice, the model usually stops once the parameters or the log-likelihood have stopped changing very much.

$$\log p(\underline{X}) = \sum_i \log \left(\sum_c \pi_c \mathcal{N}(x_i|\mu_c, \Sigma c) \right)$$

A foolish consistency is the hobgoblin of little minds.

Ralph Waldo Emerson, *Self-Reliance*

3 Data Preparation

In order the produce reliable result, we prepare the data under the following consistent criteria.

3.1 Quantity and Content

Each team member provides at least 13 audio files, 8 for training and 5 for testing. The data is recorded by each member of the group independently, using their own hardware.

We have chosen the sample length in the range of 10 s to 30 s which we thought is neither too long or too short for automated as well as manual experimentation. Most of the content is daily conversation, which is close to the use case of the speaker recognition system.

3.2 Preprocessing

Audio files have been preprocessed by using Audacity in 3 steps:

1. Noise reduction to get rid of background noise frequencies [9].
2. Normalization the audio amplitude.
3. Standardization the audio file format. We chose the WAV format [10] and sampling rate of 44.1 kHz [11] for their simplicity and wide compatibility. By Nyquist–Shannon sampling theorem [12], the sampling rate is more than enough to cover the audible range of 20 Hz to 20 000 Hz [13].

There should be one—and preferably only one—obvious way to do it. Although that way may not be obvious at first unless you're Dutch.

Tim Peters, *The Zen of Python*

4 Implementation and Verification

Fortunately, the complicated algorithms described above are already implemented in third-party libraries for Python. Therefore, the system can be programmed *naively*, resulting the following modules:

- `naive.helpers` providing two functions extracting MFCCs from a given audio (with the FFT size set to 2048) and loading models from a given directory.
- `naive.train` applying MFCCs to GMM to decompose a set of data into multiple normally-distributed clusters with different means and standard deviations and output the models as Python `pickle`s.
- `naive.test` automatically testing the correctness of the models.
- `naive.run` performing interactive speaker identification using the trained models.

In the first test, we used 1 voice file around 35 seconds long for each member of the group to train the model. Then, we test the model with the exact files and it got 100% accuracy. This result is expected as the file used for training and testing are the same.

In the second test, we used file number 1, 2, 4 for model training and file number 3 for testing and the result of 100% accuracy still holds (the file numbers are noted in the GitHub project README).

In the next test, we had 8 training sample and 5 test audios for each member. With significantly more resources to train the model (in vocabulary as well as reading paces for paragraphs/dialogues), the system performed surprisingly well with one out of 30 guesses was incorrect. Given many test audios were not preprocessed for noise cancellation, we considered the system to be quite noise-resilient.

In the verification phase, with the deadline being short, we did not experiment with different parameter for the MFCC and GMM function. However, initial results with the interactive system seems to be promising enough, yielding mostly correct results unless there was voice interference from other people.

5 Conclusion

With the above results and the understandings we gained during the execution, we consider this project as a success. We believe that with further finetuning, a GMM-MFCCs model like the one we used can give sufficient accuracy for common use cases of speaker recognition.

6 References

[1] Hugo Fastl, Eberhard Zwicker. “Section 6.2: Critical-Band Rate Scale”. *Psychoacoustics: Facts and Models*. Springer-Verlag Berlin Heidelberg, 2007. ISBN 978-3-540-68888-4

[2] Todor Ganchev, Nikos Fakotakis, George Kokkinakis. *Comparative Evaluation of Various MFCC Implementations on the Speaker Verification Task*. SPECOM, 2005.

[3] Md. Sahidullah, Goutam Saha. “Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition”. *Speech Communication*. doi:10.1016/j.specom.2011.11.004.

[4] Voice Qualities. *The National Center for Voice and Speech.*

[5] Julius O. Smith III. “Hamming Window”. *Spectral Audio Signal Processing*. W3K Publishing, 2011. ISBN 978-0-9745607-3-1.

[6] Minh N. Do. 1How to Buildan AutomaticSpeaker Recognition System.

[7] James Lyon *Voice Gender Detection using GMMs : A Python Primer*, 2017.

[8] Brian Keng. *The Expectation-Maximization Algorithm*, 2016.

[9] Noise Reduction. *Audacity Manual.*

[10] Multimedia Programming Interface and Data Specifications 1.0. *IBM, Microsoft.*

[11] AES5-2018. *Audio Engineering Society.*

[12] Claude E. Shannon “Communication in the presence of noise”. *Proceedings of the Institute of Radio Engineers.* doi:10.1109/jrproc.1949.232969.

[13] Stuart Rosen. *Signals and Systems for Speech and Hearing*, 2nd ed., p. 163. BRILL, 2011.