How Can We Differentiate Between Voices?

Written by AlephZero for CCExtractor, Google Code-In 2019

Abstract

We would like to solve the problem of separating an audio input with multiple people speaking into different files, or tracks, each representing a different voice. A solution to this problem would lead to many useful applications, for example producing subtitles for a debate show, or a transcription for a court hearing.

In this research, I present different solutions for this problem, each with different strengths, or different assumptions about the content and format of the recording.

1 Possible solutions

1.1 Sound localization

An audio file can either be **mono** or **stereo**. In a **mono** recording, there is only one microphone being used, and only one single input track in the audio. This makes it much more difficult to separate voices. In a **stereo** recording, however, there are essentially two microphones that are recording the audio from different angles, or positions. This leads to a perception of space - as listeners, without even thinking about it when the audio reaches our brain we differentiate between sounds based on their perceived location.

So, how do we do that in code? Given a stereo recording, or different recordings from a microphone array, we can use different techniques to measure the location of the sound based on the location of the microphones. For example, we can use a method called **time difference of arrival**, or **TDOA**. [1] To apply this method we need to measure the time it takes for sound to arrive to each of the microphones, and then assuming we know the position of the microphones, we can calculate the exact position of each object producing sound. Once we can differentiate between different positions, assuming people don't move throughout the recording, we can theoretically differentiate different people, or voices, talking.

1.2 Deep learning

Another approach is using artificial intelligence, specifically deep learning techniques.

1.2.1 Non-simultaneous voices

The first case is that people aren't talking simultaneously (this should occur in a court hearing, for example). In this case, the problem is significantly simpler (but still very complicated).

Here is a possible approach: first, we will separate the audio into small chunks. Because people aren't talking simultaneously, we assume that most of the chunks contain only one person speaking. For each of those chunks, we can use the **Fourier Transform (FFT)** to convert it from sound to frequencies and amplitudes. Then, we can use **unsupervised learning** techniques (clustering) to try and find similarities between the different frequencies, and cluster them into groups - each corresponding to a different voice. After that, we can see which chunk corresponded to which voice, and see exactly when each person talked. [2]

1.2.2 Simultaneous voices - "The Cocktail Party Problem"

But what if different people are talking at the same time? With these constraints, the problem becomes much more difficult. A relevant problem in the field is **"The Cocktail Party Problem"**.

The name is borrowed from the following phenomenon: suppose you are at a cocktail party, with music and many different people talking all around you. You are having a conversation with your friend, and somehow you are able to tune out every other noise and listen only to your friend's voice.

The natural question is, can we replicate the same effect using computers. This is an active research question, and different research teams have solved parts of the problem - for example, separating a singer's voice from the background music, using machine learning techniques. [3]

2 Existing tools

- pyAudioAnalysis - https://github.com/tyiannak/pyAudioAnalysis
  This python library can solve this problem, as well as other similar problems such as classifying sounds, removing silence from sound and visualizing audio data.
- Spleeter by Deezer - https://github.com/deezer/spleeter
  Not exactly related to this problem, but interesting nonetheless - solves a part of the cocktail party problem I mentioned earlier, separating vocals and instruments from a song
- A few more smaller tools (or implementations of research papers):
  - SpkDiarization (https://projets-lium.univ-lemans.fr/spkdiarization/)
  - Audioseg (http://audioseg.gforge.inria.fr/)
  - github.com/yinruiqing/change_detection
  - SIDEKIT for speaker diarization (https://projets-lium.univ-lemans.fr/s4d/)

References

[1] https://en.wikipedia.org/wiki/Acoustic_location#Time_difference_of_arrival

[2] Anguera, Xavier. "Speaker diarization: A review of recent research"

[3] Simpsion, Andrew J.R. "Deep Karaoke: Extracting Vocals from Musical Mixtures Using a Convolutional Deep Neural Network"

[4] https://en.wikipedia.org/wiki/Speaker_diarisation