Week 10 - Assignment 2

Question 3: improvements

Stripping silence

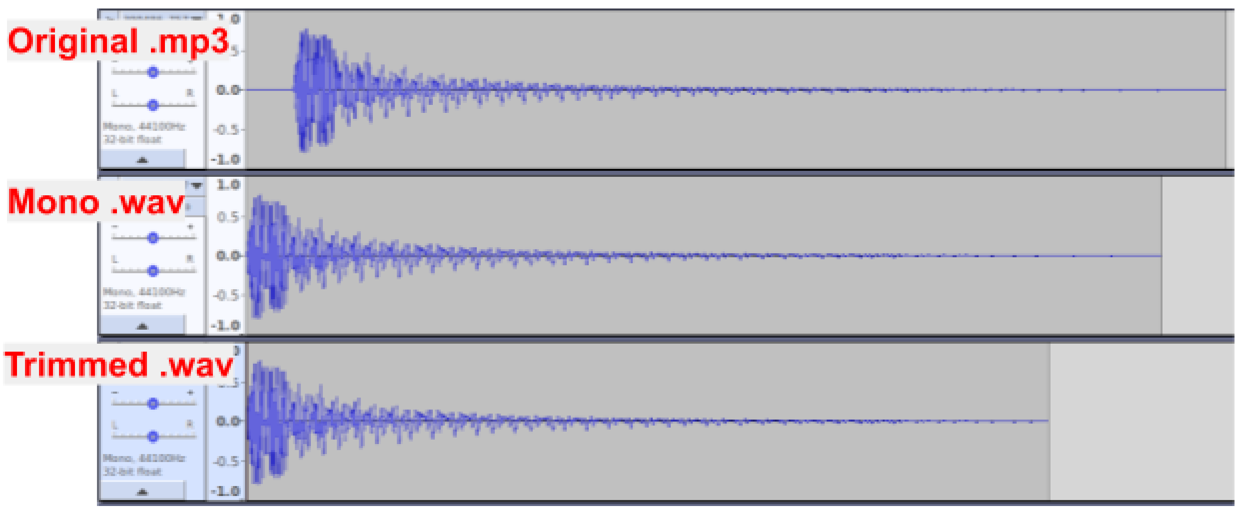
To remove the low energy parts of the audio, I used the function computeEngEnv() that we coded in week 4 but modified it to remove the frames whose energy was below some threshold and stored the clipped audio in a file with suffix _trimmed.wav. The complete code is in trim_silence.py.

I first converted the original HQ mp3 files to mono (1 channel) 44.1k Hz so they could be processed by the SMS code. Then, using trim_silence.py, I removed the audio < -50 dB (I determined this threshold by visually inspecting in Audacity a couple of files). I played and

Here are a couple of samples showing the differences of each transformation:

A snare sound (https://freesound.org/search/?q=373477)

A flute sound (https://freesound.org/search/?q=354398)

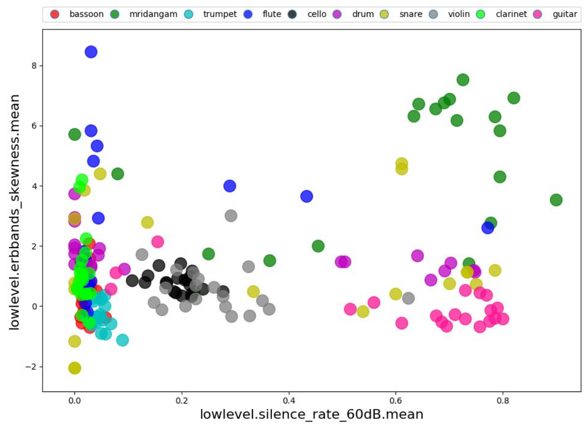


A guitar sound (https://freesound.org/search/?q=399486)

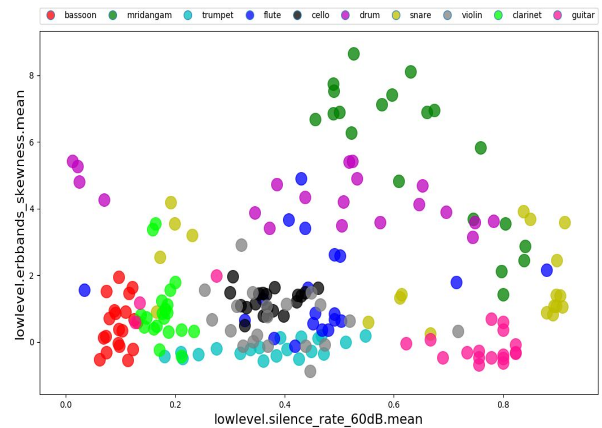
Some audios simply disappeared after being trimmed. For example, https://freesound.org/search/?q=373472, an inaudible snare recording, was clipped to 0.

Clustering using Essentia features

I then plotted the trimmed files (see descriptorPairScatterPlot() in essentia_descriptors_clustering.py) to see if removing the low energy frames made a difference and it certainly did (at least for some descriptors):



With silence trimmed



Original sound

To obtain the Essentia descriptors and do the clustering I used parts of soundAnalysis.py used in the previous assignment and example_musicextractor.py (an example script to use extractors located in the Essentia GitHub repo). The result is in essentia_descriptors_clustering.py.

To find the best descriptor I looped through all of the descriptors (just the mean ones) and plotted them using descriptorPairScatterPlot() and shortlisted them to the following based on how well they visually clustered apart different classes of sound:

- lowlevel.erbbands_crest.mean
- lowlevel.barkbands_skewness.mean
- lowlevel.erbbands_flatness_db.mean
- lowlevel.erbbands_skewness.mean
- lowlevel.erbbands_spread.mean
- lowlevel.loudness_ebu128.momentary.mean
- lowlevel.melbands_flatness_db.mean
- lowlevel.melbands_crest.mean
- lowlevel.pitch_salience.mean
- lowlevel.silence_rate_60dB.mean
- lowlevel.spectral_rms.mean
- rhythm.beats_loudness.mean

spectral_decrease.mean single point

1. [pitch_salience, silence_rate_60dB] -> 52%
2. [erbbands_flatness_db, erbbands_skewness] -> 50.5%
3. [erbbands_flatness_db, erbbands_skewness, pitch_salience] -> 49.5%
4. [erbbands_flatness_db, erbbands_skewness, spectral_decrease] -> 56%
5. [loudness_ebu128, silence_rate_60dB, erbbands_skewness, erbbands_flatness_db, erbbands_skewness, spectral_decrease] -> 61%

In some cases, the accuracy varied significantly from one run to another. For example, for the combination no. 3, I got 48.5%, 54%, and 49.5% in three successive runs.

I was quite surprised to see how adding a seemingly bad descriptor as the spectral_decrease increased the accuracy by ~6%. And although I thought that specifying more than 2 descriptors would simply confuse the algo and would make the classification worse, it turned out to be the opposite: in combination no. 5 above there are 6 descriptors and the accuracy is significantly increased.