PMC Write up

My project is trying to make an audio feature extraction tool to try and identify a piece of music from a small segment of sound. I’ve only used a small database of songs (3 songs) for testing purposes. You can either record a buffer from your phone into the laptop speakers like you would in ‘shazam’ or something similar, or you can load a small section of one of the songs into it. From there once the code is executed each song gets analysed by using sliding window fft analysis so an fft is taken at certain time intervals throughout the song and 3 features are identified. It takes the most prominent wave of each fft frame, it takes the centroid values throughout the song, and finally takes the RMS value of each frame.

The fft is also windowed. This is to reduce the leakage of the fft signal, so some signals don’t leak into adjacent energy bins (G. Lyons, 1998). The challenge I had with the feature extraction analysis was finding a way to compare the relevant time slots. The programme had to find the relevant part of the song in which to compare the data, before it compared the feature extraction data to see which song has the closest match. To do this I analysed the whole song against a segment (test buffer) of the song, against the whole array of each feature for every window and find the difference. With the difference I was able to see which window is the closest fit and take that data to compare against the rest of the songs. This is what is happening in the second last cell.

I stored the value of the window that had the lowest difference for centroid, RMS and most prominent frequency and stored what value that was in an array. I did this for each song so that I had the lowest window value for each feature and each song stored in an array. That way I could calculate the Euclidean mean using the best matching window to find out what was the overall closest match (Euclidean space - Wikipedia, 2021).

Overall I’ve made an application that computes audio similarity by extracting different features from audio files. From what I’ve researched I think there is an element of locally sensitive hashing by how it finds only the closest matching points of data to analyse (Hari, 2021). It works very well when there is clean snippets of audio being compared however when it comes to recording lower quality audio from a phone into laptop speakers it can easily get confused, although mostly it does it successfully. This could be made better by comparing more features, using some other methods like beat tracking (Smith, Campbell and Goldstein, 2015), or doing some other research into finding ways to compare noisy/clouded samples of sound.

After giving it to my peer to test I made the interface a bit more clear and added the function where you can record it playing from your phone to make it more interactive. I added some extra annotations so that the functioning of it is more easily, even if someone’s not been told.

Note: you need the sounddevice library installed if you want the record function to work. I have linked below:

“conda install -c conda-forge python-sounddevice”

(https://python-sounddevice.readthedocs.io/en/0.3.12/installation.html)

Smith, L., Campbell, O. and Goldstein, M., 2015. *index*. [online] Musicinformationretrieval.com. Available at: <https://musicinformationretrieval.com/> [Accessed 10 May 2021].

En.wikipedia.org. 2021. *Euclidean space - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Euclidean\_space> [Accessed 10 May 2021].

G. Lyons, R., 1998. *Windowing Functions Improve FFT Results, Part I*. [online] EDN. Available at: <https://www.edn.com/windowing-functions-improve-fft-results-part-i/> [Accessed 10 May 2021].

Hari, S., 2021. *Locality Sensitive Hashing*. [online] Locality Sensitive Hashing. Available at: <https://santhoshhari.github.io/Locality-Sensitive-Hashing/> [Accessed 10 May 2021].