Tools for Feature Extraction - exploring and using essentia library

MUS-17

Akshay Paranjape

July 5, 2017

1 Introduction

Essentia, Open Source C++ library for Music has been developed by Muisc Technology Group at *Universitat Pompeu Fabra* for the analysis for music. Its fairly young library, the first version was released in 2008. Sebastian Streich *et al* [2] have explained Detrented Fluctuation approach for analyzing music and suggested a musical attribute described as danceability. We make use of all this and the essentia library to find Danceability and other features for different kind of music.

To work with Essentia we have to import this library in Python interface

```
import essentia
import essentia.standard
```

This report wil guide you through implementation of danceability factor feature and other possible feature from Essentia.

2 Detrended Fluctuation Function

Detrented Fluctuation Analysis is method for scaling long-term autocorrelation of non-stationary signals [1] . This process was first introduced in analysis of music by [2]. DFA is statistical analysis of the data of audio file. Audio signal is amplitude verses time graph. DFA computes the variance of this graph for fixed time interval. Using this DFA approach we can build an impression of how music evolves over time.

Computational procedure for DFA is described very well by Jenings et all [3]. We first segment the audio signal into non-overlapping blocks of small interval. For each block standard deviation $s(n)$ of the amplitude is calculated. In order to obtain the unbounded time series $y(m)$, $s(n)$ is integrated

$$y(m) = \sum_{n=1}^{m} s(n)$$

Time interval has to be small so as not to lose the flow of the music. If we take very large time interval $s(n)$ would turn out to be zero. y(m) is now segmented into blocks of τ elements. In the manner of sliding window we sample from one block of length τ to other. This approach of τ number of block will give us better visualization of audio signal[2].

We now compute y_k mean of $y(m)$ and take mean of square residual and denote it by $D(k, \tau)$, here k is number of blocks

$$D(k, \tau) = \frac{1}{\tau} \sum_{m=0}^{\tau-1} (y(k+m) - y_k(m))^2$$

Detrended Flutuation function is then obtained by computing the square root mean of $D(k, \tau)$ for all k blocks.

$$F(\tau) = \sqrt{\frac{1}{K} \sum_{k=1}^{K} D(k, \tau)}$$

From DFA we get to know the correlation properties on different time scale. Ideal method would be to repeat the process for different number of τ values and check the pattern. τ is altered by fixed multiplication factor. We have used the

application of this algorithm in essentia and altered the code to get more parameters as output, Section of Danceability

$F(\tau)$ is related to τ via exponentially. We define DFA exponent to visualize the graph better. We get DFA exponent by taking log of F over τ denoted by α.

$$\alpha(i) = \log_\tau F(\tau)$$

$$\alpha(i) = \frac{\log_{10} F(\tau)}{\log_{10} \tau}$$

A constant value of 0.5 is obtained for a completely random series (white noise), a value of 1 for a series with 1/f-type noise, and 1.5 for a Brown noise series (integrated white noise)[4]

We have used audio music file for different dance style like Paso, Tango, Rumba, ChaChaCha, Goodbeat etc. Detrended algorithm analyzed this music according to the alpha value. Fig 1,2,3 shows graph for DFA exponent α and τ, x-component of graph denotes time in seconds and y-component shows the DFA exponent α value.

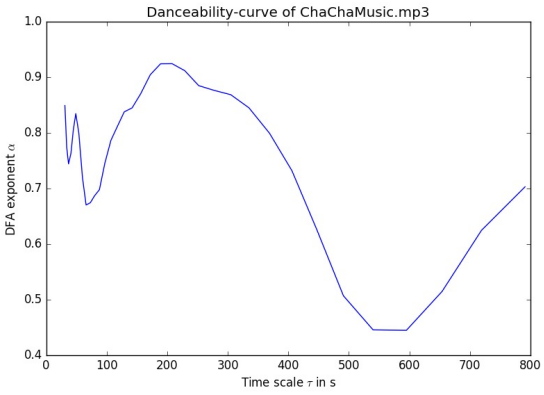

Figure 1: ChaChaCha Music

From the graph we can see that value of α varies significantly. The strong and regular beat pattern can give instability in the behaviour of graph as we can see for Tango Fig.3

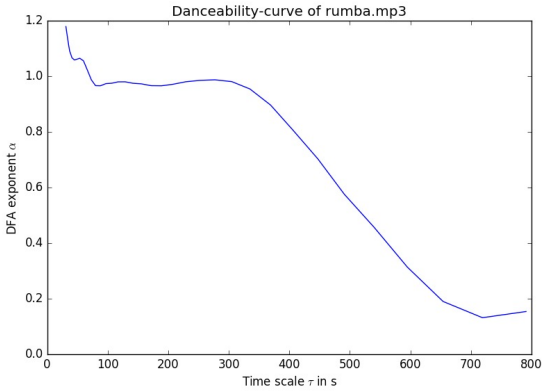

Figure 2: Rumba Music

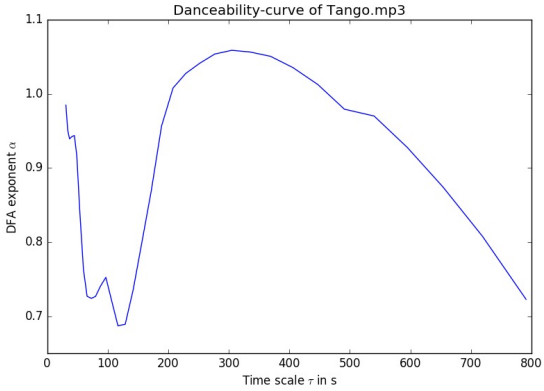

Figure 3: Tango Music

3 Danceability

The term danceability in itself gives us the hint. It tells us about the ability to dance on some audio. The presence of small and regular beat is the main characteristics of danceable music[2]. When do we feel like dancing ? When music has good variation in beats, good correlation between different time intervals denoted by τ in the above section. Presence of strong and regular beat is also the main characteristics of danceable music. Fig. 1,2,3 shows the DFA exponent for Rumba, Tango and ChaChaCha music.

Danceability is taken as the average of DFA exponent α. This can be considered as one of the major application in playlist generation or music recommendation for parties. The Essentia library gives inverse of DFA exponent value, which

is between 0 to 3. Higher the value, more it is danceable.

3.1 Computation of Danceability

Essentia provides library for finding the danceability factor of a given audio signal. The audio signal first has to be processed using function *loader*. The output of the loader can be stored in some other variable.

```
audio = loader()
dance = Danceability()
danceability_value = dance(audio)
```

The output of the function is then used as an input for Danceability function, which gives you inverse of average of α as the output.

3.2 Altering the code

In the original code by Essentia, it didn't output all the parameters. After altering the algorithm, we are able to get DFA Exponent vector as output parameter, so that we can plot the above graph. We defined this function as DanceabilityDetailed.cpp . Usage of the function in python based interface is given below :

```
dance2 =  DanceabilityDetailed()
DFA_exp = dance2(audio)
```

The output parameters that we get here are vector and average value of α

4 Classifying the songs

Essentia has lot many features. My focus was mainly on features related to Rhytm. Except danceability, I have looked upon other features like BeatTrackerDegara, Rubato, Meter, Tempo. This features can be used as classifier for songs.

```
BeatTrackerDegara()
Rubato()
Meter()
Temp()
```

In the appendix I have mentioned the code in Python for our algorithm

5 Conclusion

Essentia is a powerful library which can be used to extract music features from Spectral, Segmentation, Tonal, Duration/Silence, Loudness, Rhytm, Pitch, Filter and few others. It's python inbuild interface makes it easier to implement. It is been used world wide for music classification both in research and commercial use. Companies like freesound, dunya etc. uses Essentia. Danceability algorithm is easy to implement and can be used to get an overview of any song. Essentia is highly recommended for them who wish to play around with music and have fun.

6 Acknowledgement

I would like to thank Prof. P.Bientinesi for discussion over possibe implementation in Essentia library. This project was planned and worked out by me and Andrea Hanke under the seminar course 'Topics in Computer Music'.

References

[1] Agnieszka Kitlas Goliska. *Detrended Fluctuation Analysis (DFA) in biomedical signal processing:selected examples*. Studies in logic, Grammer and Rhetoric 29, 42 (2012).

[2] Sebastian Streich, Perfecto Herrera *Detrended Fluctuation Analysis of Music Signals: Danceability Estimation and further Semantic Characterization*. AES 118th Convention, Barcelona, Spain, (2006)

[3] H. D. Jennings et al. *Variance fluctuations in nonstationary time series: a comparative study of music genres*. Condensed Matter (2003),http://xxx.lanl.gov/abs/cond-mat/0312380

[4] C.-K. Peng et al *Fractal Mechanisms in Neural Control: Human Heartbeat and Gait Dynamics in Health and Disease*. Online Tutorial http://www.physionet.org/tutorials/fmnc/

Appendix

Code for dancebility

```
// Danceability Detailed
import essentia
import essentia.standard
import essentia.streaming
import matplotlib.pyplot as plt
import numpy as np

#music play

# let's define a small utility function
def play(audiofile):
    import os, sys

    # NB: this only works with linux!! mplayer rocks!
    if sys.platform == 'linux2':
        if playMusic == 'yes':
            cmd = 'mplayer ' + audiofile + ' -ss ' + str(myMusicStarts[musicIndex]
            os.system(cmd)
        else:
            print 'Not playing audio because you asked me not to do so'
    else:
        print 'Not playing audio...'
#play(musicfile)

# Mainstep-we start by instantiating the audio loader:
loader = essentia.standard.EasyLoader(filename = musicfile, startTime = musicStart

# and then we actually perform the loading:
audio = loader()

from pylab import *
print 'Plotting Audio file:'
print 'audio array has size '
print len(audio)
plot(audio[1*44100:2*44100]) #[1*44100:2*44100]
show()

from essentia.standard import *
w = Windowing(type = 'hann')
spectrum = Spectrum()  # FFT() would give the complex FFT, here we just want the m
mfcc = MFCC()
beatTracker = BeatTrackerDegara()
print 'Computing Danceability of file ' + musicfile
danceability = Danceability()
```

```
danciness = danceability(audio[:])
print danciness
danceabilityDetailed = DanceabilityDetailed()

(dancinessDetailed, taus, coefficients) = danceabilityDetailed(audio[:])

print dancinessDetailed
print "The coefficients:"
print coefficients
print "The corresponding tau values:"
print taus

print "Output of beat tracker:"
print beatTracker(audio[:])
frame = audio[5*44100 : 5*44100 + 1024]
fig = plt.figure(figsize=(9,6))
ax  = fig.add_subplot(111)
ax.plot(coefficients, taus)
plt.title('Danceability-curve of ' + myMusicFiles[musicIndex])
plt.xlabel(r"Time scale ${\tau}$ in s")
plt.ylabel(r"DFA exponent ${\alpha}$")
#fig.savefig("DFA_"+myMusicFiles[musicIndex]+".png")
def onclick(event):
    play(musicfile)
cid1=fig.canvas.mpl_connect('button_press_event',onclick) # when i click the mouse
#show()
```