Tools for Feature Extraction

Exploring and using the essentia library

A. Hanke[1] A. Paranjape[2]

[1]andrea.hanke@rwth-aachen.de
[2]akshay.paranjape@rwth-aachen.de

Topics in Computer Music, RWTH Aachen, July 2017

Outline

Section 1

Section 1

A few remarks and little bit of history

Essentia is

- a C++ library for audio analysis with Python bindings

A few remarks and little bit of history

Essentia is

- a C++ library for audio analysis with Python bindings
- fairly young (first version released in 2008, newest version 2.1 beta3 released on September 2016)

A few remarks and little bit of history

Essentia is

- a C++ library for audio analysis with Python bindings
- fairly young (first version released in 2008, newest version 2.1 beta3 released on September 2016)
- open-source, with continuing additions

Some features of essentia

Essentia provides algorithms for

- spectral analysis

Some features of essentia

Essentia provides algorithms for

- spectral analysis
- extraction and segmentation

Some features of essentia

Essentia provides algorithms for

- spectral analysis
- extraction and segmentation
- statistics and math

Some features of essentia

Essentia provides algorithms for

- spectral analysis
- extraction and segmentation
- statistics and math
- tonal and pich analysis

Some features of essentia

Essentia provides algorithms for

- spectral analysis
- extraction and segmentation
- statistics and math
- tonal and pich analysis
- duration, silence

Some features of essentia

Essentia provides algorithms for

- spectral analysis
- extraction and segmentation
- statistics and math
- tonal and pich analysis
- duration, silence
- loudness, dynamics and rythm

Some features of essentia

Essentia provides algorithms for

- spectral analysis
- extraction and segmentation
- statistics and math
- tonal and pich analysis
- duration, silence
- loudness, dynamics and rythm
- filters
- ...

Operating modes

Essentia provides two operating modes:

- standard, recommended for

Operating modes

Essentia provides two operating modes:

- standard, recommended for
  - maximum control in C++
  - research with Python (interactive environment)

Operating modes

Essentia provides two operating modes:

- standard, recommended for
  - maximum control in C++
  - research with Python (interactive environment)
- streaming, recommended for

Operating modes

Essentia provides two operating modes:

- standard, recommended for
  - maximum control in C++
  - research with Python (interactive environment)
- streaming, recommended for
  - easy-to-write extractors in C++ and Python
  - porting from Python to C++

Section 2

Using essentia in Python

- Step 0: Import the library

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

Using essentia in Python

- Step 0: Import the library

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

- Step 1: Instantiate algorithms from the library

```
loader = essentia.standard.EasyLoader(
                    filename = musicfile,
                    startTime = musicStart,
                    endTime = musicEnd)
```

Using essentia in Python

- Step 0: Import the library

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

- Step 1: Instantiate algorithms from the library

```
loader = essentia.standard.EasyLoader(
                    filename = musicfile,
                    startTime = musicStart,
                    endTime = musicEnd)
```

- Step 2: Use the algorithms

```
audio = loader()
```

Algorithms in essentia

can have multiple

- parameters, set during the instantiation

Algorithms in essentia

can have multiple

- parameters, set during the instantiation
- inputs, set when using an algorithm

Algorithms in essentia

can have multiple

- parameters, set during the instantiation
- inputs, set when using an algorithm
- outputs

Section 3

Step 1:

Create the .h and .cpp files:

- danceabilityDetailed.h
- danceabilityDetailed.cpp

Step 2: danceabilityDetailed.h

start of file, class declaration

```
#ifndef ESSENTIA_DANCEABILITYDETAILED_H
#define ESSENTIA_DANCEABILITYDETAILED_H

#include "algorithm.h"
#include "essentiamath.h"

namespace essentia {
namespace standard {

class DanceabilityDetailed : public Algorithm{
```

Step 2: danceabilityDetailed.h

declare all protected variables, especially input and output:

```
protected:
  Input<std::vector<Real> > _signal;
  Output<Real> _danceability;
  Output<std::vector<Real> > _dfaExponents;
  Output<std::vector<Real> > _dfaTaus;
  int _preferredSize, _actualSize;
```

Step 2: danceabilityDetailed.h

constructor declaration, here input and output are declared

```
public:
  DanceabilityDetailed() {
    _preferredSize = 36; //as seen in the paper
    _actualSize = _preferredSize;
    declareInput(_signal, "signal", "#d");
    declareOutput(_dfaTaus, "dfaTaus", "#d");
  }
```

Step 2: danceabilityDetailed.h

declare parameters

```
void declareParameters() {
  declareParameter("minTau", "#d",
                   "(0,inf)", 310.);
  declareParameter("maxTau", "#d",
                   "(0,inf)", 8800.);
}
```

Step 2: danceabilityDetailed.h

declare other functions or variables

```
void compute();
void configure();

static const char* name;
static const char* category;
static const char* description;
```

Step 2: danceabilityDetailed.h

declare other functions or variables

```
  protected:
    std::vector<int> _tau;
    Real stddev(const std::vector
          <Real>& array, int start,
              int end) const;
};

} // namespace standard
} // namespace essentia
```

Step 2: danceabilityDetailed.h

declare other functions or variables

```
  protected:
    std::vector<int> _tau;
    Real stddev(const std::vector
          <Real>& array, int start,
              int end) const;
};

} // namespace standard
} // namespace essentia
```

Repeat for streaming...

Step 3: danceabilityDetailed.cpp

File start and fill inherited variables

```
#include "danceabilityDetailed.h"

using namespace std;
namespace essentia {
namespace standard {

const char* DanceabilityDetailed
    ::name= "DanceabilityDetailed";
```

Step 3: danceabilityDetailed.cpp

File start and fill inherited variables

```
const char* DanceabilityDetailed
    ::category = "Rhythm";
const char* DanceabilityDetailed
    ::description
    = DOC("Long\n\nDescription");
```

Step 3: danceabilityDetailed.cpp

Fill the declared functions with meaning

```
Real DanceabilityDetailed::
        stddev(const vector<Real>& array,
        int start, int end) const {
    ...
}
void DanceabilityDetailed::configure() {
    ...
}
void DanceabilityDetailed::compute() {
    ...
}
```

Section 2

Section 1

Sample frame title

- Detrended Fluctuation Function - statistical self affinity $y(m) = \sum_{n=1}^{m} s(n)$
- Computation procedure
  - Segmentation
  - Standard Deviation
- Danceability taken as the average of the DFT
- Higher the Danceability value, more is the music danceable

Section 2

Modifying Essentia algorithm

- Essentia provides open source C++ library with Python interface
- Danceability() function
  - Input - signal(vector real)
  - Output - danceability, DFA(real)
- Modified the code as according to get vector as output
- Danceability values for Music
  - ChaChaCha - 1.18
  - Rumba - 1.05
  - Tango - 1.28
- Inference - Tango is more Danceable

Section 3

Results

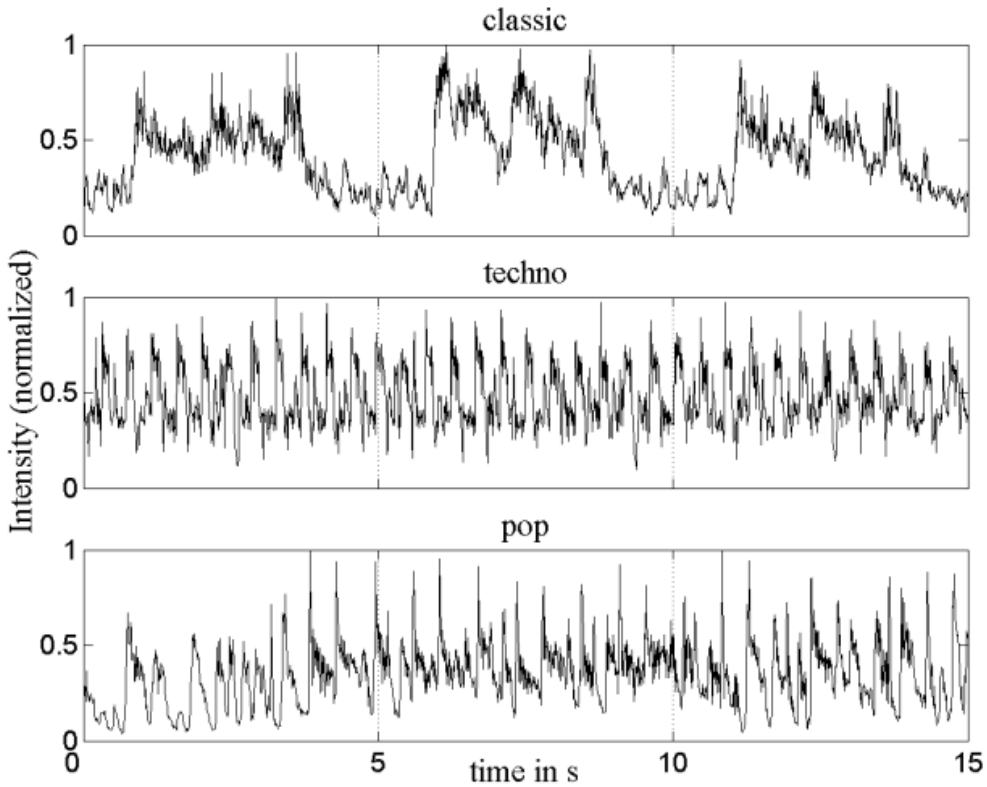

Figure: From the research paper: time series s(n)

Results

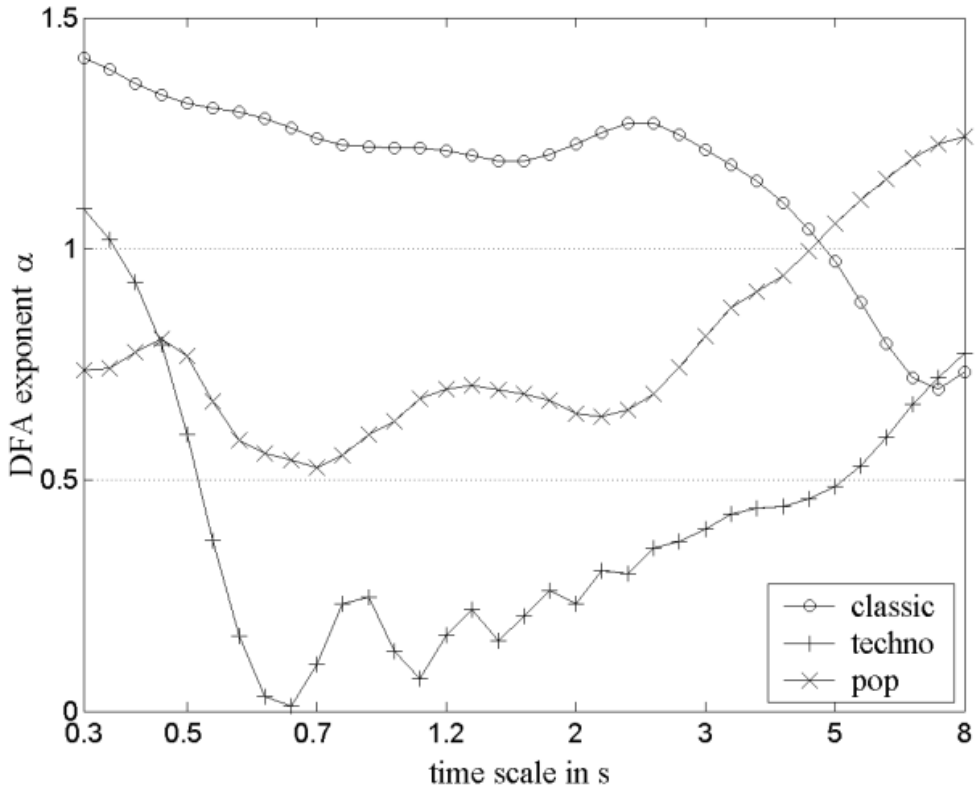

Figure: From the research paper: DFA Exponent Function

Let's take a look and try it out!

Section 3

Section 1

Purpose of the algorithm

is to

- extract features from songs

Purpose of the algorithm

is to

- extract features from songs
  - beats per minute
  - meter
  - danceability
  - ...

Purpose of the algorithm

is to

- extract features from songs
  - beats per minute
  - meter
  - danceability
  - ...
- classify them according to their features (possibly with machine learning)

Purpose of the algorithm

is to

- extract features from songs
  - beats per minute
  - meter
  - danceability
  - ...
- classify them according to their features (possibly with machine learning)
- be used as classification for DJ-algorithm

Section 2

Extraction demonstration of

- beats per minute

Extraction demonstration of

- beats per minute
- rubato

Extraction demonstration of

- beats per minute
- rubato
- meter

Extraction demonstration of

- beats per minute
- rubato
- meter
- novelty

Extraction demonstration of

- beats per minute
- rubato
- meter
- novelty
- danceability

Let's take a look and try it out!

Section 4

Conclusion and Summary

- Essentia is a powerful tool to extract music features with many options and algorithms ranging from Spectral, tonal, Rhythm

Conclusion and Summary

- Essentia is a powerful tool to extract music features with many options and algorithms ranging from Spectral, tonal, Rhythm
- In Python interface which makes it easy for beginners

Conclusion and Summary

- Essentia is a powerful tool to extract music features with many options and algorithms ranging from Spectral, tonal, Rhythm
- In Python interface which makes it easy for beginners
- Inbuilt C++ function giving us the opportunity to alter the code

Conclusion and Summary

- Essentia is a powerful tool to extract music features with many options and algorithms ranging from Spectral, tonal, Rhythm
- In Python interface which makes it easy for beginners
- Inbuilt C++ function giving us the opportunity to alter the code
- Essentia Library is being used world wide for Music Classification based on tempo, rubato etc.

Conclusion and Summary

- Essentia is a powerful tool to extract music features with many options and algorithms ranging from Spectral, tonal, Rhythm
- In Python interface which makes it easy for beginners
- Inbuilt C++ function giving us the opportunity to alter the code
- Essentia Library is being used world wide for Music Classification based on tempo, rubato etc.
- Danceability approach mentioned by us is easy and well defined to get a quick overview of any song

Conclusion and Summary

- Essentia is a powerful tool to extract music features with many options and algorithms ranging from Spectral, tonal, Rhythm
- In Python interface which makes it easy for beginners
- Inbuilt C++ function giving us the opportunity to alter the code
- Essentia Library is being used world wide for Music Classification based on tempo, rubato etc.
- Danceability approach mentioned by us is easy and well defined to get a quick overview of any song
- Overall we recommend this library for music lovers and who wish to play around, especially rhythm