

Speechresearch/gmisclib

API Documentation

September 13, 2011

Contents

Contents	1
1 Package lib	2
1.1 Modules	2
1.2 Variables	2
2 Module lib.grabdata2	3
2.1 Functions	3
2.2 Variables	4
2.3 Class faxis	5
2.3.1 Methods	5
2.3.2 Properties	5
3 Module lib.readDB	6
3.1 Functions	6
3.2 Variables	6

1 Package lib

1.1 Modules

- **grabdata2**: This module computes the uncompressed feature vectors for music rhythm determination.
(Section 2, p. 3)
- **readDB** (Section 3, p. 6)

1.2 Variables

Name	Description
--package--	Value: None

2 Module *lib.grabdata2*

This module computes the uncompressed feature vectors for music rhythm determination.

This can be run as a script for debugging/exploratory purposes. When run as a script, it produces plots of various stages of the signal processing sequence.

It is also imported as a module, in which case the plotting is off. The normal call is to `make_vector()`.

2.1 Functions

s_preprocess(*f*, *Dt*=None, *plotit*=False)

First stage. This does the signal processing to (essentially) compute a spectrogram of the music. The spectrogram is a two-dimensional representation of the sound: time is one axis, and the other axis is frequency.

It uses a "perceptual spectrum" which is more representative of what humans actually hear than the raw spectrogram. Essentially, it matches the spectral bandwidth to the ear: it uses a filterbank where each filter is 0.7 erb wide. Then, it half-wave rectified the signal and takes the cube root, in order to match the amplitude compression in the ear.

Parameters

f: filename for the input audio file.

(*type*=*str*)

Dt: The desired sampling interval of the output feature vectors. (E.g. 10ms.)

(*type*=*float*)

Return Value

A sequence of feature vectors that span the specified region. This is a 2-dimensional spectrogram.

(*type*=*numpy.ndarray* 2-dimensional.)

plot_speech(*smd*)

Plot the acoustic data.

spectral(*sd*, *Dt*, *plotaudio=False*)

Compute beat rates from a spectrogram.

Parameters

sd: his starts from a spectrogram with one time axis and one frequency axis,
(*type=*`numpy.ndarray` 2-dimensional array of floats.)

Return Value

It computes an array where the time axis is converted to a beat-rate axis. So, the value of each element tells you how strong a particular beat is, at a particular pitch. An element might reveal the strength of a 60BPM beat in the high pitch range near 3 kHz (e.g. snare drum). So, the frequency axis separates the track by instrument, and the beats-per-minute axis lets you look at the rhythm of each instrument. (Of course, that's a somewhat idealized view...)
(*type=*`numpy.ndarray`, 2-dimensional array of floats.)

stage2(*s*, *fa*, *plotit=False*)

make_vector(*f1*, *name*, *plotaudio=False*)

The basic idea here is that we should not expect a song to have a consistent rhythmic pattern throughout. So, we divide the track into blocks, and inside each block, we assume a consistent pattern. (Obviously, this is not perfect: we chop it into blocks without regard for the song. A better scheme might be to try to find boundaries in the song where the rhythm changes.) If the rhythm changes within a block, you'll get a blurred rhythmic pattern.

At the end, we report the average rhythm pattern, over all the blocks.

Parameters

f1: name of the audio file
(*type=*`str`)

name: the track's label. (Note: this is just passed through to the output.)
(*type=**normally str*)

plotaudio: display plots (true) or run silently (false). This particular routine plots the averaged beat rate as a function of frequency, and also the averaged feature vector.
(*type=*`boolean`)

Return Value

It returns an opaque blob (a **pickle**) that you can write to a file and easily read back. The pickle contains the track's label, along with the uncompressed feature vector. This feature vector contains the beat rate as a function of frequency, followed by the part of the feature vector that describes the averaged rhythmic pattern.
(*type=*A *pickle* of (*name*, *feature vector*))

2.2 Variables

Name	Description
CACHEROOT	Where to cache intermediate results. Value: <code>'/tmp/artdist'</code>
CWT	Value: 1.0
MEDRATE	Value: 1.0
<code>--package--</code>	Value: <code>'lib'</code>

2.3 Class *faxis*

object  **lib.grabdata2.faxis**

This is a logarithmically spaced frequency axis.

2.3.1 Methods

`--init--(self, f0, r, fmx)`

`x.--init--(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.--init--` `exitit`(inherited documentation)

`vals(self, i=None)`

Parameters

`i`: ask for the frequency at a certain index. Or, if `i=None` (the default), yield an array of all the frequencies.

(*type=int or None.*)

Inherited from object

`--delattr--()`, `--format--()`, `--getattribute--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

2.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

3 Module *lib.readDB*

3.1 Functions

<code>read(<i>fn</i>)</code>

<code>read_as_datum_tr(<i>fn</i>)</code>
--

<code>read_list_as_datum_tr(<i>fn</i>)</code>

3.2 Variables

Name	Description
<code>--package--</code>	Value: 'lib'

Index

- lib (*package*), 2
 - lib.grabdata2 (*module*), 3–5
 - lib.grabdata2.faxis (*class*), 5
 - lib.grabdata2.make_vector (*function*), 4
 - lib.grabdata2.plot_speech (*function*), 3
 - lib.grabdata2.s_preprocess (*function*), 3
 - lib.grabdata2.spectral (*function*), 3
 - lib.grabdata2.stage2 (*function*), 4
 - lib.readDB (*module*), 6
 - lib.readDB.read (*function*), 6
 - lib.readDB.read_as_datum_tr (*function*), 6
 - lib.readDB.read_list_as_datum_tr (*function*), 6