樹莓派, 樹莓派之學習, 樹莓派之教育

【鼎革．革鼎】：RASPBIAN STRETCH《六之 K.3-言語界面-6.1》

2018-02-03 | 懸鉤子 | 發表迴響

眼見 Python Pocketsphinx 之預設組構：

Default config

If you don’t pass any argument while creating an instance of the Pocketsphinx, AudioFile or LiveSpeech class, it will use next default values:

```
verbose = False
logfn = /dev/null or nul
audio_file = site-packages/pocketsphinx/data/goforward.raw
audio_device = None
sampling_rate = 16000
buffer_size = 2048
no_search = False
full_utt = False
hmm = site-packages/pocketsphinx/model/en-us
lm = site-packages/pocketsphinx/model/en-us.lm.bin
dict = site-packages/pocketsphinx/model/cmudict-en-us.dict
```

Any other option must be passed into the config as is, without using symbol -.

If you want to disable default language model or dictionary, you can change the value of the corresponding options to False:

```
lm = False
dict = False
```

……

道明其語音檔格式原始乎？

Raw audio format

RAW Audio format or just **RAW Audio** is an audio file format for storing uncompressed audio in raw form. Comparable to WAV or AIFF in size, RAW Audio file does not include any header information (sampling rate, bit depth, endian, or number of channels). Data can be written in PCM, IEEE 754 or ASCII.

Extensions

Raw files can have a wide range of file extensions, common ones being **.raw**, **.pcm**, or **.sam**. They can also have no extension.

Playing

As there is no header, compatible audio players require information from the user that would normally be stored in a header, such as the encoding, sample rate, number of bits used per sample, and the number of channels.

.....

該怎樣以『檔案為介面』界接 librosa 耶？！

一時想起☆

Advanced I/O Use Cases

This section covers advanced use cases for input and output which go beyond the I/O functionality currently provided by *librosa*.

Read specific formats

librosa uses audioread for reading audio. While we chose this library for best flexibility and

support of various compressed formats like MP3: some specific formats might not be supported. Especially specific WAV subformats like 24bit PCM or 32bit float might cause problems depending on your installed audioread codecs. *libsndfile* covers a bunch of these formats. There is a neat wrapper for *libsndfile* called PySoundFile which makes it easy to use the library from python.

Note

See installation instruction for PySoundFile here.

Reading audio files using PySoundFile is similmar to the method in *librosa*. One important difference is that the read data is of shape `(nb_samples, nb_channels)` compared to `(nb_channels, nb_samples)` in <`librosa.core.load`>. Also the signal is not resampled to 22050 Hz by default, hence it would need be transposed and resampled for further processing in *librosa*. The following example is equivalent to `librosa.load(librosa.util.example_audio_file())`:

```
import librosa
import soundfile as sf

# Get example audio file
filename = librosa.util.example_audio_file()

data, samplerate = sf.read(filename, dtype='float32')
data = data.T
data_22k = librosa.resample(data, samplerate, 22050)
```

何不直搗黃龍

PySoundFile

PySoundFile is an audio library based on libsndfile, CFFI and NumPy. Full documentation is available on http://pysoundfile.readthedocs.org/.

PySoundFile can read and write sound files. File reading/writing is supported through libsndfile, which is a free, cross-platform, open-source (LGPL) library for reading and writing

many different sampled sound file formats that runs on many platforms including Windows, OS X, and Unix. It is accessed through CFFI, which is a foreign function interface for Python calling C code. CFFI is supported for CPython 2.6+, 3.x and PyPy 2.0+. PySoundFile represents audio data as NumPy arrays.

一探究竟呢◎

soundfile.available_formats

`soundfile.available_formats()`

Return a dictionary of available major formats.

Examples

```
In [1]: import soundfile as sf

In [2]: sf.available_formats()
Out[2]:
{'AIFF': 'AIFF (Apple/SGI)',
 'AU': 'AU (Sun/NeXT)',
 'AVR': 'AVR (Audio Visual Research)',
 'CAF': 'CAF (Apple Core Audio File)',
 'FLAC': 'FLAC (Free Lossless Audio Codec)',
 'HTK': 'HTK (HMM Tool Kit)',
 'IRCAM': 'SF (Berkeley/IRCAM/CARL)',
 'MAT4': 'MAT4 (GNU Octave 2.0 / Matlab 4.2)',
 'MAT5': 'MAT5 (GNU Octave 2.1 / Matlab 5.0)',
 'MPC2K': 'MPC (Akai MPC 2k)',
 'NIST': 'WAV (NIST Sphere)',
 'OGG': 'OGG (OGG Container format)',
 'PAF': 'PAF (Ensoniq PARIS)',
 'PVF': 'PVF (Portable Voice Format)',
 'RAW': 'RAW (header-less)',
 'RF64': 'RF64 (RIFF 64)',
 'SD2': 'SD2 (Sound Designer II)',
 'SDS': 'SDS (Midi Sample Dump Standard)',
 'SVX': 'IFF (Amiga IFF/SVX8/SV16)',
 'VOC': 'VOC (Creative Labs)',
```

```
 'W64': 'W64 (SoundFoundry WAVE 64)',
 'WAV': 'WAV (Microsoft)',
 'WAVEX': 'WAVEX (Microsoft)',
 'WVE': 'WVE (Psion Series 3)',
 'XI': 'XI (FastTracker 2)'}
```

`soundfile.available_subtypes`(*format=None*)

Return a dictionary of available subtypes.

PARAMETERS:	**format** (*str*) – If given, only compatible subtypes are returned.

Examples

```
In [3]: sf.available_subtypes('RAW')
Out[3]:
{'ALAW': 'A-Law',
 'DOUBLE': '64 bit float',
 'DWVW_12': '12 bit DWVW',
 'DWVW_16': '16 bit DWVW',
 'DWVW_24': '24 bit DWVW',
 'FLOAT': '32 bit float',
 'GSM610': 'GSM 6.10',
 'PCM_16': 'Signed 16 bit PCM',
 'PCM_24': 'Signed 24 bit PCM',
 'PCM_32': 'Signed 32 bit PCM',
 'PCM_S8': 'Signed 8 bit PCM',
 'PCM_U8': 'Unsigned 8 bit PCM',
 'ULAW': 'U-Law',
 'VOX_ADPCM': 'VOX ADPCM'}
```

`soundfile.check_format`(*format*, *subtype=None*, *endian=None*)

Check if the combination of format/subtype/endian is valid.

Examples

```
In [4]: sf.check_format('RAW', 'PCM_16')
Out[4]: True
```

soundfile.default_subtype(*format*)

Return the default subtype for a given format.

Examples

```
In [5]: sf.default_subtype('RAW')

In [6]:
```

雖不免波折★

soundfile.info(*file*, *verbose=False*)

Returns an object with information about a SoundFile.

PARAMETERS:	**verbose** (*bool*) – Whether to print additional information.

soundfile.available_formats()

Return a dictionary of available major formats.

Examples

```
In [6]: sf.info('./test/goforward.RAW')
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-9b0cdef67ec7> in <module>()
----> 1 sf.info('./test/goforward.RAW')

/usr/local/lib/python3.5/dist-packages/soundfile.py in info(file, verbose)
    550         Whether to print additional information.
    551     """
--> 552     return _SoundFileInfo(file, verbose)
    553
    554

/usr/local/lib/python3.5/dist-packages/soundfile.py in __init__(self, file, verbose)
    497     def __init__(self, file, verbose):
    498         self.verbose = verbose
--> 499         with SoundFile(file) as f:
    500             self.name = f.name
```

```
    501             self.samplerate = f.samplerate

/usr/local/lib/python3.5/dist-packages/soundfile.py in __init__(self, file, mode, sampl
    737         self._mode = mode
    738         self._info = _create_info_struct(file, mode, samplerate, channels,
--> 739                                          format, subtype, endian)
    740         self._file = self._open(file, mode_int, closefd)
    741         if set(mode).issuperset('r+') and self.seekable():

/usr/local/lib/python3.5/dist-packages/soundfile.py in _create_info_struct(file, mode,
   1520     if 'r' not in mode or format.upper() == 'RAW':
   1521         if samplerate is None:
-> 1522             raise TypeError("samplerate must be specified")
   1523         info.samplerate = samplerate
   1524         if channels is None:

TypeError: samplerate must be specified
```

恰逢即將『狗來富』之時

RAW Files

Pysoundfile can usually auto-detect the file type of sound files. This is not possible for RAW files, though:

Pysoundfile can usually auto-detect the file type of sound files. This is not possible for RAW files, though:

```
In [7]: data, samplerate = sf.read('./test/goforward.RAW', channels=1, samplerate=16000,
```

Note that on x86, this defaults to `endian='LITTLE'`. If you are reading big endian data (mostly old PowerPC/6800-based files), you have to set `endian='BIG'` accordingly.

You can write RAW files in a similar way, but be advised that in most cases, a more expressive format is better and should be used instead.