

# 《用 Python 玩转数据》之 WAV 音频处理入门研究

By Dazhuang@NJU

Python 在许多领域有重要的应用。例如，

本实验的主要目的是学习使用 Python 对音频信号进行简单的处理。

实验的主要步骤如下：

1. 网络数据读取与保存
2. 使用 scipy 中的 wavfile 模块对音频信号进行简单的幅度处理
3. 使用 matplotlib 中的 pylab 模块对音频信号进行简单的频谱分析

## 1. 网络数据读取与保存

主要使用 urllib.request 模块

此处选择位于 <http://www.nch.com.au/acm/11k16bitpcm.wav> 的音频作为数据源，使用 urllib.request.urlopen() 函数取得该文件，并以 'english.wav' 的文件名保存在本地的程序所在路径。

## 2. 简单幅度处理

使用 wavfile 模块可以比较简单地从 wav 格式的文件中读取相应的采样率、数据等信息，而不必关心文件的格式细节。python 的 wave 模块也有相应的操作，示例详见代码。

读取后使用 pyplot 模块，可以绘制出音频文件的波形图。

在本例中，将信号幅值按比例减小，表现出来的效果就是声音强度减小。

使用 numpy 模块，将原始数据与常数相乘，得到一个值减小的新数组。这里利用了 numpy 的广播机制，如果两个相乘的对象长度不等，numpy 会根据一定的规则将对象扩展为相同的类型，再做运算。

计算完成后，使用 wavfile.write() 函数将修改后的音频信号写入一个新的文件中，命名为 'silent.wav'，采样率未修改，与原音频相同。

最后使用 matplotlib.pyplot 绘制波形图对比。

## 3. 简单时频分析

音频信号是一种常见的非平稳信号，频域特性随时间变化。时频图是分析音频信号的常用工具，在一幅图中表示出信号的频率、幅度随时间的变化。pylab 模块提供了 specgram 函数，可以简单地通过配置相应参数进行短时傅里叶变换，并输出时频图。

扩展阅读：

使用 Python 进行声音处理：

[http://old.sebug.net/paper/books/scipydoc/wave\\_pyaudio.html](http://old.sebug.net/paper/books/scipydoc/wave_pyaudio.html)

numpy 官方文档的广播机制介绍：

<https://docs.scipy.org/doc/numpy/reference/ufuncs.html#broadcasting>

specgram 函数官方文档中的参数介绍：

[http://matplotlib.org/api/pyplot\\_api.html#matplotlib.pyplot.specgram](http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.specgram)

【代码】

```
import scipy.io.wavfile
import wave
import matplotlib.pyplot
import matplotlib.pyplot
import urllib.request
import numpy

response =
urllib.request.urlopen('http://www.nch.com.au/acm/11k16bitpcm.wav')

WAV_FILE = 'english.wav'
file = open(WAV_FILE, 'wb+')
file.write(response.read())
file.close()

wavefile = wave.open(WAV_FILE, 'r')
params = wavefile.getparams()
nchannels, sample_width, framerate, numframes = params[:4]

sample_rate, data = scipy.io.wavfile.read(WAV_FILE)

matplotlib.pyplot.subplot(2,1,1)
matplotlib.pyplot.title('Original')
matplotlib.pyplot.plot(data)

newdata = data * 0.2
newdata = newdata.astype(numpy.int16)

scipy.io.wavfile.write('silent.wav', sample_rate, newdata)

matplotlib.pyplot.subplot(2,1,2)
matplotlib.pyplot.title('Quiet')
matplotlib.pyplot.plot(newdata)

matplotlib.pyplot.show()

result = matplotlib.pyplot.specgram(newdata, NFFT=1024, Fs = sample_rate,
noverlap=900)
#y:音频信号
#NFFT: 每个进行快速傅里叶变换的数据块大小, 一般取 2 的幂次
#Fs:采样率
#noverlap: 数据块之间重叠数据点的个数
```

```
matplotlib.pyplot.show()
```

```
'''
```

#注：Python 的 wave 模块也可以实现相应的 wav 文件读取，示例代码如下：

```
wavefile = wave.open(WAV_FILE,'r')
```

```
params = wavefile.getparams()
```

```
nchannels, sample_width, framerate, numframes = params[:4]
```

```
#nchannels:声道数
```

```
#sample_width:采样宽度,每个采样的字节数
```

```
#framerate:采样率
```

```
#numframes:总采样数
```

```
y_data = wavefile.readframes(numframes)
```

```
y = numpy.fromstring(y_data, dtype=numpy.int16)
```

```
result = matplotlib.pyplot.specgram(y, NFFT=1024, Fs = framerate, noverlap=900)
```

```
'''
```