**NLTK自然语言处理利器**

NLTK是Python很强大的第三方库，可以很方便的完成很多自然语言处理（NLP）的任务，包括分词、词性标注、命名实体识别（NER）及句法分析等等。也是很多书籍的御用库，比如：《用Python进行自然语言处理(中文)》。

NLTK的官网：http://www.nltk.org/

NLTK的官方指导书籍：http://www.nltk.org/book/

NLTK的github主页位于：https://github.com/nltk/nltk

## NLTK库安装

相关的入门安装教程（包括语料库下载）

<http://www.cnblogs.com/huangcong/archive/2011/08/29/2157437.html>

如：需要下载一些数据文件：

*import  nltk*

*nltk.download()*

         也可到http://nltk.org/nltk\_data/手动下载，然后放到根目录下。

（python安装根目录）

## NLTK库轻松入门NLP

### **一、NLTK加载语料库**

针对一批语料，nltk提供了一些处理函数。对自己收集的文本文件， 并且想使用nltk提供的方法访问它们， 可以在NLTK中的PlaintextCorpusReader帮助下载入*它们。*

*>>>rom nltk.corpus import PlaintextCorpusReader*

*>>>corpus\_root = '/usr/share/dict'*

*>>>wordlists = PlaintextCorpusReader(corpus\_root, '.\*')*

其中corpus\_root是自己文本文件的目录。nltk的分词是针对英语的，可以自己处理好分词后，再将语料交给nltk来处理。

加载好了语料，究竟能作什么样的处理呢？nltk提供的基本语料库处理函数如下：

|  |  |
| --- | --- |
| fileids() | 语料库中的文件 |
| fileids([categories]) | 这些分类对应的语料库中的文件 |
| categories() | 语料库中的分类 |
| categories([fileids]) | 这些文件对应的语料库中的分类 |
| raw() | 语料库的原始内容 |
| raw(fileids=[f1,f2,f3]) | 指定文件的原始内容 |
| raw(categories=[c1,c2]) | 指定分类的原始内容 |
| words() | 整个语料库中的词汇 |
| words(fileids=[f1,f2,f3]) | 指定文件中的词汇 |
| words(categories=[c1,c2]) | 指定分类中的词汇 |
| sents() | 指定分类中的句子 |
| sents(fileids=[f1,f2,f3]) | 指定文件中的句子 |
| sents(categories=[c1,c2]) | 指定分类中的句子 |
| abspath(fileid) | 指定文件在磁盘上的位置 |
| encoding(fileid) | 文件的编码（如果知道的话） |
| open(fileid) | 打开指定语料库文件的文件流 |
| root() | 到本地安装的语料库根目录的路径 |

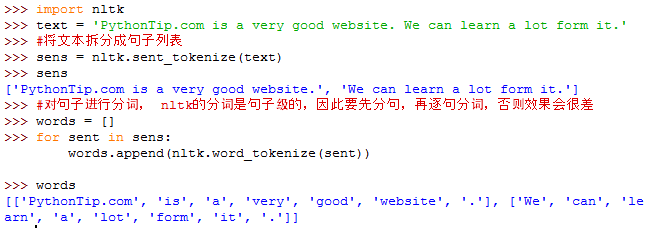
这些是基本的函数，更多的功能还要结合其他模块使用。同时，nltk提供了很多的语料库：古腾堡语料库，网络和聊天文本，布朗语料库等。在教程中，利用这些语料库可以尝试nltk提供的其他功能。

### **二、NLTK进行分词**

**用到的函数：**

**nltk.sent\_tokenize(text) #对文本按照句子进行分割**

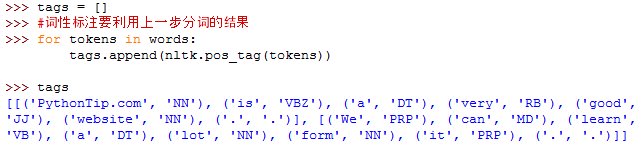
**nltk.word\_tokenize(sent) #对句子进行分词**



### **三、NLTK进行词性标注**

**用到的函数：**

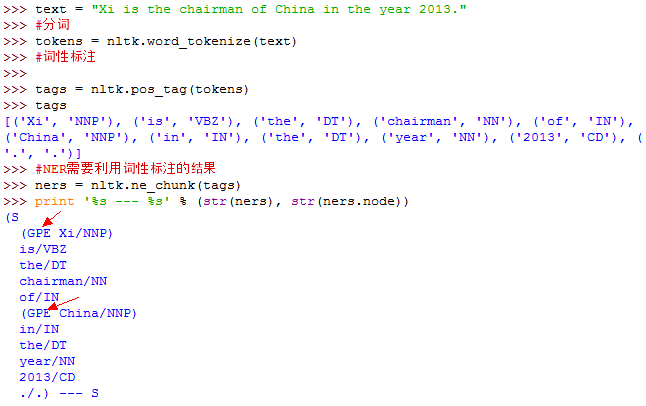
**nltk.pos\_tag(tokens)#tokens是句子分词后的结果，同样是句子级的标注**

****

### **四、NLTK进行命名实体识别（NER）**

**用到的函数：**

**nltk.ne\_chunk(tags)#tags是句子词性标注后的结果，同样是句子级**



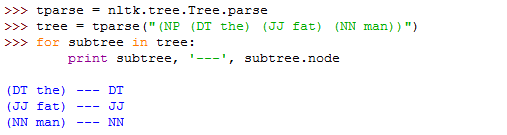
上例中，有两个命名实体，一个是Xi,这个应该是PER，被错误识别为GPE了； 另一个事China，被正确识别为GPE。

### **五、句法分析**

nltk没有好的parser，推荐使用Stanford parser

但是nltk有很好的树类，该类用list实现

可以利用Stanford parser的输出构建一棵python的句法树



### **六、概率统计**

如果以上功能还没发满足您的需求，那么，可以尝试使用概率统计功能来进行更为灵活的底层实现您想要的功能。本功能主要使用**FreqDist类。**

 注：下文斜体表示运行结果

**初始化实例**：fdist1=FreqDist(['abc','cd','cd']);

*("FreqDist({'a': 5, ' ': 5, 'e': 5, 's': 3, 'n': 3, 'i': 2, 'p': 1, 'T': 1,'x': 1, 'c': 1, ...})")*

                      fdist1=FreqDist('This is an example sentenceaaa')

*("FreqDist({'cd':2, 'abc': 1})")*

**计算特定样本次数**：fdist1['cd']

*2*

**计算特定样本频率**：fdist1.freq('cd')

*0.6666666666666666*

**样本总数**：fdist1.N()

*3*

                              fdist1.B()

*2*

**遍历样本**：for sample in fdist1:

                            print(sample)

*abc*

*ab*

**数值最大的样本**：fdist1.max()

                               'cd'

**绘制频率分布表**：fdist1.tabulate()

**绘制频率分布图**：fdist1.plot()