# NLTK 自然语言处理

NLTK 是构建 Python 程序以处理人类语言数据的领先平台。它为超过 50 种语料库和词汇资源（如 WordNet）提供了易于使用的界面，以及用于分类，标记化，词干化，标记，分析和语义推理的一套文本处理库，用于工业强度 NLP 库的包装器，和一个活跃的社区论坛。

由于在计算语言学方面引入了编程基础知识以及综合 API 文档的实践指南，NLTK 适用于语言学家，工程师，学生，教育工作者，研究人员和行业用户等。NLTK 适用于 Windows，Mac OS X 和 Linux。最重要的是，NLTK 是一个免费的，开源的社区驱动项目。

NLTK 被称为“使用 Python 进行计算语言学教学和工作的绝佳工具”，以及“神奇的自然语言图书馆”。

## 1. NLTK 安装

### 1.1 安装 nltk

1、下载 NLTK 包 pip install nltk

2、安装好之后可以用 NLTK 做一些简单的事情

对句子进行分词并标注词性：

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning ... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence) # 对句子进行分词
>>> tokens ['At', 'eight', "o'clock", 'on', 'Thursday', 'morning', 'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
>>> tagged = nltk.pos_tag(tokens) #标注词性，tokens 是句子分词后的结果，
>>> tagged[0:6] [('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN')]
```

### 1.2 语料库安装

1、运行 Python，并输入下面的指令

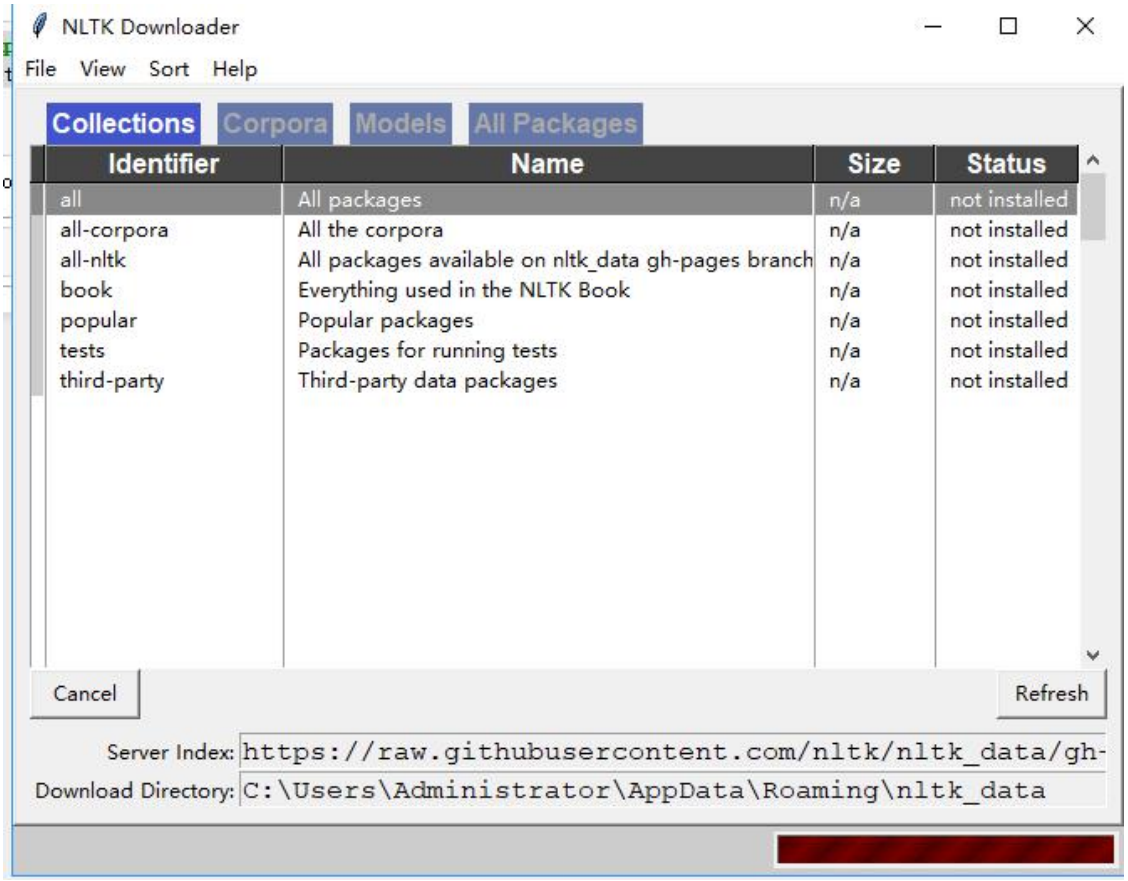
```
import nltk

nltk.download()
```

2、弹出下面的窗口，建议安装所有的包， 双击安装，即 all

或者进行覆盖的方式：具体见教学视频，覆盖文件下载地址：

链接: https://pan.baidu.com/s/1ECC3JHzW8tb0Qqip_ZSP0Q

提取码: jf65




3、测试是否按照成功,所有安装的语料库都在 nltk.corpus 下

```
In [*]: import nltk
        nltk.download()

        showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
```

```
In [12]: from nltk.corpus import brown  #导入brown语料库

         brown.readme()

Out[12]: 'BROWN CORPUS\n\nA Standard Corpus of Present-Day Edited American\nEnglish, for use with Digital Computers.\n\n
         a (1964)\nDepartment of Linguistics, Brown University\nProvidence, Rhode Island, USA\n\nRevised 1971, Revised a
         ww.hit.uib.no/icame/brown/bcm.html\n\nDistributed with the permission of the copyright holder,\nredistribution
```

```
In [13]: brown.words()[:10]   #  查看语料库中前10个单词

Out[13]: ['The',
          'Fulton',
          'County',
          'Grand',
          'Jury',
          'said',
          'Friday',
          'an',
          'investigation',
          'of']
```

```
In [14]: len(brown.words())   # 查看语料库中有多少个单词

Out[14]: 1161192
```

## 1.3 jieba 分词

jieba 分词是 python 写成的一个中文分词开源库

**特点：**

- jieba 支持三种分词模式：
  - 精确模式，试图将句子最精确地切开，适合文本分析；
  - 全模式，把句子中所有的可以成词的词语都扫描出来, 速度非常快，但是不能解决歧义；
  - 搜索引擎模式，在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。
- 支持繁体分词
- 支持自定义词典

**安装**

```
# python2
pip install jieba
```

```
# python3
pip3 install jieba
```

**jieba 三种分词模式示例**

```
import jieba

# 返回列表
seg_list = jieba.cut("我是共产主义接班人，我爱我的祖国", cut_all=True)
print("全模式: " + "/ ".join(seg_list))  # 全模式

seg_list = jieba.cut("我是共产主义接班人，我爱我的祖国", cut_all=False)
print("精确模式: " + "/ ".join(seg_list))  # 精确模式

seg_list = jieba.cut("我是共产主义接班人，我爱我的祖国")
print("默认模式: " + "/ ".join(seg_list))  # 默认是精确模式

seg_list = jieba.cut_for_search("我是共产主义接班人，我爱我的祖国")
print("搜索引擎模式: " + "/ ".join(seg_list))  # 搜索引擎模式
```

输出结果

```
全模式: 我/ 是/ 共产/ 共产主义/ 主义/ 接班/ 接班人/ / / 我/ 爱/ 我/ 的/ 祖国
精确模式: 我/ 是/ 共产主义/ 接班人/ ，/ 我/ 爱/ 我/ 的/ 祖国
默认模式: 我/ 是/ 共产主义/ 接班人/ ，/ 我/ 爱/ 我/ 的/ 祖国
搜索引擎模式: 我/ 是/ 共产/ 主义/ 共产主义/ 接班/ 接班人/ ，/ 我/ 爱/ 我/ 的/ 祖国
```

**词性标注**

jieba.posseg 为默认词性标注分词器，标注句子分词后每个词的词性。

```
import jieba.posseg as pseg

words = pseg.cut("上海自来水来自海上")
for word, flag in words:
    print('%s %s' % (word, flag))
```

输出：

```
上海 ns
自来水 l
来自 v
海上 s
```

### 1.4 词频分析案例

抓取国务院关于印发 新一代人工智能发展规划的通知，并利用 jieba 分词进行文本分析，统计全文词频。

原文链接 http://www.gov.cn/zhengce/content/2017-07/20/content_5211996.htm

```
import requests
import jieba
# xml 模块提供数据提取
from lxml import etree

# collections 是一个提供了一些集合类的集合模块
# Counter 类是一个简单的计数器，比如统计字符出现的个数
from collections import Counter


def load_page(url):
    """
    抓取新一代人工智能发展规划的通知
    """

    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.86 Safari/537.36"}
    resp = requests.get(url=url, headers=headers)
    resp.encoding = resp.apparent_encoding  # 解决编码问题
    html = resp.text

    # 提取通知文本内容
    selector = etree.HTML(html)
    content_list = selector.xpath('//td[@class="b12c"]//p/text()')
    # 将文本拼接成一个文本
```

```
    text = "\n".join(content_list)

    return text


def word_statistical(text):
    """统计文本的词频"""
    # 利用 jieba 进行分词，返回所有分词后长度大于等于 2 的词的列表
    seg_list = [word for word in jieba.cut(text, cut_all=True) if len(word) >= 2]

    # Counter 将分词后的列表将转化为字典，并统计字符出现的个数
    # 字典的键为字符，值为字符出现的个数，并按个数降序排序
    count_dict = Counter(seg_list)

    # most_common(10)获取字典前十个数据并返回列表，列表的每个元素都是一个包含字符和个数的元组
    for word_count in count_dict.most_common(10):
        # word 为词，count 为个数
        word, count = word_count
        print(word, count)


if __name__ == "__main__":
    text = load_page('http://www.gov.cn/zhengce/content/2017-07/20/content_5211996'
                     '.htm')
    word_statistical(text)
```

输出

```
智能 538
人工 319
人工智能 319
技术 120
发展 101
应用 86
创新 73
基础 63
```

```
社会 61
系统 61
```

## 2. 自然语言处理基本流程

### 2.1、语料库的使用

```
import nltk
from nltk.corpus import brown  # 需要下载 brown 语料库

# 引用布朗大学的语料库

# 查看语料库包含的类别
print(brown.categories())

# 查看 brown 语料库
print('共有{}个句子'.format(len(brown.sents())))
print('共有{}个单词'.format(len(brown.words())))
```

执行结果：

```
['adventure', 'belles_lettres', 'editorial', 'fiction', 'government', 'hobbies', 'humor',
'learned', 'lore', 'mystery', 'news', 'religion', 'reviews', 'romance', 'science_fiction']

共有 57340 个句子
共有 1161192 个单词
```

### 2.2、分词 (tokenize)

- 分词的意义
    - 将句子拆分成 具有语言语义学 上意义的词
- 中英文分词的区别
    - 英文文本，单词之间是以空格作为自然分界符的
    - 中文中没有一个形式上的分界符，分词比英文复杂的多，中文分词需要引用 jieba 强大的中文分词工具
- 分词结束之后，后续操作中英文基本没区别

**英文分词：**

英文分词需要先安装 punkt

```
import nltk

nltk.download('punkt')
# 导入 nltk
import nltk

# 需要进行分词的文本
text = 'Python is a programming language that lets you work quickly and integrate systems more effectively'

# 对文本进行分词
sel_list = nltk.word_tokenize(text)

# 打印分词结果
print(sel_list)
```

分词结果

```
['Python', 'is', 'a', 'programming', 'language', 'that', 'lets', 'you', 'work', 'quickly', 'integrate', 'systems', 'more', 'effectively', '.']
```

**中文分词**

中文分词采用 jieba 它是最好的中文分词工具

安装 jieba

```
pip install jieba
```

简单应用：

```
import jieba

seg_list = jieba.cut('我是共产主义接班人', cut_all=True)

print("全模式: " + "/ ".join(seg_list))  # 全模式
```

```
seg_list = jieba.cut('我是共产主义接班人', cut_all=False)
print("精确模式: " + "/ ".join(seg_list))  # 精确模式
```

输出结果

```
全模式: 我/ 是/ 共产/ 共产主义/ 主义/ 接班/ 接班人
精确模式: 我/ 是/ 共产主义/ 接班人
```

### 2.3、词性归一化

**词干提取（stemming）**

- 词干提取，如将单词 looking, looked 中的 ing, ed 去掉，只保留单词主干
- nltk 中常用的 stemer 有 Portmanteau, SnowballStemmer, LancasterStemmer

示例：

```
# PorterStemmer
from nltk.stem.porter import PorterStemmer

porter_stemmer = PorterStemmer()
print(porter_stemmer.stem('looked'))
print(porter_stemmer.stem('looking'))
```

输出结果

```
look
look
```

```
# SnowballStemmer
from nltk.stem import SnowballStemmer

snowball_stemmer = SnowballStemmer('english')
print(snowball_stemmer.stem('looked'))
print(snowball_stemmer.stem('looking'))
```

输出结果

```
look
look
```

```
# LancasterStemmer
from nltk.stem.lancaster import LancasterStemmer

lancaster_stemmer = LancasterStemmer()
print(lancaster_stemmer.stem('looked'))
print(lancaster_stemmer.stem('looking'))
```

输出结果

```
look
look
```

**词形还原(lemmatization)**

- 词形还原，将单词的各种词形归并成一种形式，是把一个任何形式的语言词汇还原为一般形式（能表达完整语义），如 am, is, are -> be, went->go
- nltk 中使用 WordNetLemmatizer 进行词形还原

```
from nltk.stem import WordNetLemmatizer

# 需要下载 wordnet 语料库

wordnet_lematizer = WordNetLemmatizer()
print(wordnet_lematizer.lemmatize('cats'))
print(wordnet_lematizer.lemmatize('boxes'))
print(wordnet_lematizer.lemmatize('are'))
print(wordnet_lematizer.lemmatize('went'))
```

运行结果：

```
cat
box
are
went
```

指明词性可以更准确地进行 lemma

```
# lemmatize 默认为名词
print(wordnet_lematizer.lemmatize('are', pos='v'))
print(wordnet_lematizer.lemmatize('went', pos='v'))
```

运行结果：

```
be
go
```

### 2.4、词性标注

NLTK 中的词性标注

```
tokens = nltk.word_tokenize() # 分词
nltk.pos_tag(tokens) #标注词性，tokens 是句子分词后的结果，
```

### 2.5、去除停用词

- 为节省存储空间和提高搜索效率，NLP 中会自动过滤掉某些字或词
- 停用词都是人工输入、非自动化生成的，形成停用词表

分类：

- 语言中的功能词，如 the, is…
- 词汇词，通常是使用广泛的词，如 want

NLTK 去除停用词：

```
stopwords.words()
```

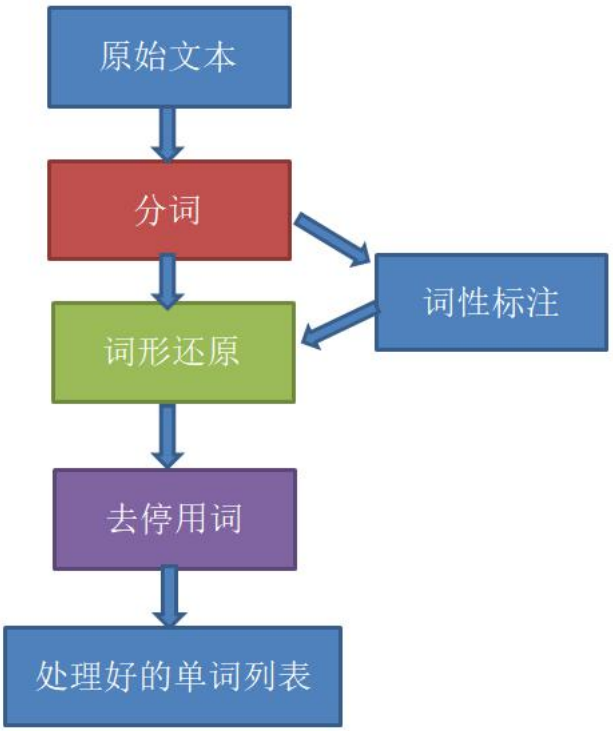
示例：

```
from nltk.corpus import stopwords # 需要下载 stopwords

words = nltk.word_tokenize('Python is a widely used programming language.')

filtered_words = [word for word in words if word not in stopwords.words('english')]
print('原始词：', words)
print('去除停用词后：', filtered_words)
```

输出结果

```
原始词： ['Python', 'is', 'a', 'widely', 'used', 'programming', 'language', '.']
去除停用词后： ['Python', 'widely', 'used', 'programming', 'language', '.']
```

### 2.6、典型的文本处理流程




### 2.7、应用案例

```
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords

# 原始文本 ，你若不离不弃，我必生死相依。
raw_text = 'If  you do not leave me .I will by your side untill the life end.'

# 分词
raw_words = nltk.word_tokenize(raw_text)

# 词形归一化
wordnet_lematizer = WordNetLemmatizer()
```

```
words = [wordnet_lematizer.lemmatize(raw_word) for raw_word in raw_words]

# 去除停用词
filtered_words = [word for word in words if word not in stopwords.words('english')]

print('原始文本：', raw_text)
print('预处理结果：', filtered_words)
```

**输出结果**

```
原始文本： If  you do not leave me .I will by your side untill the life end.
预处理结果： ['If', 'leave', '.I', 'side', 'untill', 'life', 'end', '.']
```