# NLP学习小结

## 1、语料库

### 书籍相关

In [1]:
```
import nltk
```

In [9]:
```
from nltk.corpus import gutenberg
gutenberg.fileids()    # 获取语料库标识符
```

Out[9]:
```
['austen-emma.txt',
 'austen-persuasion.txt',
 'austen-sense.txt',
 'bible-kjv.txt',
 'blake-poems.txt',
 'bryant-stories.txt',
 'burgess-busterbrown.txt',
 'carroll-alice.txt',
 'chesterton-ball.txt',
 'chesterton-brown.txt',
 'chesterton-thursday.txt',
 'edgeworth-parents.txt',
 'melville-moby_dick.txt',
 'milton-paradise.txt',
 'shakespeare-caesar.txt',
 'shakespeare-hamlet.txt',
 'shakespeare-macbeth.txt',
 'whitman-leaves.txt']
```

In [10]:
```
emma = gutenberg.words('austen-emma.txt')  # 获取《爱玛》语料
len(emma)
```

Out[10]:
```
192427
```

```
In [11]: emma = nltk.Text(emma)  # 获取单个文本检索信息对象
         emma.concordance('surprize')
```

```
Displaying 25 of 37 matches:
er father , was sometimes taken by surprize at his being still able to pity `
hem do the other any good ." " You surprize me ! Emma must do Harriet good : a
Knightley actually looked red with surprize and displeasure , as he stood up ,
r . Elton , and found to his great surprize , that Mr . Elton was actually on
d aid ." Emma saw Mrs . Weston ' s surprize , and felt that it must be great ,
father was quite taken up with the surprize of so sudden a journey , and his f
y , in all the favouring warmth of surprize and conjecture . She was , moreove
he appeared , to have her share of surprize , introduction , and pleasure . Th
ir plans ; and it was an agreeable surprize to her , therefore , to perceive t
talking aunt had taken me quite by surprize , it must have been the death of m
f all the dialogue which ensued of surprize , and inquiry , and congratulation
 the present . They might chuse to surprize her ." Mrs . Cole had many to agre
the mode of it , the mystery , the surprize , is more like a young woman ' s s
 to her song took her agreeably by surprize -- a second , slightly but correct
" " Oh ! no -- there is nothing to surprize one at all .-- A pretty fortune ;
t to be considered . Emma ' s only surprize was that Jane Fairfax should accep
of your admiration may take you by surprize some day or other ." Mr . Knightle
ation for her will ever take me by surprize .-- I never had a thought of her i
```

```
In [15]: print(len(gutenberg.raw('austen-emma.txt')))  # 原文
         print(gutenberg.words('austen-emma.txt'))  # 词汇集
         print(gutenberg.sents('austen-emma.txt'))  # 句子
```

```
887071
['[', 'Emma', 'by', 'Jane', 'Austen', '1816', ']', ...]
[['[', 'Emma', 'by', 'Jane', 'Austen', '1816', ']'], ['VOLUME', 'I'], ...]
```

## 网络文本

```
In [18]: from nltk.corpus import webtext  # 网络文本集
         for fileid in webtext.fileids():
             print(fileid, webtext.raw(fileid)[:50])
```

```
firefox.txt Cookie Manager: "Don't allow sites that set remove
grail.txt SCENE 1: [wind] [clop clop clop]
KING ARTHUR: Who
overheard.txt White guy: So, do you have any plans for this even
pirates.txt PIRATES OF THE CARRIBEAN: DEAD MAN'S CHEST, by Ted
singles.txt 25 SEXY MALE, seeks attrac older single lady, for
wine.txt Lovely delicate, fragrant Rhone wine. Polished lea
```

## 即时聊天回话

```
In [19]: from nltk.corpus import nps_chat  # 即时聊天回话预料库
         chatroom = nps_chat.posts('10-19-20s_706posts.xml')
         chatroom[123]
```

```
Out[19]: ['i',
          'do',
          "n't",
          'want',
          'hot',
          'pics',
          'of',
          'a',
          'female',
          ',',
          'I',
          'can',
          'look',
          'in',
          'a',
          'mirror',
          '.']
```

## 布朗语料库

```
In  [20]: from nltk.corpus import brown  # 布朗预料库
          brown.categories()
```

```
Out[20]: ['adventure',
          'belles_lettres',
          'editorial',
          'fiction',
          'government',
          'hobbies',
          'humor',
          'learned',
          'lore',
          'mystery',
          'news',
          'religion',
          'reviews',
          'romance',
          'science_fiction']
```

```
In  [21]: brown.words(categories='news')
```

```
Out[21]: ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
```

```
In  [22]: brown.words(fileids=['cg22'])
```

```
Out[22]: ['Does', 'our', 'society', 'have', 'a', 'runaway', ',', ...]
```

```
In  [23]: brown.sents(categories=['news', 'editorial', 'reviews'])
```

```
Out[23]: [['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'investigation',
          'of', "Atlanta's", 'recent', 'primary', 'election', 'produced', '``', 'no', 'evidence',
          "''", 'that', 'any', 'irregularities', 'took', 'place', '.'], ['The', 'jury', 'further
          ', 'said', 'in', 'term-end', 'presentments', 'that', 'the', 'City', 'Executive', 'Commi
          ttee', ',', 'which', 'had', 'over-all', 'charge', 'of', 'the', 'election', ',', '``', '
          deserves', 'the', 'praise', 'and', 'thanks', 'of', 'the', 'City', 'of', 'Atlanta',
          "''", 'for', 'the', 'manner', 'in', 'which', 'the', 'election', 'was', 'conducted',
          '.'], ...]
```

## 路透社语料库

```
In  [33]: from nltk.corpus import reuters  # 路透社预料库
          reuters.fileids()[:10]
```

```
Out[33]: ['test/14826',
          'test/14828',
          'test/14829',
          'test/14832',
          'test/14833',
          'test/14839',
          'test/14840',
          'test/14841',
          'test/14842',
          'test/14843']
```

```
In  [36]: reuters.categories()[:20]
```

```
Out[36]: ['acq',
          'alum',
          'barley',
          'bop',
          'carcass',
          'castor-oil',
          'cocoa',
          'coconut',
          'coconut-oil',
          'coffee',
          'copper',
          'copra-cake',
          'corn',
          'cotton',
          'cotton-oil',
          'cpi',
          'cpu',
          'crude',
          'dfl',
          'dlr']
```

```
In  [38]: reuters.categories(['training/9865', 'test/14826'])  # 获取指定集的类别信息
Out[38]: ['barley', 'corn', 'grain', 'trade', 'wheat']
```

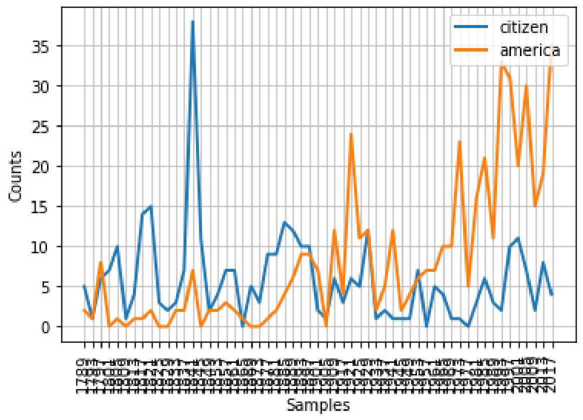
```
In  [41]: reuters.fileids(['barley', 'corn'])[:10]  # 指定类别查询语料集
Out[41]: ['test/14832',
          'test/14858',
          'test/15033',
          'test/15043',
          'test/15106',
          'test/15287',
          'test/15341',
          'test/15618',
          'test/15648',
          'test/15649']
```

```
In  [42]: reuters.words('training/9865')[:15]  # 开头大写的是题目
Out[42]: ['FRENCH',
          'FREE',
          'MARKET',
          'CEREAL',
          'EXPORT',
          'BIDS',
          'DETAILED',
          'French',
          'operators',
          'have',
          'requested',
          'licences',
          'to',
          'export',
          '675']
```

```
In  [44]: reuters.words(['training/9865', 'training/9880'])[:15]
Out[44]: ['FRENCH',
          'FREE',
          'MARKET',
          'CEREAL',
          'EXPORT',
          'BIDS',
          'DETAILED',
          'French',
          'operators',
          'have',
          'requested',
          'licences',
          'to',
          'export',
          '675']
```

```
In  [47]: reuters.words(categories=['barley'])[:15]
Out[47]: ['FRENCH',
          'FREE',
          'MARKET',
          'CEREAL',
          'EXPORT',
          'BIDS',
          'DETAILED',
          'French',
          'operators',
          'have',
          'requested',
          'licences',
          'to',
          'export',
          '320']
```

## 就职演说

In [49]:
```
from nltk.corpus import inaugural  # 就职演说预料库
inaugural.fileids()[:10]
```
Out[49]:
```
['1789-Washington.txt',
 '1793-Washington.txt',
 '1797-Adams.txt',
 '1801-Jefferson.txt',
 '1805-Jefferson.txt',
 '1809-Madison.txt',
 '1813-Madison.txt',
 '1817-Monroe.txt',
 '1821-Monroe.txt',
 '1825-Adams.txt']
```
In [51]:
```
[fileid[:4] for fileid in inaugural.fileids()][:10]  # 获取时间
```
Out[51]:
```
['1789',
 '1793',
 '1797',
 '1801',
 '1805',
 '1809',
 '1813',
 '1817',
 '1821',
 '1825']
```
In [54]:
```
# 绘制不同词在随时间演讲时的变换
cfd = nltk.ConditionalFreqDist(
    (target, fileid[:4]) for fileid in inaugural.fileids()
                             for w in inaugural.words(fileid)
                                 for target in ['america', 'citizen']
                                     if w.lower().startswith(target))
cfd.plot()
```


Out[54]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x20062ed7040>
```

## 使用自己的语料库

In [77]:
```
from nltk.corpus import PlaintextCorpusReader  # 导入自己的预料库
corpus_root = '.'
wordlists = PlaintextCorpusReader(corpus_root, ".*")
wordlists.fileids()
```
Out[77]:
```
['.ipynb_checkpoints/获得文本语料和词汇资料-checkpoint.ipynb',
 '.ipynb_checkpoints/语言处理与Python-checkpoint.ipynb',
 'my_text.txt',
 '获得文本语料和词汇资料.ipynb',
 '语言处理与Python.ipynb']
```
In [80]:
```
wordlists.words('my_text.txt')  # 使用
```
Out[80]:
```
['I', 'am', 'hnuer', ',', 'and', 'you', '?', 'Me', ',', ...]
```

```
from urllib.request import urlopen

```

```
# 访问本地文本可以直接使用```open('xxx').read()```来获取raw
url = "http://www.gutenberg.org/files/2554/2554-0.txt"
raw = urlopen(url).read()
raw = str(raw)

tokens = nltk.word_tokenize(raw)  # 生成token列表
nltk.tokenwrap(raw)  # 生成token字符串

text = nltk.Text(tokens)
text.collocations()  # 检测高频双连词

raw.find("PART I")
raw.rfind("the subject of a new story")

raw = raw[5866: 1338204]
raw.find("PART I")

raw = raw[5866: 1338204]
raw.find("the subject of a new story")

'''
需要清除html情况
'''
url = "http://www.gutenberg.org/files/2554/2554-h/2554-h.htm"
html = urlopen(url).read()

from bs4 import BeautifulSoup

raw = BeautifulSoup(html).get_text()
tokens = nltk.word_tokenize(raw)

text = nltk.Text(tokens)
text.concordance('forbidden')  # 检索指定词

'''
读取NLTK语料库文件
'''
path = nltk.data.find('corpora/gutenberg/melville-moby_dick.txt')
raw = open(path, 'r').read()
```

## 2、词性标注器

### 默认标注器

（以词频为依据进行标注）

```
In  [87]: brown_tagged_sents = brown.tagged_sents(categories='news')
          brown_sents = brown.sents(categories='news')
```

```
In  [88]: tags = [tag for (word, tag) in brown.tagged_words(categories='news')]
          nltk.FreqDist(tags).max()

Out[88]: 'NN'
```

```
In  [89]: raw = 'i do not like green eggs and ham, i do not like them sam i am!'
          tokens = nltk.word_tokenize(raw)
          default_tagger = nltk.DefaultTagger('NN')
          default_tagger.tag(tokens)

Out[89]: [('i', 'NN'),
          ('do', 'NN'),
          ('not', 'NN'),
          ('like', 'NN'),
          ('green', 'NN'),
          ('eggs', 'NN'),
          ('and', 'NN'),
          ('ham', 'NN'),
          (',', 'NN'),
          ('i', 'NN'),
          ('do', 'NN'),
          ('not', 'NN'),
          ('like', 'NN'),
          ('them', 'NN'),
          ('sam', 'NN'),
          ('i', 'NN'),
          ('am', 'NN'),
          ('!', 'NN')]
```

```
In  [90]: default_tagger.evaluate(brown_tagged_sents)

Out[90]: 0.13089484257215028
```

## 正则表达式标注器

（正则匹配为依据进行标注）

In [91]:
```
patterns = [
    (r'.*ing$', 'VBG'),
    (r'.*ed$', 'VBD'),
    (r'.*es$', 'VBZ'),
    (r'.*ould$', 'MD'),
    (r'.*\'s$', 'NN$'),
    (r'.*s$', 'NNS'),
    (r'^-?[0-9]+(.[0-9]+)?$', 'CD'),
    (r'.*', 'NN')
]

regexp_tagger = nltk.RegexpTagger(patterns)
regexp_tagger.tag(brown_sents[3])
```

Out[91]:
```
[('``', 'NN'),
 ('Only', 'NN'),
 ('a', 'NN'),
 ('relative', 'NN'),
 ('handful', 'NN'),
 ('of', 'NN'),
 ('such', 'NN'),
 ('reports', 'NNS'),
 ('was', 'NNS'),
 ('received', 'VBD'),
 ("''", 'NN'),
 (',', 'NN'),
 ('the', 'NN'),
 ('jury', 'NN'),
 ('said', 'NN'),
 (',', 'NN'),
 ('``', 'NN'),
 ('considering', 'VBG'),
 ('the', 'NN'),
```

In [92]:
```
regexp_tagger.evaluate(brown_tagged_sents)
```

Out[92]:
```
0.20326391789486245
```

## 查询标注器

（以高频词为依据标注，不匹配则标注为空）

```
In  [94]: fd = nltk.FreqDist(brown.words(categories='news'))  # 统计词频
          cfd = nltk.ConditionalFreqDist(brown.tagged_words(categories='news'))  # 词 - 词性 统计

          most_freq_words = list(fd.keys())[:100]  # 最高频的前100词
          likely_tags = dict((word, cfd[word].max()) for word in most_freq_words)  # 高频词及高频词

          baseline_tagger = nltk.UnigramTagger(model=likely_tags)

          baseline_tagger.evaluate(brown_tagged_sents)
```

```
Out[94]: 0.3329355371243312
```

```
In  [101]: sent = brown.sents(categories='news')[3]
           baseline_tagger.tag(sent)
```

```
Out[101]: [('``', '``'),
           ('Only', 'RB'),
           ('a', 'AT'),
           ('relative', 'JJ'),
           ('handful', 'NN'),
           ('of', 'IN'),
           ('such', 'JJ'),
           ('reports', 'NNS'),
           ('was', 'BEDZ'),
           ('received', 'VBD'),
           ("''", "''"),
           (',', ','),
           ('the', 'AT'),
           ('jury', 'NN'),
           ('said', 'VBD'),
           (',', ','),
           ('``', '``'),
           ('considering', 'IN'),
           ('the', 'AT'),
```

```
In  [102]: # 从上可知有些词被标注为空，因为我们需要将未查到的词性标注为默认的
           baseline_tagger = nltk.UnigramTagger(model=likely_tags, backoff=nltk.DefaultTagger('NN'))
```

## N-gram

（挑选上下文中最可能的词性进行标注；一元以该词词频为准，其他以前N-1词性为准，存在数据稀疏问题）

```
In  [107]: brown_tagged_sents = brown.tagged_sents(categories='news')
           brown_sents = brown.sents(categories='news')

           unigram_tagger = nltk.UnigramTagger(brown_tagged_sents)  # 一元标注
           unigram_tagger.tag(brown_sents[2007])
```

```
Out[107]: [('Various', 'JJ'),
           ('of', 'IN'),
           ('the', 'AT'),
           ('apartments', 'NNS'),
           ('are', 'BER'),
           ('of', 'IN'),
           ('the', 'AT'),
           ('terrace', 'NN'),
           ('type', 'NN'),
           (',', ','),
           ('being', 'BEG'),
           ('on', 'IN'),
           ('the', 'AT'),
           ('ground', 'NN'),
           ('floor', 'NN'),
           ('so', 'QL'),
           ('that', 'CS'),
           ('entrance', 'NN'),
           ('is', 'BEZ'),
           ('direct', 'JJ'),
           ('.', '.')]
```

```
In  [108]: unigram_tagger.evaluate(brown_tagged_sents)
```

```
Out[108]: 0.9349006503968017
```

```
In  [111]: bigram_tagger = nltk.BigramTagger(train_sents)  # 二元标注（未出现的不会统计）
           bigram_tagger.tag(brown_sents[2007])
```

```
Out[111]: [('Various', 'JJ'),
           ('of', 'IN'),
           ('the', 'AT'),
           ('apartments', 'NNS'),
           ('are', 'BER'),
           ('of', 'IN'),
           ('the', 'AT'),
           ('terrace', 'NN'),
           ('type', 'NN'),
           (',', ','),
           ('being', 'BEG'),
           ('on', 'IN'),
           ('the', 'AT'),
           ('ground', 'NN'),
           ('floor', 'NN'),
           ('so', 'CS'),
           ('that', 'CS'),
           ('entrance', 'NN'),
           ('is', 'BEZ'),
```

```
In  [112]: unseen_sent = brown_sents[4203]
           bigram_tagger.tag(unseen_sent)
```

```
Out[112]: [('The', 'AT'),
           ('population', 'NN'),
           ('of', 'IN'),
           ('the', 'AT'),
           ('Congo', 'NP'),
           ('is', 'BEZ'),
           ('13.5', None),
           ('million', None),
           (',', None),
           ('divided', None),
           ('into', None),
           ('at', None),
           ('least', None),
           ('seven', None),
           ('major', None),
           ('``', None),
           ('culture', None),
           ('clusters', None),
           ("''", None),
```

## 组合标注器

（类似于if-else形式来选定标注器）

```
In  [114]: t0 = nltk.DefaultTagger('NN')
           t1 = nltk.UnigramTagger(train_sents, backoff=t0)
           t2 = nltk.BigramTagger(train_sents, backoff=t1)
           t2.evaluate(test_sents)

Out[114]: 0.8452108043456593
```

```
In  [120]: # 将会丢弃那些只看到一次或两次的上下文
           t2 = nltk.BigramTagger(train_sents, cutoff=2, backoff=t1)
           t2.evaluate(test_sents)

Out[120]: 0.8424200139539519
```

## Brill基于转换的标注

（猜词性，然后根据转换规则进行修改）

```
In  [190]: from nltk.tbl import demo as brill
           brill.demo()
```

```
Loading tagged data from treebank...
Read testing data (200 sents/5251 wds)
Read training data (800 sents/19933 wds)
Read baseline data (800 sents/19933 wds) [reused the training set]
Trained baseline tagger
    Accuracy on test set: 0.8366
Training tbl tagger...
TBL train (fast) (seqs: 800; tokens: 19933; tpls: 24; min score: 3; min acc: None)
Finding initial useful rules...
    Found 12799 useful rules.

           B      |
   S   F   r   O  |        Score = Fixed - Broken
   c   i   o   t  |  R     Fixed = num tags changed incorrect -> correct
   o   x   k   h  |  u     Broken = num tags changed correct -> incorrect
   r   e   e   e  |  l     Other = num tags changed incorrect -> incorrect
   e   d   n   r  |  e
------------------+-------------------------------------------------------
  23  23   0   0  | POS->VBZ if Pos:PRP@[-2,-1]
  18  19   1   0  | NN->VB if Pos:-NONE-@[-2] & Pos:TO@[-1]
  14  14   0   0  | VBP->VB if Pos:MD@[-2,-1]
  12  12   0   0  | VBP->VB if Pos:TO@[-1]
  11  11   0   0  | VBD->VBN if Pos:VBD@[-1]
  11  11   0   0  | IN->WDT if Pos:-NONE-@[1] & Pos:VBP@[2]
  10  11   1   0  | VBN->VBD if Pos:PRP@[-1]
   9  10   1   0  | VBD->VBN if Pos:VBZ@[-1]
   8   8   0   0  | NN->VB if Pos:MD@[-1]
   7   7   0   1  | VB->NN if Pos:DT@[-1]
   7   7   0   0  | VB->VBP if Pos:PRP@[-1]
   7   7   0   0  | IN->WDT if Pos:-NONE-@[1] & Pos:VBZ@[2]
   7   8   1   0  | IN->RB if Word:as@[2]
   6   6   0   0  | VBD->VBN if Pos:VBP@[-2,-1]
   6   6   0   1  | IN->WDT if Pos:-NONE-@[1] & Pos:VBD@[2]
   5   5   0   0  | POS->VBZ if Pos:-NONE-@[-1]
   5   5   0   0  | VB->VBP if Pos:NNS@[-1]
   5   5   0   0  | VBD->VBN if Word:be@[-2,-1]
   4   4   0   0  | POS->VBZ if Pos:``@[-2]
   4   4   0   0  | VBP->VB if Pos:VBD@[-2,-1]
   4   6   2   3  | RP->RB if Pos:CD@[1,2]
   4   4   0   0  | RB->JJ if Pos:DT@[-1] & Pos:NN@[1]
   4   4   0   0  | NN->VBP if Pos:NNS@[-2] & Pos:RB@[-1]
   4   5   1   0  | VBN->VBD if Pos:NNP@[-2] & Pos:NNP@[-1]
   4   4   0   0  | IN->WDT if Pos:-NONE-@[1] & Pos:MD@[2]
   4   8   4   0  | VBD->VBN if Word:*@[1]
```

## 存储与使用

```
In  [122]: from pickle import dump
           from pickle import load
```

```
In  [123]: output = open('t2.pkl', 'wb')
           dump(t2, output, -1)
           output.close()
```

```
In  [125]: inputs = open('t2.pkl', 'rb')
           tagger = load(inputs)
           inputs.close()
```

```
In  [126]: text = '''
           One notable effort in increasing the interoperability of biomedical ontologies has been t
           '''
           tokens = text.split()
           tagger.tag(tokens)
```

```
Out[126]: [('One', 'CD'),
           ('notable', 'JJ'),
           ('effort', 'NN'),
           ('in', 'IN'),
           ('increasing', 'VBG'),
           ('the', 'AT'),
           ('interoperability', 'NN'),
           ('of', 'IN'),
           ('biomedical', 'NN'),
           ('ontologies', 'NN'),
           ('has', 'HVZ'),
           ('been', 'BEN'),
           ('the', 'AT'),
           ('creation', 'NN'),
           ('of', 'IN'),
           ('logical', 'JJ'),
           ('definitions[71].', 'NN'),
           ('This', 'DT'),
           ('is', 'BEZ'),
           ('an', 'AT'),
           ('initiative', 'NN')]
```

## 3、文本分类

- 编写特征提取规则，生成特征字典：（特征，标签）
- 使用指定分类器进行训练

# 性别鉴定

In [1]:
```python
import nltk
import random
from nltk.corpus import names
```

In [2]:
```python
# 提取特征
def gender_features(word):
    return {'last_letter': word[-1]}
```

In [3]:
```python
name_set = ([(name, 'male') for name in names.words('male.txt')] +
            [(name, 'female') for name in names.words('female.txt')])
random.shuffle(name_set)
```

In [4]:
```python
featuresets = [(gender_features(n), g) for (n, g) in name_set]
train_set, test_set = featuresets[500:], featuresets[:500]

classifier = nltk.NaiveBayesClassifier.train(train_set)
```

In [5]:
```python
classifier.classify(gender_features('Neo'))
```

Out[5]:
```
'male'
```

In [6]:
```python
classifier.classify(gender_features('Trinity'))
```

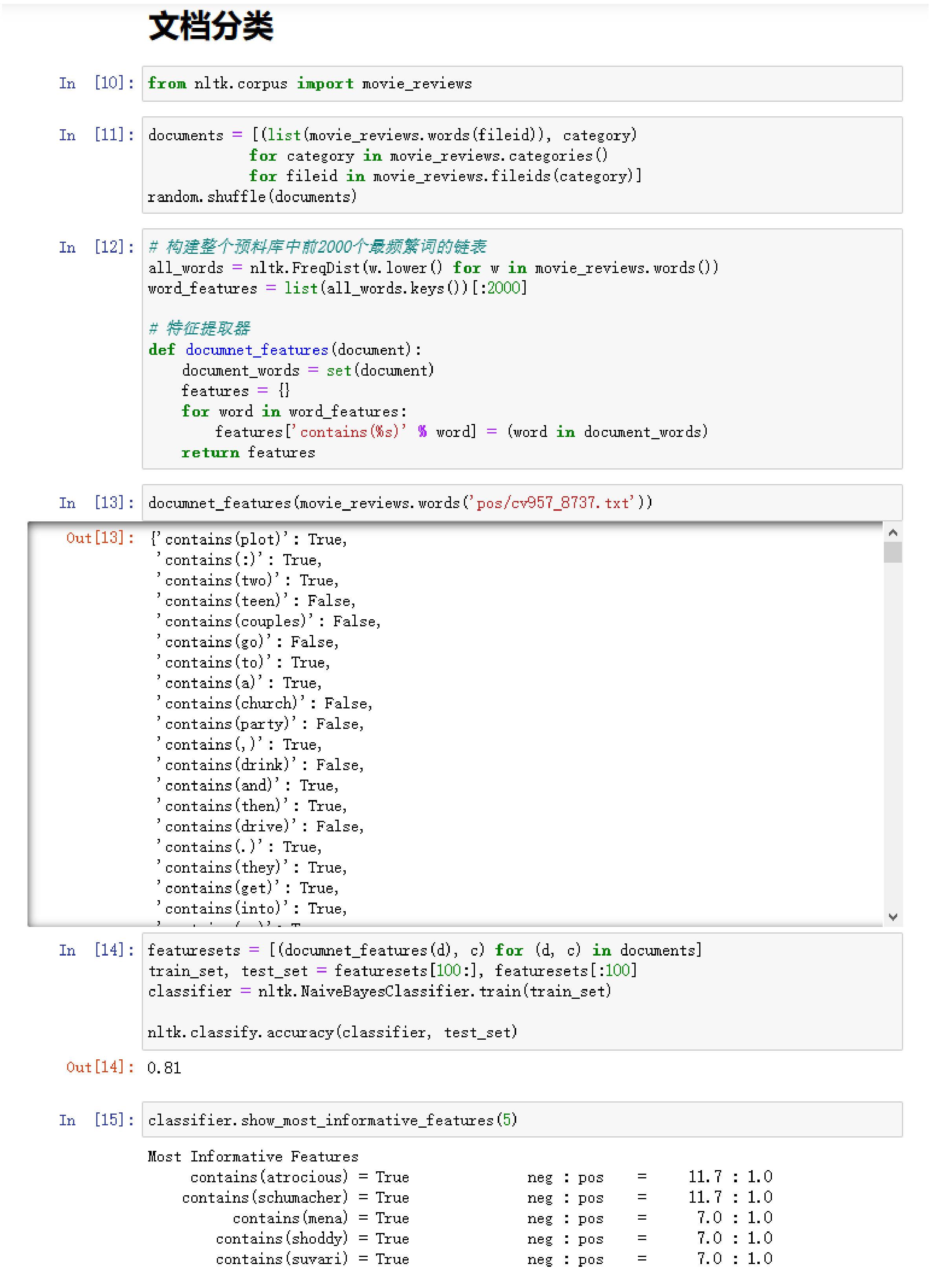
Out[6]:
```
'female'
```

In [7]:
```python
nltk.classify.accuracy(classifier, test_set)
```

Out[7]:
```
0.76
```

In [8]:
```python
# 显示的比率为似然比，用于比较不同特征-结果关系
classifier.show_most_informative_features(5)
```

```
Most Informative Features
             last_letter = 'a'            female : male   =     34.6 : 1.0
             last_letter = 'k'              male : female =     31.7 : 1.0
             last_letter = 'f'              male : female =     28.8 : 1.0
             last_letter = 'p'              male : female =     12.5 : 1.0
             last_letter = 'd'              male : female =      9.4 : 1.0
```



## 文档分类

```
In [10]: from nltk.corpus import movie_reviews
```

```
In [11]: documents = [(list(movie_reviews.words(fileid)), category)
                      for category in movie_reviews.categories()
                      for fileid in movie_reviews.fileids(category)]
         random.shuffle(documents)
```

```
In [12]: # 构建整个预料库中前2000个最频繁词的链表
         all_words = nltk.FreqDist(w.lower() for w in movie_reviews.words())
         word_features = list(all_words.keys())[:2000]

         # 特征提取器
         def documnet_features(document):
             document_words = set(document)
             features = {}
             for word in word_features:
                 features['contains(%s)' % word] = (word in document_words)
             return features
```

```
In [13]: documnet_features(movie_reviews.words('pos/cv957_8737.txt'))

Out[13]: {'contains(plot)': True,
          'contains(:)': True,
          'contains(two)': True,
          'contains(teen)': False,
          'contains(couples)': False,
          'contains(go)': False,
          'contains(to)': True,
          'contains(a)': True,
          'contains(church)': False,
          'contains(party)': False,
          'contains(,)': True,
          'contains(drink)': False,
          'contains(and)': True,
          'contains(then)': True,
          'contains(drive)': False,
          'contains(.)': True,
          'contains(they)': True,
          'contains(get)': True,
          'contains(into)': True,
```

```
In [14]: featuresets = [(documnet_features(d), c) for (d, c) in documents]
         train_set, test_set = featuresets[100:], featuresets[:100]
         classifier = nltk.NaiveBayesClassifier.train(train_set)

         nltk.classify.accuracy(classifier, test_set)

Out[14]: 0.81
```

```
In [15]: classifier.show_most_informative_features(5)

         Most Informative Features
              contains(atrocious) = True              neg : pos    =     11.7 : 1.0
             contains(schumacher) = True              neg : pos    =     11.7 : 1.0
                   contains(mena) = True              neg : pos    =      7.0 : 1.0
                 contains(shoddy) = True              neg : pos    =      7.0 : 1.0
                 contains(suvari) = True              neg : pos    =      7.0 : 1.0
```

```
# 在处理大型预料库时，构建一个包含每一个实例的特征的单独的链表会使用大量的内存。
# 下述方式不会在内容中存储所有的特征集对象
from nltk.classify import apply_features
train_set = apply_features(gender_features, name_set[500:])
test_set = apply_features(gender_features, name_set[:500])
```

## 4、NLP中文数据集

GITHUB：https://github.com/InsaneLife/ChineseNLPCorpus

- 任务型对话（语音+文本理解）
- 文本分类（新闻、评论、微博等）
- 实体识别/标注（微博、人民日报、微软亚洲研究院数据集等）
- 搜索匹配
- 推荐系统（电影、餐馆、商品等）
- 百科数据（百度百科、维基百科）
- 指代消岐
- 完形填空
- 中华诗词
- 保险行业
- 汉语拆字字典
- 预训练模型BERT及词向量
- NLP工具
  - THULAC：中文分词和标注工具
  - HanLP：多语种分词、标注、实体识别、依存文法分析，提供TF2训练模型
  - LTP4：中文分词、标注、实体识别、依存文法分析和语义角色等
  - NLPIR：无说明文档（含系统设计论文等理论资料）
  - Jieba：中文分词