作业三 朴素贝叶斯情感分类
任务描述：利用nltk语料库中的影评来进行朴素贝叶斯情感分类训练。
影评导入：from nltk.corpus import movie_reviews
文件具体目录在…\nltk_data\corpora\movie_reviews，已做好分类标注，消极与积极影评各1000条。
步骤：经过文本预处理（去噪、分句、分词、去停词、取词干、修剪）和特征选择，生成特征词表，之后利用朴素贝叶斯模型进行训练。（每个步骤最好注释一下）
选择前80%（即前800条消极影评与前800条积极影评）作为训练集，后20%作为测试集。
输出：准确率Accuracy，精确率Precision，召回率Recall和F1值，精确到小数点后两位。其中，F1 = (2 * Precision * Recall) / (Precision + Recall)。
例如：
Accuracy = 0.98
Precision = 0.67
Recall = 0.32
F1 = 0.43

Part1 库的导入

In [75]:

```
import os
import re
import nltk
import nltk.tokenize as tk
import nltk.stem as ns
from nltk.corpus import stopwords
from nltk.corpus import movie_reviews
from collections import Counter
```

Part2 文件的读取

In [76]:

```
path = str(movie_reviews)   #得到movie_reviews的路径，但还包含其他杂项信息
start = path.find("'")      #其中路径被包含在两个引号之间，因此先找到第一个引号的下标
end = path.rfind("'")       #再找到下一个引号的下标
path = path[start+1:end]    #然后取子串
pos_path = path + "\\\\pos" #添加pos为积极影评的路径
neg_path = path + "\\\\neg" #添加neg为消极影评的路径

pos_comments = []           #存储所有积极评论的文本
neg_comments = []           #存储所有消极评论的文本

for root,dirs,files in os.walk(pos_path):   #循环目录
    for file in files:                      #循环所有文件
        fileName = root+"\\\\"+file
        f = open(fileName,'r')              #获取文本的绝对路径
        pos_comments.append(f.read())       #读取其内容并存储到对应数组中

for root,dirs,files in os.walk(neg_path):  #同上
    for file in files:
        fileName = root+"\\\\"+file
        f = open(fileName,'r')
        neg_comments.append(f.read())
#print(pos_comments)
#print(neg_comments)
```

Part3 文本处理函数
参照作业2的文本处理办法，将800条积极影评/消极影评的文本进行文本处理
与作业2不同的是，并不是800条影评的词汇全部加在一起之后进行文本处理
而是对每一条进行单独处理，之后再合并存储，目的是方便删去低频词
在此基础上，以出现频率为特征进行特征选择
并删去低频词，选取排名前20的单词作为函数的返回值，返回类型为Counter()

In [117]:

```
def TextProcess(text):
    #Step 1 Tokenization:
    Tokenization = tk.word_tokenize(text) #首先逐个拆分

    #Step 2 Normalization:
    pattern=re.compile("[^a-zA-Z0-9\n ]")       #数字字符的正则匹配
    Normalization = []
    for e in Tokenization:
        e = re.sub(pattern,"",e).lower()       #将所有非数字字符的符号转化为空，大小写转换
        e = tk.word_tokenize(e)                #文本标记化/分词
        e = [w for w in e if(w not in stopwords.words('english'))]  #去停用词
        Normalization += e                     #t添加到输出

    #Step3 Stemming:                           #词干提取
    Stemming = []
    for token in Normalization:
        pt_stem= nltk.stem.porter.PorterStemmer().stem(token)
        Stemming.append(token)

    #Step 4 Lemmatization:
    Lemmatization = []
    lemmatizer = ns.WordNetLemmatizer()
    for token in Stemming:
        Lemmatization.append(lemmatizer.lemmatize(token))

    result = Counter(Lemmatization).most_common(20)    #取最高词频的20个
    return result
```

Part4 训练集的训练，对于每一个数据
获取其文本处理函数的返回值，然后依次添加到字典中
其中pos_dict为积极影评的字典，key为单词，value为出现次数
neg_dict则是消极影评的字典，key为单词，value为出现次数
因为是字典的建立需要大量时间，所以整个训练过程花费时间较多，还请耐心等待

In [126]:

```
pos_dict = {}
print("训练可能要花费较长时间，还请耐心等待")
for i in range(800):
    counter = TextProcess(pos_comments[i])  #
    if i%50 == 0:
        print("积极影评已经训练到第%d个数据"%i)
    for e in counter:
        if e[0] in pos_dict:
            pos_dict[e[0]] += e[1]
        else:
            pos_dict[e[0]] = e[1]
neg_dict = {}
for i in range(800):
    if i%50 == 0:
        print("消极影评已经训练到第%d个数据"%i)
    counter = TextProcess(neg_comments[i])
    for e in counter:
        if e[0] not in neg_dict:
            neg_dict[e[0]] = e[1]
        else:
            neg_dict[e[0]] += e[1]
print("训练完成")
```

```
训练可能要花费较长时间，还请耐心等待
积极影评已经训练到第0个数据
积极影评已经训练到第50个数据
积极影评已经训练到第100个数据
积极影评已经训练到第150个数据
积极影评已经训练到第200个数据
积极影评已经训练到第250个数据
积极影评已经训练到第300个数据
积极影评已经训练到第350个数据
积极影评已经训练到第400个数据
积极影评已经训练到第450个数据
积极影评已经训练到第500个数据
积极影评已经训练到第550个数据
积极影评已经训练到第600个数据
积极影评已经训练到第650个数据
积极影评已经训练到第700个数据
积极影评已经训练到第750个数据
消极影评已经训练到第0个数据
消极影评已经训练到第50个数据
消极影评已经训练到第100个数据
消极影评已经训练到第150个数据
消极影评已经训练到第200个数据
消极影评已经训练到第250个数据
消极影评已经训练到第300个数据
消极影评已经训练到第350个数据
消极影评已经训练到第400个数据
消极影评已经训练到第450个数据
消极影评已经训练到第500个数据
消极影评已经训练到第550个数据
消极影评已经训练到第600个数据
消极影评已经训练到第650个数据
消极影评已经训练到第700个数据
消极影评已经训练到第750个数据
训练完成
```

Part5 测试集的检验
C1为积极影评字典中的单词总数，V1为积极影评字典的规模
C2位消极影评字典中的单词总数，V2位消极影评字典的规模
检验方法为:
对于测试集的每个影评，首先通过文本处理函数获取其最高的20个单词以及其出现次数
然后对于每个单词，计算其相应的概率
P1表示其为积极影评的概率
P2表示其为消极影评的概率
在计算的过程中不断更新
最后通过比较P1和P2的大小来增加TP，FN，FP，TN的值

In [131]:

```
C1 = 0
C2 = 0
V1 = len(pos_dict)
V2 = len(neg_dict)
for e in pos_dict:
    C1 += pos_dict[e]
for e in neg_dict:
    C2 += neg_dict[e]
#首先计算好C1,C2,V1,V2的值

#积极影评的测试
TP = 0
FN = 0
for i in range(800,1000):
    if(i%50 == 0):
        print("已经测试完第%d个积极影评"%(i-800))
    counter = TextProcess(pos_comments[i]) #得到最高频的20个单词的键值对
    P1 = 1
    P2 = 1
    for e in counter:                  #对于每个键值对
        word = e[0]                    #获取单词
        if word not in pos_dict:       #若不在字典中
            val = 1                    #val为1
        else:
            val = pos_dict[word]+1   #否则val = 1+该单词的出现次数
        P1 *= pow(val/(C1+V1),e[1])  #更新P1的值

        if word not in neg_dict:    #同上
            val = 1
        else:
            val = neg_dict[word]+1
        P2 *= pow(val/(C2+V2),e[1])

    #根据P1和P2的大小比较来判断测试结果
    if P1 >= P2:
        TP += 1
    else:
        FN += 1


#消极影评的测试，其步骤同上
FP = 0
TN = 0
for i in range(800,1000):
    counter = TextProcess(neg_comments[i])
    if(i%50 == 0):
        print("已经测试完第%d个消极影评"%(i-800))
    P1 = 1
    P2 = 1
    for e in counter:
        word = e[0]
        if word not in pos_dict:
            val = 1
        else:
            val = pos_dict[word]+1
        P1 *= pow(val/(C1+V1),e[1])

        if word not in neg_dict:
            val = 1
```

```
        else:
            val = neg_dict[word]+1
        P2 *= pow(val/(C2+V2),e[1])

    if P1 <= P2:
        TN += 1
    else:
        FP += 1
print("测试完成")
```

```
已经测试完第0个积极影评
已经测试完第50个积极影评
已经测试完第100个积极影评
已经测试完第150个积极影评
已经测试完第0个消极影评
已经测试完第50个消极影评
已经测试完第100个消极影评
已经测试完第150个消极影评
测试完成
```

In [109]:

```
Part6 结果输出
```

In [132]:

```
print("TP = ",TP,"FP = ",FP)
print("FN = ",FN,"TN = ",TN)
Accuracy = (TP+TN) / (TP+TN+FN+FP)
Precision = TP / (TP+FP)
Recall = TP / (TP+FN)
F1 = (2*Precision*Recall) / (Precision+Recall)
print("Accuracy = ",Accuracy)
print("Precision = ",Precision)
print("Recall = ",Recall)
print("F1 = ",F1)
```

```
TP =  139 FP =  48
FN =  61 TN =  152
Accuracy =  0.7275
Precision =  0.7433155080213903
Recall =  0.695
F1 =  0.7183462532299741
```