TF-IDF 算法和词袋模型的朴素贝叶斯在聊天文本挖掘中的实践

李国夏，16051216

1 问题提出

1.1 问题背景

文本挖掘是一个对具有丰富语义的文本进行分析从而理解其所包含的内容和意义的过程，已经成为数据挖掘中一个日益流行而重要的研究领域。文本挖掘所研究的文本数据库，由来自各种数据源的大量文档组成，包括新闻文章，研究论文，聊天文本等。

1.2 问题的提出

针对聊天文本的挖掘问题的研究，目前来说还是一片空白。本文利用作者与其女朋友的五年内数万条 QQ 消息记录作为数据集，利用 TF-IDF 算法和基于词袋模型的朴素贝叶斯算法进行文本挖掘，以发现该数据集背后隐藏的相关信息和模式。

2 TF-IDF 算法原理

反比文档频数权重评价算法(Term Frequency Inverse Document Frequency, TFIDF)是 Gerald Salton 和 Mc Gill 针对向量空间信息检索范例提出的广泛应用在对文本集合进行特征选取的方法。它依据某个词的词频和其出现过的文本的频率来计算该词在整个文本集中的权重，然后依据权重来进行特征选区。权重越高，说明该词对该文本集合的区分能力越强，否则其区分能力越弱。

聊天文本集可利用其特征向量来表示，即文本被看作是一系列项t的集合。对每个项t可以加上一个对应的权值w,这样文本就可由形如(t,w)的对组成。项$(t_1,t_2,t_3,\dots,t_n)$代表文本内容的特征项，可以看成一个n维的坐标系。权值$(w_1,w_2,w_3,\dots,w_n)$表示文本特征项对应的权重，每个文本集d都可映射成此空间上的一个特征向量：

$$V(d)=(t_1,w_1,t_2,w_2,\dots,t_n,w_n)$$

对于词t和某一文本d来说，t在d中权重的计算公式为：

$$w(d,t)=tf(d,t)*\log\left(\frac{|D|}{df(t)}\right)$$

其中，$tf(d,t)$是词t在文本d中出现的数目，$df(t)$是词t在文本集合中出现过的文本数目,$|D|$是整个文本集合中文本的数目。从上式可以看出，可以根据$w(d,t)$值从大到小选择用户指定数目的词作为某篇文本的特征，生成文本的特征向量。

3　TF-IDF 算法在聊天文本挖掘中的应用

在聊天文本中，人们常常关注双方聊天内容的关键词，即在一段时间内双方经常聊天的主题是什么。此时，可以使用 TF-IDF 算法进行关键词提取。

一个容易想到的思路，就是找到出现次数最多的词。如果某个词很重要，它应该在这篇文章中多次出现。于是，我们进行"词频"（Term Frequency，TF）统计。出现次数最多的词是----"的"、"是"、"在"----这一类最常用的词。它们叫做"停用词"（stop words），表示对找到结果毫无帮助、必须过滤掉的词。假设我们把它们都过滤掉了，只考虑剩下的有实际意义的词。这样又会遇到了另一个问题，我们可能发现某些词的出现次数一样多。这是不是意味着，作为关键词，它们的重要性是一样的？显然不是这样，我们需要一个重要性调整系数，衡量一个词是不是常见词。如果某个词比较少见，但是它在这篇文章中多次出现，那么它很可能就反映了这篇文章的特性，正是我们所需要的关键词。用统计学语言表达，就是在词频的基础上，要对每个词分配一个"重要性"权重。最常见的词（"的"、"是"、"在"）给予最小的权重，较常见的词给予较小的权重，较少见的词给予较大的权重。这个权重叫做"逆文档频率"（Inverse Document Frequency，缩写为 IDF），它的大小与一个词的常见程度成反比。知道了"词频"（TF）和"逆文档频率"（IDF）以后，将这两个值相乘，就得到了一个词的 TF-IDF 值。某个词对文章的重要性越高，它的 TF-IDF 值就越大。所以，排在最前面的几个词，就是这篇文章的关键词。

其算法步骤一般分为以下几步。

Step1：计算词频

$$词频(TF) = \frac{某个词在文章中的出现次数}{文章的总词数}$$

Step2：计算逆文档频率

$$逆文档频率(IDF) = \log(\frac{语料库的文档总数}{包含该词的文档数+1})$$

Step3：计算 TF-IDF

TF-IDF 与一个词在文档中的出现次数成正比，与该词在整个语言中的出现次数成反比。所以，自动提取关键词的算法就很清楚了，就是计算出文档的每个词的 TF-IDF 值，然后按降序排列，取排在最前面的几个词。

jieba 分词这个模块提供上述算法的实现，基于 TF-IDF 算法抽取关键词的主调函数是 TFIDF.extract_tags 函数，主要是在 jieba/analyse/tfidf.py 中实现。其中 TFIDF 是为 TF-IDF 算法抽取关键词所定义的类。类在初始化时，默认加载了分词函数 tokenizer =jieba.dt、词性标注函数 postokenizer = jieba.posseg.dt、停用词 stop_words = self.STOP_WO

RDS.copy()、idf 词典 idf_loader = IDFLoader(idf_path or DEFAULT_IDF)等，并获取 idf 词典及 idf 中值（如果某个词没有出现在 idf 词典中，则将 idf 中值作为这个词的 idf 值）

```
def wordCloud(person='all'):
    if person == 'all':
        content = ''
        for item in data['content']:
            content += (item+' ')
    elif person == 'jean':
        content = ''
        temp_data = data[data['name'] == 'Jean']
        for item in temp_data['content']:
            content += (item+' ')
    else:
        content = ''
        temp_data = data[data['name'] == 'Gordon']
        for item in temp_data['content']:
            content += (item+' ')
    for word in stop_words:
        content = content.replace(word, '')
    tag_list = jieba.analyse.extract_tags(content, topK=50)
    wl = ','.join(tag_list)
    wc = WordCloud(background_color="white",mask=None,max_words=100,font_path
='font.ttf',max_font_size=100,random_state=30,scale=2.5)
    myword = wc.generate(wl)
    plt.imshow(myword)
    plt.axis("off")
```

4 朴素贝叶斯算法简介

我们现在有一个数据集，它由两类数据组成，数据分布如下图所示：

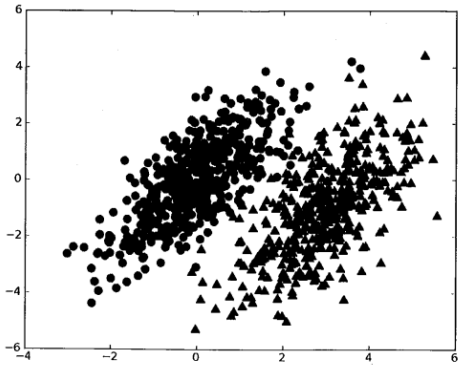


图4-1　两个参数已知的概率分布，参数决定了分布的形状

我们现在用 p1(x,y) 表示数据点 (x,y) 属于类别 1（图中用圆点表示的类别）的概率，用 p2(x,y) 表示数据点 (x,y) 属于类别 2（图中三角形表示的类别）的概率，那么对于一个新数据点 (x,y)，可以用下面的规则来判断它的类别：

- 如果 p1(x,y) > p2(x,y) ，那么类别为 1
- 如果 p2(x,y) > p1(x,y) ，那么类别为 2

也就是说，我们会选择高概率对应的类别。这就是贝叶斯决策理论的核心思想，即选择具有最高概率的决策。

$$p(c_i|x,y) = \frac{p(x,y|c_i)p(c_i)}{p(x,y)}$$

使用上面这些定义，可以定义贝叶斯分类准则为:

- 如果 P(c1|x, y) > P(c2|x, y), 那么属于类别 c1;
- 如果 P(c2|x, y) > P(c1|x, y), 那么属于类别 c2.

5　词袋模型和朴素贝叶斯算法进行文档分类

机器学习的一个重要应用就是文档的自动分类。在文档分类中，整个文档（如一封电子邮件）是实例，而电子邮件中的某些元素则构成特征。我们可以观察文档中出现的词，并把每个词作为一个特征，而每个词的出现或者不出现作为该特征的值，这样得到的特征数目就会跟词汇表中的词的数目一样多。朴素贝叶斯是上面介绍的贝叶斯分类器的一个扩展，是用于文档分类的常用算法。

- **朴素贝叶斯**

提取所有文档中的词条并进行去重
获取文档的所有类别
计算每个类别中的文档数目
对每篇训练文档:
　　对每个类别:
　　　　如果词条出现在文档中-->增加该词条的计数值（for 循环或者矩阵相加）

```
        增加所有词条的计数值（此类别下词条总数）
对每个类别:
    对每个词条:
        将该词条的数目除以总词条数目得到的条件概率（P(词条|类别)）
返回该文档属于每个类别的条件概率（P(类别|文档的所有词条)）
```

- **词袋模型**

我们将每个词的出现与否作为一个特征，这可以被描述为**词集模型(set-of-words model)**。如果一个词在文档中出现不止一次，这可能意味着包含该词是否出现在文档中所不能表达的某种信息，这种方法被称为**词袋模型(bag-of-words model)**。在词袋中，每个单词可以出现多次，而在词集中，每个词只能出现一次。

为适应词袋模型，需要对函数 setOfWords2Vec() 稍加修改，修改后的函数为 bagOfWords2Vec() 。如下给出了基于词袋模型的朴素贝叶斯代码。它与函数 setOfWords2Vec() 几乎完全相同，唯一不同的是每当遇到一个单词时，它会增加词向量中的对应值，而不只是将对应的数值设为 1 。

```
def bagOfWords2VecMN(vocaList, inputSet):
    returnVec = [0] * len(vocabList)
    for word in inputSet:
        if word in vocaList:
            returnVec[vocabList.index(word)] += 1
    return returnVec
```

6 基于词袋模型的朴素贝叶斯算法在聊天文本挖掘中的应用

我们可以基于朴素贝叶斯算法对聊天文本进行训练，从而得到可以通过输入一段文本，预测其说话者是谁的模型。

其预测函数代码主要部分见下文。

```
def prefunc(string):
    testWords = string
    testEntry = " ".join(jieba.cut(testWords)).split()
    thisDoc = np.array(bagOfwords2MN(vocabList,testEntry))
    predict = classifyNB(thisDoc,p0V,p1V,pAb)
    if predict==0:
        print ("这句话是 Jean 说的")
    else:
        print("这句话是 Gordon 说的")
```

7　聊天文本挖掘结果

除了上述通过 TF-IDF 寻找聊天文本出现最多的关键词，以及用朴素贝叶斯算法预测说话者身份，我们**同时挖掘了其他相关信息，下文一并给出**。（注：聊天数据集的双方为：gordon，jean,训练:测试=7:3），**朴素贝叶斯在测试集上的表现为：error_rate:0.23240262787423746**

聊天总词频

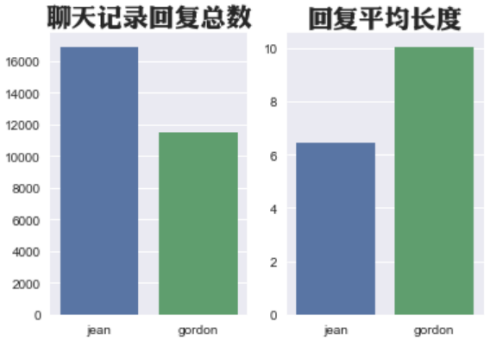
```
wordCloud()
```

Jean的总词频

```
wordCloud('jean')
```

Gordon的总词频

```
wordCloud('gordon')
```




```
In [367]: prefunc('卧槽')
```

这句话是Gordon说的

```
In [368]: prefunc('大佬')
```

这句话是Jean说的

```
In [370]: prefunc('学习')
```

这句话是Gordon说的

```
In [371]: prefunc('晚安')
```

这句话是Jean说的

```
In [372]: prefunc('拜拜')
```

这句话是Jean说的

```
In [374]: prefunc('GPA')
```

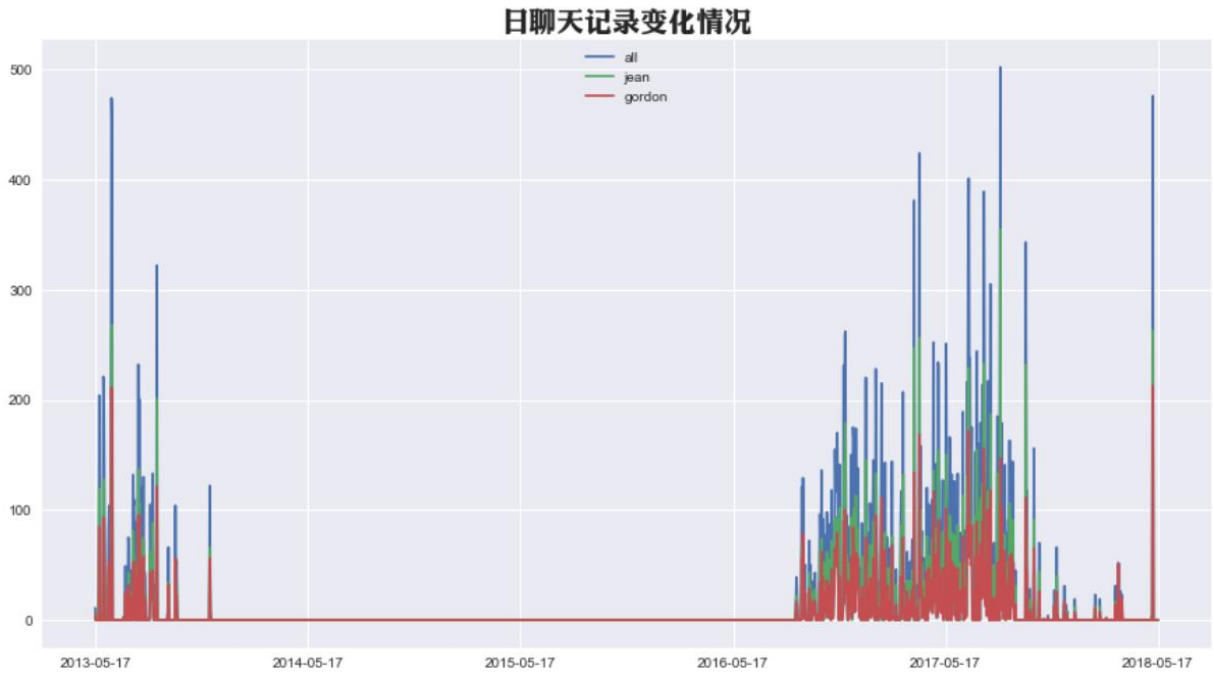
这句话是Gordon说的

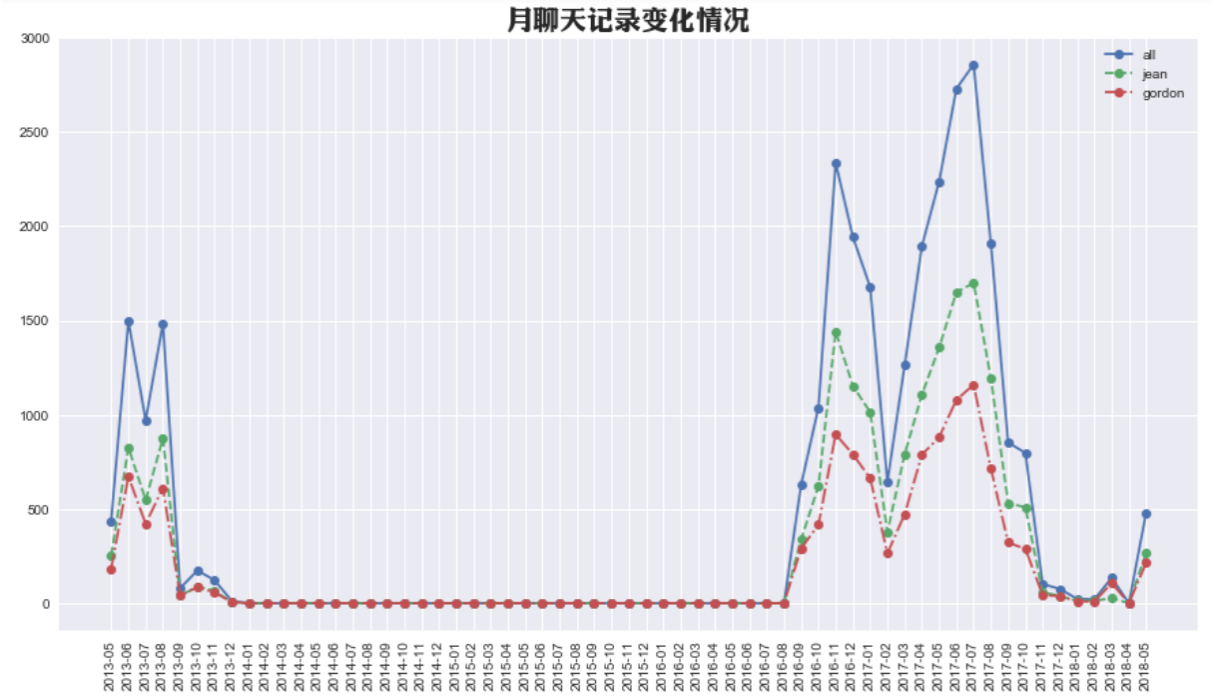
```
In [375]: prefunc('竞赛')
```

这句话是Gordon说的

```
In [376]: prefunc('男神')
```

这句话是Jean说的





Jean发起聊天时，常首先谈论的主题

```
subjectWordCloud('jean','create')
```

Gordon结束聊天时，常说的结束语

```
subjectWordCloud('gordon','end')
```

聊天发起者次数统计

Gordon发起聊天时，常首先谈论的主题

Jean结束聊天时，常说的结束语

0]: subjectWordCloud('gordon','create')

subjectWordCloud('jean','end')

聊天终结者次数统计

8 附录：完整代码

```
import numpy as np
import pandas as pd
import jieba
import jieba.analyse
import datetime
import random
from wordcloud import WordCloud
from PIL import Image
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv('out.csv')
data['time'] = data['time'].apply(lambda x: datetime.datetime.strptime(x, "%Y-%m-%d %H:%M:%S"))
data['date'] = data['time'].apply(lambda x: x.date())
data['hour'] = data['time'].apply(lambda x: x.hour)
data['year-month'] = data['time'].apply(lambda x:str(x.year) + '-' + str(x.month).zfill(2))
stop_words = []
with open('stop_words.txt') as f:
    for line in f.readlines():
        stop_words.append(line.strip())
#词云，前 50 个经常出现的词(有意义的词)
def wordCloud(person='all'):
    if person == 'all':
        content = ''
        for item in data['content']:
            content += (item+' ')
    elif person == 'jean':
        content = ''
        temp_data = data[data['name'] == 'Jean']
        for item in temp_data['content']:
            content += (item+' ')
    else:
        content = ''
        temp_data = data[data['name'] == 'Gordon']
        for item in temp_data['content']:
            content += (item+' ')
```

```
36        for word in stop_words:
37            content = content.replace(word, '')
38        tag_list = jieba.analyse.extract_tags(content, topK=50)
39        wl = ",".join(tag_list)
40        wc = WordCloud(background_color="white",mask=None,max_words=100,font_
    path='font.ttf',max_font_size=100,random_state=30,scale=2.5)
41        myword = wc.generate(wl)
42        plt.imshow(myword)
43        plt.axis("off")
44    #聊天记录总数
45    totGordonNum = len(data[data['name'] == 'Gordon'])
46    totJeanNum = len(data[data['name'] == 'Jean'])
47    #回复平均长度
48    avgGordonMsg = data[data['name'] == 'Gordon']['len'].mean()
49    avgJeanMsg = data[data['name'] == 'Jean']['len'].mean()
50    def dayCount(person='all'):
51        if person == 'all':
52            #按日统计聊天记录总数
53            dayChatNum = data['content'].groupby(data['date']).count()
54            dayFrame = pd.DataFrame(pd.date_range('2013-05-17','2018-05-17').astype
    (str))
55            dayFrame.columns = ['date']
56            tempFrame = pd.DataFrame(dayChatNum).reset_index()
57            tempFrame.columns = ['date', 'cnt']
58            tempFrame['date'] = tempFrame['date'].astype(str)
59            dayFrame = pd.merge(dayFrame, tempFrame, on='date',how='outer')
60            dayFrame = dayFrame.fillna(0)
61            return dayFrame
62        elif person == 'jean':
63            temp_data = data[data['name'] == 'Jean'].copy()
64            dayChatNum = temp_data['content'].groupby(temp_data['date']).count()
65            dayFrame = pd.DataFrame(pd.date_range('2013-05-17','2018-05-17').astype
    (str))
66            dayFrame.columns = ['date']
67            tempFrame = pd.DataFrame(dayChatNum).reset_index()
68            tempFrame.columns = ['date', 'cnt']
69            tempFrame['date'] = tempFrame['date'].astype(str)
70            dayFrame = pd.merge(dayFrame, tempFrame, on='date',how='outer')
71            dayFrame = dayFrame.fillna(0)
```

```
72              return dayFrame
73          elif person=='gordon':
74              temp_data = data[data['name'] == 'Gordon'].copy()
75              dayChatNum = temp_data['content'].groupby(temp_data['date']).count()
76              dayFrame = pd.DataFrame(pd.date_range('2013-05-17','2018-05-17').astype
    (str))
77              dayFrame.columns = ['date']
78              tempFrame = pd.DataFrame(dayChatNum).reset_index()
79              tempFrame.columns = ['date', 'cnt']
80              tempFrame['date'] = tempFrame['date'].astype(str)
81              dayFrame = pd.merge(dayFrame, tempFrame, on='date',how='outer')
82              dayFrame = dayFrame.fillna(0)
83              return dayFrame
84          else:
85              print('error')
86              return 0
87  def monCount(person='all'):
88          if person == 'all':
89              #按月统计聊天记录总数
90              monChatNum = data['content'].groupby(data['year-month']).count()
91              monFrame = pd.DataFrame([x[:7] for x in list(pd.date_range('2013-05','2
    018-05').astype(str))]).drop_duplicates()
92              monFrame.columns = ['month']
93              tempFrame = pd.DataFrame(monChatNum).reset_index()
94              tempFrame.columns = ['month', 'cnt']
95              tempFrame['month'] = tempFrame['month'].astype(str)
96              monFrame = pd.merge(monFrame, tempFrame, on='month',how='outer')
97              monFrame = monFrame.fillna(0)
98              return monFrame
99          elif person == 'jean':
100             #按月统计聊天记录总数
101             temp_data = data[data['name'] == 'Jean'].copy()
102             monChatNum = temp_data['content'].groupby(temp_data['year-month']).cou
    nt()
103             monFrame = pd.DataFrame([x[:7] for x in list(pd.date_range('2013-05','2
    018-05').astype(str))]).drop_duplicates()
104             monFrame.columns = ['month']
105             tempFrame = pd.DataFrame(monChatNum).reset_index()
106             tempFrame.columns = ['month', 'cnt']
```

```
107             tempFrame['month'] = tempFrame['month'].astype(str)
108             monFrame = pd.merge(monFrame, tempFrame, on='month',how='outer')
109             monFrame = monFrame.fillna(0)
110             return monFrame
111         elif person == 'gordon':
112             #按月统计聊天记录总数
113             temp_data = data[data['name'] == 'Gordon'].copy()
114             monChatNum = temp_data['content'].groupby(temp_data['year-month']).cou
    nt()
115             monFrame = pd.DataFrame([x[:7] for x in list(pd.date_range('2013-05','2
    018-05').astype(str))]).drop_duplicates()
116             monFrame.columns = ['month']
117             tempFrame = pd.DataFrame(monChatNum).reset_index()
118             tempFrame.columns = ['month', 'cnt']
119             tempFrame['month'] = tempFrame['month'].astype(str)
120             monFrame = pd.merge(monFrame, tempFrame, on='month',how='outer')
121             monFrame = monFrame.fillna(0)
122             return monFrame
123 #发起聊天人统计
124 createChat = data.drop_duplicates('date').copy()
125 createChat['flag'] = createChat['name']=='Gordon'
126 cjeanY = createChat[createChat['flag'] == False].groupby('year-month').count()['flag
    ']
127 cgordonY = createChat[createChat['flag'] == True].groupby('year-month').count()['fl
    ag']
128 #结束聊天人统计
129 endChat = data.drop_duplicates('date',keep='last').copy()
130 endChat['flag'] = endChat['name']=='Gordon'
131 ejeanY = endChat[endChat['flag'] == False].groupby('year-month').count()['flag']
132 egordonY = endChat[endChat['flag'] == True].groupby('year-month').count()['flag']
133 #聊天主题
134 def subjectWordCloud(person='jean', method='create'):
135     if method == 'create':
136         temp_data = data.drop_duplicates('date',keep='first').copy()
137     else:
138         temp_data = data.drop_duplicates('date',keep='last').copy()
139     if person == 'jean':
140         content = ''
141         temp_data = temp_data[temp_data['name'] == 'Jean']
```

```
142             for item in temp_data['content']:
143                 content += (item+' ')
144     else:
145             content = ''
146             temp_data = temp_data[temp_data['name'] == 'Gordon']
147             for item in temp_data['content']:
148                 content += (item+' ')
149     for word in stop_words:
150             content = content.replace(word, '')
151     tag_list = jieba.analyse.extract_tags(content, topK=50)
152     wl = ','.join(tag_list)
153     wc = WordCloud(background_color="white",mask=None,max_words=100,font_
    path='font.ttf',max_font_size=100,random_state=30,scale=2.5)
154     myword = wc.generate(wl)
155     plt.imshow(myword)
156     plt.axis("off")
157 def func(a):
158     if a == 'Gordon':
159             return 1;
160     else:
161             return 0
162 data['label'] = data.name.map(func)
163 def createVocabList(dataSet):
164     vocabSet = set()
165     for document in dataSet:
166             vocabSet = vocabSet | set(document)
167     return list(vocabSet)
168 def bagOfwords2MN(vocabList, inputSet):
169     returnVec = [0]*len(vocabList)
170     for word in inputSet:
171             if word in vocabList:
172                 returnVec[vocabList.index(word)] += 1
173     return returnVec
174 def divSet(vocablist, dataSet, classVec):
175     trainingSet = range(len(dataSet)); testSet=[]
176     for i in range(int(len(dataSet)*0.3)):
177             randIndex = int(random.uniform(0,len(trainingSet)))
178             testSet.append(trainingSet[randIndex])
179             del(list(trainingSet)[randIndex])
```

```
    trainMat=[]; trainClasses = []
    for docIndex in trainingSet:
        trainMat.append(bagOfwords2MN(vocabList, dataSet[docIndex]))
        trainClasses.append(classVec[docIndex])
    return trainMat,trainClasses,testSet
def trainNB1(trainMatrix,trainCategory):
    numTrainDocs = len(trainMatrix)
    numWords = len(trainMatrix[0])
    pAbusive = sum(trainCategory)/float(numTrainDocs) #p(辱骂的=1)的概率

    p0Num = np.ones(numWords);p1Num = np.ones(numWords) #n 列
    p0Denom = 2.0; p1Denom = 2.0 #分母
    for i in range(numTrainDocs):
        if trainCategory[i] == 1:
            p1Num += trainMatrix[i] #n 列同时计数
            p1Denom += sum(trainMatrix[i]) ##标量 分母是该类的总词条数目
        else:
            p0Num += trainMatrix[i]
            p0Denom += sum(trainMatrix[i])
    p1Vect = np.log(p1Num/p1Denom)
    p0Vect = np.log(p0Num/p0Denom)
    return p0Vect,p1Vect,pAbusive
def classifyNB(vec2Classify, p0Vec, p1Vec, pClass1):
    p1 =sum(vec2Classify*p1Vec) + np.log(pClass1) #转为 log 后*全部为+
    p0 =sum(vec2Classify*p0Vec) + np.log(1 - pClass1)
    if p1>p0:
        return 1
    else:
        return 0
def prefunc(string):
    testWords = string
    testEntry = " ".join(jieba.cut(testWords)).split()
    thisDoc = np.array(bagOfwords2MN(vocabList,testEntry))
    predict = classifyNB(thisDoc,p0V,p1V,pAb)
    if predict==0:
        print ("这句话是 Jean 说的")
    else:
        print("这句话是 Gordon 说的")
```

```
def test(testSet,vocabList, dataSet, classVec, p0V,p1V,pAb):
    errorCount = 0
    for docIndex in testSet:
        wordVector = bagOfwords2MN(vocabList, dataSet[docIndex])
        if classifyNB(np.array(wordVector),p0V,p1V,pAb) != classVec[docIndex]:
            errorCount += 1
            print ("classification error",dataSet[docIndex])
    print ('the error rate is: ',float(errorCount)/len(testSet))
dataSet = []
for i in data.index:
    dataSet.append(" ".join(jieba.cut(data.content[i])).split())
classVec = data.label.tolist()
vocabList = createVocabList(dataSet)
trainMat,trainClasses,testSet = divSet(vocabList,dataSet,classVec)
p0V,p1V,pAb = trainNB1(trainMat,trainClasses)
test(testSet,vocabList, dataSet, classVec, p0V,p1V,pAb)
prefunc('测试这句话是谁说的')
```