**东北财经大学本科生论文形式课程考试**

**成绩评定单**

**论文题目 酒店评论情感分析**

**学生姓名 全深深 学 号 2016210645**

**专 业 大数据商务分析创新实验班**

**年 级 2016**

**课程名称 文本挖掘**

|  |
| --- |
| **论文评语：** |

**酒店评论情感分析**

**摘要：**

本文采用酒店评论数据集进行情感分析，通过机器学习和基于情感词典两种方法进行分析比较。其中，机器学习方法采用了多种算法，有支持向量机、神经网络、朴素贝叶斯以及逻辑回归四种，并比较各分类器的准确率，得到准确率最高的模型。

**关键词**：酒店评论 情感分析 机器学习 情感词典

1. **绪论**
2. **情感分析**

情感分析（Sentiment analysis，SA），又称意见挖掘（Opinion mining）、情感挖掘（Sentiment mining），是对带有情感色彩的主观性文本进行分析、处理、归纳和推理的过程。情感分析的目的是为了找出说话者在某些话题上做出言论时的情绪状态。

文本情感分析的应用非常广泛，如网络舆情风险分析，信息预测等。如通过Twitter用户情感预测股票走势，电影票房、选举结果等，均是将公众情绪与社会事件对比，发现一致性，并用于预测。

情感分析的方法一般有两种，一是通过机器学习的方法，把情感分析当成分类问题来解决，要求有分好类的训练数据集，一般是有监督学习。二是基于情感词典的方法，计算句子的情感得分，推测出情感状态，多用于无监督学习的问题。

1. **机器学习**

基于机器学习的情感分类问题，它的处理过程大致可以分为两个部分，一部分是学习过程，另一部分是情感分类过程。其中，学习过程包括训练过程和测试过程，训练过程中对训练集进行训练得到分类器，用其对测试集进行情感分类，将测试的结果反馈给分类器，进一步改进训练方法，生成新的分类器，最后利用最终生成的分类器对新的文本进行情感分类。

其优点是准确性较高，适用场景更多样。缺点是作为有监督学习，对训练语料依赖性高，针对不同领域，需要人工标注训练数据，并且训练周期长。

1. **基于情感词典**

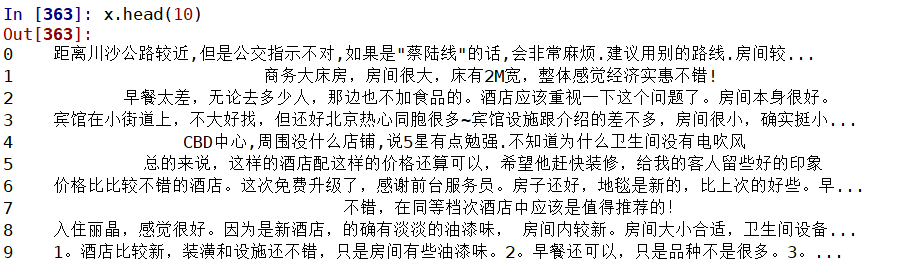
要构建合适的情感词典，为情感分析而建立的特殊的字典或词汇表，其中包括了情感词和情感短语的情感倾向性和情感强度。要定位句中的情感词，处理否定词及程度副词，综合计算句子的情感得分，最后给出结论。

其优点是面向大规模、多领域的实际应用场景，鲁棒性强，不需要依赖人工标注的训练集。缺点是受到自然语言处理技术的限制，影响分析的精度。

1. **数据获取及词典构建**
2. **数据来源**

数据来源于携程网上的酒店评论，由谭松波教授多年实践工作中收集得到的评论语料，并做好了情感标签，很适合用机器学习算法。我选用了其中的6000条评论，其中正负评论各3000条。

（<https://download.csdn.net/download/qq_40989940/10856258>）

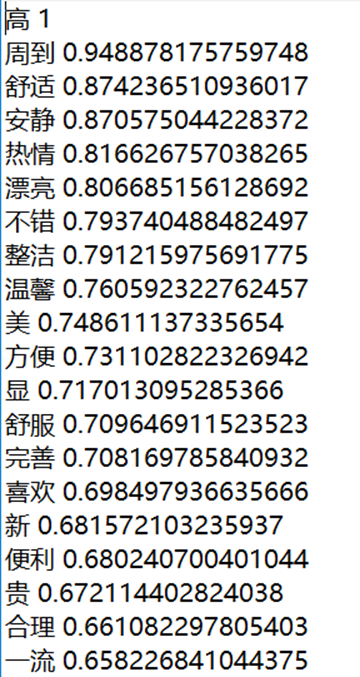
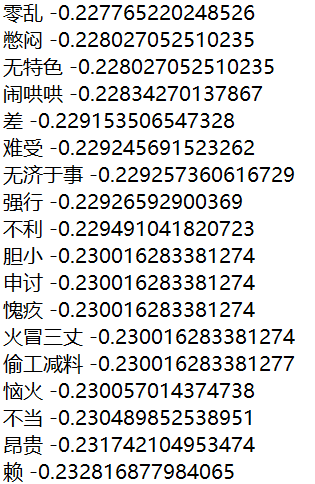


1. **构建停用词表**

为获得更好的分类效果，需要更加精准的去除不相关的词语。此实验整合了常用的哈工大停用词表、四川大学机器学习智能实验室停用词库、百度停用词表。并根据文本的词频统计，加入了酒店相关的停用词，如‘说’、‘住’‘入住’、‘订’、‘元’等。

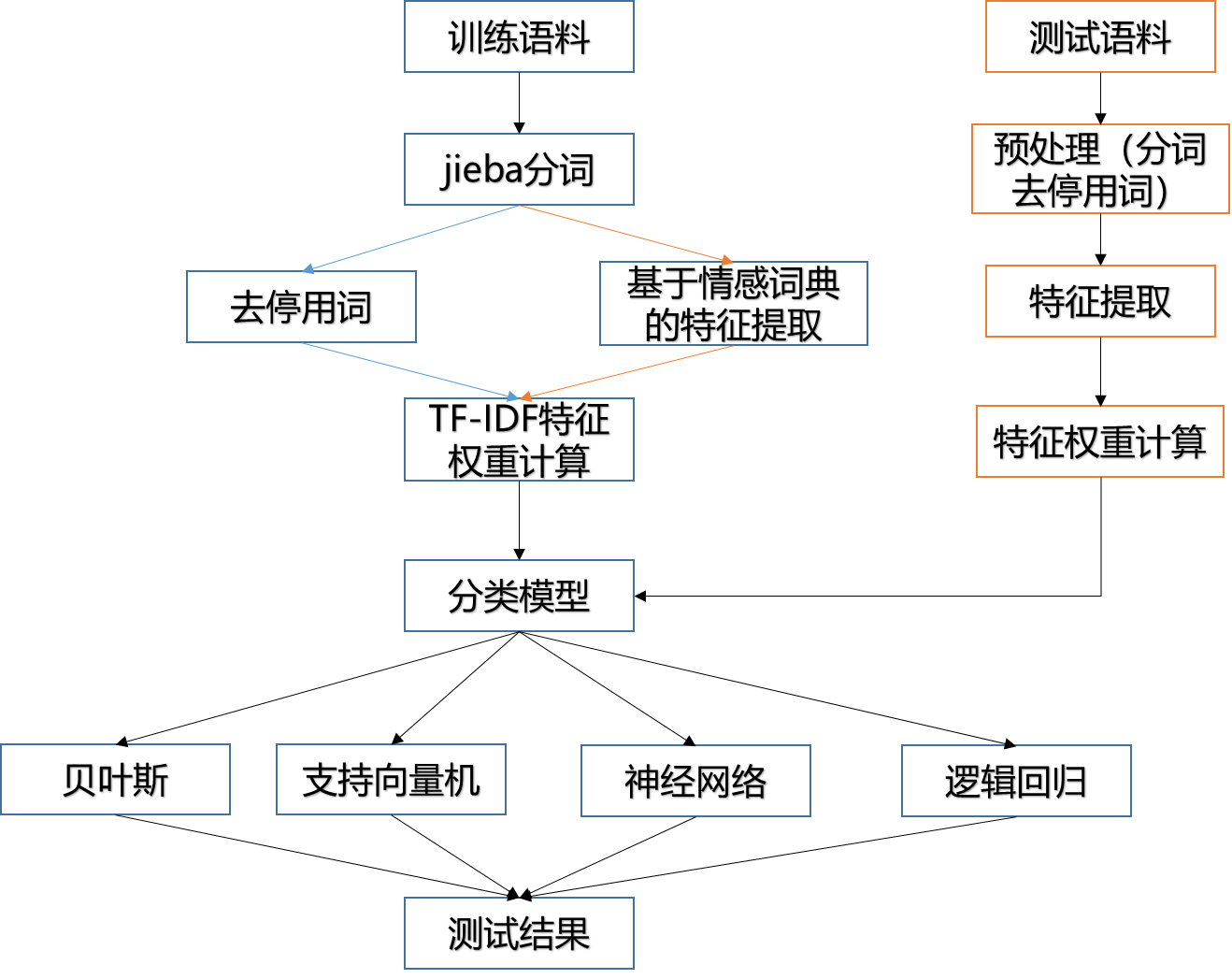
1. **构建情感词典**

此次实验整合了知网和台湾大学的情感词典，并加入了从网上找到的酒店相关的情感词典。整合之后，正向词共5161个，负向词共4776个。程度副词、否定词词典均选自知网。

1. **基于机器学习的文本分类方法**

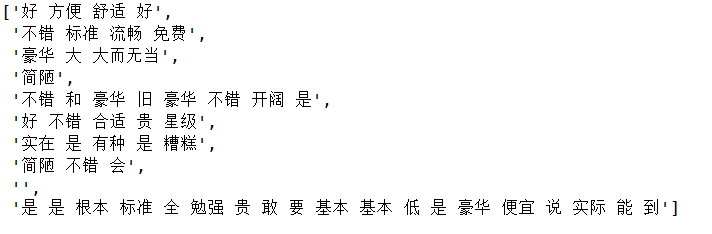
流程图如下：



1. **文本预处理**

将所有的酒店评论语料整合在一起，并按1：3的比例随机划分测试集和训练集。先用jieba中文分词工具进行分词，在基于构建好的停用词库去停用词。

第二种方法是jieba分词后，只提取包含在情感词典中的词作为特征词。最后将两种方法的测试结果作比较。



1. **词频统计**

在分词、停用词处理完后，先进行词频统计，找出高频却对文本分类无意义的词加入停用词表，来提高算法的准确度，下图是根据词频做的词云图：



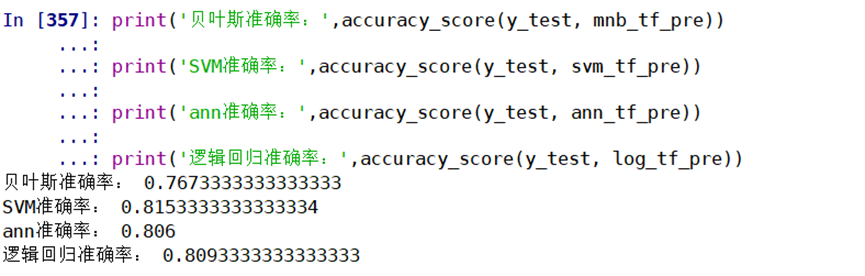
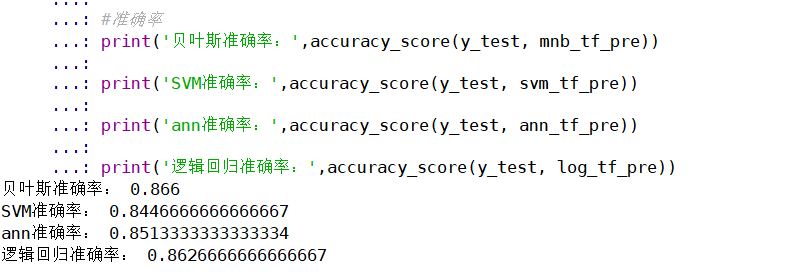
1. **特征提取与文本表示**

文本是一种非结构化的数据，由大量字符构成，计算机无法直接处理字符类型的数据，因此需要将普通文本的内容转变为计算机能够读懂的数据形式，即将文本进行形式化表示。实验采用向量空间模型来表示文本，它对文本的表示效果较好，可以将文档表示成空间向量进行运算，且具有较强的可计算性和可操作性。它缺点也十分明显，忽略了特征的次序和位置关系，且不考虑文本的长度，不考虑语义联系，认为所有的特征项都是独立的，只考虑文本所属类别的文档中出现特征项的频率，在情感分类的应用中存在一定的局限性。

文本向量化后，采用TF-IDF模型进行特征选择和特征提取，提取出有效的特征词进行训练。

1. **分类模型**

分类器是文本分类问题中的核心部分，在进行文本分类过程中常用的分类器有朴素贝叶斯分类器(NB)，支持向量机(SVM)，最大熵分类器(ME)，K近邻分类器(KNN)等。此实验使用了SVM、ANN、贝叶斯、逻辑回归等多种方法来进行分类训练，并对比各自的测试结果。



1. **模型评估**

通过训练集训练好分类器后，用测试集进行检验，并以准确率作为评估结果。通过实验，各方法结果对比如下表：

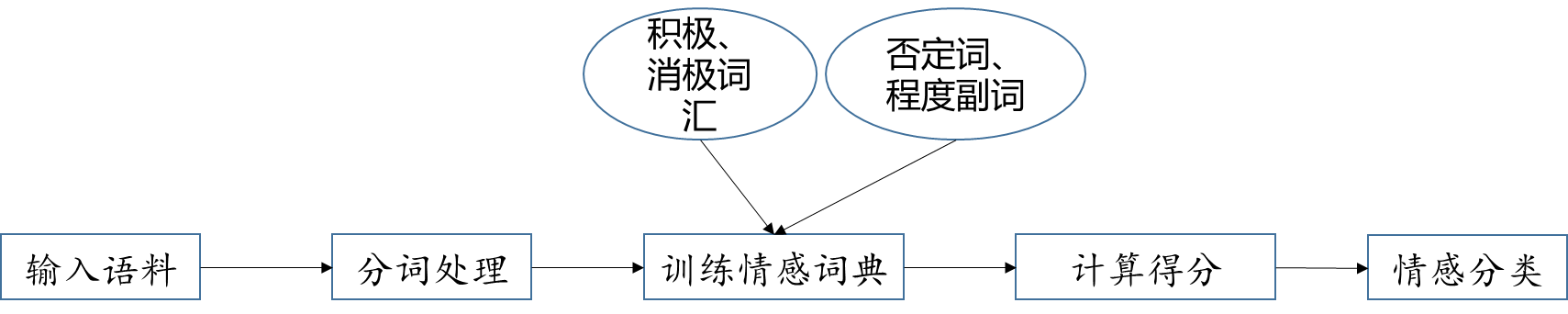


1. **总结**

可以看到，同一种方法中各模型的准确率差异不是很大，其中贝叶斯分类的准确率最高，达到了0.866。本以为基于情感词典的特征提取方法会更加准确，但结果普遍低于普通方法，准确率最高的SVM模型只有0.815。

1. **基于情感词典的方法**

实验又尝试了基于情感词典的方法，此方法相比于机器学习就比较简单了，流程如下：



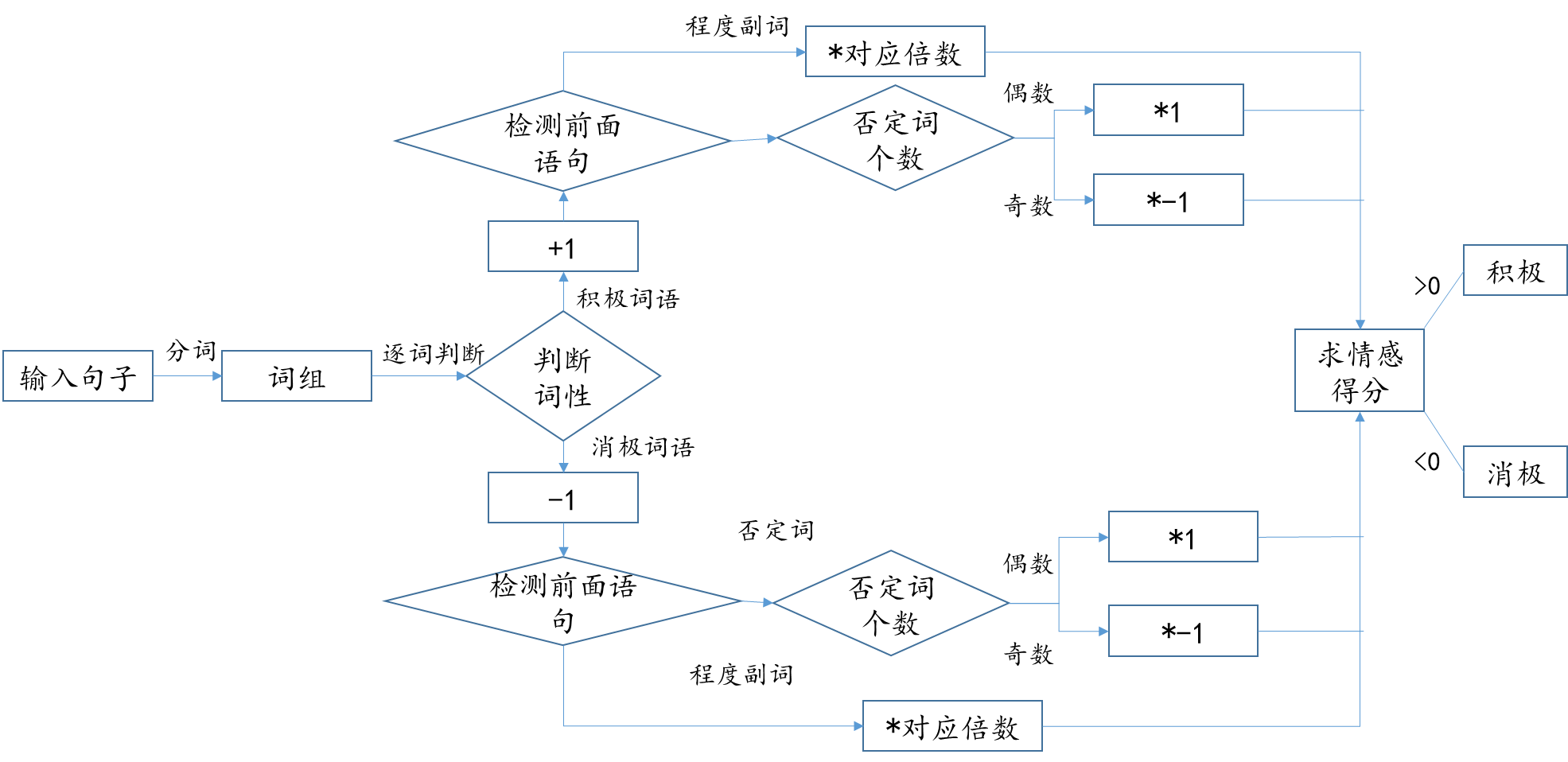
其中，最重要的就是词典的构造以及得分的计算，词典在前文中已经叙述了，下面来看看如何计算情感分值。

* 1. 、对于情感词汇，并没有考虑每一个词的权值，而是同等对待，如果是积极词汇，分值加一，反之，则减一；
  2. 、句子中检测出程度副词，得分\*程度指标；
  3. 、句子中的否定词，计算其个数

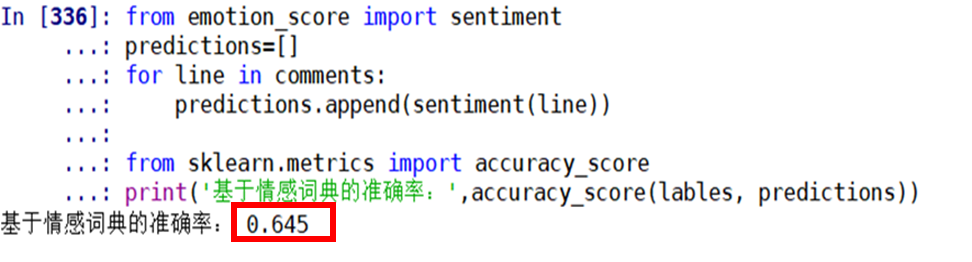
偶数：得分\*1

奇数：得分\*-1

具体模型如下图所示：



但模型准确率不是很好，只有0.645。



1. **小结**

情感分类是一个具有挑战性的研究课题，其问题更加复杂，同一方法应用于不同语料所产生的结果存在较大差异。本实验使用两种方法对文本情感分析做了基础的研究，取得了一定的研究结论。从实验结果看，机器学习的方法要远好于基于情感词典的方法。但得到的结果还有很大的提升空间，就本文而言，在许多方面还需要进一步的研究和改进。

在机器学习算法中，可以通过调节参数，来提高准确率；在基于情感词典的方法中，可以进一步考虑不同词汇的具体权值，优化计分模型。

**附录：**

【主程序：run.py】

#%%库引入

**import** os

**import** jieba

**import** collections

**from** wordcloud **import** WordCloud

**import** matplotlib**.**pyplot **as** plt

**import** pandas **as** pd

#%%情感词典构建

stopwords**=[**line**.**strip**()** **for** line **in** open**(**'C:/Users/dell/Desktop/文本挖掘期末/hotel\_comment/stopwords/stopword.txt'**,** 'r'**,**encoding**=**'utf-8'**,**errors**=**'ignore'**).**readlines**()]**

#结合了多个停用词表

**def** open\_dict**(**Dict**=**'hahah'**,**path**=**'C:/Users/dell/Desktop/文本挖掘期末/hotel\_comment/emotion\_dict/'**):**

path **=** path **+** '%s.txt' **%**Dict

dictionary **=** open**(**path**,** 'r'**,** encoding**=**'utf-8-sig'**,**errors**=**'ignore'**)**#encoding='utf-8-sig',检查是否有文件头，并去掉

dict **=** **[]**

**for** word **in** dictionary**:**

word**=**word**.**strip**(**'\n'**)**

word**=**word**.**strip**(**' '**)**

dict**.**append**(**word**)**

**return** dict

posdict **=** open\_dict**(**Dict**=**'posdict'**)**#积极情感词典

negdict **=** open\_dict**(**Dict**=**'negdict'**)**#消极情感词典

f**=**open**(**'C:/Users/dell/Desktop/文本挖掘期末/hotel\_comment/emotion\_dict/酒店情感词典.txt'**,**'r'**,**encoding**=**'utf-8'**)**

words **=** **[]**

value**=[]**

**for** word **in** f**.**readlines**():**

words**.**append**(**word**.**split**(**' '**)[**0**])**

value**.**append**(**float**(**word**.**split**(**' '**)[**1**].**strip**(**'\n'**)))**

c**={**'words'**:**words**,**

'value'**:**value**}**

fd**=**pd**.**DataFrame**(**c**)**

pos**=**fd**[**'words'**][**fd**.**value**>**0**]**

posdict**=**posdict**+**list**(**pos**)** ##加入酒店相关的正向情感词

neg**=**fd**[**'words'**][**fd**.**value**<**0**]**

negdict**=**negdict**+**list**(**neg**)** ##加入酒店相关的负向情感词

alldict**=**posdict**+**negdict

f**.**close**()**

#%%预处理函数

**def** remove\_characters**(**sentence**):** #去停用词

cleanwordlist**=[**word **for** word **in** sentence **if** word**.**lower**()** **not** **in** stopwords**]**

filtered\_text**=**' '**.**join**(**cleanwordlist**)**

**return** filtered\_text

**def** remove\_emotion\_characters**(**sentence**):** #抽取情感词

wordlist**=[**word **for** word **in** sentence **if** word **in** alldict**]**

filtered\_text**=**' '**.**join**(**wordlist**)**

**return** filtered\_text

**def** text\_normalize**(**text**):**

text\_split**=[]**

**for** line **in** text**:**

text\_split**.**append**(**list**(**jieba**.**cut**(**line**)))**

text\_normal**=[]**

**for** word\_list **in** text\_split**:**

text\_normal**.**append**(**remove\_characters**(**word\_list**))**

**return** text\_normal

**def** text\_normalize2**(**text**):** #基于情感词典的预处理

text\_split**=[]**

**for** line **in** text**:**

text\_split**.**append**(**list**(**jieba**.**cut**(**line**)))**

text\_normal**=[]**

**for** word\_list **in** text\_split**:**

text\_normal**.**append**(**remove\_emotion\_characters**(**word\_list**))**

**return** text\_normal

**def** get\_content**(**path**):**

**with** open**(**path**,**'r'**,**encoding**=**'utf-8'**,**errors**=**'ignore'**)** **as** f**:**

content**=**''

**for** l **in** f**:**

l**=**l**.**strip**().**replace**(**u'\u3000'**,**u''**)**

content**+=**l

**return** content

**def** get\_file\_content**(**path**):**

flist**=**os**.**listdir**(**path**)**

flist**=[**os**.**path**.**join**(**path**,**x**)** **for** x **in** flist**]**

corpus**=[**get\_content**(**x**)** **for** x **in** flist**]**

**return** corpus

#读取语料文本

pos\_comment**=**get\_file\_content**(**'C:/Users/dell/Desktop/文本挖掘期末/hotel\_comment/6000/pos'**)**

neg\_comment**=**get\_file\_content**(**'C:/Users/dell/Desktop/文本挖掘期末/hotel\_comment/6000/neg'**)**

#%%词频统计、词云图

all\_comment**=**''

**for** x **in** pos\_comment**:**

all\_comment**+=**x

**for** x **in** neg\_comment**:**

all\_comment**+=**x

split\_words**=**list**(**jieba**.**cut**(**all\_comment**))** #分词

filtered\_corpus**=**remove\_characters**(**split\_words**)** #去停用词

filtered\_corpus**=[**word **for** word **in** split\_words **if** word **not** **in** stopwords**]**

##词频统计

word\_counts **=** collections**.**Counter**(**filtered\_corpus**)** # 对分词做词频统计

word\_counts\_top10 **=** word\_counts**.**most\_common**(**10**)** # 获取前10最高频的词

**print** **(**word\_counts\_top10**)** # 输出检查

##词云制作

wordcloud\_1 **=** WordCloud**(**

font\_path**=**'C:/Windows/Fonts/simkai.ttf'**,**#设置字体，为电脑自带黑体

max\_words**=**200**,** # 最多显示词数

max\_font\_size**=**100**,** # 字体最大值)

background\_color**=**'white'**,**

mask**=**plt**.**imread**(**'C:/Users/dell/Desktop/文本挖掘期末/hotel\_comment/house.jpg'**)**

**)**

wordcloud**=**wordcloud\_1**.**generate\_from\_frequencies**(**word\_counts**)**

wordcloud**.**to\_file**(**'C:/Users/dell/Desktop/文本挖掘期末/hotel\_comment/wordcloud.jpg'**)**

plt**.**figure**(**figsize**=(**16**,**12**))**

plt**.**imshow**(**wordcloud**,** interpolation**=**'bilinear'**)**

plt**.**axis**(**"off"**)**

#%%数据整理与划分

pos\_lable**=[**1 **for** i **in** range**(**3000**)]** #加入标签

neg\_lable**=[-**1 **for** i **in** range**(**3000**)]**

comments**=**pos\_comment**+**neg\_comment #语料整合

lables**=**pos\_lable**+**neg\_lable

c**={**'comment'**:**comments**,**

'value'**:**lables**}**

df**=**pd**.**DataFrame**(**c**)**

x**=**df**[**'comment'**]**

y**=**df**[**'value'**]**

**from** sklearn**.**model\_selection **import** train\_test\_split

#分割数据集

X\_train**,** X\_test**,** y\_train**,** y\_test **=** train\_test\_split**(**x**,** y**,** test\_size**=**1**/**4**,** random\_state**=**0**)**

X\_train**=**text\_normalize**(**X\_train**)**

X\_test**=**text\_normalize**(**X\_test**)**

#X\_train=text\_normalize2(X\_train) #方法二：基于情感词典的特征提取

#X\_test=text\_normalize2(X\_test)

#%%机器学习部分

**from** sklearn**.**feature\_extraction**.**text **import** TfidfVectorizer

**from** sklearn**.**naive\_bayes **import** MultinomialNB

**from** sklearn**.**linear\_model **import** SGDClassifier

**from** sklearn**.**neural\_network **import** MLPClassifier

**from** sklearn**.**linear\_model **import** LogisticRegression

**from** sklearn**.**metrics **import** accuracy\_score

#TF-IDF计算权重

tfidf\_vectorizer**=**TfidfVectorizer**(**min\_df**=**1**,**norm**=**'l2'**,**smooth\_idf**=True,**use\_idf**=True,**ngram\_range**=(**1**,**1**))**

tf\_train\_features**=** tfidf\_vectorizer**.**fit\_transform**(**X\_train**)**

tf\_test\_features**=** tfidf\_vectorizer**.**transform**(**X\_test**)**

#a=tfidf\_vectorizer.vocabulary\_

#b=sorted(a.items(),key=lambda item:item[1],reverse=True)

##预测函数

**def** train\_predict\_evaluate\_model**(**classifier**,**train\_features**,**train\_labels**,**test\_features**,**test\_labels**):**

classifier**.**fit**(**train\_features**,**train\_labels**)**

predictions**=**classifier**.**predict**(**test\_features**)**

**return** predictions

##贝叶斯

mnb**=**MultinomialNB**()**

mnb\_tf\_pre**=**train\_predict\_evaluate\_model**(**classifier**=**mnb**,**train\_features**=**tf\_train\_features**,**train\_labels**=**y\_train**,**test\_features**=**tf\_test\_features**,**test\_labels**=**y\_test**)**

##SVM

svm**=**SGDClassifier**(**loss**=**'hinge'**,**max\_iter**=**1000**)**

svm\_tf\_pre**=**train\_predict\_evaluate\_model**(**classifier**=**svm**,**train\_features**=**tf\_train\_features**,**train\_labels**=**y\_train**,**test\_features**=**tf\_test\_features**,**test\_labels**=**y\_test**)**

###ann

ann\_model **=** MLPClassifier**(**hidden\_layer\_sizes**=**1**,** activation**=**'logistic'**,** solver**=**'lbfgs'**,** random\_state**=**0**)**

ann\_tf\_pre**=**train\_predict\_evaluate\_model**(**classifier**=**ann\_model**,**train\_features**=**tf\_train\_features**,**train\_labels**=**y\_train**,**test\_features**=**tf\_test\_features**,**test\_labels**=**y\_test**)**

##逻辑回归

logreg **=** LogisticRegression**(**C**=**1**,**penalty**=**'l2'**)**

log\_tf\_pre**=**train\_predict\_evaluate\_model**(**classifier**=**logreg**,**train\_features**=**tf\_train\_features**,**train\_labels**=**y\_train**,**test\_features**=**tf\_test\_features**,**test\_labels**=**y\_test**)**

#准确率

**print(**'贝叶斯准确率：'**,**accuracy\_score**(**y\_test**,** mnb\_tf\_pre**))**

**print(**'SVM准确率：'**,**accuracy\_score**(**y\_test**,** svm\_tf\_pre**))**

**print(**'ann准确率：'**,**accuracy\_score**(**y\_test**,** ann\_tf\_pre**))**

**print(**'逻辑回归准确率：'**,**accuracy\_score**(**y\_test**,** log\_tf\_pre**))**

#%%基于情感词典的分析

**from** emotion\_score **import** sentiment

predictions**=[]**

**for** line **in** comments**:**

predictions**.**append**(**sentiment**(**line**))**

**from** sklearn**.**metrics **import** accuracy\_score

**print(**'基于情感词典的准确率：'**,**accuracy\_score**(**lables**,** predictions**))**

【情感得分计算：emotion\_score.py】

**import** jieba

**import** pandas **as** pd

#载入情感词典

# 打开词典文件，返回列表

**def** open\_dict**(**Dict**=**'hahah'**,**path**=**'C:/Users/dell/Desktop/文本挖掘期末/hotel\_comment/emotion\_dict/'**):**

path **=** path **+** '%s.txt' **%**Dict

dictionary **=** open**(**path**,** 'r'**,** encoding**=**'utf-8-sig'**,**errors**=**'ignore'**)**#encoding='utf-8-sig',检查是否有文件头，并去掉

dict **=** **[]**

**for** word **in** dictionary**:**

word**=**word**.**strip**(**'\n'**)**

word**=**word**.**strip**(**' '**)**

dict**.**append**(**word**)**

**return** dict

posdict **=** open\_dict**(**Dict**=**'posdict'**)**#积极情感词典

negdict **=** open\_dict**(**Dict**=**'negdict'**)**#消极情感词典

inversedict**=**open\_dict**(**Dict**=**'inversedict'**)**

mostdict **=** open\_dict**(**Dict**=**'mostdict'**)**

verydict**=** open\_dict**(**Dict**=**'verydict'**)**

moredict **=** open\_dict**(**Dict**=**'moredict'**)**

ishdict **=** open\_dict**(**Dict**=**'ishdict'**)**

insufficientdict **=** open\_dict**(**Dict**=**'insufficientdict'**)**

f**=**open**(**'C:/Users/dell/Desktop/文本挖掘期末/hotel\_comment/emotion\_dict/酒店情感词典.txt'**,**'r'**,**encoding**=**'utf-8'**)**

words **=** **[]**

value**=[]**

**for** word **in** f**.**readlines**():**

words**.**append**(**word**.**split**(**' '**)[**0**])**

value**.**append**(**float**(**word**.**split**(**' '**)[**1**].**strip**(**'\n'**)))**

c**={**'words'**:**words**,**

'value'**:**value**}**

fd**=**pd**.**DataFrame**(**c**)**

pos**=**fd**[**'words'**][**fd**.**value**>**0**]**

posdict**=**posdict**+**list**(**pos**)** ##加入酒店相关的正向情感词

neg**=**fd**[**'words'**][**fd**.**value**<**0**]**

negdict**=**negdict**+**list**(**neg**)** ##加入酒店相关的负向情感词

f**.**close**()**

#分句

**def** cut\_sentence**(**words**):**

start **=** 0

i **=** 0

sents **=** **[]**

token**=[]**

punt\_list **=** ',.!?:;~，。！？：；～'

**for** word **in** words**:**

**if** word **in** punt\_list **:** #检查标点符号下一个字符是否还是标点

sents**.**append**(**words**[**start**:**i**+**1**])**

start **=** i**+**1

i **+=** 1

**else:**

i **+=** 1

token **=** list**(**words**[**start**:**i**+**2**]).**pop**()** # 取下一个字符

**if** start **<** len**(**words**):**

sents**.**append**(**words**[**start**:])**

**return** sents

#定义判断奇偶的函数

**def** judgeodd**(**num**):**

**if** num**%**2**==**0**:**

**return** 'even'

**else:**

**return** 'odd'

#计算正、负和总的情感得分

**def** sentiment**(**review**):**

sents**=**cut\_sentence**(**review**)**

#print(sents)

pos\_senti**=**0#段落的情感得分

neg\_senti**=**0

total\_senti**=**0

**for** sent **in** sents**:**

pos\_count**=**0#句子的情感得分

neg\_count**=**0

seg**=**jieba**.**lcut**(**sent**,**cut\_all**=False)**

#print(sent)

i **=** 0 #记录扫描到的词的位置

a **=** 0 #记录情感词的位置

poscount **=** 0 #正向词的第一次分值

poscount2 **=** 0 #正向词反转后的分值

poscount3 **=** 0 #正向词的最后分值

negcount **=** 0 #负向词的第一次分值

negcount2 **=** 0 #负向词反转后的分值

negcount3 **=** 0 #负向词的最后分值

**for** word **in** seg**:**

#print(word)

poscount**=**0

negcount**=**0

**if** word **in** posdict**:** #判断词语是否是情感词

poscount **+=** 1

c **=** 0 #情感词前否定词的个数

**for** w **in** seg**[**a**:**i**]:** #扫描情感词前的程度词

**if** w **in** mostdict**:**

poscount **\*=** 4.0

**elif** w **in** verydict**:**

poscount **\*=** 3.0

**elif** w **in** moredict**:**

poscount **\*=** 2.0

**elif** w **in** ishdict**:**

poscount **/=** 2.0

**elif** w **in** insufficientdict**:**

poscount **/=** 4.0

**elif** w **in** inversedict**:**

c **+=** 1

**if** judgeodd**(**c**)** **==** 'odd'**:** #扫描情感词前的否定词数

poscount **\*=** **-**1.0

poscount2 **+=** poscount

poscount **=** 0

poscount3 **=** poscount **+** poscount2 **+** poscount3

poscount2 **=** 0

**else:**

poscount3 **=** poscount **+** poscount2 **+** poscount3

poscount **=** 0

a **=** i **+** 1 #情感词的位置变化

**elif** word **in** negdict**:** #消极情感的分析，与上面一致

negcount **+=** 1

d **=** 0

**for** w **in** seg**[**a**:**i**]:**

**if** w **in** mostdict**:**

#print(w)

negcount **\*=** 4.0

**elif** w **in** verydict**:**

#print(w)

negcount **\*=** 3.0

**elif** w **in** moredict**:**

#print(w)

negcount **\*=** 2.0

**elif** w **in** ishdict**:**

#print(w)

negcount **/=** 2.0

**elif** w **in** insufficientdict**:**

#print(w)

negcount **/=** 4.0

**elif** w **in** inversedict**:**

d **+=** 1

**if** judgeodd**(**d**)** **==** 'odd'**:**

negcount **\*=** **-**1.0

negcount2 **+=** negcount

negcount **=** 0

negcount3 **=** negcount **+** negcount2 **+** negcount3

negcount2 **=** 0

**else:**

negcount3 **=** negcount **+** negcount2 **+** negcount3

negcount **=** 0

a **=** i **+** 1

i **+=** 1 #扫描词位置前移

**if** poscount3 **<** 0 **and** negcount3 **>=**0**:**

neg\_count **+=** negcount3 **-** poscount3

pos\_count **=** 0

**elif** negcount3 **<** 0 **and** poscount3 **>=** 0**:**

pos\_count **=** poscount3 **-** negcount3

neg\_count **=** 0

**elif** poscount3 **<** 0 **and** negcount3 **<** 0**:**

neg\_count **=** **-**poscount3

pos\_count **=** **-**negcount3

**else:**

pos\_count **=** poscount3

neg\_count **=** negcount3

#print(pos\_count,neg\_count)

pos\_senti**=**pos\_senti**+**pos\_count

neg\_senti**=**neg\_senti**+**neg\_count

total\_senti**=**pos\_senti**-**neg\_senti

**if** total\_senti**>**0**:**

predictions**=**1

**else:**

predictions**=-**1

**return** **(**predictions**)**