中山大学计算机学院人工智能本科生实验报告（2022学年春季学期）

课程名称：Artificial Intelligence

教学班级	专业（方向）	学号	姓名
2班	计算机科学与技术	21307174	刘俊杰

一、实验题目

Week 10 Naïve Bayes 朴素贝叶斯

二、实验内容

实验内容

- 用朴素贝叶斯法完成文本信息情感分类训练，要求使用拉普拉斯(Laplace)平滑技5，理论课件中为所有计数加1，这里可以尝试加λ(λ>0)。
- 使用压缩包中的数据进行训练和测试，计算各类别准确率与总准确率，可借助 sklearn 机器学习库完成文本特征(TF-IDF)提取

1. 算法原理

朴素贝叶斯

特征条件假设：假设每个特征之间没有联系，给定训练数据集，其中每个样本x都包括n维特征，即x = (x1,x2……xn)，类标记集合含有k种类别，即y=(y1,y2……yk) 对于给定的新样本x，判断其属于哪个标记的类别，根据贝叶斯定理,可以得到x属于yk类别的概率 可以得到x属于{y_k}类别的概率

$$P(y_k|x) = \frac{P(x|y_k) \times P(y_k)}{\sum_k P(x|y_k) \times P(y_k)}$$

朴素贝叶斯算法对条件概率分布作出了独立性的假设，通俗地讲就是说假设各个维度的特征{x_1},{x_2}, \cdots ,{x_n}互相独立，在这个假设的前提上，条件概率可以转化为：

$$P(x|y_k) = P(x_1, x_2, \cdots, x_n|y_k) = \prod_{i=1}^{n} P(x_i|y_k)$$

代入上面贝叶斯公式中，得到：

$$P(y_k|x) = \frac{P(y_k) \times \prod_{i=1}^{n} P(x_i|y_k)}{\sum_k P(y_k) \times \prod_{i=1}^{n} P(x_i|y_k)}$$

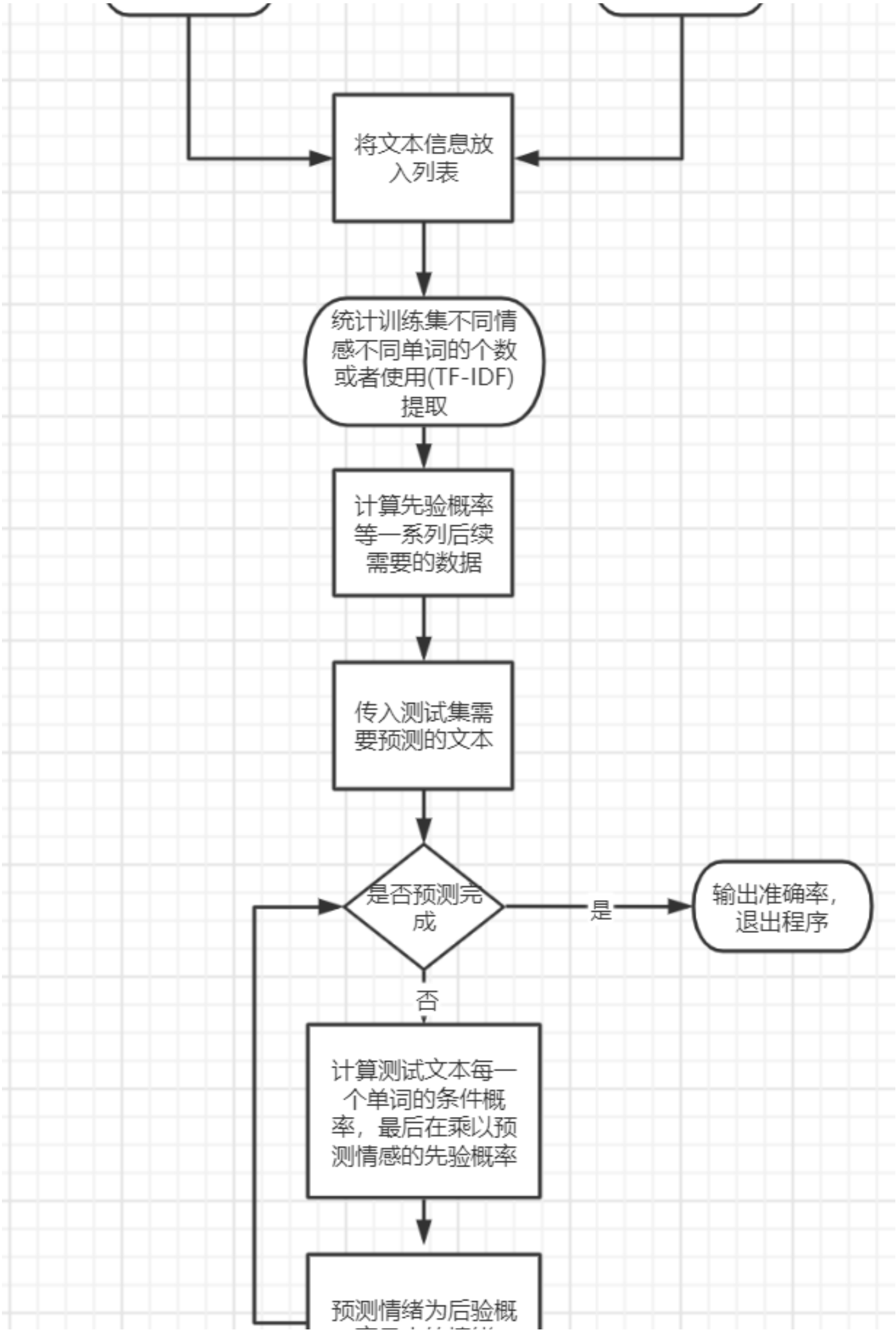
于是，朴素贝叶斯分类器可表示为：

$$f(x) = \arg\max_{y_k} P(y_k|x) = \arg\max_{y_k} \frac{P(y_k) \times \prod_{i=1}^{n} P(x_i|y_k)}{\sum_k P(y_k) \times \prod_{i=1}^{n} P(x_i|y_k)}$$

因为对所有的yk，上式中的分母的值都是一样的，所以可以忽略分母部分，朴素贝叶斯分类器最终表示为：

$$f(x) = \arg\max_{y_k} P(y_k) \times \prod_{i=1}^{n} P(x_i|y_k)$$

2.流程图


将文本信息放入列表
统计训练集不同情感不同单词的个数或者使用(TF-IDF)提取
计算先验概率等一系列后续需要的数据
传入测试集需要预测的文本
是否预测完成
是
输出准确率，退出程序
否
计算测试文本每一个单词的条件概率，最后在乘以预测情感的先验概率
预测情绪为后验概

3.关键代码展示(本次实验写了利用单词频次(出现次数)和利用TfidfVectorizer的两份代码，这里展示TfidfVectorizer的代码)

1.读取文本信息

```
def read_file(f):#读取文本信息
    totalnum=0#不同单词的数目
    d={}#每种情感不同单词的数目
    d_total={}#辅助计算不同单词的数目
    category_num=[0 for i in range(6)]#不同情感的文本数目
    total=[0 for i in range(6)]#每种情感不重复单词的数目

    data=[]#记录文本信息
    sentence=[]#记录文本
    for line in f:
        tmp=line.strip().split(" ")
        if tmp[0]=="documentId":#跳过无关信息
            continue
        tag=int(tmp[0])#标号
        category=int(tmp[1])#情感标号
        emotion=tmp[2]#情感
        sentence_tmp=''
        category_num[category-1]+=1
        for i in range(3,len(tmp)):
            if tmp[i] not in d_total:
                d_total[tmp[i]]=1
                totalnum+=1
            if (tmp[i],category-1) in d:
                d[(tmp[i],category-1)]+=1
                total[category-1]+=1
            else :
                d[(tmp[i],category-1)]=1
            sentence_tmp+=tmp[i]
            if i!=len(tmp)-1:sentence_tmp+=" "
        sentence.append(sentence_tmp)
        data.append([tag,category,emotion,sentence_tmp])
    return data,sentence,d,total,totalnum,category_num
```

2. TfidfVectorizer提取文本特征

```
#TfidfVectorizer提取文本信息，得到train_matrix计算
t=TfidfVectorizer()
train=t.fit_transform(train_sentence)
train_matrix=train.toarray()
```

3. 计算后续需要的数据

```
#将测试集文本放入列表中
for i in range(len(test_sentence)):
    tmp=test_sentence[i].split(' ')
    test_sentence[i]=tmp
#计算一些准备用到的数据
prob_train=[0 for i in range(6)]#训练集每个情感的单词的tdidf和
prob_word=[[0]*len(train_matrix[0]) for i in range(6)]
pro_total_word=[0 for i in range(len(train_matrix[0]))] #训练集每个单词的tiidf和
for i in range(len(train_matrix)):#计算数据
    for j in range(len(train_matrix[i])):
        prob_train[train_data[i][1]-1]+=train_matrix[i][j]
        prob_word[train_data[i][1]-1][j]+=train_matrix[i][j]
        pro_total_word[j]+=train_matrix[i][j]

sum=0
for i in range(6):
    sum+=prob_train[i]#整个文档的tdidf总和
```

4. 朴素贝叶斯情感分类

```
correct_total=0#总共预测对的数目
correct_pred=[0 for i in range(6)]#每种情感预测对的数目
test_category=[0 for i in range(6)]#测试集每种情感的数目

for i in range(len(test_data)):
    pre=-1#预测的情感标号
    max=0#后验概率
    test_category[test_data[i][1]-1]+=1
    vocabulary=t.vocabulary_#TfidfVectorizer()的字典
    for j in range(6):
        prob=1
        for k in range(len(test_sentence[i])):
            if test_sentence[i][k] in vocabulary:#在预测的情感文本单词集中
                if prob_word[j][vocabulary[test_sentence[i][k]]]!=0:
                    #prob_word[j][vocabulary[test_sentence[i][k]]]该单词在预测情感中的tdidf和
                    #prob_train[j]预测情感的tdidf和
                    #total1训练集单词集合大小
                    #train_category_num[j]预测情感的单词集合大小
                    prob_tmp=(prob_word[j][vocabulary[test_sentence[i][k]]]+lam)/(prob_train[j]+train_category_num[j]*lam)
                else :#不在预测的情感文本单词集中，在训练集单词集合中
                    prob_tmp=(prob_word[j][vocabulary[test_sentence[i][k]]]+lam)/(prob_train[j]+total1*lam)
            else :#不在训练集单词集合中，计算时忽略
                prob_tmp=1
            prob*=prob_tmp#prob_tmp每个单词的条件概率
        prob*=train_category_num[j]/len(train_data)#乘以先验概率
        if max<prob:
            max=prob
            pre=j
    if pre==test_data[i][1]-1:
        correct_total+=1
        correct_pred[pre]+=1
```

5. 输出结果

```
#输出结果
for i in range(6):
    print(emotion[i+1],":",correct_pred[i]/test_category[i])
print("Total accuracy:",correct_total/len(test_data))
```

4.创新点&优化

无

三、实验结果及分析

1. 实验结果展示示例

对于选取不同的λ值和选用计算方法(利用单词的频次计算和利用TfidfVectorizer计算）得到的准确率的结果放在了result文件中。同时也将实验结果放入评测指标中进行对比展示。

2. 评测指标展示及分析

(1)利用单词的频次计算

λ	anger	disgust	fear	joy	sad	surprise	total
1	0.2878787878787879	0.6923076923076923	0.09375	0.24033149171270718	0.16831683168316833	0.09782608695652174	0.191
0.1	0.19696969696969696	0.4230769230769231	0.275	0.4005524861878453	0.36633663366336633	0.16304347826086957	0.317
0.01	0.19696969696969696	0.38461538461538464	0.29375	0.4171270718232044	0.3811881188118812	0.16304347826086957	0.328
0.001	0.18181818181818182	0.38461538461538464	0.2875	0.4198895027624309	0.3910891089108911	0.16304347826086957	0.329

(2)利用TfidfVectorizer计算

λ	anger	disgust	fear	joy	sad	surprise	total
1	0.21212121212121213	0.4230769230769231	0.29375	0.3867403314917127	0.3712871287128713	0.16847826086956522	0.318
0.1	0.16666666666666666	0.4230769230769231	0.325	0.4088397790055249	0.3910891089108911	0.18478260869565216	0.335
0.01	0.18181818181818182	0.4230769230769231	0.3	0.4005524861878453	0.3910891089108911	0.16847826086956522	0.326
0.001	0.16666666666666666	0.4230769230769231	0.2875	.40607734806629836	0.3910891089108911	0.15760869565217392	0.323

(3)λ取0.001,将给定的测试集和训练集反过来

计算方式	anger	disgust	fear	joy	sad	surprise	total
单词的频次	0.047619047619047616	0.125	0.4117647058823529	0.6075949367088608	0.4126984126984127	0.21212121212121213	0.3983739837
TfidfVectorizer	0.047619047619047616	0.125	0.4411764705882353	0.5949367088607594	0.42857142857142855	0.21212121212121213	0.4024390243

分析：

(1)从上述数据的分析中可以看出,基本上当λ越小时，预测情感的准确率会提高，这是因为加λ的作用是避免出现0次的单词对最终概率的影响，所以λ只要大于0就能发挥这作用，但当λ越大会对原来的概率造成较大的影响，从而影响预测的准确率。

(2)从两种计算方式的比较，可以发现TfidfVectorizer计算受λ的变化影响较小。

(3)将测试集和训练集反转运行对比，发现准确率提高了许多，这是因为训练集元素个数提高了，训练集反馈给我们的提取的信息更多了，有利于我们做出更到的预测。

四、参考资料

实验课PPT

【朴素贝叶斯】深入浅出讲解朴素贝叶斯算法（公式、原理）:https://blog.csdn.net/kevinjin2011/article/details/125099177