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	人工智能（陈川）	专业（方向）	计算机科学与技术人工智能与大数据
学号	20337025	姓名	崔璨明

一、实验题目

在给定的文本数据集完成文本情感分类训练，在测试集完成测试，计算准确率。

1. 文本的特征可以使用TF-IDF或TF，对TF均使用拉普拉斯平滑技巧（可以使用sklearn库提取特征）。
2. 利用朴素贝叶斯完成对测试集的分类，并计算准确率。
3. 需要提交简要报告+代码。

二、实验内容

1、算法原理

TF-IDF:

TF（Term Frequency，缩写为TF）指词频，即一个词在文中出现的次数。计算公式为$TF=某个词在文章中的出现次数/文章总词数$。

IDF（Inverse Document Frequency），指逆文档频率，它的大小与一个词的常见程度成反比。计算公式为：$IDF=log(文档总数/包含该词的文档数+1)$。如果一个词越常见，那么分母就越大，逆文档频率就越小越接近0。分母之所以要加1，是为了避免分母为0（即所有文档都不包含该词）。

TF-IDF值即为将TF值和IDF值相乘。

拉普拉斯平滑：

在文本分类算法中，如果测试文本中的单词没有在训练文本中出现则会影响到后验概率的计算结果，使分类产生偏差。解决这一问题的方法是采用贝叶斯估计。具体方法为：

$p(x_k|d_j,e_i)=\frac{x_k+\lambda}{\sum_{k=1}^{K}x_k+K\lambda}$ 式中 $\lambda \geq 0$。等价于在随机变量各个取值的频数上赋予一个 正数 $\lambda \geq 0$。当 $\lambda = 0$ 时就是极大似然估计。尝取 $\lambda = 1$，这 时称为拉普拉斯平滑。

朴素贝叶斯算法：

朴素贝叶斯算法的核心为贝叶斯公式：$P(B|A)=\frac{P(A|B)P(B)}{P(A)}$，具体做法是根据贝叶斯定理来估计每个类别的后验概率，然后比较属于各个情感类别的概率，选取最大者作为判断的结果。

2、算法伪代码

计算TF-IDF矩阵的伪代码：

```
输入：句子集set；未初始化的TF-IDF矩阵arr,矩阵的行编号代表句子编号，列编号代表单词编号；单词数组name，保存编号到单词的映射。
输出：计算后的TF-IDF矩阵
def caculate_TF():
for i1 in range(len(arr)):
        for i2 in range(len(arr[i1])):
                total_len=len(set[i1]) #该句子的长度
                cnt=0 #初始化计数为0
                for i3 in range(total_len):
                        if set[i1][i3]==name[i2]:
                                cnt+=1     #若在句子set[i1]中找到单词，则计数加一
                arr[i1][i2]=cnt
#计算每个单词对应的IDF
dic1={}
all=len(anger_set)
for index1 in range(0,len(name1)):
        appear_times=0
        for sentance in anger_set:
                if name1[index1] in sentance:
                        appear_times+=1
        dic1[name1[index1]]=float(math.log10(all/(appear_times+1)))

#TF矩阵乘上idf值得到tf-idf矩阵
for i in range(0,len(name1)):
        for j in range(0,len(arr1)):
                arr1[j][i]*=dic1[name1[i]]
```

对矩阵进行归一、拉普拉斯平滑操作的伪代码：

```
输入：TF矩阵arr,矩阵的行编号代表句子编号，列编号代表单词编号
输出：经过归一操作和拉普拉斯平滑后的TF矩阵
```

```
def fun():
sum=0
for i in range(len(arr)):
        sum=0
        for j in range(len(arr[i])):
                        sum+=arr[i][j]
        sum+=len(arr[i]) #加上K,K即特征维数
        for j in range(len(arr[i])):
                arr[i][j]=(arr[i][j]+0.0001)/(0.0001*sum)#对每个值进行拉普拉斯平滑的归一
```

朴素贝叶斯分类算法伪代码：

```
输入：待分类的句子test_sentabce，
输出：将该句子划分的类别（1，2，3，4，5，6分别代表anger、disgust、fear、joy sad、surprise）
def classify(test_sentabce):
max=-9999
flag=-1
for i in range(6):
        res=caculate(i,arr,test_sentabce,name,count)#计算属于第i类的概率
        if res>max:
                max=res
                flag=i  #保存概率最大值
return flag+1

#计算概率
def caculate(num,arr,test_sentabce,name,count):
res=0
for i in range(0,len(arr[num])):
        mul=1
        for j in range(0,len(test_sentabce)):#累乘
                if test_sentabce[j] in name[num]:
                        now=name[num].index(test_sentabce[j])
                        mul*=arr[num][i][now]
                else:
                        continue
        res+=mul
        res=res*count[num]#根据公式，乘上先验概率
return res
```

3、关键代码展示

计算TF-IDF矩阵的关键代码如下，首先调用`sklearn`库中的模块`CountVectorizer`生成TF矩阵和对应的单词名字矩阵，然后统计次数、计算每个词的idf值，最后相乘得到tf-

idf矩阵：

```
    #得到TF矩阵
    vectorizer = CountVectorizer()
    X1 = vectorizer.fit_transform(anger_set)
    name1=vectorizer.get_feature_names_out().tolist()
    arr1=X1.toarray().tolist()
    #归一和平滑的过程
    sum=0
    for i in range(len(arr1)):
        sum=0
        for j in range(len(arr1[i])):
            sum+=arr1[i][j]
        sum+=len(arr1[i])
        for j in range(len(arr1[i])):
            arr1[i][j]=(arr1[i][j]+1)/(0.0001*sum)
    #计算idf
    dic1={}
    all=len(anger_set)
    for index1 in range(0,len(name1)):
        appear_times=0
        for sentance in anger_set:
            if name1[index1] in sentance:
                appear_times+=1
        dic1[name1[index1]]=float(math.log10(all/(appear_times+1)))
    #乘上idf，得到tf-idf矩阵
    for i in range(0,len(name1)):
        for j in range(0,len(arr1)):
            arr1[j][i]*=dic1[name1[i]]
```

依照伪代码，编写基于朴素贝叶斯分类进行分类的关键代码如下：

```
    #-------------------------------------------------------------------------------
    file = open("test.txt","r")
    test_list = file.readlines()#每一行数据写入到test_list中
    file.close()
    right=0     #统计正确的数量
    wrong=0     #统计错误的数量
    for i in range(1,len(test_list)):
        test_list[i]=test_list[i].strip("\n")
        tmp1=test_list[i].split()
        tmp2=tmp1[2:len(tmp1)]
        correct=tmp1[1]
        charge=classify(arr, tmp2, name)    #分类
        if charge == int(correct):
            right+=1
        else:
            wrong+=1
        print('判断:',charge,'正确:',int(correct),tmp2)#输出分类结果
    print("rate: ",100*right/len(test_list),"%",sep='')#输出正确率
```

其中，计算概率总和的函数 caculate() 和分类函数classify() 的实现如下：

```
def caculate(num,arr,test_sentabce,name,count):
    res=0
    for i in range(0,len(arr[num])):
        mul=1
        for j in range(0,len(test_sentabce)):
            if test_sentabce[j] in name[num]:
                now=name[num].index(test_sentabce[j])
                mul*=arr[num][i][now]
            else:
                continue
        res+=mul
    #print(res)
    res*=count[num]
    return res


def classify(arr,test_sentabce,name,count):
    max=-9999
    flag=-1
    for i in range(6):
        res=caculate(i,arr,test_sentabce,name,count)
        if res>max:
            max=res
            flag=i

    print("max:",max)
    return flag+1
```

三、实验结果分析&&对比

经过TF值和TF-IDF值的对比实验，我最终采用了使用TF值来作为文本特征值的方法，且对原有程序进行了改进，按照实验题目的实验要求，使用训练集train.txt训练模型，然后在测试集test.txt上进行测试，当采用拉普拉斯平滑时，最终得到的**分类正确率约为36.00%**，实验结果如下：

```
问题  输出  终端  调试控制台

判断: 3 正确: 5 ['dryer', 'blame', 'for', 'fire', 'kill', 'cat', 'dog']
判断: 4 正确: 4 ['roddick', 'murrai', 'score', 'in', 'san', 'jose']
判断: 4 正确: 4 ['appl', 'trademark', 'disput', 'settl']
判断: 4 正确: 5 ['u', 's', 'onlin', 'love', 'broker', 'ey', 'china']
判断: 5 正确: 1 ['u', 's', 'divert', 'troop', 'to', 'fight', 'taliban']
判断: 4 正确: 5 ['hous', 'of', 'card', 'actor', 'ian', 'richardson', 'dead']
判断: 5 正确: 5 ['taliban', 'leader', 'kill', 'in', 'airstrik']
判断: 4 正确: 4 ['escap', 'to', 'pragu', 'without', 'the', 'summer', 'hord']
判断: 4 正确: 6 ['kathmandu', 'first', 'snow', 'in', 'year']
判断: 4 正确: 5 ['nasdaq', 'fail', 'in', 'bid', 'for', 'lse']
判断: 5 正确: 5 ['ex', 'pastor', 'get', 'death', 'sentenc']
判断: 5 正确: 6 ['babi', 'born', 'on', 'turnpik', 'after', 'dad', 'miss', 'exit']
判断: 3 正确: 6 ['studi', 'link', 'chimp', 'and', 'hammer']
判断: 6 正确: 4 ['un', 'googl', 'earth', 'map', 'climat', 'chang']
rate: 35.964035964035965%
PS E:\VSCODE\py>
```

四、创新点&优化：

1、增加停用表

在原有的程序基础上增加了英文单词停用表，即在原始文本集中去掉不需要的词汇，例如和情感无关的介词、主语、连接词等等，参考哈工大的一个停用词表，将其导入到一个停用列表delset中，具体代码如下：

```
delset=[]
file =open("ban.txt","r")
dlist=file.readlines()
file.close()
for i in range(0,len(dlist)):
    dlist[i]=dlist[i].strip("\n")
    delset.append(dlist[i])
```

在导入训练集句子时，对其中单词进行判断即可，若某一单词存在于停用集，则删除该单词。

2、改善参数λ

对TF值进行拉普拉斯平滑的公式为：$p(x_k|d_j,e_i)=\frac{x_k+\lambda}{\sum_{k=1}^{K}x_k+K\lambda}$，其中$\lambda$取值为1，但经过多次实验，发现$\lambda$=1并不是效果最好的参数，最后经过多次调试，取$\lambda$=0.000001，准确

率有了提高：

```
判断：3 正确：5 ['dryer', 'blame', 'for', 'fire', 'kill', 'cat', 'dog']
判断：4 正确：4 ['roddick', 'murrai', 'score', 'in', 'san', 'jose']
判断：4 正确：4 ['appl', 'trademark', 'disput', 'settl']
判断：4 正确：5 ['u', 's', 'onlin', 'love', 'broker', 'ey', 'china']
判断：5 正确：1 ['u', 's', 'divert', 'troop', 'to', 'fight', 'taliban']
判断：4 正确：5 ['hous', 'of', 'card', 'actor', 'ian', 'richardson', 'dead']
判断：5 正确：5 ['taliban', 'leader', 'kill', 'in', 'airstrik']
判断：4 正确：4 ['escap', 'to', 'pragu', 'without', 'the', 'summer', 'hord']
判断：4 正确：6 ['kathmandu', 'first', 'snow', 'in', 'year']
判断：4 正确：5 ['nasdaq', 'fail', 'in', 'bid', 'for', 'lse']
判断：5 正确：5 ['ex', 'pastor', 'get', 'death', 'sentenc']
判断：5 正确：6 ['babi', 'born', 'on', 'turnpik', 'after', 'dad', 'miss', 'exit']
判断：5 正确：6 ['studi', 'link', 'chimp', 'and', 'hammer']
判断：6 正确：4 ['un', 'googl', 'earth', 'map', 'climat', 'chang']
rate: 36.06393606393606%
PS E:\VSCODE\py>
```

五、参考资料

- [哈工大停用词表](GitHub - goto456/stopwords: 中文常用停用词表（哈工大停用词表、百度停用词表等）)