作业一 实验报告

一、 实验目的

对 20 Newsgroups dataset 文档集建立 VSM 模型，用 KNN 实现分类，通过 N-fold 交叉验证的方式研究 k 的取值对分类结果的影响。

二、 实验环境

Windows 10 + Python3.6.5

三、 实验步骤

首先对文档进行初始化操作，包括去除停用词，过滤非字母字符，提取词干等。实验中是使用 nltk 工具包中的函数实现的。

```
def lineProcess(line):
    line = re.sub(r'[^a-zA-Z]'," ",line)
    stopwords = nltk.corpus.stopwords.words('english') #去停用词
    porter = nltk.PorterStemmer() #提取词干
    splitter = re.compile('[^a-zA-z]') #过滤非字母字符
    words = [porter.stem(word.lower()) for word in line.split(" ") if len(word) > 0 and word.lower() not in stopwords
    return words
```

将初始化后的文档保存，用于下一步词典的构建。在构建词典时，首先选取每个类别前 80%的文档，过滤掉出现次数小于 10 的词（避免词典规模过大），计算剩下所有词的文档频率（DF）并保存。

然后对每一篇文档计算其向量表示，具体做法为初始化一个矩阵 W，行向量代表词典中的词，列向量代表文档索引，W[i,j]保存的是词典中第 i 个词在第 j 篇文档中的 TF×IDF 值，计算公式为：

$$tf(t,d) = c(t,d)$$
$$IDF(t) = \log(\frac{N}{df(t)})$$

其中 t,d,c(t,d),N 分别代表词的索引，文档索引，文档长度和总文档数

```
for i in range(len(data)):
    print("构建完成{0}%".format((i + 1) * 5))
    cateDir = 'data/' + data[i]
    cateList = listdir(cateDir)
    for j in range(len(cateList)):
        article = open('data/' + data[i] + '/' + cateList[j], encoding='ISO-8859-1').readlines()
        article = [word.split()[0] for word in article]
        for word in article:
            if word in lexicon:
                tf = article.count(word) / len(article)
                w[lexicon.index(word), num] = tf * np.log(N / df[word])
        label[num] = i
        num += 1
```

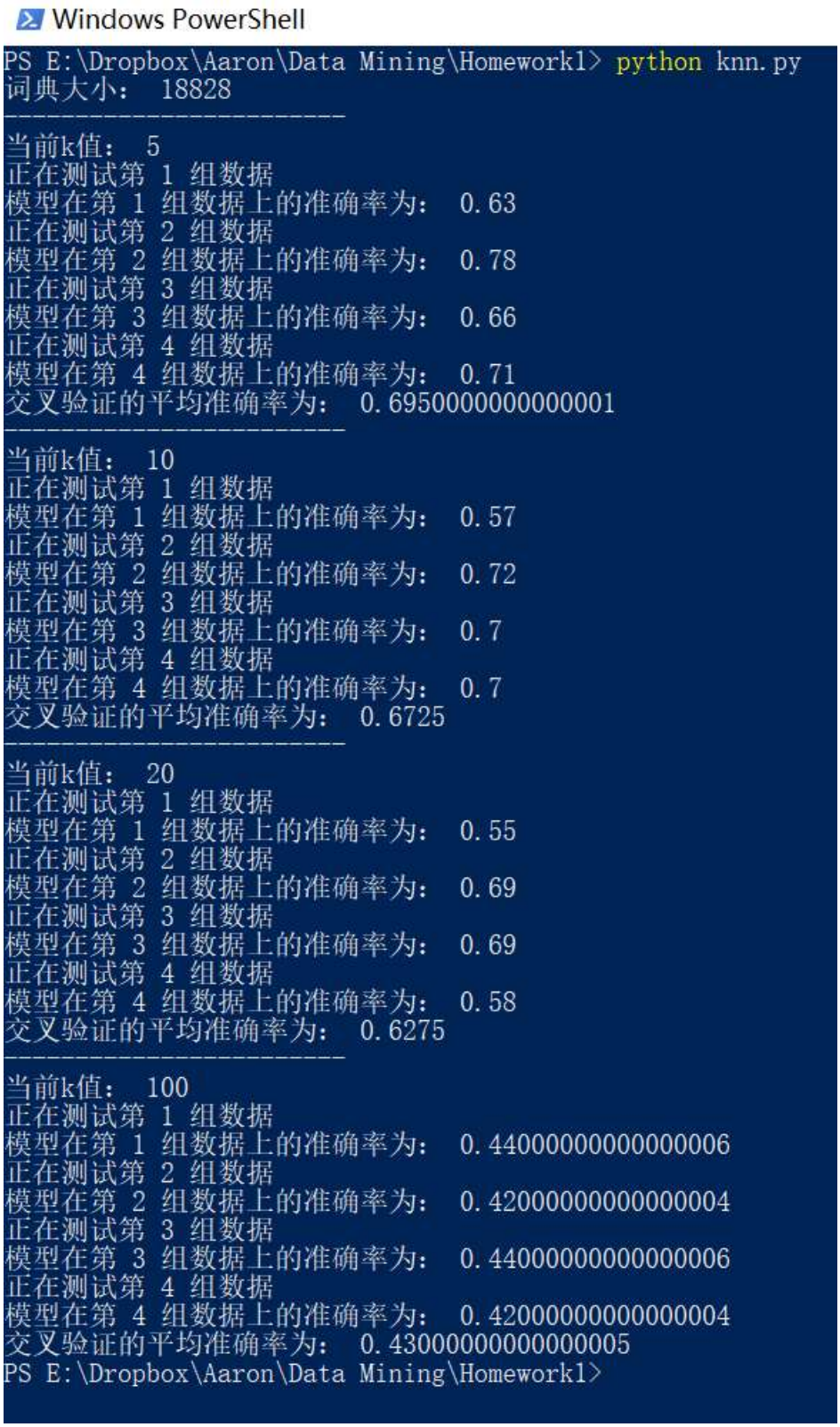
至此已经成功建立了向量空间模型，接下来使用 KNN 进行文本分类。实验使用 4 折交叉验证的方式进行，将所有文档分成 4 份，每次取 3 份作为测试集，1 份作为训练集，KNN 中的参数 K 分别取 5,10,20,100 进行实验。

对于测试集的每个向量，与训练集中所有向量分别计算余弦相似度，取相似度最高的 K 个样本，对他们的类别标签进行投票，票数最多的类被当作该测试样本的类别

$$cosine(d_i, d_j) = \frac{V_{d_i}^T V_{d_j}}{|V_{d_i}|_2 \times |V_{d_j}|_2}$$

```python
def predict(train, train_label, test, test_label, k):
    result = np.zeros([test.shape[1],])
    #for i in range(train.shape[0]):
        #print(test[i,0], train[i,0])
    for i in range(test.shape[1]):
        similarity = np.zeros([train.shape[1]])
        for j in range(train.shape[1]):
            similarity[j] = similar(test[:,i], train[:,j])
        closest_y = []
        #print(similarity[np.argsort(similarity)[len(similarity)-k:]])
        closest_y = train_label[np.argsort(similarity)[len(similarity)-k:]].flatten()
        #print(closest_y)
        c = Counter(closest_y)
        result[i]=c.most_common(1)[0][0]
    return result
```

四、 实验结果



可以看出，k 的选取不宜过大，当 k 增加时准确率逐渐降低，但在小范围内的变化对分类结果影响不大。