实验报告

1. 构建向量空间模型（VSmodel）

1.1 数据预处理

读取文档，利用 nltk 相关工具对文档进行分词处理，然后将大写变为小写，去停用词，并对单词进行词干提取和词形还原的操作（不考虑顺序）。在这之后，统计词频并将结果写入建好的 train_make 和 test_make 中的相应文档，同时建立字典 allword。

1.2 TF-IDF 算法

W=TF * IDF,利用 TF-IDF 算法，将数据进一步处理，并根据字典将文档构建成向量，写进相应的文件夹：向量和向量_test。由于数据量庞大，字典较大，可以通过过滤 W 的大小来进行筛选，设置不同的 W 值，字典的大小会相应的变化。选取合适的 W 值来确定合适的字典。

2. KNN 算法

2.1 整体思想：

是通过测量不同特征值之间的距离进行分类。采用余弦距离的近邻度量的方法，利用多维公式计算距离。公式如下：

2.余弦值（Cosine）

我们学过二维空间两个向量的夹角余弦公式为：

$$\cos\theta = \frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2}\sqrt{x_2^2 + y_2^2}}$$

扩展到多维空间中，对于 $\mathbf{x}_i = (x_i^{(1)}, x_i^{(2)}, x_i^{(3)}, \ldots, x_i^{(n)})^T$ 和 $\mathbf{x}_j = (x_j^{(1)}, x_j^{(2)}, x_j^{(3)}, \ldots, x_j^{(n)})^T$，同样可以计算多维向量的余弦值：

$$\cos\theta = \frac{\sum_{k=1}^{n} x_{1k} x_{2k}}{\sqrt{\sum_{k=1}^{n} x_{1k}^2}\sqrt{\sum_{k=1}^{n} x_{2k}^2}}$$

当两个向量的方向重合时夹角余弦取最大值 1，此时两个向量的相似性最高，当两个向量的方向完全相反夹角余弦取最小值-1，此时两个向量的相似性为负。

2.2　K 值的选择：

K 值一般取一个比较小的数值，通常采用交叉验证法（简单来说，就是一部分样本做训练集，一部分做测试集）通过观察 K 值不同时模型的分类效果来选择最优的 K 值。

2.3　K 值的不同，正确率的差异如下：

2.3.1 K=3

```
/usr/local/bin/python3.7 /Users/ima/PycharmProjects/201844897fangguimei/
正确率: 0.7584745762711864

Process finished with exit code 0
```

2.3.2　K=7

```
/usr/local/bin/python3.7 /Users/ima/PycharmProjects/201844897fangguimei
正确率: 0.7641242937853108

Process finished with exit code 0
```

2.3.3 K=10

```
/usr/local/bin/python3.7 /Users/ima/PycharmProjects/201844897fangguimei
正确率: 0.7796610169491526

Process finished with exit code 0
```

2.3.4 K=15

```
/usr/local/bin/python3.7 /Users/ima/PycharmProjects/201844897fangguimei/
正确率: 0.769774011299435

Process finished with exit code 0
```

3. 实验总结

在构建字典的时候，由于字典过大，在后续的构建向量空间模型的时候，十分缓慢，等待时间太长。所以在计算 w=tf *idf 之后，通过限制 w 的大小来控制字典的大小。

将训练集与测试集划分之后，又分别利用 vmc 来进行数据预处理，tf,idf 的计算等等构建向量空间模型，可尝试将数据统一处理，构建向量空间模型之后再划分。

KNN 分类算法在运行时十分缓慢。所以在查看不同的 K 值的情况的好坏时耗费了大量的时间。并没有实现将训练数据划分成 N 份，其中 N-1 份用来训练，1 份用来测试。只是通过测试数据与训练数据整体划分来进行整个实验。在尝试用 KD 树来优化 KNN 算法是未成功实现。