VSM_KNN 实验报告

一、实验内容

1、构建 VSM

首先读取每个文本，对其分词、去停用词等预处理，初步建立词典。同时计算每个文本中各单词出现的频率。因为频率受文本大小的影响，所以使用 Sub-linear TF scaling 公式对其标准化。

其次，遍历词典，计算每个单词的 IDF，从而根据 w=IF*IDF 得到单词的权重。

最后，设置权重阈值 w_min 重新建立词典，单词权重大于 w_min 的放入单词中，从而达到缩小词典规模的效果，再根据词典构建文本的向量。

表 1.1 阈值 w_min 与词典规模的对应关系

阈值 w_min	词典规模（个）
0	110000
5	50586
15	31472
20	6184
25	2200

2、KNN

按照 9/1 的比例划分训练集和测试集，使用向量间的余弦值度量相似性。计算测试集样本与训练集每个样本的相似度，按照相似度从

大到小排序后，看排在前 k 个的训练集样本属于哪个类型的最多，就把测试集样本分到哪个类型。K 值设定为 9，得到分类正确率 0.763，k 值最好选奇数，为了避免出现“平局”情况。

二、实验日志

1、遇到的困难

(1)、文本数量太多，构建 VSM 时生成的词典规模过大，算起来非常慢。

(2)、KNN 分类时，运行时间特别长。

2、解决方案

(1)、对于词典规模过大的问题，可以通过调整最小权重阈值，从全局考虑过滤词典，从而达到缩小词典规模的目的。

(2)、对于 KNN 分类运行时间过长的问题，有两种优化方案：一是划分训练集和测试集的时候，缩小测试集的规模；二是使用部分训练数据估算分类的准确率。我采用的是第一种方案把测试集的规模从 2/8 缩小到 1/9。

NBC 实验报告

一、实验内容

首先读取每个文本的内容，进行分词、去停用词等预处理，同时按一定比例划分训练集和测试集。在这里我是把每个文件夹下前 90%的文本作为训练集，剩下的作为测试集。程序中函数 data_process 完成这些工作。函数返回两个字典：dict_train,dict_test。Dict_train 中 key 为文件夹名（类名），value 值是个二级字典，二级字典 key_2:value_2 对应该文件夹下所有单词及其频率。dict_test 的 key 是每个文本的路径，value 也是个二级字典，二级字典的 key_2:value_2 对应该文本所有的单词及其频率。

然后，对测试集进行分类，具体就是计算测试文本分到每个类的概率，将该文本分到概率最高的那个类。我分别使用了多项式模型和伯努利模型，分类的正确率分别为 82.8%和 84.3%。

二、实验日志

NBC 相对于 KNN 简单些，但我却用了挺长时间才完成。因为使用字典的 clear()不当，导致程序的运算结果及其不合理。时间都用在定位这个 bug 上面了。最后发现 python 中有个容易犯错的地方：一个对象同时又充当了类似于指针或引用的功能。所以 clear 一个字典的同时，会影响之前的赋值。

之所以出现上述错误，还是因为对 python 了解的不够所以今后还要继续深入的学习 python。

Sklearn 实验报告

一、实验步骤

首先处理给定文本，把其中”text”对应的内容放到 Data 这个列表中，同时把”cluster”对应的数据提取出来，放到 Labels 这个列表中，然后，使用 TfidfVectorizer 把 Data 中的元素转为向量形式表示，最后依次调用 sklearn 中的聚类方法，用 NMI 评价聚类结果。

表 1.1 各种聚类方法的 NMI

聚类方法	评价指标 NMI
K-Means	0.7830779694677665
Affinity propagation	0.7836988975391975
Mean-shift	-0.7265625
Spectral clustering	0.6528537755966615
Ward hierarchical clustering	0.7823244114906182
Agglomerative	0.6198448865824127
DBSCAN	0.8962440814686098

二、实验日志

这些聚类方法大部分只需要指定分类的数量 n_cluster 这个参数，其余使用默认值就可以。但是 DBSCAN 使用默认参数值分类效果就很差，默认值为：eps=0.5，min_simples=5，将这两个参数分别调整为 1.13，6 后效果最佳。