Data Mining 实验报告

张浩 计算机技术 专硕班 201834886

实验名称

实验二 朴素贝叶斯分类器实现文本分类

实验要求

实现朴素贝叶斯分类器，测试其在 20Newsgroups 上的效果。

实验环境

硬件环境: Intel(R) Core(TM) i5-6260U CPU @1.8GHz 4G RAM
软件环境: windows 10 1803
编程语言: python 3.7

实验数据

The 20 Newsgroups dataset is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups.
20news-18828.tar.gz - 20 Newsgroups

数据计算方式

贝叶斯公式

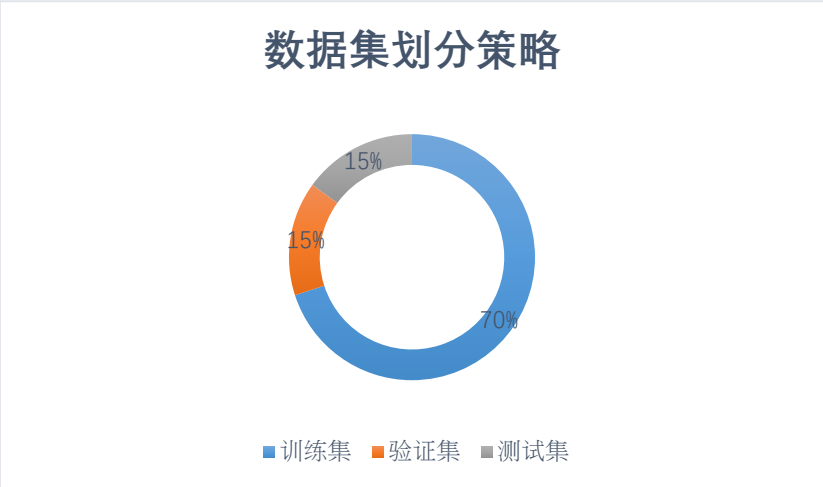
$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

先验概率 P(h)
P_h = [训练集该类文本的数量]/[训练集所有文本的数量]
条件概率(多项式模型)
P_D_h = ([训练集中出现该词的频数(包括重复)]+0.00001)/([该类文本的总长度]+[当前分类文本长度])
为了防止出现分子为 0 而使得 P(D1|h) P(D2|h) P(D3|h) P(D4|h) P(D5|h)…… P(Dn|h)为 0 的情况，进行了平滑处理。
为了方便计算对分子取对数运算
math.log(P(D1|h))+ math.log(P(D2|h))+ math.log(P(D3|h))+ math.log(P(D4|h))+……+ math.log(P(Dn|h))+ math.log(P(h))

实验过程

划分数据集测试集

将数据集划分为两类
训练集
作用：统计词频
验证集
作用：调整词典过滤系数 dict_f 的取值大小，优化分类器模型
[在建立词典时，根据该词在当前文本词频过滤某些单词]
测试集
作用：检验分类器性能



划分数据集时使用其在文件夹名字顺序选取（即选区前 n 个文件）因为文本存放方式与其内容直接不存在直接的关系，所以这样的选择方法可以视为随机选择。

规则化数据集文本

在建立词典时首先要考虑每个 word 之间尽量不要在意思上重复，所以要提取词干，即单复数、大小写、时态的转化。其次，
考虑因素（单复数、大小写、时态、特殊符号）
注意：词干提取时常用方法有

1. **Porter Stemmer 基于 Porter 词干提取算法**
2. **Lancaster Stemmer 基于 Lancaster 词干提取算法**
3. **Snowball Stemmer 基于 Snowball 词干提取算法**
   本次实验使用的是 Porter Stemmer 基于 Porter 词干提取算法，在实验时使用不同的算法最终效果也不相同，这个有待进一步实验。

建立各类文本词典

建立词典策略
单词长度大于 2 AND 小于 14 的词 AND 非数字的词 AND 词频大于当前类型文本长度* dict_f(dict_f 为过滤系数)

合并各类词典，建立全局词典

统计全局词典词频

计算先验概率

P_h = [训练集该类文本的数量]/[训练集所有文本的数量]

读入验证集数据，统计实验结果优化 dict_f 的取值

dict_f 从 0.000002 ~~ 0.0002 步长为 0.000002 选择正确率最高的值作为 dict_f
[当参数为 0.00002 正确率最高为 85.19138755980862 %]

统计测试集正确率

实验结果

当 dict_f = 0.00002　　正确率为　　86.6267942583732 %

总结

1. 由于第一次作业积累了一些 python 的基础，这次作业完成的比较顺利。在规则化文本，过滤词典上有了一定的经验，并且在该基础上进行了改进，上次时根据某类文本选择一个固定词频进行过滤词典，这次实验，采用了一个过滤系数，根据当前文本长度过滤掉出现次数小于当前文本长度一定百分比的词。相比于固定频率，这个方法更合适。

附录（主要函数说明）

```
#按照类型归类文本
Texts = travel_all_file_merge(file_dir_train)
#计算总词数
all_words_f = count_all_words(Texts)
#分别建立词典
Dict_all = build_dict(Texts)
#统计各词典词频
Dict_count = build_dict_count_text(Texts,Dict_all)
#建立统一词典
Dict = dict_marge(Dict_count)
#统计统一词典词频
Dict_fin = fin_dict(Dict,Texts)
#词典大小
dict_f = len(Dict_fin[0])
#先验概率
P_h = p_h_compute(Label)
#建立词典字母与序号映射
dict_map = dict_map_num_name()
#读取测试集
Texts_test,Label_test = travel_all_file(file_dir_test)
#测试集范围
range_test = Find_Label_range(Label_test)
#统计正确率
test_rate(Texts_test)
```