山东大学 计算机科学与技术 学院

信息检索与数据挖掘 课程实验报告

学号：201600130077	姓名： 于海洋	班级： 16 人工智能
实验题目：Classification-NBC		

实验内容:

1、 测试集合训练集的建立

从每个类中选取一定比例的文件，加入 test_set，剩余的加入 train_set，分别作为训练集和测试集并记录测试集数量，以便后面做各种 rate 的计算

```
def get_set():
    train_num = {}
    file_list = []
    root_1dir = r"20news-18828"
    for pack in os.listdir(root_1dir):
        class_dir = root_1dir + "\\" + pack
        for file in os.listdir(class_dir):
            file_list.append(file)
        len_class_doc = len(file_list)
        train_num[pack] = len_class_doc
        file_list.clear()
        num_doc = 0
        for file in os.listdir(class_dir):
            file_name = class_dir + "\\" + file
            num_doc += 1
            if num_doc <= train_num[pack]/5:
                test_set.append(file_name)
            else:
                train_set.append(file_name)
```

2、 朴素贝叶斯各数值的获取:

$$P(\text{类}|\text{词}) = \frac{P(\text{词}|\text{类}) * P(\text{类})}{P(\text{词})}$$

P（类）均为 1/20，P（词）利用 homework1 中 word bag 统计好的词频，去与总的词数做除法，关键点落在了 P（词|类）上。

3、 对于每个类建立一个 word bag，建立起一个某类中某词出现次数的字典

```
def get_word_class():
    word_class = {}
    root_dir = r"20news-18828"
    for pack in os.listdir(root_dir):
        class_dir = root_dir + "\\" + pack
        for file in os.listdir(class_dir):
            file_name = class_dir + "\\" + file
            ff = open(file_name, "rb")
            for line in ff:
                str_line = str(line).strip().lower()
                for word in re.findall(letters, str_line):
                    if word in p_word:
                        if (pack, word) not in word_class:
                            word_class[pack, word] = 1
                        else:
                            word_class[pack, word] += 1
    return word_class
```

4、 由 test_set 中每一个 doc 建立一个 word list，然后直接对词分别与 20 个类进行贝叶斯概率计算，选出最大的类，然后与 word 所在的类作比较，如果匹配相同，为 True，记录下来，最后汇总计算准确率

```
for test_file in test_set:
    test_ff = open(test_file, "rb")
    for line in test_ff:
        str_line = str(line).strip().lower()
        for word in re.findall(letters, str_line):
            if word in p_word:
                doc_word.append(word)
    for pack in os.listdir(root_1dir):
        p_doc[test_file, pack] = 0
        for test_word in doc_word:
            if (pack, test_word) in word_class:
                p_doc[test_file, pack] += math.log((word_class[pack, test_word]+4)/class_word_num[pack]+1)
                p_doc[test_file, pack] -= math.log(int(p_word[test_word])/word_sum)
            else:
                p_doc[test_file, pack] += math.log(4/class_word_num[pack]+1)
                p_doc[test_file, pack] -= math.log(int(p_word[test_word])/word_sum)
```

实验过程中遇到和解决的问题：

平滑处理：还没应用其他的平滑处理，只做了 0 的特判

随机训练集：目前只取了遍历的前百分之几的文档

结论分析与体会：

准确率会因为 train 与 test 的比例不同而不同，多次测试后，准确率基本稳定在 80%~85%，

之后考虑对单词进行去重和不去重两种作比较，这就要修改之前的 word bag 的生成方式。

下一步，考虑学习 json，操作和处理方面要更方便。