**数字内容安全**

**实验报告**

****

**姓 名 詹冲**

**学 号 2023211616**

**指导教师 杨震**

**学 院 网络空间安全学院**

**2025年 5 月 14 日**

实验名称 垃圾邮件过滤系统 实验日期： 5月14日 指导老师 杨震 得分

学院 网络空间安全学院 专业 信息安全 班次 2023211801 姓名 詹冲 学号 2023211616

1. **实验目的**

（1）掌握在WINDOWS下安装和使用垃圾邮件过滤系统

（2）掌握垃圾邮件过滤系统主要功能模块

（3）文本内容过滤的原理

1. **实验内容**

（1）分析并调试垃圾邮件过滤系统程序主要功能模块

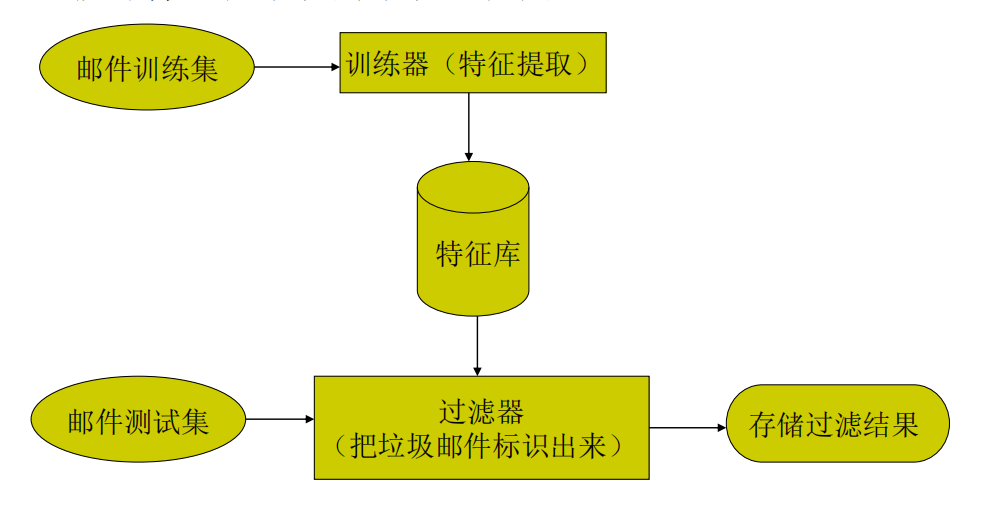
（2）选取实验数据集

（3）运行WINDOWS下的垃圾邮件过滤系统

（4）用垃圾邮件过滤系统对实验数据集进行过滤实验

1. **系统整体描述和分功能描述**

**1.系统整体描述**



**2.分功能描述**

（1）获取文件列表:获取指定目录下所有文件的名称列表，用于识别需要处理的邮件文件。

用到的函数:get\_File\_List(filePath)

1. def get\_File\_List(self, filePath):
2. filenames = os.listdir(filePath)
3. return filenames

（2）获取停用词表:加载预定义的中文停用词列表，用于在分词后过滤掉常见且无实际意义的词语。

用到的函数:get\_stop\_words()

1. def get\_stop\_words(self):
2. stop\_list = []
3. for line in open("data/中文停用词表.txt",encoding="gbk"):
4. stop\_list.append(line[:len(line) - 1])
5. return stop\_list

（3）处理训练邮件:遍历训练集中的正常邮件和垃圾邮件文件，对清洗后的文本进行分词，去除停用词，并收集每封邮件中出现的唯一词语列表。

用到的函数:get\_word\_list(content,words\_list,stop\_list)

1. def get\_word\_list(self, content, words\_list, stop\_list):
2. *# 分词结果放入res\_list*
3. res\_list = list(jieba.cut(content))
4. for i in res\_list:
5. if i not in stop\_list and i.strip() != '' and i != None:
6. if i not in words\_list:
7. words\_list.append(i)

（4）统计词频:统计每个词语在所有正常邮件或垃圾邮件中出现的邮件数量。

用到的函数:addToDict(words\_list,words\_dict)

1. def addToDict(self, words\_list, words\_dict):
2. for item in words\_list:
3. if item in words\_dict.keys():
4. words\_dict[item] += 1
5. else:
6. words\_dict.setdefault(item, 1)

（5）处理测试邮件:遍历测试集中的邮件文件，对清洗后的文本进行分词，去除停用词，并收集每封邮件中出现的唯一词语列表，并统计这些词语在当前邮件中的出现情况。

使用的函数:get\_word\_list(content,words\_list,stop\_list)，

addToDict (words\_list,words\_dict)

与训练集上的相同，故此不再赘述

（6）计算词语的垃圾邮件概率P(S|w):对于测试邮件中的每个词语，使用训练阶段得到的文档频率计算该词语出现在垃圾邮件中的概率P(w|S)和出现在正常邮件中的概率P(w|N)，进而根据贝叶斯公式计算P(S|w)=P(w|S)/(P(w|S)+P(w|N))。考虑了词语未在训练集中出现的情况（使用了平滑处理，概率设为0.01或0.4）。

用到的函数:getTestWords(testDict,spamDict,normDict,normFilelen,spamFilelen)

1. def getTestWords(self, testDict, spamDict, normDict, normFilelen, spamFilelen):
2. wordProbList = {}
3. for word, num in testDict.items():
4. if word in spamDict.keys() and word in normDict.keys():
5. *# 该文件中包含词个数*
6. pw\_s = spamDict[word] / spamFilelen
7. pw\_n = normDict[word] / normFilelen
8. ps\_w = pw\_s / (pw\_s + pw\_n)
9. wordProbList.setdefault(word, ps\_w)
10. if word in spamDict.keys() and word not in normDict.keys():
11. pw\_s = spamDict[word] / spamFilelen
12. pw\_n = 0.01
13. ps\_w = pw\_s / (pw\_s + pw\_n)
14. wordProbList.setdefault(word, ps\_w)
15. if word not in spamDict.keys() and word in normDict.keys():
16. pw\_s = 0.01
17. pw\_n = normDict[word] / normFilelen
18. ps\_w = pw\_s / (pw\_s + pw\_n)
19. wordProbList.setdefault(word, ps\_w)
20. if word not in spamDict.keys() and word not in normDict.keys():
21. *# 若该词不在脏词词典中，概率设为0.4*
22. wordProbList.setdefault(word, 0.4)
23. sorted(wordProbList.items(), key=lambda d: d[1], reverse=True)[0:15]
24. return (wordProbList)

（7）计算邮件的垃圾邮件总概率P(S|邮件):将测试邮件中各词语的P(S|w)进行组合，计算整封邮件是垃圾邮件的总概率。这里使用了对P(S|w)和P(N|w)(即1-P(S|w))连乘后再归一化的简化贝叶斯方法。

用到的函数:calBayes(wordList,spamdict,normdict)

1. def calBayes(self, wordList, spamdict, normdict):
2. ps\_w = 1
3. ps\_n = 1
4. for word, prob in wordList.items():
5. print(word + "/" + str(prob))
6. ps\_w \*= (prob)
7. ps\_n \*= (1 - prob)
8. p = ps\_w / (ps\_w + ps\_n)
9. *#         print(str(ps\_w)+"////"+str(ps\_n))*
10. return p

（8）计算准确率:遍历测试结果，根据文件名判断真实的邮件类别（小于1000的为正常，大于1000的为垃圾），然后与预测结果进行比较，统计预测正确的邮件数量，计算准确率。

用到的函数:calAccuracy(testResult)

1. def calAccuracy(self, testResult):
2. rightCount = 0
3. errorCount = 0
4. for name, catagory in testResult.items():
5. if (int(name) < 1000 and catagory == 0) or (int(name) > 1000 and catagory == 1):
6. rightCount += 1
7. else:
8. errorCount += 1
9. return rightCount / (rightCount + errorCount)
10. **实验步骤、结果及分析**

**1.分析并调试垃圾邮件过滤系统程序主要功能模块**

本系统主要基于朴素贝叶斯算法构建，程序主要分为以下几个功能模块：

（1）数据处理模块

get\_stop\_words:从文件中读取停用词表，去除对分类无显著意义的高频常用词。

get\_word\_list:对邮件内容进行分词，过滤停用词，构建词列表。

addToDict:统计词频，生成词典（词:出现次数）。

（2）文件管理模块

get\_File\_List:获取指定路径下所有文件名，用于读取训练集与测试集。

（3）特征提取模块

getTestWords:针对每一封测试邮件，计算其包含的各个词语在垃圾邮件和正常邮件中出现的概率，并根据贝叶斯公式计算每个词为垃圾邮件特征词的可能性（P(spam|word)）。

（4）分类与判断模块

calBayes:对邮件中出现的15个代表性词语进行概率组合，根据贝叶斯公式计算整封邮件是垃圾邮件的概率。

calAccuracy:基于测试集的命名规则（小于1000为正常邮件，大于1000为垃圾邮件），评估分类正确率。

**2.选取实验数据集**

本实验采用自定义的中文邮件数据集，包括：

data/normal:正常邮件训练集；

data/spam:垃圾邮件训练集；

data/test:待分类的测试邮件集；

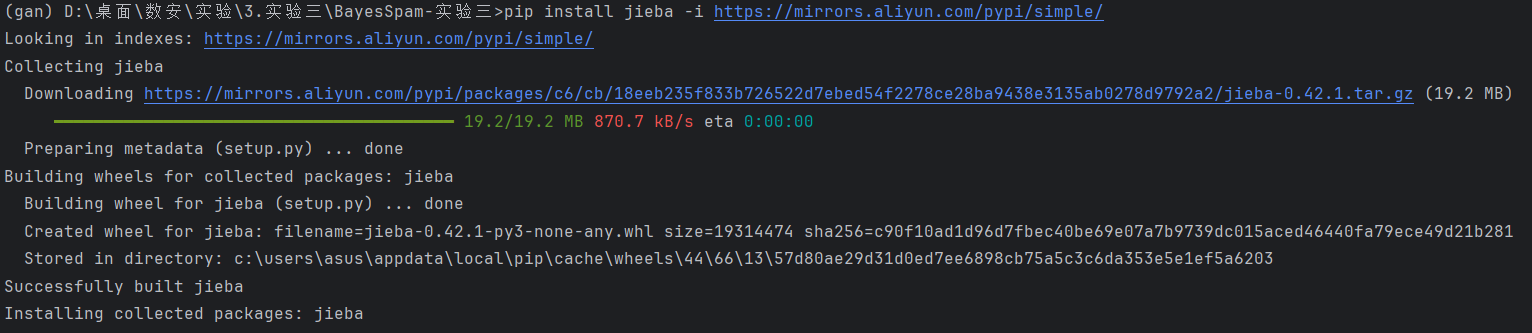
data/中文停用词表.txt:停用词列表。

所有邮件均为文本格式，编码统一为GBK，内容为中文，确保系统能正确处理中文文本。

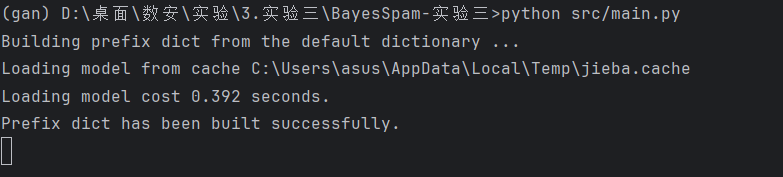
**3.运行WINDOWS下的垃圾邮件过滤系统并进行过滤实验**

在Windows环境下，执行主程序，系统按如下步骤运行：

（1）配置系统运行环境，发现系统中没有jieba库，所以使用pip install jieba -i https://mirrors.aliyun.com/pypi/simple/命令安装该库。

****

（2）在终端使用python src\main.py命令运行该系统。



（3）输出分类结果与总体分类准确率,得到最终正确率为0.9515。



**4.实验结果及分析**

（1）结果输出

①系统使用15个最具代表性的词语作为特征；

②对于每封测试邮件，通过P(spam|word)组合计算得出邮件属于垃圾邮件的概率；

③阈值设置为0.9，超过此值判为垃圾邮件；

④系统输出每封邮件的分类结果，并统计整体准确率。

⑤系统准确率由calAccuracy()函数评估，基于测试集文件名规则进行标签判断。实际准确率取决于训练集质量与测试集覆盖面。

（2）结果分析

显而易见，垃圾邮件过滤系统达到95.15%的准确率是一个非常好的结果。这意味着该系统能够正确地判断绝大多数邮件是正常邮件还是垃圾邮件，在现实中能够有效的过滤垃圾邮件。

达到如此正确率的原因如下：

（1）算法选择合适：朴素贝叶斯（Naive Bayes）算法在文本分类领域是一个非常经典且效果很好的算法，尤其适合处理词频特征。它基于词语出现的概率来判断文档的类别，这非常符合垃圾邮件和正常邮件在用词习惯上的差异。

（2）数据预处理有效：首先，使用了jieba进行中文分词，这是处理中文文本的关键第一步。其次，过滤掉了非中文字符，减少了噪声。最后，使用了停用词表，移除了“的”、“是”等对分类没有帮助的高频词，使得模型更关注那些有区分度的词语。

（3）统计词频，将文本数据转化为算法可以处理的数值特征。

（4）处理了未知词汇：getTestWords函数对测试邮件中出现的词语进行了处理，考虑了词语只在垃圾邮件字典中、只在正常邮件字典中、或都不在的情况，并赋予了基础概率（0.01或0.4），避免了概率为零的问题，这增加了模型的鲁棒性。

**五、实验总结**

本次实验通过实现基于贝叶斯的垃圾邮件过滤系统，掌握了以下知识点：

1.中文文本处理：分词、去除停用词、词频统计；

2.朴素贝叶斯分类器原理与应用；

3.文件操作与数据集构建技巧；

4.系统准确率评估方法。

**六、实验中遇到的问题及改正的方法**

该实验进行顺利，并未遇到相关问题。