

数字内容安全 实验报告



姓 名 项 枫
学 号 2022211570
指导教师 张 茹
学 院 网络空间安全学院

2024 年 5 月 16 日

实验名称 垃圾邮件过滤算法设计实验 实验日期: 2024 年 5 月 16 日 指导老师 张茹 得分
学院 网络空间安全学院 专业 信息安全 班次 2022211801 姓名 项枫 学号 2022211570

一、实验目的

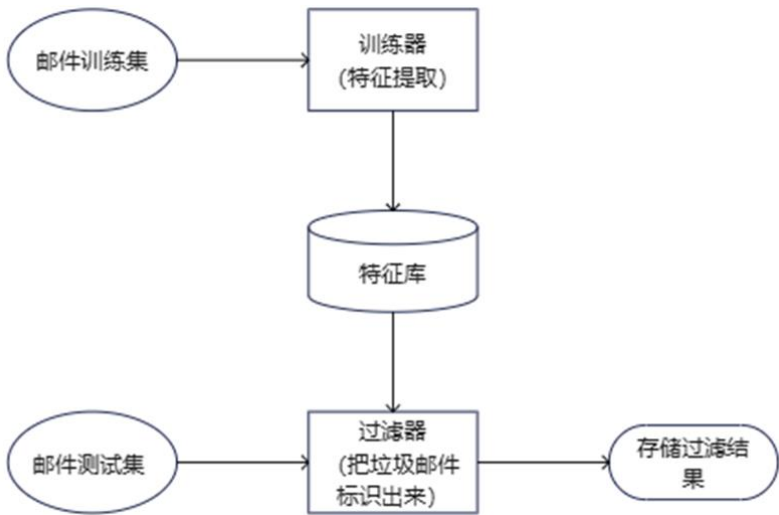
- (1) 掌握在 WINDOWS 下安装和使用垃圾邮件过滤系统
- (2) 掌握垃圾邮件过滤系统主要功能模块
- (3) 文本内容过滤的原理

二、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

三、系统整体描述和分功能描述

系统整体描述



分功能描述

- 1) 分功能 1: 分词处理
用 到 的 函 数 : `get_stop_words(self)` , `get_word_list(self, content, words_list, stop_list)` , `addToDict(self, words_list, words_dict)` , `addToDict(self, words_list, words_dict)` , `get_File_List(self, filePath)`
- 2) 分功能 2: 获取对邮件分类影响最大的 15 个词
用 到 的 函 数 : `getTestWords(self, testDict, spamDict, normDict, normFilelen, spamFilelen)`
- 3) 分功能 3: 计算贝叶斯概率
用到的函数: `calBayes(self, wordList, spamdict, normdict)`
- 4) 分功能 4: 计算预测结果正确率
用到的函数: `calAccuracy(self, testResult)`

四、实验步骤、结果及分析

实验步骤

1、安装 jieba。

```
C:\Users\项枫>pip install jieba
Collecting jieba
  Downloading jieba-0.42.1.tar.gz (19.2 MB)
    19.2/19.2 MB 46.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: jieba
  Building wheel for jieba (setup.py) ... done
  Created wheel for jieba: filename=jieba-0.42.1-py3-none-any.whl size=19314474 sha256=c1529cec6ff88dc8c71e0abc3137f50fc2e59e1588f42d68f4e5a18fda8fa309
  Stored in directory: c:\users\项枫\appdata\local\pip\cache\wheels\fe\aa\79\217233b7c4512e033cb980e378a27a73ff3c5359d2a48740f5
Successfully built jieba
Installing collected packages: jieba
Successfully installed jieba-0.42.1
```

2、对训练集用结巴分词，并用停用表进行简单过滤，然后使用正则表达式过滤掉邮件中的非中文字符。

3、分别保存正常邮件与垃圾邮件中出现的词有多少邮件出现该词，得到两个词典。例如词“疯狂”在 8000 封正常邮件中出现了 20 次，在 8000 封垃圾邮件中出现了 200 次。

4、对测试集中的每一封邮件做同样的处理，并计算得到 $P(s|w)$ 最高的 15 个词，在计算过程中，若该词只出现在垃圾邮件的词典中，则令 $P(w|s')=0.01P$ ，反之亦然；若都未出现，则令 $P(s|w)=0.4P$ 。

5、对得到的每封邮件中重要的 15 个词利用式 2 计算概率，若概率 > 阈值 α (设为 0.9)，则判为垃圾邮件，否则判为正常邮件。

以上步骤代码如下：

spamEmail.py

```
# encoding=utf-8
import jieba
import os

class SpamEmailBayes:
    # 获得停用词表
    def get_stop_words(self):
        stop_list = []
        for line in open(r'D:\DCS-LAB\3\BayesSpam_LAB3\data\中文停用词表.txt', encoding="gbk"):
            stop_list.append(line[:len(line) - 1])
        return stop_list

    # 获得词典
    def get_word_list(self, content, words_list, stop_list):
        # 分词结果放入 res_list
        res_list = list(jieba.cut(content))
        for i in res_list:
            if i not in stop_list and i.strip() != '' and i != None:
                if i not in words_list:
                    words_list.append(i)
```

```

# 若列表中的词已在词典中，则加 1，否则添加进去
def addToDict(self, words_list, words_dict):
    for item in words_list:
        if item in words_dict.keys():
            words_dict[item] += 1
        else:
            words_dict.setdefault(item, 1)

def get_File_List(self, filePath):
    filenames = os.listdir(filePath)
    return filenames

# 通过计算每个文件中  $p(s|w)$  来得到对分类影响最大的 15 个词
def getTestWords(self, testDict, spamDict, normDict, normFilelen,
spamFilelen):
    wordProbList = {}
    for word, num in testDict.items():
        if word in spamDict.keys() and word in normDict.keys():
            # 该文件中包含词个数
            pw_s = spamDict[word] / spamFilelen
            pw_n = normDict[word] / normFilelen
            ps_w = pw_s / (pw_s + pw_n)
            wordProbList.setdefault(word, ps_w)
        if word in spamDict.keys() and word not in normDict.keys():
            pw_s = spamDict[word] / spamFilelen
            pw_n = 0.01
            ps_w = pw_s / (pw_s + pw_n)
            wordProbList.setdefault(word, ps_w)
        if word not in spamDict.keys() and word in normDict.keys():
            pw_s = 0.01
            pw_n = normDict[word] / normFilelen
            ps_w = pw_s / (pw_s + pw_n)
            wordProbList.setdefault(word, ps_w)
        if word not in spamDict.keys() and word not in
normDict.keys():
            # 若该词不在脏词词典中，概率设为 0.4
            wordProbList.setdefault(word, 0.4)
    sorted(wordProbList.items(), key=lambda d: d[1],
reverse=True) [0:15]
    return (wordProbList)

# 计算贝叶斯概率
def calBayes(self, wordList, spamdict, normdict):
    ps_w = 1

```

```

        ps_n = 1

        for word, prob in wordList.items():
            print(word + "/" + str(prob))
            ps_w *= (prob)
            ps_n *= (1 - prob)
        p = ps_w / (ps_w + ps_n)
        #         print(str(ps_w)+"////"+str(ps_n))
        return p

# 计算预测结果正确率

def calAccuracy(self, testResult):
    rightCount = 0
    errorCount = 0
    for name, catagory in testResult.items():
        if (int(name) < 1000 and catagory == 0) or (int(name) >
1000 and catagory == 1):
            rightCount += 1
        else:
            errorCount += 1
    return rightCount / (rightCount + errorCount)

```

main.py

```

# encoding=utf-8

from spamEmail import SpamEmailBayes
import re

# spam 类对象
spam = SpamEmailBayes()
# 保存词频的词典
spamDict = {}
normDict = {}
testDict = {}
# 保存每封邮件中出现的词
wordsList = []
wordsDict = {}
# 保存预测结果, key 为文件名, 值为预测类别
testResult = {}
# 分别获得正常邮件、垃圾邮件及测试文件名称列表
normFileList = spam.get_File_List("D:/DCS-
LAB/3/BayesSpam_LAB3/data/normal")
spamFileList = spam.get_File_List("D:/DCS-
LAB/3/BayesSpam_LAB3/data/spam")

```

```

testFileList = spam.get_File_List("D:/DCS-
LAB/3/BayesSpam_LAB3/data/test")
# 获取训练集中正常邮件与垃圾邮件的数量
normFilelen = len(normFileList)
spamFilelen = len(spamFileList)
# 获得停用词表，用于对停用词过滤
stopList = spam.get_stop_words()
# 获得正常邮件中的词频
for fileName in normFileList:
    wordsList.clear()
    for line in open("D:/DCS-LAB/3/BayesSpam_LAB3/data/normal/" +
fileName, encoding="gbk"):
        # 过滤掉非中文字符
        rule = re.compile(r"^[^u4e00-\u9fa5]")
        line = rule.sub("", line)
        # 将每封邮件出现的词保存在 wordsList 中
        spam.get_word_list(line, wordsList, stopList)
        # 统计每个词在所有邮件中出现的次数
        spam.addToDict(wordsList, wordsDict)
normDict = wordsDict.copy()

# 获得垃圾邮件中的词频
wordsDict.clear()
for fileName in spamFileList:
    wordsList.clear()
    for line in open("D:/DCS-LAB/3/BayesSpam_LAB3/data/spam/" +
fileName, encoding="gbk"):
        rule = re.compile(r"^[^u4e00-\u9fa5]")
        line = rule.sub("", line)
        spam.get_word_list(line, wordsList, stopList)
        spam.addToDict(wordsList, wordsDict)
spamDict = wordsDict.copy()

# 测试邮件
for fileName in testFileList:
    testDict.clear()
    wordsDict.clear()
    wordsList.clear()
    for line in open("D:/DCS-LAB/3/BayesSpam_LAB3/data/test/" +
fileName, encoding="gbk"):
        rule = re.compile(r"^[^u4e00-\u9fa5]")
        line = rule.sub("", line)
        spam.get_word_list(line, wordsList, stopList)
        spam.addToDict(wordsList, wordsDict)

```

```

testDict = wordsDict.copy()
# 通过计算每个文件中  $p(s|w)$  来得到对分类影响最大的 15 个词
wordProbList = spam.getTestWords(testDict, spamDict, normDict,
normFilelen, spamFilelen)
# 对每封邮件得到的 15 个词计算贝叶斯概率
p = spam.calBayes(wordProbList, spamDict, normDict)
if (p > 0.9):
    testResult.setdefault(fileName, 1)
else:
    testResult.setdefault(fileName, 0)
# 计算分类准确率 (测试集中文件名低于 1000 的为正常邮件)
testAccuracy = spam.calAccuracy(testResult)
for i, ic in testResult.items():
    print(i + "/" + str(ic))

print("Accuracy:" + str(testAccuracy))

```

实验结果及分析

1、实验结果

(1) 影响每封邮件分类关键词的贝叶斯概率 (概率大于 0.9 判为垃圾邮件, 否则判为正常邮件), 部分结果截图如下:

```

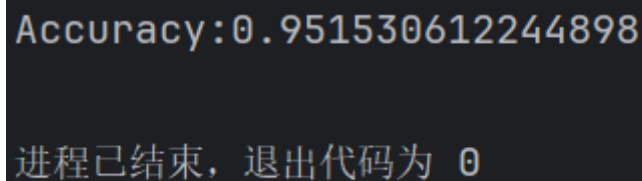
运行  main x
成绩/0.20809550052833954
持/0.3663218884756161
怀疑/0.09829563377929523
态度/0.08370362225855067
是不是/0.012127133151217433
考试/0.21308195295035876
特殊性/0.9592557381502105
普通/0.8811813777664128
考研/0.05788303858732651
大不相同/0.4
这种/0.03161952958906795
专业/0.5604872126354211
背景/0.08227172305075928
将来/0.008227854368259097
毕业/0.011090732651003864
就业/0.1523932059727662

```

(2) 根据贝叶斯概率对测试集邮件判别情况 (1 为垃圾邮件, 0 为正常邮件), 部分结果截图如下:

```
运行 main x
结构
60/0
61/0
62/0
63/0
65/0
66/1
67/0
68/0
69/0
70/0
71/0
72/0
73/0
74/0
75/0
76/0
7801/1
7802/1
7803/0
7804/1
7805/1
7806/1
7807/1
7808/1
7809/1
7810/1
7811/1
7812/1
7813/1
7814/1
7815/1
7816/1
```


(3) 邮件分类判别的正确率:



```
Accuracy:0.951530612244898  
进程已结束，退出代码为 0
```

完整代码运行结果请见附件“实验三_代码运行结果.txt”

2、分析

(1) 显而易见，在该测试集中测试结果为分类准确率 95.15%，朴素贝叶斯分类器的分类结果还是相当好的。

(2) 该方法优点:

①对待预测样本进行预测，过程简单速度快(想想邮件分类的问题，预测就是分词后进行概率乘积，在 \log 域直接做加法更快)。

②对于多分类问题也同样很有效，复杂度也不会有大程度上升。

③在分布独立这个假设成立的情况下，贝叶斯分类器效果奇好，会略胜于逻辑回归，同时我们需要的样本量也更少一点。

④对于类别类的输入特征变量，效果非常好。对于数值型变量特征，我们是默认它符合正态分布的。

五、实验中遇到的问题及改正的方法

最初，打开文件代码写成了“D:\DCS-LAB\3\BayesSpam_LAB3\data\normal\”，最后改成了“D:/DCS-LAB/3/BayesSpam_LAB3/data/normal/”运行成功。