信 息 内 容 安 全 实 验 报 告

实验项目名称：	基于朴素贝叶斯的垃圾邮件过滤
班级：	SC011701
姓名：	廖凯华 谢希 郑博文 裴嘉琨
学号：	2017302219 2017302220 2017302240 2017302242
指导教师：	杨黎斌
实验时间：	2020 年 3 月 30 日

摘要

随着网络通信技术的迅速进步，人与人之间的通信手段也逐渐便捷、迅速，作为一种传统的网络通信手段，电子邮件凭借其速度快、广域性、费用低等优点被人们广泛利用，但是随之而来的便是日益增长的垃圾邮件数量，并带来各种各样的经济损失以及生活干扰。因此，我们将基于第三代反垃圾邮件技术，通过**朴素贝叶斯**以及 **SVM 支持向量机**进行邮件分类。在本份报告中，我们会从朴素贝叶斯的理论推导以及 SVM 支持向量机分类的实现原理出发，并依次实现贝叶斯多项式模型以及 SVM 模型，对搜集到的邮件样本进行分类。然后通过二维图像形式直观地反映两种模型在邮件分类的准确率、查全率、召回率上的对比，并且发现对比朴素贝叶斯算法，SVM 算法虽然能更好地将垃圾邮件进行了过滤，但是依然存在召回率低下的问题。最后我们总结认为，只有采取多种分类模型结合、取长补短的方法，才能进一步提高邮件的分类效果。

关键词：垃圾邮件过滤　朴素贝叶斯　SVM 支持向量机

Abstract

Among with the rapid development of Network Communication Technology, the contact between people is getting increasingly convenient and fast as well. As one of the traditional network communication method, E-mail is extensively used due to its several advantages of high speed, wide performance and low cost. However, what follow the merits of E-mail are the various problems caused by spams, including economic losses and life interference. Therefore, we will classify the E-mails by using **Naive Bayes** and **SVM (Support Vector Machine)** which are based on the 3rd generation of Anti-Spam Technology. In this experiment report, we start with the theoretical derivation of Naive Bayes and the implementation principle of SVM. Next, we will classify the E-mail samples into two categories, Hams and Spams, by realizing the Multinomial Naive Bayes and SVM model. We also use the 2D images in order to reflect the comparison of classification results at the aspects of accuracy, precision and recall directly. Finally, we find that, compared with Naive Bayes, the model of SVM has more advantages on filtering Spams as well as the problem of low recall. After all, we conclude that, only if a combination of various classification models is adopted, can we improve the classification effect of E-mails much further.

Key Words: Spam Filtering; Naive Bayes; SVM (Support Vector Machine)

目录

工作分配：

学号	姓名	工作分工
2017302219	廖凯华	贝叶斯多项式模型代码编写
2017302220	谢希	SVM 向量机模型代码编写
2017302240	郑博文	理论模型分析与报告撰写
2017302242	裴嘉琨	实验结果分析与报告撰写

1. 引言

垃圾邮件一般具有批量发送的特征。其内容包括赚钱信息、成人广告、商业或个人网站广告、电子杂志、连环信等。垃圾邮件可以分为良性和恶性的。良性垃圾邮件是各种宣传广告等对收件人影响不大的信息邮件。恶性垃圾邮件是指具有破坏性的电子邮件。有些垃圾邮件发送组织或是非法信息传播者，为了大面积散布信息，常采用多台机器同时巨量发送的方式攻击邮件服务器，造成邮件服务器大量带宽损失，并严重干扰邮件服务器进行正常的邮件递送工作。[1]因此，我们必须要采取必要的、有效的技术手段进行垃圾邮件阻挡。

在垃圾邮件的泛滥中，垃圾邮件拦截对抗已经发展产生了第四代技术，典型的反垃圾邮件技术包括基本过滤技术、智能过滤技术以及前端认证技术等。第一代技术采用关键字（词）过滤、黑白名单、IP 过滤等技术；第二代技术主要是实时黑名单和电子签名技术；第三代则主要是运用智能方法，包括贝叶斯过滤、人工智能、机器语言学习等技术；而目前的第四代为多技术整合分层过滤。[2]

本次实验，我们主要利用第三代反垃圾邮件技术中的贝叶斯过滤的方法以及 SVM 向量机分类法设计并实现一个垃圾邮件过滤系统。

2. 理论概述

2.1 贝叶斯公式

在得出贝叶斯公式之前，我们需要列出以下定义：

边缘概率：又称**先验概率**，即某件事情发生的概率。比如事件 A 的边缘概率就是$P(A)$。

联合概率：两个或多个事件同时发生的概率。比如事件 A 和 B 的联合概率就是$P(A \cap B)$。

条件概率：某一事件在另一事件已经发生的情况下发生的概率。比如$P(A|B)$表示为事件 B 发生的条件下，事件 A 发生的概率。因此，结合边缘概率以及联合概率，我们给出条件概率的定义公式：

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \quad (1)$$

下面我们进行贝叶斯公式的推导。

根据上式（1）我们可以同样得出$P(B|A)$，即事件 A 发生的条件下，事件 B 发生的概率为：

$$P(B|A) = \frac{P(A \cap B)}{P(A)} \quad (2)$$

[1] 百度百科：https://baike.baidu.com/item/%E5%9E%83%E5%9C%BE%E9%82%AE%E4%BB%B6%E8%BF%87%E6%BB%A4/9642958?fr=aladdin

[2] 百度百科：https://baike.baidu.com/item/%E5%8F%8D%E5%9E%83%E5%9C%BE%E9%82%AE%E4%BB%B6%E6%8A%80%E6%9C%AF/5445064?fr=aladdin

结合式（1）和式（2），我们可以得到：

$$P(A \cap B) = P(A|B) \cdot P(B) = P(B|A) \cdot P(A) \tag{3}$$

在$P(B) \neq 0$的条件下，我们可以得到贝叶斯定理：

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{4}$$

接下来，我们对上式（4）进行拓展，我们假设$A_1, \cdots, A_n$为事件 A 的完备事件组，即$\cup_{i=1}^{n} A_i = \Omega,\ A_i \cap A_j = \emptyset, P(P_i) \neq 0$，可以得到贝叶斯法则公式：

$$P(A_i|B) = \frac{P(B|A_i) \cdot P(A_i)}{\sum_j P(B|A_j) \cdot P(A_j)} \tag{5}$$

结合前文中列出的定义，贝叶斯法则可表述为：

$$\text{后验概率} = \frac{\text{似然度} \times \text{先验概率}}{\text{标准化常量}} \tag{6}$$

也就是说，后验概率与先验概率和似然度的乘积成正比。另外，比例$\frac{P(B|A)}{P(B)}$有时也被称作标准似然度（standardized likelihood），因此贝叶斯法则可表述为：后验概率 = 标准似然度 × 先验概率。

2.2 朴素贝叶斯

给定训练数据集(X, Y)，其中每个样本X都包括n维特征，即$x = (x_1, x_2, \cdots x_n)$，类标记集合含有 k 种类别，即$y = (y_1, y_2, \cdots, y_k)$。那么对于一个给定的样本 X 并判断它所属的类别，也就是给定样本 X，求它属于哪一个类别的概率最大。也就转化为求解$P(y_1|x), P(y_2|x), \cdots, P(y_k|x)$中的最大值。这里根据 2.1 推导出的贝叶斯公式（5）可以计算，即：

$$P(y_k|x) = \frac{P(x|y_k)P(y_k)}{\sum_k P(x|y_k)P(y_k)} \tag{7}$$

通过利用上式（7），我们可以实现简单的垃圾邮件过滤，而这里所指的简单垃圾邮件过滤，是在只有一个因素影响下，对邮件进行过滤，也就是说，我们只考虑一个词汇对邮件是否是垃圾邮件的影响。但是事实并非如此，单单只靠一个词汇就将一个邮件评判为垃圾邮件或者是非垃圾邮件，是完全不够的，一般的情况，都需要我们根据多个词语才能判断一个邮件是否是垃圾邮件。而具体到数据集中的样本 X，就是考虑样本 X 中的每一个有可能使邮件被判断为垃圾邮件的特征，即 n 维特征$x = (x_1, x_2, \cdots x_n)$，同时对判断垃圾邮件产生的影响。那么对上式（7）进行推广，便可以得到多维特征对垃圾邮件判断的影响（概率）：

$$P(y_k|x_1, x_2, \cdots, x_n) = \frac{P(x_1, x_2, \cdots, x_n|y_k)P(y_k)}{P(x_1, x_2, \cdots, x_n)} \tag{8}$$

式（8）中的分母是一个 n 维空间向量的联合概率，如果每一个特征有 t 种取值，那么所有的情况将会有t^n种，则求解该问题将会变成一个 NP 问题。因此，朴素贝叶斯的方法便是基于这种情况，将其进行简化。

我们假设 n 维特征$x = (x_1, x_2, \cdots x_n)$中每一个特征都相互独立，于是，$n$ 维空间向量的联合概率分布将会简化为 n 个特征单独出现的概率之积，也就是如下式所示：

$$P(x_1, x_2, \cdots, x_n) \to P(x_1) \cdot P(x_2) \cdot \cdots \cdot P(x_n) \tag{9}$$

因此，判断邮件是否是垃圾邮件的概率公式（8）便可以简化为

$$P(y_k|x_1,x_2,\cdots,x_n)=\frac{P(x_1,x_2,\cdots,x_n|y_k)P(y_k)}{P(x_1)\cdot P(x_2)\cdot\cdots\cdot P(x_n)} \quad (10)$$

即

$$P(x|y_k)P(y_k)=P(y_k)\prod_{i=1}^{n}P(x_i|y_k) \quad (11)$$

这就是朴素贝叶斯，也就是对原先的多维贝叶斯公式推广的一种简化，这将会大大降低计算量。

3. 计算模型

3.1 贝叶斯多项式模型

当特征值是离散的情况时，我们采用多项式模型，对前面的朴素贝叶斯进行处理，同时，该多项式模型，在计算先验概率$P(y_k)$和条件概率$P(x_i|y_k)$会进行平滑处理，具体公式为：

$$P(y_k)=\frac{N_{y_k}+\alpha}{N+k\alpha} \quad (12)$$

其中：N是总的样本个数，k是总的类别个数，N_{y_k}是类别为y_k的样本个数，α是平滑值。

$$P(x_i|y_k)=\frac{N_{y_k,x_i}+\alpha}{N_{y_k}+n\alpha} \quad (13)$$

其中：N_{y_k}是类别为y_k的样本个数，n 是特征的维数，N_{y_k,x_i}是类别为y_k的样本中，第 i 维特征的值是x_i的样本个数，α是平滑值。

当$\alpha=1$时，称作 Laplace 平滑，当$0<\alpha<1$时，称作 Lidstone 平滑，$\alpha=0$时不做平滑。如果不做平滑，当某一维特征的值x_i没在训练样本中出现过时，会导致$P(x_i|y_k)=0$，从而导致后验概率为 0，而加上平滑就可以克服这个问题。

3.2 比较模型——SVM 模型简述

支持向量机（Support Vector Machine, SVM）是一类按监督学习方式对数据进行二元分类的广义线性分类器其决策边界是对学习样本求解的最大边距超平面[3]。

SVM 使用铰链损失函数（hinge loss）计算经验风险（empirical risk）并在求解系统中加入了正则化项以优化结构风险（structural risk），是一个具有稀疏性和稳健性的分类器[4]。SVM 可以通过核方法（kernel method）进行非线性分类，是常见的核学习（kernel learning）方法之一[5]。

[3] Vapnik, V.. Statistical learning theory. 1998 (Vol. 3). . New York, NY：Wiley，1998：Chapter 10-11, pp.401-492
[4] 周志华．机器学习．北京：清华大学出版社，2016：pp.121-139, 298-300
[5] 李航．统计学习方法．北京：清华大学出版社，2012：第七章，pp.95-135

3.2.1 线性可分性

在分类问题中给定输入数据和学习目标：$X=\{X_1,X_2,\cdots,X_N\}$，$y=\{y_1,y_2,\cdots,y_N\}$，其中输入数据的每个样本都包含多个特征并由此构成特征空间：$X_i=[x_1,x_2,\cdots,x_n]\in\chi$，而学习目标为二元变量$y\in\{-1,1\}$，表示负类和正类。

若输入数据所在的特征空间存在作为决策边界的超平面将学习目标按正类和负类分开，并使任意样本的点到平面距离大于等于 1：

$$w^TX+b=0 \tag{14}$$

$$y_i(w^TX_i+b)\geq 1 \tag{15}$$

则称该分类问题具有线性可分性，参数w,b 分别为超平面的法向量和截距。

满足该条件的决策边界实际上构造了 2 个平行的超平面作为间隔边界以判别样本的分类：

$$w^TX_i+b\geq+1\Rightarrow y_i=+1 \tag{16}$$

$$w^TX_i+b\leq-1\Rightarrow y_i=-1 \tag{17}$$

所有在上间隔边界上方的样本属于正类，在下间隔边界下方的样本属于负类。两个间隔边界的距离$d=\frac{2}{\|w\|}$被定义为边距，位于间隔边界上的正类和负类样本为支持向量。

3.2.2 损失函数

在一个分类问题不具有线性可分性时，使用超平面作为决策边界会带来分类损失，即部分支持向量不再位于间隔边界上，而是进入了间隔边界内部，或落入决策边界的错误一侧。损失函数可以对分类损失进行量化，其按数学意义可以得到的形式是 0-1 损失函数：

$$L(p)=\begin{cases}0 & p<0\\ 1 & p\geq 0\end{cases} \tag{18}$$

0-1 损失函数不是连续函数，不利于优化问题的求解，因此通常的选择是构造代理损失。可用的选择包括铰链损失函数、logistic 损失函数、和指数损失函数，其中 SVM 使用的是铰链损失函数：

$$L(p)=\max(0,1-p) \tag{19}$$

对替代损失的相合性研究表明，当代理损失是连续凸函数，并在任意取值下是 0-1 损失函数的上界，则求解代理损失最小化所得结果也是 0-1 损失最小化的解。铰链损失函数满足上述条件。

3.2.3 经验风险与结构风险

按统计学习理论，分类器在经过学习并应用于新数据时会产生风险，风险的类型可分为经验风险和结构风险：

$$empirical\ risk:\quad \varepsilon=\sum_{i=1}^{N}L(p_i)=\sum_{i=1}^{N}L[f(X_i,w),y_i] \tag{20}$$

$$structural\ risk:\quad \Omega(f)=\|w\|^p \tag{21}$$

式中f表示分类器，经验风险由损失函数定义，描述了分类器所给出的分类

结果的准确程度；结构风险由分类器参数矩阵的范数定义，描述了分类器自身的复杂程度以及稳定程度，复杂的分类器容易产生过拟合，因此是不稳定的。若一个分类器通过最小化经验风险和结构风险的线性组合以确定其模型参数：

$$L = \|w\|^p + C\sum_{i=1}^{N} L[f(X_i, w), y_i] \quad (22)$$

$$w = argminL \quad (23)$$

则对该分类器的求解是一个正则化问题，常数 C 是正则化系数。当 p=2 时，该式被称为 L_2 正则化或 Tikhonov 正则化。SVM 的结构风险按 p=2 表示，在线性可分问题下，硬边界 SVM 的经验风险可以归 0，因此其是一个完全最小化结构风险的分类器；在线性不可分问题中，软边界 SVM 的经验风险不可归 0，因此其是一个 L_2 正则化分类器，最小化结构风险和经验风险的线性组合。

3.2.4 核方法

由于映射函数具有复杂的形式，难以计算其内积，因此可使用核方法（kernel method），即定义映射函数的内积为核函数（kernel function）。下面列出常用的核函数：

名称	解析式
多项式核（polynomial kernel）	$\kappa(X_1, X_2) = (X_1^T X_2)^n$
径向基函数核（RBF kernel）	$\kappa(X_1, X_2) = \exp\left(-\frac{\|X_1 - X_2\|^2}{2\sigma^2}\right)$
拉普拉斯核（Laplacian kernel）	$\kappa(X_1, X_2) = \exp\left(-\frac{\|X_1 - X_2\|}{\sigma}\right)$
Sigmoid 核（Sigmoid kernel）	$\kappa(X_1, X_2) = \tanh[a(X_1^T X_2) - b],\ a, b > 0$

4. 实验内容

4.1 利用多项式模型的实现内容及结果

4.1.1 多项式模型使用的函数库

•（1）***Counter***：Counter 类的目的是用来跟踪值出现的次数。它是一个无序的容器类型，以字典的键值对形式存储，其中元素作为 key，其计数作为 value。计数值可以是任意的 Interger（包括 0 和负数）。Counter 类和其他语言的 bags 或 multisets 很相似。

4.1.2 对已有数据集进行提取

调取已有文件库中的文件，随机选取文件进行后面的检测。其中 Hamfile 表示正常邮件资料库；Spamfile 表示垃圾邮件资料库。

```
def getFileDiv():
    hamFile = './/email//ham'
    spamFile = './/email//spam'
    hamEmails = [os.path.join(hamFile, f) for f in os.listdir(hamFile)]#取 ham 正常邮件
    spamEmails = [os.path.join(spamFile, f) for f in os.listdir(spamFile)]#取 spam 垃圾邮件
    label = np.zeros(50)
    label[len(hamEmails):50] = 1
    hamEmails.extend(spamEmails)
    randnum = os.urandom(8)   # 以同样方式对路径和标签打乱
    random.seed(randnum)
    random.shuffle(label)
    random.seed(randnum)
    random.shuffle(hamEmails)
    return hamEmails, label
```

4.1.3 提取相关词汇

```
def getWordsProb(filepath, label):
    spamList = []
    hamList = []
    for index, path in enumerate(filepath):
        with open(path, 'r', encoding='gb18030', errors='ignore') as f:
            lines = f.readlines()
            for line in lines:
                line = re.sub('[^a-zA-Z]', ' ', line)
                words = line.split()#将邮件中每行文字进行分词
                if label[index] == 0:
                    hamList.extend(words)
                else:
                    spamList.extend(words)
    spamCounter = Counter(spamList)#构建垃圾邮件词汇列表
    hamCounter = Counter(hamList)#构建正常邮件词汇列表
    for item in list(spamCounter):   #避免单字符对最终的影响 处理单字符
        if len(item) == 1:
            del spamCounter[item]
    for item in list(hamCounter):
        if len(item) == 1:
            del hamCounter[item]
    spamSet = set(spamCounter)
    hamSet = set(hamCounter)
    spamSet.update(hamSet)
    allWordList = Counter(spamSet)
```

为了可以更好的将垃圾邮件测试出来，针对已有的训练集，分别建立相对应的语料库。

在构建语料库时，需要将字数较小的单词消除避免影响。

```
spamCounter = allWordList + spamCounter    # 消除概率为零相乘的影响
    spamCnt = sum(spamCounter.values())
    for k, v in spamCounter.items():
        spamDict[k] = v / spamCnt
    hamCounter = allWordList + hamCounter
    hamCnt = sum(hamCounter.values())
    for k, v in hamCounter.items():
        hamDict[k] = v / hamCnt
```

消除概率为零相乘的影响，利用拉普拉斯平滑处理解决。通过整个函数可以得到最终的垃圾邮件和正常邮件的词汇集。

4.1.4 垃圾邮件测试

对已有邮件测试库进行垃圾邮件测试，先设定好收到正常邮件和垃圾邮件的先验概率，随后针对随机抽取到的邮件进行相关的测试。

计算抽取到的邮件中单个词汇的在文档中的贡献率，并针对多个特征词进行公式计算得到最终的垃圾邮件可能性。通过该函数可以通过公式，将垃圾邮件分离开。

```
def mulNBTest(hamDict, spamDict, testEmail, testLabel):
    result = []    # 记录判断结果，之后与 Label 对比
    spamProb = 0.5    # P(spam) = 0.5
    hamProb = 0.5    # P(ham) = 0.5
    for testFile in testEmail:
        testWords = []
        with open(testFile, 'r', encoding='gb18030', errors='ignore') as f:
            lines = f.readlines()
            for line in lines:
                line = re.sub('[^a-zA-Z]', ' ', line)
                words = line.split()
                testWords.extend(words)
        testCounter = Counter(testWords)
        for item in list(testCounter):    # 处理单字符
            if len(item) == 1:
                del testCounter[item]
        pureWords = list(testCounter)    # 得到邮件内字符列表
        probList = []    # 存储每个字符的贡献
        mediumFre1 = np.median(list(hamDict.values()))
        mediumFre2 = np.median(list(spamDict.values()))
```

```
            for word in pureWords:
                pwh = hamDict.get(word, mediumFre1)    # P(word|ham)
                pws = spamDict.get(word, mediumFre2)    # P(word|spam)
                psw = (spamProb * pws) / (pwh * hamProb + pws * spamProb)
                                        # P(spam|word) = P(spam)*P(word|spam)/P(word)
                probList.append(psw)
            numerator = 1    #  分子
            denominator = 1    #  分母
            for psw in probList:
                numerator *= psw
                denominator *= (1 - psw)
            #P(spam|word1word2…wordn)= P1P2…Pn/(P1P2…Pn+(1-P1)(1-P2)…(1-Pn))
            resProb = numerator / (numerator + denominator)
            if resProb > 0.9:
                result.append(1)
            else:
                result.append(0)
```

4.1.5 测试结果统计

为了可以直观看到该方式可以将垃圾邮件，该函数部分计算检测垃圾邮件的准确率、精确率和召回率。并通过 Python 的 Matplotlib 库，得到直观的二维图像。

```
    TP = 0    #  将正类预测为正类数
        FN = 0    #  将正类预测为负类数
        FP = 0    #  将负类预测为正类数
        for index in range(len(testLabel)):
            if testLabel[index] == 1:
                if result[index] == 1:
                    rightCnt += 1
                    TP += 1
                else:
                    FN += 1
            else:
                if result[index] == 0:
                    rightCnt += 1
                else:
                    FP += 1
        accuracy = rightCnt / len(testLabel)
        precision = TP / (TP + FP)
        recall = TP / (TP + FN)
```

4.1.6 测试结果分析

该部分实验我们小组主要从贝叶斯多项式方法对垃圾邮件进行过滤。在分析最后结果实现时的情况时，我们主要运用准确率、精确率、召回率三者进行比较分析，三者计算公式在本次实验的实际情况为：

$$准确率 = \frac{所有预测正确的样本}{所有样本}$$

$$精确率 = \frac{准确预测为垃圾邮件的数量}{所有预测为垃圾邮件的数量}$$

$$召回率 = \frac{将垃圾邮件预测为垃圾邮件}{所有垃圾邮件}$$

在使用贝叶斯多项式方法时，在每次执行代码时，均会对样本集进行 100 次的测试，对其准确率、精确率、召回率进行统计并制图得到结果。为了确保代码实现效果是良好的，我们针对代码运行多次，下图为其中一次的结果图：

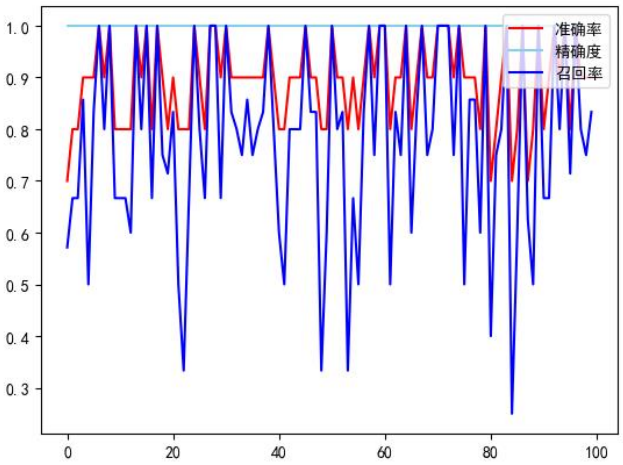

图 1. 多项式模型准确率、精确率、召回率折线统计图

可以从结果来看，针对这 100 次的垃圾邮件过滤结果来看，经过贝叶斯多项式方法，识别到的垃圾邮件是没有错误的。但是同时在部分时候，并未能将所有的垃圾邮件全部识别并过滤出。说明通过贝叶斯多项式方法的检测还是存在的一定的容错率，仍会有些新颖的词汇的输入会导致最终判断垃圾邮件结果的出入情况。

4.2 利用 SVM 模型的实验内容及结果

4.2.1 SVM 使用的函数库

•（1）***Matplotlib***：是一个 Python 的 2D 绘图库，它以各种硬拷贝格式和跨平台的交互式环境生成出版质量级别的图形[6]。

•（2）***pandas***：是基于 NumPy 的一种工具，该工具是为了解决数据分析任务而创建的[7]。

4.2.2 统计邮件数量

```
path = '..\\spam.csv'
data = pd.read_csv(path, encoding='latin-1')
count_Class = pd.value_counts(data["v1"], sort= True)
#统计 V1 处的值 ham 和 spam 的数量
count_Class.plot(kind= 'bar', color= ["blue", "orange"])
plt.title('Bar chart')
plt.show()
```

通过 python 中 matplotlib 函数库，调用文件 spam.scv，实现统计 ham 以及 spam 并输出两项的数量对比图，如下：

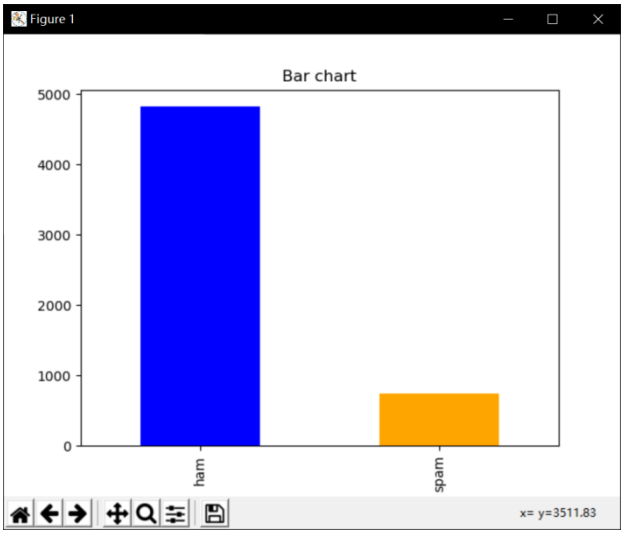

图 2. Ham 和 Spam 的数量对比图

[6] 百度百科：https://baike.baidu.com/item/Matplotlib/20436231?fr=aladdin
[7] 百度百科：https://baike.baidu.com/item/pandas

4.2.3 文本词频统计

```
count1 = Counter(" ".join(data[data['v1'] =='ham']["v2"]).split()).most_common(20)
df1 = pd.DataFrame.from_dict(count1)
df1 = df1.rename(columns={0: "words in non-spam", 1 : "count"})
print(df1) #计算 ham 中的词语频率
```

通过上面代码段实现对 Ham 邮件中词语频率最高的 20 个词汇的统计，输出文字统计结果并依据 plt 函数库绘制可视性结果（柱状图）：

```
df1.plot.bar(legend = False)
y_pos = np.arange(len(df1["words in non-spam"]))
plt.xticks(y_pos, df1["words in non-spam"])
plt.title('More frequent words in non-spam messages')
plt.xlabel('words')
plt.ylabel('number')
plt.show()
```

	words in non-spam	count
0	to	1530
1	you	1458
2	I	1436
3	the	1019
4	a	969
5	and	738
6	i	736
7	in	734
8	u	645
9	is	638
10	my	619
11	me	537
12	of	498
13	for	475
14	that	398
15	it	375
16	your	373
17	on	352
18	have	346
19	at	333

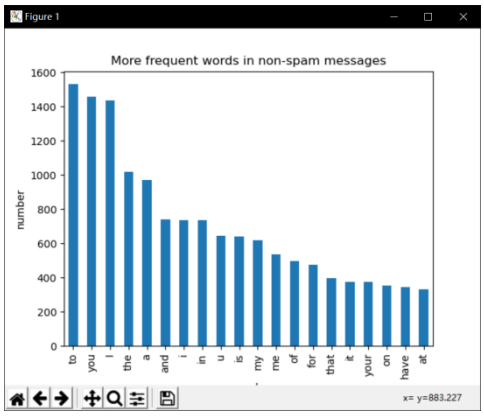

图 3（左）. Non-spam 邮件的词频统计情况（前 20）

图 4（右）. Non-spam 邮件词频统计柱状图（前 20）

再次统计 spam 邮件中的词语特征值，绘制词频统计柱状图：

```
#统计 spam 邮件的词语特征值：
    count2 = Counter(" ".join(data[data['v1'] == 'spam']["v2"]).split()).most_common(20)
    df2 = pd.DataFrame.from_dict(count2)
    df2 = df2.rename(columns={0: "words in spam", 1 : "count_"})
```

```
#绘制词频统计柱状图：
    df2.plot.bar(legend = False, color = 'orange')
    y_pos = np.arange(len(df2["words in spam"]))
    plt.xticks(y_pos, df2["words in spam"])
    plt.title('More frequent words in spam messages')
    plt.xlabel('words')
    plt.ylabel('number')
    plt.show()
```

绘制结果如下图所示：

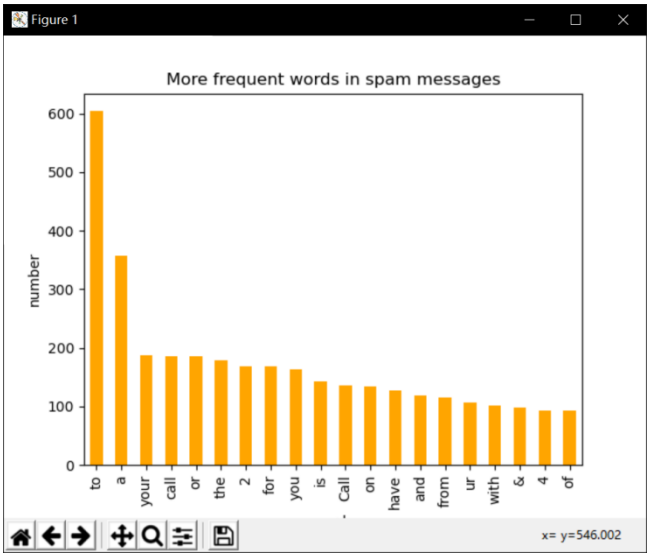

图 5. Spam 邮件词频统计柱状图（前 20）

通过对比图 4 和图 5，我们不难发现，Ham 邮件和 Spam 邮件中出现次数较高的词语几乎相同，也就是说这些词语对于筛选 Ham 邮件以及 Spam 邮件没有任何作用，因此我们需要考虑对这些文本进行预处理。

4.2.4 文本预处理

我们将文本中大量的停用词（stop_word）进行删除，使其留下真正对我们进行邮件内容分析并进行邮件筛选有用的内容。

```
f = feature_extraction.text.CountVectorizer(stop_words = 'english')
X = f.fit_transform(data["v2"])
print(np.shape(X))#得到处理后的特殊值
```

同时我们将变量 spam/non-spam 转换为二进制变量，然后将数据集分为训练集和测试集。

```
data["v1"]=data["v1"].map({'spam':1,'ham':0})
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, data['v1'], test_size=0.33,
random_state=42)
print([np.shape(X_train), np.shape(X_test)])
```

4.2.5 朴素贝叶斯进行预先分析

为了更好显示出朴素贝叶斯与 SVM 向量机之间的对比，我们先对同样的样本进行朴素贝叶斯分析，并得出结果。

```
list_alpha = np.arange(1/100000, 20, 0.11) #样本容量
score_train = np.zeros(len(list_alpha))
score_test = np.zeros(len(list_alpha))
recall_test = np.zeros(len(list_alpha))
precision_test= np.zeros(len(list_alpha))
count = 0 #参数调入
for alpha in list_alpha:
    bayes = naive_bayes.MultinomialNB(alpha=alpha)
    bayes.fit(X_train, y_train)
    score_train[count] = bayes.score(X_train, y_train)
    score_test[count]= bayes.score(X_test, y_test)
    recall_test[count] = metrics.recall_score(y_test, bayes.predict(X_test))
    precision_test[count] = metrics.precision_score(y_test, bayes.predict(X_test))
    count = count + 1                                    #贝叶斯具体算法
matrix = np.asmatrix(np.c_[list_alpha, score_train, score_test, recall_test, precision_test])
models = pd.DataFrame(data = matrix, columns =
                ['alpha', 'Train Accuracy', 'Test Accuracy', 'Test Recall', 'Test Precision'])
print(models.head(n=10))                                 #将数据存入 dataframe 中
best_index = models['Test Precision'].idxmax()
print(models.iloc[best_index, :])
m_confusion_test = metrics.confusion_matrix(y_test, bayes.predict(X_test))
print(pd.DataFrame(data = m_confusion_test, columns = ['ham', 'spam'],index = ['ham',
'spam']))
accuracy = []
precision = []
recall = []
accuracy = models['Test Accuracy']
precision = models['Test Precision']
recall = models['Test Recall']
plt.plot( accuracy, color='red', label='accuracy')
plt.plot( precision, color='skyblue', label='precisio')
plt.plot( recall, color='blue', label='recall')
plt.legend(loc='upper right')
my_x_ticks = np.arange(0, 9, 1) #X 轴的范围和刻度值
plt.xticks(my_x_ticks)
plt.xlim((0, 9))#X 轴那个线的范围
plt.show()
```

并绘制出朴素贝叶斯的分类结果的准确率、精准率、召回率的折线图如下：

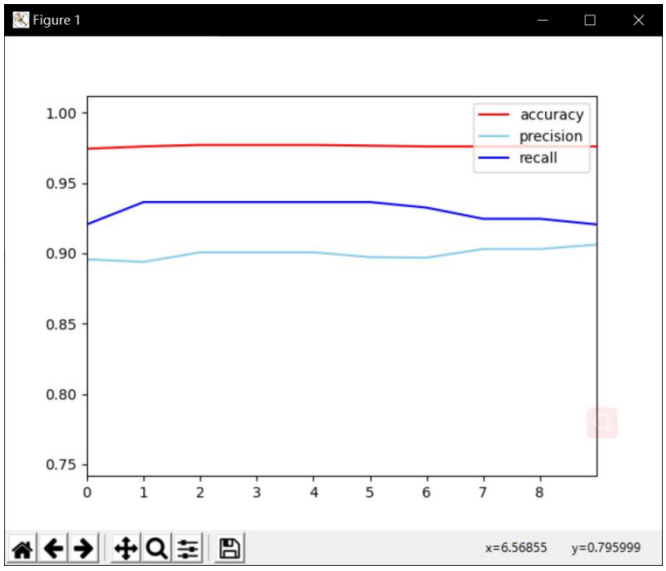

图 6. 朴素贝叶斯分类的准确率、精准率、召回率折线图

结合其结果进行分析，朴素贝叶斯分类整体的精确率在 90%以上，召回率在 92%-94%区间内微小浮动。通过图 6 的折线图，我们可以发现朴素贝叶斯虽然可以较好的将垃圾邮件从大量的邮件中过滤出来，但同时也存在着将一部分正常邮件判定为垃圾邮件进行过滤。

4.2.6 SVM 向量机处理结果

我们采用 SVM 的 SVC 模块进行样本分类。在 100000 个样本，取第 500-2000 个样本，并以 100 为间隔分组。

C：C-SVC 的惩罚参数 C 的默认值是 1.0。C 越大，相当于惩罚松弛变量，希望松弛变量接近 0，即对误分类的惩罚增大，趋向于对训练集全分对的情况，这样对训练集测试时准确率很高，但泛化能力弱。C 值小，对误分类的惩罚减小，允许容错，将他们当成噪声点，泛化能力较强。

kernel ：核函数，默认是 rbf，也可以是‘linear’，‘poly’，‘rbf’。

```
for C in list_C:
    svc = svm.SVC(C=C)
    svc.fit(X_train, y_train)
    score_train[count] = svc.score(X_train, y_train)
    score_test[count]= svc.score(X_test, y_test)
    recall_test[count] = metrics.recall_score(y_test, svc.predict(X_test))
    precision_test[count] = metrics.precision_score(y_test, svc.predict(X_test))
    count = count + 1                                    #SVM 具体算法
matrix = np.asmatrix(np.c_[list_C, score_train, score_test, recall_test, precision_test])
models = pd.DataFrame(data = matrix, columns =
                ['C', 'Train Accuracy', 'Test Accuracy', 'Test Recall', 'Test Precision'])
print(models.head(n=10))                              #数据转为 dataframe 的形式
best_index = models['Test Precision'].idxmax()
```

```
print(models.iloc[best_index, :])
m_confusion_test = metrics.confusion_matrix(y_test, svc.predict(X_test))
print(pd.DataFrame(data = m_confusion_test, columns = ['ham', 'spam'],index = ['ham', 'spam']))
accuracy = []
precision = []
recall = []
accuracy = models['Test Accuracy']
precision = models['Test Precision']
recall = models['Test Recall']
plt.plot( accuracy, color='red', label='accuracy')
plt.plot( precision, color='skyblue', label='precisio')
plt.plot( recall, color='blue', label='recall')
plt.legend(loc='upper right')
plt.xlim((0, 9))#调 X 轴范围
my_x_ticks = np.arange(0, 9, 1)#调刻度
plt.xticks(my_x_ticks)
plt.show()
```

并通过 plt 绘图模块进行 SVM 分类结果的绘制：

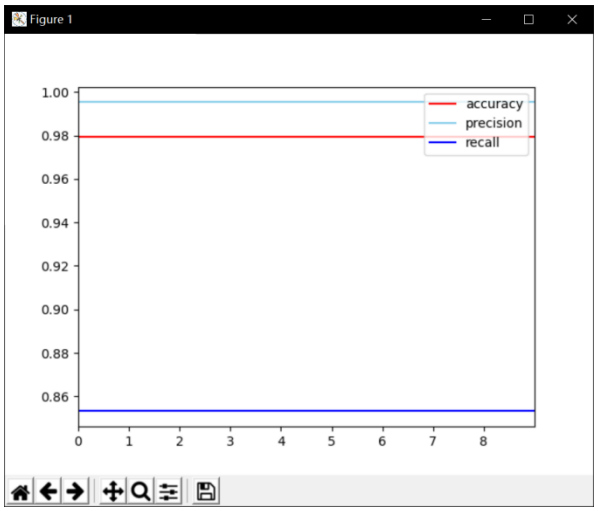

图 7. SVM 模型的准确率、精准率、召回率折线图

结合其结果进行分析，可以看出，通过 SVM 向量机模型进行垃圾邮件过滤结果整体情况是要相较朴素贝叶斯或者是贝叶斯多项式模型要好的。我们通过了大量的数据进行分析和检测，整体的精确率达到 98%以上，召回率也稳定在 85%-86%区间之内。针对于此可以看出，SVM 向量机模型相比于朴素贝叶斯模型，能够更好地将垃圾邮件从大量的邮件中过滤出来，

5. 总结及未来工作展望

5.1 总结

本次的垃圾邮件过滤实验主要是针对静态的有一定数量的邮件训练集进行的学习、过滤。根据两种不同方式的计算情况来看，贝叶斯多项式可以提供更高的精确率，保证每一个被过滤的邮件或是信息均是垃圾邮件没有错误，但同时也会有一部分的垃圾邮件因为词汇库的不充足或是词汇的新颖程度导致未被能分辨并过滤出来。在现实生活中，有点像手机短信的过滤，针对明显的垃圾短信是会被直接过滤，但是被包装后的垃圾短信还是可以不被过滤而被接受到。

另外的 SVM 算法在过滤垃圾邮件时体现出了更好的性能。对比朴素贝叶斯算法，SVM 算法更好的将垃圾邮件进行了过滤，得到的垃圾邮件中混淆的正常邮件仅在不到 1%之内。同时，随着精确率的提高，最终的召回率有所下降，维持在 85%左右，说明仍然存在着一部分垃圾邮件无法过滤出来。整体比较来看，要比贝叶斯多项式的效果好很多，也更加稳定。现在的市场上很多进行垃圾邮件过滤方法不会单单只使用 SVM 这一种单一的方法，而是更多的将多种方式融合来提高其召回率的前提下，保证其高精准率。

5.2 未来工作展望

（1）在实际网络环境中的应用。本次主要使用静态垃圾邮件样本集进行实验,这并不能完全仿真实际应用中的邮件信息。实际中的邮件内容千差万别,是否为垃圾邮件的标准也很难确定,因此,下一步工作应考虑将算法应用到实际邮件捕获、识别过程中;

（2）随着邮件逐渐增加，训练样本集会不断增大,增加系统开销。提高整体过滤垃圾邮件的工作效率也是一个难题。