doi 10.3969/j.issn.1671-1122.2012.09.006

基于贝叶斯算法和费舍尔算法的垃圾邮件过滤系统设计与实现

范仕伦，薛天俊，夏玮
（天津师范大学，天津，300387）

摘　要：贝叶斯过滤算法和费舍尔过滤算法均是利用统计学知识对于垃圾邮件进行过滤的算法，有着良好的过滤效果。该文设计将某一词组（单词）出现概率使用加权计算的方法，改善了朴素贝叶斯算法和朴素费舍尔的邮件过滤算法对于出现较少的单词误判情况，使系统对于垃圾邮件判断的准确率上升。设计可以使用个性化的垃圾邮件过滤方案，支持使用邮件下载协议（POP3、IMAP 协议）从邮件服务器下载邮件，以及使用邮件解析协议（MIME 协议）对于邮件进行解析，支持邮件发送协议（SMTP 协议）帮助用户发送邮件。

关键词：垃圾邮件过滤；贝叶斯算法；费舍尔算法
中图分类号：TP393.08　文献标识码：A　文章编号：1671-1122（2012）09-0018-05

Spam Email Filter System based on Bayesian Algorithm and Fisher Algorithm Design and Implementation

FAN Shi-lun, XUE Tian-jun, XIA Wei
(*Tianjin Normal University, Tianjin* 300387, *China*)

Abstract: Bayesian filtering algorithm and Fisher filtering algorithm which are use of statistical knowledge for the spam filtering algorithm have a good filtering effect. The design which uses weighted method to calculate words probability improves situations which the Naive Bayesian algorithm and the Naive Fisher algorithm are misjudged when they find few words in emails and increases spam judgment accuracy rate. The design which uses user's personalized filtering scheme filters spam emails.The design which uses POP3 protocol or IMAP protocol supports to download emails from the mail server,analyzes emails which use MIME protocol and helps users to send emails which uses SMTP protocol.

Key words: spam filtering; bayesian algorithm; fisher algorithm

1 相关工作

1.1 算法比较器的设计与实现

1.1.1 算法模拟器概述

在进行系统设计之前，首先制作算法模拟器，该算法模拟器从 9272 封正常邮件和 25088 封垃圾邮件中随机选择需要进行训练和过滤的邮件，在挑选邮件的过程中，使用哈希表数据结构，保证抽取邮件的唯一性，即训练邮件和过滤邮件每封不同，同时为了体现出算法的随机性，采用随机抽取阈值（贝叶斯算法[1-3]）和上下限概率值（费舍尔算法）的方法，每个算法选用 5 个不同的参数对同样邮件进行过滤，最后对过滤算法的查准率、查全率、计算时间进行对比，得出实验结果。

1.1.2 算法模拟器的设计实现

算法模拟器制作过程中使用的编程工具是 Visual Studio 2010，采用 C# 语言进行编程，整个算法模拟器的代码数量在 5000 行，制作过程中使用了 C# 的窗体编程知识、线程知识、IO 操作知识、贝叶斯概率知识、数据结构的哈希表知识。

1.2 邮件客户端程序的设计与实现

1.2.1 邮件客户端程序概述

邮件客户端模拟程序完整实现了市面上流行的 Foxmail、Outlook 客户端的主体功能，并且配备有完整的垃圾邮件过滤体系，

收稿时间 2012-07-28
作者简介 范仕伦（1989-），男，天津，本科，主要研究方向：软件工程；薛天俊（1989-），男，河南，本科，主要研究方向：软件工程；夏玮（1973-），女，河北，副教授，博士，主要研究方向：信息安全。

即“四重过滤模式”：

1）黑白名单过滤：主要针对“发件人”进行过滤。

2）关键词过滤：主要针对“发件标题”和“发件时间”进行过滤。

3）高级算法过滤：主要针对“发件内容”进行过滤。

4）手动过滤：方便用户手动完成已过滤邮件的再划分，同时用户可以通过已经过滤的邮件对训练器完成再次训练，通过新训练集对邮件进行过滤，提升邮件的过滤效果。

1.2.2 邮件客户端程序的设计实现

邮件客户端程序制作过程中使用的编程工具是Visual Studio 2010，采用C#语言进行编程，整个邮件客户端程序的代码数量在3.5万行，制作过程中使用了C#的窗体编程知识、线程知识、IO操作知识、贝叶斯概率知识[4]、数据结构的哈希表[5, 6]知识。在技术框架方面采用三层架构、接口—实现模式，该架构和模式主要应用于数据存储方面，使用该架构和模式，可以增强程序扩展性、方便日后对于程序的升级和维护。系统整体框架设计如图1所示。

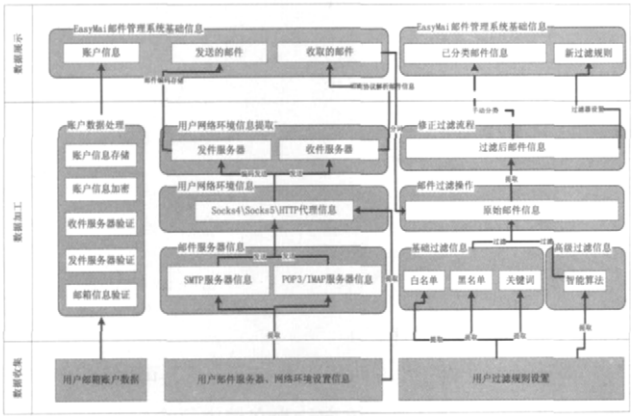

图1 邮件系统整体架构设计图

系统整体上分为数据收集部分、数据加工部分、数据展示部分共三部分。

1）数据收集部分

（1）收集用户邮箱信息：主要完成用户邮箱账户信息的收集工作；（2）收集用户使用的网络环境信息：主要完成用户网络代理信息的收集工作，即完成对于SOCKS4代理、SOCKS5代理[7, 8]、HTTP代理的信息收集；（3）收集用户邮件过滤规则信息：主要完成用户设置的白名单信息、黑名单信息、关键词信息、邮件训练集信息的收集工作。

2）数据加工部分

数据加工部分的应用场合主要是用户进行邮件发送操作、邮件下载操作、邮件过滤操作。

3）数据展示部分

从存储文件中读取用户的收件信息、用户的发件信息、用户的过滤设置信息、用户账户信息、用户网络设置信息等相关信息，如果数据经过加密的话，按照解密规则解密，没有进行加密的，直接提取相关信息，并显示在窗体界面上。

系统总体流程框架设计如图2所示。

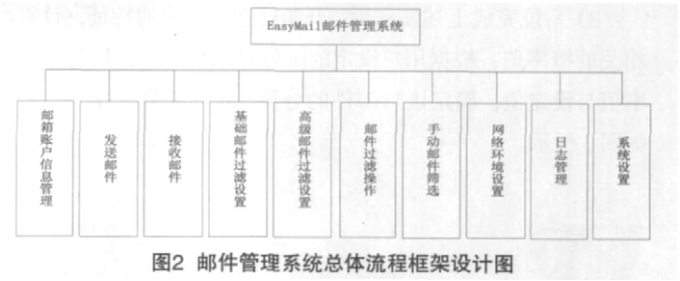

图2 邮件管理系统总体流程框架设计图

系统的使用者为个人用户，个人用户属于用户和管理员身份，为个人用户提供邮箱账户信息管理、发送邮件、接收邮件、基础邮件过滤设置、高级邮件过滤设置、邮件过滤操作、手动邮件筛选、网络环境设置、日志管理、系统设置共10项功能。

2 算法概述

2.1 概念

2.1.1 特征

本文所涉及的“特征”指进行概率计算的最小单元，对于英文即指每个单独的英文单词，对于中文指的是拥有独立意义的单字或词组。

2.1.2 贝叶斯算法

贝叶斯分类算法利用统计学中的概率知识对于事物进行分类，具有很多的优点。贝叶斯分类算法的分类速度快，准确性较高，被广泛应用于垃圾邮件分类和文档分类设计中。本设计使用的贝叶斯算法[9, 10]对于朴素的贝叶斯算法进行了一定程度的改进，对于特征概率进行加权计算，使算法对于较少出现的特征的识别能力得到进一步的增强，改善了朴素贝叶斯算法的垃圾邮件过滤效果。

2.1.3 费舍尔算法

贝叶斯算法进行邮件分类计算时，假设邮件中的每个特征是彼此独立的，这种假设在实际情况中是不成立的。费舍尔算法针对于贝叶斯算法的假设进行了改进，可以给出更为精确的邮件概率计算值，提升邮件过滤效果。该方法为文档中每个特征都求得分类概率，然后将概率组合起来，进行分类判断。本设计中使用的费舍尔算法同样对于朴素费舍尔算法进行了改进，对于特征概率进行加权计算，改善了朴素费舍尔算法的垃圾邮件过滤效果。

2.2 算法介绍

2.2.1 贝叶斯算法

2.2.1.1 贝叶斯算法描述

在贝叶斯算法的使用过程中，主要流程为：

1）完成对于一封邮件的信息的分解，即将邮件信息进行分词处理，将分词结果保存在相关数据结构中，在本设计中

主要将数据写入到泛型变量中。

2）将已确定的邮件过滤集合（即邮件训练集），加载到内存中，使用哈希表的数据结构完成邮件训练集的加载。

3）系统完成上述工作后，开始处理邮件中的特征，计算相关的概率值，根据用户设定的邮件过滤参数，比较计算概率值与设定值，确定该封邮件的分类。贝叶斯算法实现流程如图 3 所示。

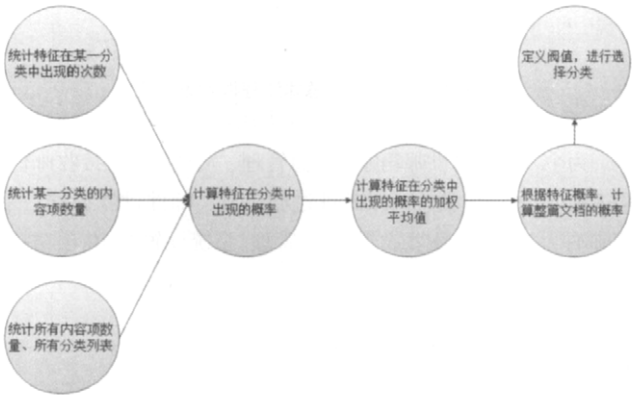

图3 贝叶斯算法实现流程图

2.2.1.2 贝叶斯算法改进

朴素贝叶斯算法假设每个特征概率彼此独立，通过计算每个特征属于某类邮件的概率，组合计算出该封邮件属于某类邮件概率，最后根据设定的阈值对于邮件进行分类。在计算过程中需要计算某个特征属于某类邮件的概率，计算公式如下：

Pr（特征 | 分类）= 某一特征在某类邮件中出现的次数 / 某类邮件数。

不难发现，如果训练集合样本过少，所计算特征并没有在某类邮件中出现过，将会导致 Pr（特征 | 分类）计算结果为 0，将对于最终的邮件分类产生影响。本设计参考文献 [11，12]，决定在计算 Pr（特征 | 分类）时，使用加权计算方法，避免出现计算概率为 0 的情况，假定单词出现权值为 1，代表一个单词，随机出现概率为 0.5。改进后的计算公式如下：

Pr（特征 | 分类）=（某一特征在某类邮件中出现的次数 + 权值 * 概率）/ 某类邮件数。

如果在训练集足够充分的条件下，某特征在某类邮件中出现次数仍然很少，Pr（特征 | 分类）概率值将逐渐接近于 0，使用这种方法，改善了朴素贝叶斯算法 [13-14] 对于垃圾邮件的分类效果。

2.2.1.3 核心代码

```
public string CalculateEmailCategory(List<string> features, ClassificationParamNumber classificationParamNumber, DictionnaryFeatureLibrary dictionnaryFeatureLibrary, double ProbabilityThreshold)
        {
            try
            {
              string emailCategory = EmailCategoryEnums.Unknow.ToString();
                double regularCategoryProbability = CalculateCategoryProbability(features, classificationParamNumber, dictionnaryFeatureLibrary, EmailCategoryEnums.Regular.ToString());// 正常邮件的分类
                double spamCategoryProbability = CalculateCategoryProbability(features, classificationParamNumber, dictionnaryFeatureLibrary, EmailCategoryEnums.Spam.ToString());// 垃圾邮件的分类
                if (regularCategoryProbability > spamCategoryProbability)
                {
                    double max = regularCategoryProbability;
                  if (max > spamCategoryProbability * ProbabilityThreshold)
                    {
                      return EmailCategoryEnums.Regular.ToString();
                    }
                    else
                    {
                        return emailCategory;
                    }
                }
              else if (spamCategoryProbability > regularCategoryProbability)
                {
                    double max = spamCategoryProbability;
                  if (max > (regularCategoryProbability * ProbabilityThreshold))
                    {
                      return EmailCategoryEnums.Spam.ToString(); ;
                    }
                    else
                    {
                        return emailCategory;
                    }
                }
                else
                {
                    return emailCategory;
                }
            }
            catch (Exception)
            {
                throw;
            }
        }
```

2.2.2 费舍尔算法

2.2.2.1费舍尔算法描述

费舍尔算法对于贝叶斯算法进行了改进，可以给出更为精确的垃圾邮件过滤结果。该方法为文档中每个特征 [15] 都求得分类概率，然后将概率组合起来，进行分类判断。

在费舍尔算法的使用过程中，其主要处理流程与贝叶斯算法的处理流程一致。

1）邮件的分词处理，将分词结果保存在泛型变量中。

2）将邮件训练集导入到内存中，在此过程中主要使用哈希表数据结构。

3）根据分词结果匹配训练集，按照费舍尔算法提供的计算公式计算邮件中的每个特征的概率，并将相关的概率进行组合计算，根据计算邮件的整体概率判断邮件的所属分类，完成邮件的过滤。

费舍尔算法实现流程如图 4 所示。

2.2.2.2 费舍尔算法改进

贝叶斯算法在进行特征概率计算之前，假设每个特征概率彼此独立，但是这种假设在实际情况下是不成立的，费舍尔算法在此方面对于贝叶斯算法进行了改进。首先，费舍尔

算法同贝叶斯算法一样会计算某特征在分类中出现的概率，然后为了反映特征概率的关联情况，根据计算结果，计算某特征在分类中随机出现的概率，即分别计算某一特征属于正常分类概率、属于垃圾分类概率，然后分别使用这两种概率除以两类概率的和。

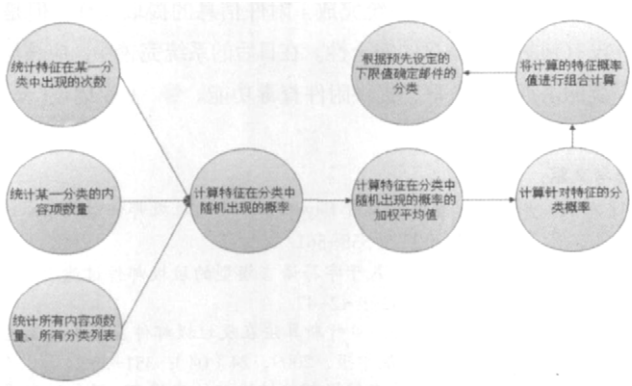

图4 费舍尔算法实现流程图

计算公式如下：

$Pr(特征|分类)_{随机} = Pr(特征|分类)_{正常} | Pr(特征|分类)_{垃圾} / (Pr(特征|分类)_{正常} + Pr(特征|分类)_{垃圾})$。

然后通过计算 Pr(特征|分类)随机的加权概率值，改善朴素费舍尔算法对于较少出现的特征判别能力较弱的情况，使用该方法进行计算，改善了朴素费舍尔算法对于垃圾邮件的分类效果。

2.2.2.3 核心代码

```
public string CalculateEmailCategory(List<string> features, ClassificationParamNumber classificationParamNumber, DictionnaryFeatureLibrary dictionnaryFeatureLibrary, double RegularLowerLimit, double SpamLowerLimit)
        {
            try
            {
              string MailResults = EmailCategoryEnums.Unknow.ToString();
                double max = 0.0;
                string[] emailCategoryText = new string[2];
               emailCategoryText[0] = EmailCategoryEnums.Regular.ToString();
                emailCategoryText[1] = EmailCategoryEnums.Spam.ToString();
                double[] collectionLimit = new double[2];
               collectionLimit[0] = RegularLowerLimit;//正常邮件概率下限
                collectionLimit[1] = SpamLowerLimit;//垃圾邮件概率下限
                double[] collectionProbability = new double[2];
               collectionProbability[0] = CalculateFisherCombinProbability(features, classificationParamNumber, dictionnaryFeatureLibrary, EmailCategoryEnums.Regular.ToString());//计算一封邮件是正常邮件概率;
                collectionProbability[1] = CalculateFisherCombinProbability(features, classificationParamNumber, dictionnaryFeatureLibrary, EmailCategoryEnums.Spam.ToString());//计算一封邮件是垃圾邮件概率
                for (int i = 0; i < 2; i++)
                {
                  if (collectionProbability[i] > collectionLimit[i] && collectionProbability[i] > max)
                    {
                        MailResults = emailCategoryText[i];
                        max = collectionProbability[i];
                    }
                }
                return MailResults;
            }
            catch (Exception)
            {
                throw;
            }
        }
```

3 实验及结果

3.1 贝叶斯算法实验及结果分析

在挑选算法的过程中，进行了相关实验，实验数据如下：共有正常邮件 9272 封、垃圾邮件 25088 封，根据用户设定的参数，随机抽选邮件进行训练、过滤，本次实验中，共抽取正常邮件和垃圾邮件各 1000 封，用来生成过滤集。各抽取正常邮件、垃圾邮件 1000 封，用来进行过滤。对实验数据采用随机阈值，共计采用 5 组阈值：4、6、10、9、2。

表1 朴素贝叶斯算法和改进贝叶斯算法过滤结果对比

阈值	朴素贝叶斯算法	查准率	改进贝叶斯算法	查准率
4	531|732|737	63.15%	1056|25|919	93.80%
6	531|732|737	63.15%	1053|29|918	93.75%
10	530|733|737	63.10%	1048|37|915	93.60%
9	531|732|737	63.15%	1048|35|917	93.65%
2	531|732|737	63.15%	1056|24|920	93.80%

注：531|732|737 代表正常邮件 531 封，未知类型邮件 732 封，垃圾邮件 737 封。

从所得的实验结果可以看出，改进的贝叶斯算法[16]比传统的贝叶斯算法性能好，在准确率上有所提高。传统的贝叶斯算法没有考虑较少特征词出现的情况。而改进后的贝叶斯算法考虑了较少特征词[17]出现的情况，使用加权概率方法进行计算，充分利用改进提高过滤系统的性能。图 5 说明贝叶斯算法查准率情况。

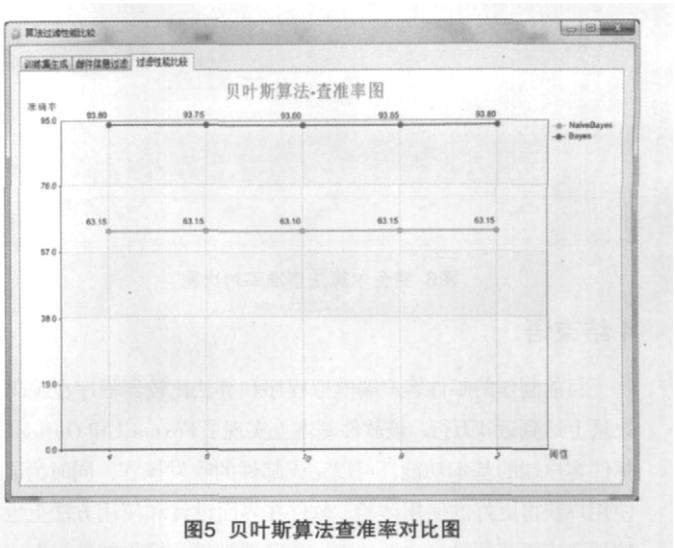

图5 贝叶斯算法查准率对比图

3.2 费舍尔算法实验及结果分析

费舍尔算法对于贝叶斯算法进行了改进，可以给出更为精确的垃圾邮件过滤结果。该方法为文档中每个特征都求得分类概率，然后将概率组合起来，进行分类判断。在挑选算法的过程中，进行了相关实验，实验数据为：共有正常邮件 9272 封、垃圾邮件 25088 封，根据用户设定的参数，随机抽

选邮件[18]进行训练、过滤，本次试验中，共抽取正常邮件和垃圾邮件各1000封，用来生成过滤集。各抽取正常邮件、垃圾邮件1000封，用来进行过滤。对试验数据采用随机邮件下限值，共计采用5组下限值：(0.5|0.6)、(0.1|0.4)、(0.9|0.8)、(0.7|0.7)、(0.3|0.5)。

(0.5|0.6)代表正常邮件概率下限值为0.5，垃圾邮件概率下限值为0.6。

表2 朴素费舍尔算法和改进费舍尔算法过滤结果对比

概率下限值	朴素费舍尔算法	查准率	改进费舍尔算法	查准率
(0.5|0.6)	1678|7|315	35.45%	1031|16|953	95.05%
(0.1|0.4)	1685|0|315	35.45%	1047|0|953	95.05%
(0.9|0.8)	1677|8|315	35.40%	1010|46|944	94.25%
(0.7|0.7)	1677|8|315	35.40%	1026|23|951	94.90%
(0.2|0.9)	1684|1|315	35.45%	1046|1|953	95.05%

注：1678|7|315 代表正常邮件1678封，未知类型邮件7封，垃圾邮件315封。

从所得的实验结果可以看出，改进的费舍尔算法比传统的费舍尔算法性能好，在准确率上都有所提高。传统的费舍尔算法没有考虑较少特征词[19]出现的情况。而改进后的费舍尔算法考虑了较少特征词出现的情况，使用加权概率方法进行计算，充分利用改进提高过滤系统的性能[20]。图6说明费舍尔算法查准率情况。

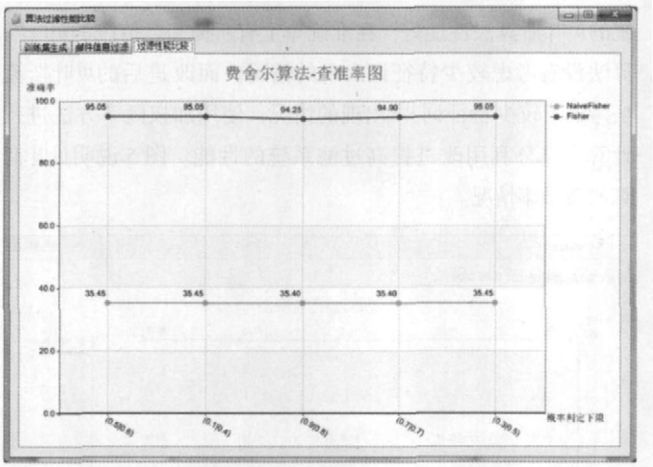

图6 费舍尔算法查准率对比图

4 结束语

目前制作的邮件客户端模拟程序和算法比较器程序在编码数量上达到近4万行，该软件基本上实现了Foxmail和Outlook邮件客户端的基本功能，可以完成邮件的收发操作，同时为了使用户获得更好的使用体验，软件在界面设计和使用方法上也接近于市面流行的客户端软件，在发送邮件、接收邮件、生成邮件过滤集的操作中，使用多线程技术，而且在挑选过滤算法后，针对于算法的缺点进行了改进，经过实验，改进后的贝叶斯算法和费舍尔算法在邮件过滤中取得良好的过滤效果。

本系统未来改进工作放在以下几点：

1）继续对于算法进行改进，比如将贝叶斯算法和神经网络算法结合，进一步改善贝叶斯算法的垃圾邮件过滤效果。

2）界面控件，为了给用户提供更好的使用体验，系统模仿Foxmail和Outlook进行界面设计，在设计的过程中，使用了第三方的皮肤控件，在日后的系统完善中也应该使用自己编写的皮肤控件。

3）附件问题，本系统完成了附件信息的提取工作，但是并没有判断附件信息的安全性。在日后的系统完善中，应该与系统的杀毒软件关联，提供附件查毒功能。（责编 张岩）

参考文献：
[1] 刘震，周明天．基于有监督Bayesian网络的垃圾邮件过滤[J]. 计算机应用，2006，26（03）：558-561.
[2] 苏绥，林鸿飞，叶正．基于字符语言模型的垃圾邮件过滤[J]. 中文信息学报，2009，23（02）：42-47.
[3] 薛松，张钟澍，殷知磊．贝叶斯算法在反垃圾邮件应用中的改进方案[J]．成都信息工程学院学报，2009，24（04）：351-355.
[4] 王晖．贝叶斯分类算法在垃圾邮件过滤中的应用[J]. 黑龙江交通科技，2009，182（04）：157-158.
[5] 卢杨竹，张新有，祁玉．邮件过滤中特征选择算法的研究及改进[J]. 计算机应用，2009，29(10)：2812-2815.
[6] 刘海峰，张学仁，姚泽清，等．基于类别选择的改进KNN文本分类[J]. 计算机科学，2009，36（11）：213-216.
[7] 陈静．基于SOCKS5协议的专用网络文件传输的设计与实现[J]. 大众科技，2009，121（09）：33-34.
[8] 谷发平，杨林，杨峰，等．一种增强型SSL安全通道建立方案设计与实现[J]. 军事通信技术，2010，31（02）：40-44.
[9] 刘宝萍，李爱军．基于神经网络集成的垃圾邮件过滤系统设计[J]. 人工智能及识别技术，2010，6（01）：171-173.
[10] 闫斐．基于贝叶斯模型的邮件过滤系统[J]. 太原师范学院学报，2010，9（02）：63-66.
[11] 邓蔚，秦志光，刘峤，等．抗好词攻击的中文垃圾邮件过滤模型[J]. 电子测量与仪器学报，2010，24（12）：1146-1152.
[12] 常青．基于词条时序的朴素贝叶斯垃圾邮件过滤方法[J]. 微电子学与计算机，2010，27（05）：212-216.
[13] 夏超，徐德华．一种改进的贝叶斯邮件过滤算法[J]．计算机与现代化，2010，182（10）：125-128.
[14] 李鹏．基于贝叶斯理论的神经网络算法研究[J]. 光机电信息，2011，28（01）：28-31.
[15] 赵静，刘培玉，许明英．邮件过滤中特征选择方法的性能评价与分析[J]. 计算机应用研究，2012，29（02）：693-697.
[16] D, GROBELNIK M. Feature selection for unbalanced class distribution and Naive Bayes[C]. Proc of ICML ' 99. San Francisco : Morgan Kaufmann，1999, 258-267.
[17] SEBASTIANI F. Machine learning in auomated text categorization[J]. ACM Computing Surveys, 2002, 34(01): 1-47.
[18] Sakkis G, Androutsopoulos I, Paliouras G, et al. A memo- ry-based approach to anti - spam filtering for mailing lists[J]. Information Retrieval, 2003, 6(01): 49-73.
[19] JORGENSEN Z, ZHOU Y, INGE M, et al. A Multiple Instance Learning Strategy for Combating Good Word Attacks on Spam Filters[J]. Journal of Machine Learning Research, 2008, 8: 1115-1146.
[20] LIU W, CHAWLA S. Mining adversarial patterns viaregularized loss minimization[J]. Machine Learning, 2010, 81: 69-83.