Bayesian Analysis Group Project

Text Spam Classifier / Email Spam Classifier

He Yi 1630005019

Liu Jun Jie 1630005038

Miao Shu Tong 1630005045

Sheng Wang Wang 1630005047

Table of Contents

[Abstract 3](#_Toc533595367)

[Introduction 3](#_Toc533595368)

[Naive Bayes Classification 3](#_Toc533595369)

[2.2 Filtering process 4](#_Toc533595370)

[2.3 Application 5](#_Toc533595371)

[Paul Graham’s approach 6](#_Toc533595372)

[3.1 Filtering process 6](#_Toc533595373)

[3.2 Application 7](#_Toc533595374)

[3.3 Tim Peter’s criticism 7](#_Toc533595375)

[Unknown Tokens 8](#_Toc533595376)

[Modified sample emails: 8](#_Toc533595377)

[Problem: 8](#_Toc533595378)

[Solution: 9](#_Toc533595379)

[Application 9](#_Toc533595380)

[Monte Carlo Simulation 11](#_Toc533595381)

[Setup: 12](#_Toc533595382)

[Appendix 12](#_Toc533595383)

[Data Pre-Processing 12](#_Toc533595384)

[Function Writing 12](#_Toc533595385)

[Build a text database 13](#_Toc533595386)

[Define classifiers and use normal mail tests that are not easily recognizable 13](#_Toc533595387)

[build a set of spam training data with TDM 15](#_Toc533595388)

[Bibliography 19](#_Toc533595389)

# Abstract

In this project, our group investigate a statistical e-mail filter, Bayesian e-mail filter (Gelman, 2013), which is widely used nowadays. In our project, at the beginning, we will get a lot of e-mails as the sample from our real life in order to create the training data set. And then we will get a detailed process about how to use naive Bayes in filtering process and the classification. Among all the different methods of implementing naive Bayes in classification, Paul Graham’s method is the most famous one and it is widely used. He introduced a new formula for calculating token values and overall probabilities of an email being classified as spam. But there is a unrealistic assumption in his formula which assume the number of spam and ham are equal in the data set for everyone. And we will also tell about Tim Peter adjust the formula a little correctly to fit well all the data set. Beside this method we will also explain another method which is more accurate created by Gary Robinson. Where Robinson’s method lets us consider both our background knowledge and the limited data, we have got from unknown tokens in the prediction of their token values. Again, their performances will be measured against our sample emails. In the last we will use our testing data set which also create by the e-mail sample from real life to test the methods are useful or not.

# Introduction

Naive Bayes classifiers (Rish, 2001) are a popular statistical technique of e-mail filtering. They typically use bag of words features to identify spam e-mail, an approach commonly used in text classification.

Naive Bayes classifiers work by correlating the use of tokens (typically words, or sometimes other things), with spam and non-spam e-mails and then using Bayes' theorem to calculate a probability that an email is or is not spam.

Naive Bayes spam filtering is a baseline technique for dealing with spam that can tailor itself to the email needs of individual users and give low false positive spam detection rates that are generally acceptable to users. It is one of the oldest ways of doing spam filtering, with roots in the 1990s. (Ng, 2002)

# Naive Bayes Classification

The following table shows the frequency of a token appear in spam and ham email groups. For each email received, the appearances of each token count only once. The training dataset has a total number of 432 spams and 2170 hams, we do not list all of them

This table is based on some previous work. Then, it is reproduced according to previous dataset because some words are not included in these two example emails. Multiplying the ratio in previous dataset by the total number in the new table, and here only list the appearances of words that appear in two example emails.

|  |  |  |
| --- | --- | --- |
| feature | Appearance in spam | Appearance in ham |
| a | 165 | 1235 |
| advised | 12 | 42 |
| as | 2 | 579 |
| chance | 45 | 35 |
| Clarins | 1 | 6 |
| exercise | 6 | 39 |
| for | 378 | 1829 |
| free | 253 | 137 |
| fun | 59 | 9 |
| girlfriend | 26 | 8 |
| have | 297 | 2008 |
| her | 38 | 118 |
| I | 9 | 1435 |
| just | 207 | 253 |
| much | 126 | 270 |
| now | 221 | 337 |
| paying | 26 | 10 |
| receive | 171 | 98 |
| regularly | 9 | 78 |
| take | 142 | 278 |
| tell | 76 | 89 |
| the | 185 | 930 |
| time | 212 | 446 |
| to | 389 | 1948 |
| too | 56 | 141 |
| trial | 26 | 13 |
| vehicle | 21 | 58 |
| Viagra | 39 | 19 |
| you | 391 | 786 |
| your | 332 | 450 |

Table of training data

## Filtering process

Firstly, we break the email by individual wordsand denote this email as E. then, denote a spam email as S, a ham as H. The probability of receiving the email E is equal to the probability of receiving the words .

Not that an assumption is made that the words in an email are independent of each other and randomly displayed leads to simplify the computations of probabilities in the Bayes formula and reduce the size of the training dataset we need.

It is easy to get,

P(E|S) is probability of given an email from email class S that it is the email E and P(E|H) is probability of given an email from email class H that it is the email E.

The following step is to compute the posterior probability. This is the vital part of the entire classification.

To recognize whether a spam or not, it is a good way to compare the ratio of, . Therefore, take ratio of these two probabilities,

To avoid the products in the above equations can be extremely small values if there is a big amount of words , just apply log to the probability ratio.

The conclusion can be made: classify the email is a ham if it is less than or the email is spam if it is greater than 0.

## Application

Apply the naive Bayes classification as follow:

For the first one,

and for the second one,

=−5.1473422

Therefore, the result indicates email ES is spam shows that email EH is classified as ham. Finally, by using naive Bayes classification, two sample emails are classified into email groups.

# Paul Graham’s approach

## Filtering process

Paul Graham is slight different with the last naive Bayes method on classifications. The first one is that he assume the size of his groups are almost the same. The second one is that he enlarges the domain of tokens by considering dashes, apostrophes, alphanumeric characters and dollar signs and so on as tokens. The third one is that double words’ appearances in ham when calculate the overall probability. The last one is that only pick up the most interesting 15 tokens when getting result.

At first, denote the token’s frequency of appearances in spam is SH and the token’s frequency of appearances in ham is AH. In this example, for each word,



Graham multiplies the ham two times, which shows he tries to enhance the effect of all the hammy words in overall probability. This slight bias is helpful to reduce the probability that filter misjudges a ham as spam, which leads to minimize the number of false positives.

In Graham’s classification, the filter assigns a probability of a token only when its appearance in total is more than five times. That is to say AS + 2AH > 5 for each token in the dataset. For a token’s appearance in both email groups is less than five times, this type of tokens will be considered as unknown tokens. In Graham’s method, he uses 0.4 because he treats unknown tokens are usually fairly innocent. What’s more, if one token which only appears in one of the email groups, called single corpus tokens. Assign its probability is 0.99 for a token that only appears in spam emails and 0.01 for only appears in ham emails.

When computing, only pick up the most interesting 15 words but not taking all the tokens into account. The reason is that Graham finds out a large percentage of spams in his dataset tend to contain around 15 Sammy words in them. The method about how to measure the interestingness of each token is like that. By its absolute value of

Therefore, clearly noise will add by consider more than 15 numbers. So when n=15,





The conclusion can be made: classify the email as a spam email if the overall probability P(S|E) is larger than 0.9 and classify the email as a ham email if the overall probability P(S|E) is near to 0 with Paul Graham’s approach.

## Application

Now, try to compute two over probability by Paul Graham’s approach:

= 0.9988236

=8.913810

Therefore, the conclusion can be made.

But how about only consider the top 5 most interesting tokens in this relatively small email samples?

=0.9997092

=4.993545

The conclusion also can be made successfully.

## Tim Peter’s criticism

Tim Peter claims that Graham mistakenly assumed P(S) = P(H) = 0.5 in everyone’s dataset. But this is unlikely to be true. Therefore, the way Tim Peter applies naive Bayes classification to make improvement.

Firstly, let E be the email to classify. And by Bayes theorem,







3.3.2 Tim Peter’s criticism application

For the sample spam:

=0.99999999

For the sample ham:

=0.9294468

Unfortunately, Tim Peter’s new formula makes a wrong classification. If we compare the ratio of each token’s number of appearance in both email groups, we will discover that the spammy tokens appear a lot more often in spam than the hammy ones that is the reason why doubling words’ appearances in ham when calculate the overall probability in Paul Graham’s approach he wants to minimize the false positives by lower the boundary of spam classification. I hope there is other method to improve the filters’ performance without boosting the number of spam we receive in the email inbox.

# Unknown Tokens

## Modified sample emails:

Due to investigate the consequence of containing an unknown token in the email, we add one extra word into each of the sample emails we had earlier, where these two extra words are not included in our training dataset. Word *great* is added into the sample spam and word *do* is added into the sample ham. The filters will consider them as *unknown tokens* in the classification later.

## Problem:

Before we calculate the overall probability of an email, we need to know the probabilities of these unknown tokens are equal to 0.

In the naive Bayes classification, we calculate token probabilities as follow, where AS and AH are the number of appearances of each token in spam and ham email groups respectively:

The unknown tokens cause these two equations become 0, since their probabilities must be and .

And let the overall probability of the email is calculated by this way:

And Still, the whole equation will be 0 due to one of the denominators . Hence this type of filtering method fails to classify messages containing unknown tokens.

In the Graham’s method, using a different route to measure token probabilities:

We have AS=AH=0, in that an unknown token has never appeared in the training dataset before. This method still collapses since the denominator equal to 0.

## Solution:

Firstly, I give the Graham’s solution to deal with the unknown tokens. He just attaches a probability of 0.4(not 0.5 before) to any unknown tokens. From his observations, he believes that spammy words are always familiar and unknown tokens should be relatively innocent. This neutral probability could protect filters from dictionary attacks which contain lots of random words that normally don’t appear in our datasets.

The second solutions will be Gary Robinson’s smoothing method. It is one of the greatest solutions to solve this problem. The formula is based on an assumption that the classification of an e-mails containing word is binomial random variable with a beta distribution prior. We totally have n trails, which represent the classification result of a new email containing word . If the email is classified as spam, this trail will be a success, and if the email is classified as ham, this trail will be a fail. Under the big assumption that all tokens are independent to each other, this method clearly meets all the criteria for being a binomial experiment. Then, we assume a beta distribution for the prior to help us to predict the probability of an extra token (assume being a spammy word) as follow:

Then, we give two parameters to predict the probability of unknown tokens:

𝛼+𝛽=𝐶 and 𝛼=𝐶×𝐺

𝐶: the confidence level of our general knowledge

𝐺: predicted probability of an unknown token being in a spam, based on our general knowledge of the word.

The value of C and G will be adjusted in real problem for the high accuracy in the classification and low false positives. In generally, we set a starting value for C=1 and a neutral probability for G=0.5.

In general, if a token has a few appearances in either spam or ham email groups, the probability of this token will be weakened. Then let us consider and compare both our general knowledge () and the new data we have calculated from unknown tokens by the above formula (. Finally, we can get a more reliable classification result which does not include any extreme values from the unknown tokens.

## Application

Since Robinson’s solution completely changed the way of measuring our token probabilities, we need to calculate the new probability table by this new formula.

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Appearances in Spam | Appearances in Spam |  |
| A | 165 | 1235 | 0.1181299 |
| Advised | 12 | 42 | 0.2272727 |
| As | 2 | 579 | 0.0042955 |
| Chance | 45 | 35 | 0.5617284 |
| Clarins | 1 | 6 | 0.1875000 |
| Exercise | 6 | 39 | 0.1413043 |
| For | 378 | 1829 | 0.1714221 |
| Free | 253 | 137 | 0.6483376 |
| Fun | 59 | 9 | 0.8623188 |
| Girlfriend | 26 | 8 | 0.7571429 |
| Have | 291 | 2008 | 0.1267391 |
| Her | 38 | 118 | 0.2452229 |
| I | 9 | 1435 | 0.0065744 |
| Just | 207 | 254 | 0.4501085 |
| Much | 126 | 270 | 0.3186398 |
| Now | 221 | 337 | 0.3962433 |
| Paying | 26 | 10 | 0.7162162 |
| Receive | 171 | 98 | 0.6351852 |
| Regularly | 9 | 87 | 0.0979381 |
| Take | 142 | 287 | 0.3313953 |
| Tell | 76 | 89 | 0.4608434 |
| The | 185 | 930 | 0.1662186 |
| Time | 212 | 446 | 0.3224583 |
| To | 389 | 1948 | 0.1665954 |
| Too | 56 | 141 | 0.2853535 |
| Trial | 26 | 13 | 0.6625000 |
| Vehicle | 21 | 58 | 0.2687500 |
| Viagra | 39 | 19 | 0.6694915 |
| You | 391 | 786 | 0.3323430 |
| Your | 332 | 450 | 0.4246488 |
| Great | 0 | 0 | 0.5000000 |
| Do | 0 | 0 | 0.5000000 |

Table 1. Training data table with Gary Robinson's method

Next, we use Tim Peter’s formula to calculate the overall probabilities because the number of spam and ham are not equal. In addition, we can check whether the Peter’s formula still classifies an email with the new token probabilities. It is bad to have a false positive at all time.

Classification of the sample spam:

And

*Then*

Classification of the sample ham:

And

992480

*Then*

Finally, we still get the correct classification results in Peter’s formula by using Robinson’s method to calculate new token probabilities. We also give a table to compare the and , which shows a large difference in token probabilities. Obviously, all the tokens become more hammy than before within the new formula. Robinson’s formula improves the accuracy for many neutral-looking emails. It successfully classified our sample ham as a legitimate email this time, which reduced the number of false positives.

In conclusion, Robinson’s formula solves this problem with unknown tokens smoothly and generates a more accurate classification for emails containing rare words.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | In Spam | In Ham |  |  |
| Advised | 12 | 42 | 0.2272727 | 0.5893536 |
| As | 2 | 579 | 0.0042955 | 0.0170552 |
| Clarins | 1 | 6 | 0.1875000 | 0.4556909 |
| Exercise | 6 | 39 | 0.1413043 | 0.4359180 |
| For | 378 | 1829 | 0.1714221 | 0.5093555 |
| Have | 291 | 2008 | 0.1267391 | 0.4212816 |
| Her | 38 | 118 | 0.2452229 | 0.6179742 |
| I | 9 | 1435 | 0.0065744 | 0.0305419 |
| Just | 207 | 253 | 0.4501085 | 0.8042995 |
| Regularly | 9 | 87 | 0.0979381 | 0.3419477 |
| Take | 142 | 287 | 0.3313953 | 0.7130824 |
| The | 185 | 930 | 0.1662186 | 0.4998070 |
| Time | 212 | 446 | 0.3224583 | 0.7048131 |
| To | 389 | 1948 | 0.1665954 | 0.5007694 |
| Your | 332 | 450 | 0.4246488 | 0.7875038 |
| Do | 0 | 0 | 0.5000000 | 0.4000000 |

Table 1. Comparison table of token probabilities in sample ham

# Monte Carlo Simulation

Direct simulation requires that the posterior distributions of θ1 and θ2 are directly available to be simulated.

Usually we can write down the prior and the likelihood, so that usually the function prior × likelihood is available. The problem is that the normalizing constant is missing, so direct simulation is not possible.

Idea: Start with a random sample from a guess at the posterior distribution, and modify the sample to be more like the true posterior.

### Setup:

Let g(θ) be a first guess approximation to the true posterior distribution. Let

f(θ|data) = likelihood × prior, which we can write down.

The SIR algorithm follows three steps:

1. Sample: Draw a random sample.

2. Importance: Attach a weight to each element in the sample,

3. Resample: Draw a sample with replacement from the original sample, using the weights calculated in step 2.

The sample in step 3 is an approximate random sample from the posterior distribution, without needing to directly simulate from it.

# Appendix

## Data Pre-Processing

As the beginning, we have to do the data pre-processing:the data were divided into training group and test group, and the training group was used to train the model, and the test group was used to evaluate the model. There are three types of e-mail: 1. The hams (normal emails) which are easy to classified; 2. The hams (normal emails) which are hard to classified; 3. The **spam** (inormal emails)

*# Training Dataset*

spam\_train.path <- "Email-Spam-Classifier/data/spam/" easy\_ham\_train.path <- "Email-Spam-Classifier/data/easy\_ham/" hard\_ham\_train.path <- "Email-Spam-Classifier/data/hard\_ham/"

*# Testing Dataset*

spam\_test.path <- "Email-Spam-Classifier/data/spam\_2/" easy\_ham\_test.path <- "Email-Spam-Classifier/data/easy\_ham\_2/" hard\_ham\_test.path <- "Email-Spam-Classifier/data/hard\_ham\_2/"

## Function Writing

*# Extract Email Body ; Single Element Vector*

get.msg <- **function**(path) {  
f <- **file**(path, open ="rt", encoding = "latin1")  
*#rt: Open for reading in text mode*text <- **readLines**(f)  
*# transform the text as "UTF-8"  
# The message always begins after the first full line break*msg <- **tryCatch**(text[**seq**(**which**(text **==** "")[1] **+** 1, **length**(text), 1)],

error = **function**(e)e)  
*#add a 'tryCatch' to catch the Exception***close**(f)  
**return**(**paste**(msg, collapse ="\n"))  
*# setting msg as a single element vector with '\n'* }

## Build a text database

One way to quantify the frequency of spam feature terms is to construct a Term Document Matrix (TDM).

TDM is an N\*M matrix, whose rows correspond to the word items extracted from all documents in a specific corpus, and whose column pairs correspond to all documents in the corpus

Corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus corpus ) this case USES VectorSource function to construct source object to construct corpus with email vector. Conrol parameter explanation: tell tm how to clean and organize text stopwords=TRUE: remove 488 most common English punctuation words. RemovePunctuation, removeNumbers: removePunctuation and number minDocFreq=2: words with more than 1 occurrence in text can appear in the line of TDM.

*# Term Document Matrix*get.tdm <- **function**(doc.vec) { *#doc.vec is all the files content*

doc.corpus <- **Corpus**(**VectorSource**(doc.vec)) *# set up a corpora/natural language database*

control <- **list**(stopwords = TRUE, removePunctuation = TRUE, removeNumbers = TRUE, minDocFreq = 2)

*# This control is the option to set how to extract text (extract text requirements)*

*# remove stop word + punctution + number*

doc.tdm <- **TermDocumentMatrix**(doc.corpus, control)

**return**(doc.tdm)*# returning the TMD text* }

page2image58985856

## Define classifiers and use normal mail tests that are not easily recognizable

In order to calculate the probability that an email is spam or normal, you need to find the word items that are common to the message to be sorted and the training set. Let behind using these features probability calculation of this email is training focus corresponding category of conditional probability, when need the word does not appear in the classification of mail in the training set, we can give a fixed value, or by some to a probability distribution, or use natural language processing technology to estimate a word "rubbish" in the specific context. I'm going to fix it at 1e-6.

*# Naive Bayes Classifier*

classify.email <- **function**(path, train.df, prior = 0.8, q = 1e-6) {  
*# Here, we use many of the support functions to get the  
# email text data in a workable format*msg <- **get.msg**(path) *# Read the contents of the message you want to*

*verify*msg.tdm <- **get.tdm**(msg) *# tdm*msg.freq <- **rowSums**(**as.matrix**(msg.tdm))  
*# Counting the number of occurrences of each word  
# Find intersections of words*msg.intersect <- **intersect**(**names**(msg.freq), train.df**$**term) *# get the*

*intersection*

**if**(**length**(msg.intersect) **<** 1){  
**return**(prior **\*** q **^** (**length**(msg.freq))) *# A fixed probability is given*

*if # does not appear*

} **else**{

match.prob <- train.df**$**occurrence[**match**(msg.intersect,

train.df**$**term)]  
**return**(prior **\* prod**(match.prob) **\*** q **^** (**length**(msg.freq)**-**

**length**(msg.intersect))) }

}

*# Finally, we can get the probability of testing e-mail # Training :-*

*# SPAM DATA SET*

*# Get all the SPAM Emails into a single character vector*

spam.docs <- **dir**(spam\_train.path)  
spam.docs <- spam.docs[**which**(spam.docs **!=** "cmds")] all.spam <- **sapply**(spam.docs, **function**(x) **get.msg**(**file.path**(spam\_train.path,x)))  
*#every line of 'all.spam' is the context of email text*

page3image58998528

## build a set of spam training data with TDM

*# DocumentTermMatrix from that vector*spam.tdm <- **get.tdm**(all.spam) *# training Corpus* spam.matrix <- **as.matrix**(spam.tdm)  
spam.counts <- **rowSums**(spam.matrix)

*# Data frame*

spam.df <- **data.frame**(**cbind**(**names**(spam.counts), **as.numeric**(spam.counts)), stringsAsFactors =

FALSE)  
**names**(spam.df) <- **c**("term", "frequency") spam.df**$**frequency <- **as.numeric**(spam.df**$**frequency) spam.occurrence <- **sapply**(1**:nrow**(spam.matrix),**function**(i){

**length**(**which**(spam.matrix[i, ] **>** 0)) **/**

**ncol**(spam.matrix)

})  
spam.density <- spam.df**$**frequency **/ sum**(spam.df**$**frequency)

spam.df <- **transform**(spam.df, density = spam.density, occurrence = spam.occurrence)

*# HAM DATA SET*

*# Get all the HAM Emails into a single character vector*

easy\_ham.docs <- **dir**(easy\_ham\_train.path)

easy\_ham.docs <- easy\_ham.docs[**which**(easy\_ham.docs **!=** "cmds")] all.easy\_ham <- **sapply**(easy\_ham.docs, **function**(x) **get.msg**(**file.path**(easy\_ham\_train.path,x)))

*# DocumentTermMatrix from that vector*

easy\_ham.tdm <- **get.tdm**(all.easy\_ham) easy\_ham.matrix <- **as.matrix**(easy\_ham.tdm) easy\_ham.counts <- **rowSums**(easy\_ham.matrix)

*# Data frame*

easy\_ham.df <- **data.frame**(**cbind**(**names**(easy\_ham.counts), **as.numeric**(easy\_ham.counts)), stringsAsFactors = FALSE) **names**(easy\_ham.df) <- **c**("term", "frequency") easy\_ham.df**$**frequency <- **as.numeric**(easy\_ham.df**$**frequency) easy\_ham.occurrence <- **sapply**(1**:nrow**(easy\_ham.matrix), **function**(i){

**length**(**which**(easy\_ham.matrix[i, ] **>** 0)) **/**

**ncol**(easy\_ham.matrix)

})  
easy\_ham.density <- easy\_ham.df**$**frequency **/ sum**(easy\_ham.df**$**frequency)

easy\_ham.df <- **transform**(easy\_ham.df, density = easy\_ham.density, occurrence = easy\_ham.occurrence)

*# HARD HAM*

hard\_ham.docs <- **dir**(hard\_ham\_train.path)  
hard\_ham.docs <- hard\_ham.docs[**which**(hard\_ham.docs **!=** "cmds")]

hard\_ham.spamtest <- **sapply**(hard\_ham.docs, **function**(x) **classify.email**(**file.path**(hard\_ham\_train.path, x), train.df = spam.df)) hard\_ham.hamtest <- **sapply**(hard\_ham.docs, **function**(x) **classify.email**(**file.path**(hard\_ham\_train.path, x), train.df = easy\_ham.df))

hard\_ham.res <- **ifelse**(hard\_ham.spamtest **>** hard\_ham.hamtest, TRUE, FALSE) *# Check***summary**(hard\_ham.res)

## Mode FALSE TRUE

## logical 238 11

*# Test DATA*spam.classifier <- **function**(path) *#path*指定所有文件路径，进行分类*func* {

pr.spam <- **classify.email**(path, spam.df)  
pr.ham <- **classify.email**(path, easy\_ham.df) **return**(**c**(pr.spam, pr.ham, **ifelse**(pr.spam **>** pr.ham, 1, 0)))

}

*# Get lists of all the email messages*

easy\_ham2.docs <- **dir**(easy\_ham\_test.path)  
easy\_ham2.docs <- easy\_ham2.docs[**which**(easy\_ham2.docs **!=** "cmds")]

page4image59181504page4image59180160

hard\_ham2.docs <- **dir**(hard\_ham\_test.path)  
hard\_ham2.docs <- hard\_ham2.docs[**which**(hard\_ham2.docs **!=** "cmds")]

spam2.docs <- **dir**(spam\_test.path)  
spam2.docs <- spam2.docs[**which**(spam2.docs **!=** "cmds")]

*# Classify them all!  
#*易于辨别的正常邮件  
easy\_ham2.class <- **suppressWarnings**(**lapply**(easy\_ham2.docs,

**function**(x) {

**spam.classifier**(**file.path**(easy\_ham\_test.path, x)) *#*不易于辨别的正常邮件

}))  
hard\_ham2.class <- **suppressWarnings**(**lapply**(hard\_ham2.docs,

**function**(x) {

**spam.classifier**(**file.path**(hard\_ham\_test.path, x)) }))

*#*垃圾邮件  
spam2.class <- **suppressWarnings**(**lapply**(spam2.docs,

x))

**function**(x) {

**spam.classifier**(**file.path**(spam\_test.path,

}))

*# Create a single, final, data frame with all of the classification data in it*数据整合  
easy\_ham2.matrix <- **do.call**(rbind, easy\_ham2.class)  
easy\_ham2.final <- **cbind**(easy\_ham2.matrix, "EASYHAM")

hard\_ham2.matrix <- **do.call**(rbind, hard\_ham2.class) hard\_ham2.final <- **cbind**(hard\_ham2.matrix, "HARDHAM")

spam2.matrix <- **do.call**(rbind, spam2.class) spam2.final <- **cbind**(spam2.matrix, "SPAM")

class.matrix <- **rbind**(easy\_ham2.final, hard\_ham2.final, spam2.final) class.df <- **data.frame**(class.matrix, stringsAsFactors = FALSE) **names**(class.df) <- **c**("Pr.SPAM" ,"Pr.HAM", "Class", "Type") class.df**$**Pr.SPAM <- **as.numeric**(class.df**$**Pr.SPAM)

class.df**$**Pr.HAM <- **as.numeric**(class.df**$**Pr.HAM) class.df**$**Class <- **as.logical**(**as.numeric**(class.df**$**Class)) class.df**$**Type <- **as.factor**(class.df**$**Type)

*# Create final plot of results*

class.plot <- **ggplot**(class.df, **aes**(x = **log**(Pr.HAM), **log**(Pr.SPAM))) **+ geom\_point**(**aes**(shape = Type, alpha = 0.5)) **+**

**scale\_shape\_manual**(values = **c**( "EASYHAM" = 1, "HARDHAM" = 2, "SPAM" = 3), name = "Email Type") **+**

**scale\_alpha**(guide = "none") **+  
xlab**("log[P(HAM)]") **+  
ylab**("log[P(SPAM)]") **+***#theme\_bw() +***theme**(axis.text.x = **element\_blank**(), axis.text.y =

**element\_blank**())  
**ggsave**(plot = class.plot, filename = **file.path**("Email-Spam-

Classifier/images", "result.png"), height = 10, width = 10)

get.results <- **function**(bool.vector) {

results <- **c**(**length**(bool.vector[**which**(bool.vector **==** FALSE)]) **/ length**(bool.vector),

**length**(bool.vector[**which**(bool.vector **==** TRUE)]) **/ length**(bool.vector))

**return**(results) }

*# Save results as a 2x3 table*

EasyHam <- **get.results**(**subset**(class.df, Type **==** "EASYHAM")**$**Class) HardHam <- **get.results**(**subset**(class.df, Type **==** "HARDHAM")**$**Class) Spam <- **get.results**(**subset**(class.df, Type **==** "SPAM")**$**Class)

class.res <- **rbind**(EasyHam, HardHam, Spam) **colnames**(class.res) <- **c**("NOT SPAM", "SPAM") **print**(class.res)

## NOT SPAM SPAM

## EasyHam 0.9871429 0.01285714

## HardHam 0.9556452 0.04435484

## Spam 0.2870437 0.71295634

**write.csv**(spam.df, **file.path**("Email-Spam-Classifier/data", "spam\_df.csv"), row.names = FALSE)  
**write.csv**(easy\_ham.df, **file.path**("Email-Spam-Classifier/data", "easyham\_df.csv"), row.names = FALSE)

**write.csv**(class.df, **file.path**("Email-Spam-Classifier/Results", "result\_df.csv"), row.names = FALSE)  
**write.table**(class.res, **file.path**("Email-Spam-Classifier/Results", "classify.csv"), row.names=FALSE)

*## Monte*

SIR.sim <- function(ax1, bnx1, ax2, bnx2, size=10000)  
{theta1 <- runif(size, min=0,max=1)  
epsilon <- runif(size, min=theta1-1, max=theta1)  
weight <- theta1^(ax1 - 1)\*(1 - theta1)^(bnx1 - 1)\* ( theta1 - epsilon)^(ax2 - 1)\*(1 - (theta1 - epsilon))^(bnx2 - 1)  
epsilon.post <- sample(epsilon, size, replace=T, prob=weight)  
layout(matrix(c(1,2), byrow=T,nrow=2))  
hist(epsilon.post)  
plot(density(epsilon.post))  
}

# Bibliography

Gelman, A. S. (2013). *Bayesian data analysis.* Chapman and Hall/CRC.

Rish, I. (2001). *An empirical study of the naive Bayes classifier.* IJCAI 2001 workshop on empirical methods in artificial intelligence.

Ng, A. Y. (2002). In Advances in neural information processing systems. In On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. (pp. 841-848).