SpamHamProject

Shon Mohsin, Maryam Shahini, Ramin Farhanian

June 15, 2019

# Introduction

Spam email is a constant problem that floods our email inboxes daily. When spam is sent to a user’s inbox, usually there are filters in place tha analyze the message and classify it as spam or not spam (we refer to as ham). Ham messages are included in the user’s inbox and spam is labelled and quarentined. In this paper we create a filter that can classify between spam and ham emails. In addition, we answer question 20 from “Data Science in R” from Nolan et al.

# Background

Our goal is to answer Question 20 and it’s 4 parts from our book.

Question 20:

In Section 3.6.3 we used the test set that we had put aside to both select τ, the threshold for the log odds, and to evaluate the Type I and II errors incurred when we use this threshold. Ideally, we choose τ from another set of messages that is both independent of our training data and our test data. The method of cross-validation is designed to use the training set for training and validating the model. Implement 5-fold cross-validation to choose τ and assess the error rate with our training data. To do this, follow the steps:

1. Use the sample() function to permute the indices of the training set, and organize these permuted indices into 5 equal-size sets, called folds.
2. Foreachfold,takethecorrespondingsubsetfromthetrainingdatatouseasa‘test’ set. Use the remaining messages in the training data as the training set. Apply the functions developed in Section 3.6 to estimate the probabilities that a word occurs in a message given it is spam or ham, and use these probabilities to compute the log likelihood ratio for the messages in the training set.
3. Pool all of the LLR values from the messages in all of the folds, i.e., from all of the training data, and use these values and the typeIErrorRate() function to select a threshold that achieves a 1% Type I error.
4. Apply this threshold to our original/real test set and ﬁnd its Type I and Type II errors.

Our dataset consists of over 9000 email messages that have been classified by SpamAssasin (http“//spamassasin.apache.org”) which we will use to develop our filter. Once the initial exploration of the data is complete, we will then set forth in solving question 20 in our book.

# Exploratory Data Analysis

Our first step is to load the data into R for analysis. We can view the directories where the messages are kept, classified into their respective folder.

spamPath = "data"  
list.dirs(spamPath, full.names = FALSE)

## [1] "" "messages" "messages/easy\_ham"   
## [4] "messages/easy\_ham\_2" "messages/hard\_ham" "messages/spam"   
## [7] "messages/spam\_2"

We take a look at the contents of the easy\_ham and spam\_2 folder and see that the the directories look the same. According to the SpamAssasin website, the messages are named by a message number and their MD5 checksum, which is a unique identifier derived from the contents of the file.

head(list.files(path = paste(spamPath, "messages", "easy\_ham", sep = .Platform$file.sep)))

## [1] "00001.7c53336b37003a9286aba55d2945844c"  
## [2] "00002.9c4069e25e1ef370c078db7ee85ff9ac"  
## [3] "00003.860e3c3cee1b42ead714c5c874fe25f7"  
## [4] "00004.864220c5b6930b209cc287c361c99af1"  
## [5] "00005.bf27cdeaf0b8c4647ecd61b1d09da613"  
## [6] "00006.253ea2f9a9cc36fa0b1129b04b806608"

head(list.files(path = paste(spamPath, "messages", "spam\_2", sep = .Platform$file.sep)))

## [1] "00001.317e78fa8ee2f54cd4890fdc09ba8176"  
## [2] "00002.9438920e9a55591b18e60d1ed37d992b"  
## [3] "00003.590eff932f8704d8b0fcbe69d023b54d"  
## [4] "00004.bdcc075fa4beb5157b5dd6cd41d8887b"  
## [5] "00005.ed0aba4d386c5e62bc737cf3f0ed9589"  
## [6] "00006.3ca1f399ccda5d897fecb8c57669a283"

There are 9353 files in the combined directories. Looking inside each directory, we observe that the files are not equally divided.

dirNames = list.files(path = paste(spamPath, "messages", sep = .Platform$file.sep))  
length(list.files(paste(spamPath, "messages", dirNames, sep = .Platform$file.sep)))

## [1] 9353

sapply(paste(spamPath, "messages", dirNames, sep = .Platform$file.sep), function(dir) length(list.files(dir)) )

## data/messages/easy\_ham data/messages/easy\_ham\_2 data/messages/hard\_ham   
## 5052 1401 501   
## data/messages/spam data/messages/spam\_2   
## 1001 1398

Now that we have verified file counts, we proceed with the task of importing the actual messages into R in a format appropriate for analysis.

fullDirNames = paste(spamPath, "messages", dirNames, sep = .Platform$file.sep)  
fileNames=list.files(fullDirNames[1], full.names=TRUE)  
msg = readLines(fileNames[1])  
head(msg, 10)

## [1] "From exmh-workers-admin@redhat.com Thu Aug 22 12:36:23 2002"   
## [2] "Return-Path: <exmh-workers-admin@spamassassin.taint.org>"   
## [3] "Delivered-To: zzzz@localhost.netnoteinc.com"   
## [4] "Received: from localhost (localhost [127.0.0.1])"   
## [5] "\tby phobos.labs.netnoteinc.com (Postfix) with ESMTP id D03E543C36"   
## [6] "\tfor <zzzz@localhost>; Thu, 22 Aug 2002 07:36:16 -0400 (EDT)"   
## [7] "Received: from phobos [127.0.0.1]"   
## [8] "\tby localhost with IMAP (fetchmail-5.9.0)"   
## [9] "\tfor zzzz@localhost (single-drop); Thu, 22 Aug 2002 12:36:16 +0100 (IST)"   
## [10] "Received: from listman.spamassassin.taint.org (listman.spamassassin.taint.org [66.187.233.211]) by"

Next we set aside a selection of emails to use as test cases as we build our methodology.

indx = c(1:5, 15, 27, 68, 69, 329, 404, 427, 516, 852, 971)   
fn = list.files(fullDirNames[1], full.names = TRUE)[indx]   
sampleEmail = sapply(fn, readLines)

We must access the body of the message in order to extract the words contained in the message. We must also split the message into it’s header and it’s body. To do this, we use the splitPoint method.

msg = sampleEmail[[1]]  
which(msg == "")

## [1] 63 67 68 70 72 74 80 82 85 88 91 93 96 97 101 103 106  
## [18] 107 108 113

splitPoint = match("", msg)  
msg[ (splitPoint-2):(splitPoint+6)]

## [1] "List-Archive: <https://listman.spamassassin.taint.org/mailman/private/exmh-workers/>"  
## [2] "Date: Thu, 22 Aug 2002 18:26:25 +0700"   
## [3] ""   
## [4] " Date: Wed, 21 Aug 2002 10:54:46 -0500"   
## [5] " From: Chris Garrigues <cwg-dated-1030377287.06fa6d@DeepEddy.Com>"   
## [6] " Message-ID: <1029945287.4797.TMDA@deepeddy.vircio.com>"   
## [7] ""   
## [8] ""   
## [9] " | I can't reproduce this error."

We can now create a function splitMessage() function that will output 2 vectors, composing of the header and body of the message. We apply the function to our sampleEmail

splitMessage = function(msg) {  
 splitPoint = match("", msg)  
 header = msg[1:(splitPoint-1)]  
 body = msg[-(1:splitPoint)]  
 return(list(header = header, body = body))  
}  
  
sampleSplit = lapply(sampleEmail, splitMessage)

We next need to remove the attachments from the email if there are any. We can get the content type in the header. We can then use the content type to determine if there is an attachment in the file.

header = sampleSplit[[1]]$header  
grep("Content-Type", header)

## [1] 46

grep("multi", header)

## integer(0)

grep("multi", tolower(header[46]))

## integer(0)

We can apply this grep() call to all the headers in the list of sample messages. Our expectation is a vector.

headerList = lapply(sampleSplit, function(msg) msg$header)  
CTloc = sapply(headerList, grep, patter = "Content-Type")  
CTloc

## $`data/messages/easy\_ham/00001.7c53336b37003a9286aba55d2945844c`  
## [1] 46  
##   
## $`data/messages/easy\_ham/00002.9c4069e25e1ef370c078db7ee85ff9ac`  
## [1] 45  
##   
## $`data/messages/easy\_ham/00003.860e3c3cee1b42ead714c5c874fe25f7`  
## [1] 42  
##   
## $`data/messages/easy\_ham/00004.864220c5b6930b209cc287c361c99af1`  
## [1] 30  
##   
## $`data/messages/easy\_ham/00005.bf27cdeaf0b8c4647ecd61b1d09da613`  
## [1] 44  
##   
## $`data/messages/easy\_ham/00014.cb20e10b2bfcb8210a1c310798532a57`  
## [1] 54  
##   
## $`data/messages/easy\_ham/00025.d685245bdc4444f44fa091e6620b20b3`  
## integer(0)  
##   
## $`data/messages/easy\_ham/00062.009f5a1a8fa88f0b38299ad01562bb37`  
## [1] 21  
##   
## $`data/messages/easy\_ham/00063.0acbc484a73f0e0b727e06c100d8df7b`  
## [1] 17  
##   
## $`data/messages/easy\_ham/0030.77828e31de08ebb58b583688b87524cc`  
## [1] 52  
##   
## $`data/messages/easy\_ham/00368.f86324a03e7ae7070cc40f302385f5d3`  
## [1] 31  
##   
## $`data/messages/easy\_ham/00389.8606961eaeef7b921ce1c53773248d69`  
## [1] 52  
##   
## $`data/messages/easy\_ham/0047.5c3e049737a2813d4ac6f13f02362fb1`  
## [1] 52  
##   
## $`data/messages/easy\_ham/00775.0e012f373467846510d9db297e99a008`  
## [1] 27  
##   
## $`data/messages/easy\_ham/00883.c44a035e7589e83076b7f1fed8fa97d5`  
## [1] 31

We observe that the 7th element returned in integer of zero. To account for this, we can check for missing Content-Type field and return zero or NA so that we can work with a numeric vector.

sapply(headerList, function(header){  
 CTloc = grep("Content-Type", header)  
 if (length(CTloc) == 0)   
 return (NA)  
 CTloc  
})

## data/messages/easy\_ham/00001.7c53336b37003a9286aba55d2945844c   
## 46   
## data/messages/easy\_ham/00002.9c4069e25e1ef370c078db7ee85ff9ac   
## 45   
## data/messages/easy\_ham/00003.860e3c3cee1b42ead714c5c874fe25f7   
## 42   
## data/messages/easy\_ham/00004.864220c5b6930b209cc287c361c99af1   
## 30   
## data/messages/easy\_ham/00005.bf27cdeaf0b8c4647ecd61b1d09da613   
## 44   
## data/messages/easy\_ham/00014.cb20e10b2bfcb8210a1c310798532a57   
## 54   
## data/messages/easy\_ham/00025.d685245bdc4444f44fa091e6620b20b3   
## NA   
## data/messages/easy\_ham/00062.009f5a1a8fa88f0b38299ad01562bb37   
## 21   
## data/messages/easy\_ham/00063.0acbc484a73f0e0b727e06c100d8df7b   
## 17   
## data/messages/easy\_ham/0030.77828e31de08ebb58b583688b87524cc   
## 52   
## data/messages/easy\_ham/00368.f86324a03e7ae7070cc40f302385f5d3   
## 31   
## data/messages/easy\_ham/00389.8606961eaeef7b921ce1c53773248d69   
## 52   
## data/messages/easy\_ham/0047.5c3e049737a2813d4ac6f13f02362fb1   
## 52   
## data/messages/easy\_ham/00775.0e012f373467846510d9db297e99a008   
## 27   
## data/messages/easy\_ham/00883.c44a035e7589e83076b7f1fed8fa97d5   
## 31

Now we check for attachments in our samples.

hasAttach = sapply(headerList, function(header)  
 {  
 CTloc = grep("Content-Type", header)  
 if (length(CTloc)== 0) return(FALSE)  
 grepl("multi", tolower(header[CTloc]))  
 })  
hasAttach

## data/messages/easy\_ham/00001.7c53336b37003a9286aba55d2945844c   
## FALSE   
## data/messages/easy\_ham/00002.9c4069e25e1ef370c078db7ee85ff9ac   
## FALSE   
## data/messages/easy\_ham/00003.860e3c3cee1b42ead714c5c874fe25f7   
## FALSE   
## data/messages/easy\_ham/00004.864220c5b6930b209cc287c361c99af1   
## FALSE   
## data/messages/easy\_ham/00005.bf27cdeaf0b8c4647ecd61b1d09da613   
## FALSE   
## data/messages/easy\_ham/00014.cb20e10b2bfcb8210a1c310798532a57   
## TRUE   
## data/messages/easy\_ham/00025.d685245bdc4444f44fa091e6620b20b3   
## FALSE   
## data/messages/easy\_ham/00062.009f5a1a8fa88f0b38299ad01562bb37   
## TRUE   
## data/messages/easy\_ham/00063.0acbc484a73f0e0b727e06c100d8df7b   
## TRUE   
## data/messages/easy\_ham/0030.77828e31de08ebb58b583688b87524cc   
## TRUE   
## data/messages/easy\_ham/00368.f86324a03e7ae7070cc40f302385f5d3   
## TRUE   
## data/messages/easy\_ham/00389.8606961eaeef7b921ce1c53773248d69   
## TRUE   
## data/messages/easy\_ham/0047.5c3e049737a2813d4ac6f13f02362fb1   
## TRUE   
## data/messages/easy\_ham/00775.0e012f373467846510d9db297e99a008   
## TRUE   
## data/messages/easy\_ham/00883.c44a035e7589e83076b7f1fed8fa97d5   
## TRUE

Now that we have found the attachments, we need to extract the string from the messages that have attachments by using the gsub() function included in a new function getBoundary.

We create a dropAttach() function to drop email attachments.

We create the function findMsgWords to find the words and cleanText to lower the capital letters in our messages.

```

We can combine all of the tasks of cleaning and preparing the messages into a few functions proceed with the data cleaning.

We apply the functions to all out directories and create a logical vector based on the number of elements in each list and then combine the 5 lists into 1 list.

# Method

# Results

# Discussion

# Future Work

# References