Identifying Spam Part I

1 data

9,000 emails classified by SpamAssasin ( [http://spamassasin.apache.org](http://spamassasin.apache.org/) )

1.1 Structure of Email

1.1.1 header

Include date, sender, subject, message ID, cc recipient, routing information in key: value format

(예) Content-Type: TEXT/PLAIN; charset=US-ASCII

1.1.2 body

header and body separated by a blank line

Attachments are also part of the body, and the boundaries are \_===669732====calmail-me.berkeley.edu===\_separated by, for example ,

2 Data processing 1

2.1 Retrieving email messages

Unpack the data <http://www.rdatasciencecases.org/Spam/SpamAssassinMessages.zip> into a suitable directory (eg D: / Dropbox / R\_Exercises / data science case studies /) and run the following: Show file names (email)

spamPath = "D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages"

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

## [1] "" "easy\_ham" "easy\_ham\_2" "hard\_ham" "spam"

## [6] "spam\_2"

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

## [1] "00001.317e78fa8ee2f54cd4890fdc09ba8176"

## [2] "00002.9438920e9a55591b18e60d1ed37d992b"

## [3] "00003.590eff932f8704d8b0fcbe69d023b54d"

## [4] "00004.bdcc075fa4beb5157b5dd6cd41d8887b"

## [5] "00005.ed0aba4d386c5e62bc737cf3f0ed9589"

## [6] "00006.3ca1f399ccda5d897fecb8c57669a283"

Outputs total and each email file in 5 directories and stores full path and file name in fileNames

dirNames = list.files(path = spamPath)

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

## [1] 9353

sapply(paste(spamPath, dirNames, sep = .Platform$file.sep),

**function**(dir) length(list.files(dir)) )

## D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham

## 5052

## D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham\_2

## 1401

## D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/hard\_ham

## 501

## D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/spam

## 1001

## D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/spam\_2

## 1398

fullDirNames = paste(spamPath, dirNames, sep = .Platform$file.sep)

fileNames = list.files(fullDirNames[1], full.names = TRUE)

fileNames[1]

## [1] "D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham/00001.7c53336b37003a9286aba55d2945844c"

prints the header of the first email of easy\_ham and reads the email files specified in indx into sampleEmail

msg = readLines(fileNames[1])

head(msg)

## [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)"

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)

2.2 Finding words in a message

2.2.1 Separating Messages into Headers and Bodies

The boundary between header and body is found by finding the first blank line.

msg = sampleEmail[[1]]

match("", msg)

## [1] 63

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."

header = msg[1:(splitPoint-1)]

body = msg[ -(1:splitPoint) ]

Functioning and Applying to Sample Email

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)

2.2.2 Removing attachments from the body

Boundaries if you have attachments:

‘Content-Type: MULTIPART/Mixed;  
BOUNDARY=“*===669732====calmail-me.berkeley.edu===*”’

header = sampleSplit[[1]]$header

grep("Content-Type", header)

## [1] 46

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

## integer(0)

header[46]

## [1] "Content-Type: text/plain; charset=us-ascii"

No attachments for first mail

It also checks if there is a Content-Type key in a function that checks for the presence of an attachment.

headerList = lapply(sampleSplit, **function**(msg) msg$header)

hasAttach = sapply(headerList, **function**(header) {

CTloc = grep("Content-Type", header)

**if** (length(CTloc) == 0) **return**(FALSE)

grepl("multi", tolower(header[CTloc]))

})

hasAttach

D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham/00001.7c53336b37003a9286aba55d2945844c

FALSE

D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham/00002.9c4069e25e1ef370c078db7ee85ff9ac

FALSE

D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham/00003.860e3c3cee1b42ead714c5c874fe25f7

FALSE

D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham/00004.864220c5b6930b209cc287c361c99af1

FALSE

D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham/00005.bf27cdeaf0b8c4647ecd61b1d09da613

FALSE

D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham/00014.cb20e10b2bfcb8210a1c310798532a57

TRUE

D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham/00025.d685245bdc4444f44fa091e6620b20b3

FALSE

D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham/00062.009f5a1a8fa88f0b38299ad01562bb37

TRUE

D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham/00063.0acbc484a73f0e0b727e06c100d8df7b

TRUE

D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham/0030.77828e31de08ebb58b583688b87524cc

TRUE

D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham/00368.f86324a03e7ae7070cc40f302385f5d3

TRUE

D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham/00389.8606961eaeef7b921ce1c53773248d69

TRUE

D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham/0047.5c3e049737a2813d4ac6f13f02362fb1

TRUE

D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham/00775.0e012f373467846510d9db297e99a008

TRUE

D:/Dropbox/R\_Exercises/data science case studies/SpamAssassinMessages/easy\_ham/00883.c44a035e7589e83076b7f1fed8fa97d5

TRUE

Find the bounding string in the 6th message

header = sampleSplit[[6]]$header

boundaryIdx = grep("boundary=", header)

header[boundaryIdx]

## [1] " boundary=\"==\_Exmh\_-1317289252P\";"

Example of function for finding boundary string and application for sixth message

getBoundary = **function**(header) {

boundaryIdx = grep("boundary=", header)

boundary = gsub('"', "", header[boundaryIdx])

gsub(".\*boundary= \*([^;]\*);?.\*", "\\1", boundary)

}

sampleSplit[[6]]$body

## [1] "--==\_Exmh\_-1317289252P"

## [2] "Content-Type: text/plain; charset=us-ascii"

## [3] ""

## [4] "> From: Chris Garrigues <cwg-exmh@DeepEddy.Com>"

## [5] "> Date: Wed, 21 Aug 2002 10:40:39 -0500"

## [6] ">"

## [7] "> > From: Chris Garrigues <cwg-exmh@DeepEddy.Com>"

## [8] "> > Date: Wed, 21 Aug 2002 10:17:45 -0500"

## [9] "> >"

## [10] "> > Ouch...I'll get right on it."

## [11] "> > "

## [12] "> > > From: Robert Elz <kre@munnari.OZ.AU>"

## [13] "> > > Date: Wed, 21 Aug 2002 19:30:01 +0700"

## [14] "> > >"

## [15] "> > > Any chance of having that lengthen instead? I like all my exmh stuff"

## [16] "> > > in nice columns (fits the display better). That is, I use the detache"

## [17] "> d"

## [18] "> > > folder list, one column. The main exmh window takes up full screen,"

## [19] "> > > top to bottom, but less than half the width, etc..."

## [20] "> "

## [21] "> I thought about that. The first order approximation would be to just add "

## [22] "> using pack .... -side top instead of pack ... -side left, however, since their "

## [23] "> each a different width, it would look funny."

## [24] ""

## [25] "I've done this. It's not as pretty as I think it should be, but it works. "

## [26] "I'm going to leave the cosmetic issues to others. When I update the "

## [27] "documentation, I'll add this to the exmh.TODO file."

## [28] ""

## [29] "I'm leaving for a 2 1/2 week vacation in a week, so this is the last new "

## [30] "functionality I'm going to add for a while. Also, I now have pretty much "

## [31] "everything in there that I want for my own use, so I'm probably pretty much "

## [32] "done. I'll work on bug fixes and documentation before my vacation, and "

## [33] "hopefully do nothing more afterwards."

## [34] ""

## [35] "Chris"

## [36] ""

## [37] "-- "

## [38] "Chris Garrigues http://www.DeepEddy.Com/~cwg/"

## [39] "virCIO http://www.virCIO.Com"

## [40] "716 Congress, Suite 200"

## [41] "Austin, TX 78701\t\t+1 512 374 0500"

## [42] ""

## [43] " World War III: The Wrong-Doers Vs. the Evil-Doers."

## [44] ""

## [45] ""

## [46] ""

## [47] ""

## [48] "--==\_Exmh\_-1317289252P"

## [49] "Content-Type: application/pgp-signature"

## [50] ""

## [51] "-----BEGIN PGP SIGNATURE-----"

## [52] "Version: GnuPG v1.0.6 (GNU/Linux)"

## [53] "Comment: Exmh version 2.2\_20000822 06/23/2000"

## [54] ""

## [55] "iD8DBQE9ZQJ/K9b4h5R0IUIRAiPuAJwL4mUus5whLNQZC8MsDlGpEdKNrACcDfZH"

## [56] "PcGgN9frLIM+C5Z3vagi2wE="

## [57] "=qJoJ"

## [58] "-----END PGP SIGNATURE-----"

## [59] ""

## [60] "--==\_Exmh\_-1317289252P--"

## [61] ""

## [62] ""

## [63] ""

## [64] "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

## [65] "Exmh-workers mailing list"

## [66] "Exmh-workers@redhat.com"

## [67] "https://listman.redhat.com/mailman/listinfo/exmh-workers"

## [68] ""

Find the start and end of an email in body

boundary = getBoundary(headerList[[15]])

body = sampleSplit[[15]]$body

bString = paste("--", boundary, sep = "")

bStringLocs = which(bString == body)

bStringLocs

## [1] 2 35

eString = paste("--", boundary, "--", sep = "")

eStringLoc = which(eString == body)

eStringLoc

## [1] 77

The first part of the body, excluding attachments, and the line after the attachment (empty lines in the example)

msg = body[ (bStringLocs[1] + 1) : (bStringLocs[2] - 1)]

tail(msg)

## [1] ">" ">Yuck" "> " ">" "" ""

msg = c(msg, body[ (eStringLoc + 1) : length(body) ])

tail(msg)

## [1] "> " ">" "" "" "" ""

Functions to remove attachments from the body

dropAttach = **function**(body, boundary){

bString = paste("--", boundary, sep = "")

bStringLocs = which(bString == body)

**if** (length(bStringLocs) <= 1) **return**(body)

eString = paste("--", boundary, "--", sep = "")

eStringLoc = which(eString == body)

**if** (length(eStringLoc) == 0)

**return**(body[ (bStringLocs[1] + 1) : (bStringLocs[2] - 1)])

n = length(body)

**if** (eStringLoc < n)

**return**( body[ c( (bStringLocs[1] + 1) : (bStringLocs[2] - 1),

( (eStringLoc + 1) : n )) ] )

**return**( body[ (bStringLocs[1] + 1) : (bStringLocs[2] - 1) ])

}

2.2.3 Extracting words from the message body

body 살펴보기

head(sampleSplit[[1]]$body)

## [1] " Date: Wed, 21 Aug 2002 10:54:46 -0500"

## [2] " From: Chris Garrigues <cwg-dated-1030377287.06fa6d@DeepEddy.Com>"

## [3] " Message-ID: <1029945287.4797.TMDA@deepeddy.vircio.com>"

## [4] ""

## [5] ""

## [6] " | I can't reproduce this error."

msg = sampleSplit[[3]]$body

head(msg)

## [1] "Man Threatens Explosion In Moscow "

## [2] ""

## [3] "Thursday August 22, 2002 1:40 PM"

## [4] "MOSCOW (AP) - Security officers on Thursday seized an unidentified man who"

## [5] "said he was armed with explosives and threatened to blow up his truck in"

## [6] "front of Russia's Federal Security Services headquarters in Moscow, NTV"

Original sentences and sentences processed for word extraction (lowercase conversion, punctuation and number elimination) sentences

msg[ c(1, 3, 26, 27) ]

## [1] "Man Threatens Explosion In Moscow "

## [2] "Thursday August 22, 2002 1:40 PM"

## [3] "4 DVDs Free +s&p Join Now"

## [4] "http://us.click.yahoo.com/pt6YBB/NXiEAA/mG3HAA/7gSolB/TM"

cleanMsg = tolower(gsub("[[:punct:]0-9[:blank:]]+", " ", msg))

cleanMsg[ c(1, 3, 26, 27) ]

## [1] "man threatens explosion in moscow "

## [2] "thursday august pm"

## [3] " dvds free s p join now"

## [4] "http us click yahoo com pt ybb nxieaa mg haa gsolb tm"

**library**(tm)

## Loading required package: NLP

stopWords = stopwords()

cleanSW = tolower(gsub("[[:punct:]0-9[:blank:]]+", " ", stopWords))

SWords = unlist(strsplit(cleanSW, "[[:blank:]]+"))

SWords = SWords[ nchar(SWords) > 1 ]

stopWords = unique(SWords)

words = unlist(strsplit(cleanMsg, "[[:blank:]]+"))

words = words[ nchar(words) > 1 ]

words = words[ !( words %in% stopWords) ]

head(words)

## [1] "man" "threatens" "explosion" "moscow" "thursday" "august"

Function: cleanText removes punctuation and numbers, findMsgWords removes prefixes, etc.

cleanText =

**function**(msg) {

tolower(gsub("[[:punct:]0-9[:space:][:blank:]]+", " ", msg))

}

findMsgWords =

**function**(msg, stopWords) {

**if**(is.null(msg))

**return**(character())

words = unique(unlist(strsplit(cleanText(msg), "[[:blank:]\t]+")))

*# drop empty and 1 letter words*

words = words[ nchar(words) > 1]

words = words[ !( words %in% stopWords) ]

invisible(words)

}

2.2.4 Finishing the preparation process

Functional: processAllWords takes a directory as input and processes all emails to extract words

processAllWords = **function**(dirName, stopWords)

{

*# 주어진 디렉토리의 모든 파일명*

fileNames = list.files(dirName, full.names = TRUE)

*# 이메일이 아닌 파일들 제거*

notEmail = grep("cmds$", fileNames)

**if** ( length(notEmail) > 0) fileNames = fileNames[ - notEmail ]

messages = lapply(fileNames, readLines, encoding = "latin1")

*# header와 body로 분리*

emailSplit = lapply(messages, splitMessage)

*# body와 header 각각을 리스트화*

bodyList = lapply(emailSplit, **function**(msg) msg$body)

headerList = lapply(emailSplit, **function**(msg) msg$header)

rm(emailSplit)

*# 첨부파일이 있는 이메일*

hasAttach = sapply(headerList, **function**(header) {

CTloc = grep("Content-Type", header)

**if** (length(CTloc) == 0) **return**(0)

multi = grep("multi", tolower(header[CTloc]))

**if** (length(multi) == 0) **return**(0)

multi

})

hasAttach = which(hasAttach > 0)

*# 첨부파일이 있는 이메일의 경계 문자열 찾기*

boundaries = sapply(headerList[hasAttach], getBoundary)

*# body에서 첨부파일 제거*

bodyList[hasAttach] = mapply(dropAttach, bodyList[hasAttach],

boundaries, SIMPLIFY = FALSE)

*# body에서 단어 추출*

msgWordsList = lapply(bodyList, findMsgWords, stopWords)

invisible(msgWordsList)

}

Process email in all directories

msgWordsList = lapply(fullDirNames, processAllWords,

stopWords = stopWords)

## Warning in FUN(X[[i]], ...): 'D:/Dropbox/R\_Exercises/

## data science case studies/SpamAssassinMessages/hard\_ham/

## 00228.0eaef7857bbbf3ebf5edbbdae2b30493'에서 불완전한 마지막 행이 발견되었습

## 니다

## Warning in FUN(X[[i]], ...): 'D:/Dropbox/R\_Exercises/

## data science case studies/SpamAssassinMessages/hard\_ham/

## 0231.7c6cc716ce3f3bfad7130dd3c8d7b072'에서 불완전한 마지막 행이 발견되었습

## 니다

## Warning in FUN(X[[i]], ...): 'D:/Dropbox/R\_Exercises/

## data science case studies/SpamAssassinMessages/hard\_ham/

## 0250.7c6cc716ce3f3bfad7130dd3c8d7b072'에서 불완전한 마지막 행이 발견되었습

## 니다

## Warning in FUN(X[[i]], ...): 'D:/Dropbox/R\_Exercises/data science case

## studies/SpamAssassinMessages/spam/00136.faa39d8e816c70f23b4bb8758d8a74f0'에

## 서 불완전한 마지막 행이 발견되었습니다

## Warning in FUN(X[[i]], ...): 'D:/Dropbox/R\_Exercises/data science case

## studies/SpamAssassinMessages/spam/0143.260a940290dcb61f9327b224a368d4af'에

## 서 불완전한 마지막 행이 발견되었습니다

numMsgs = sapply(msgWordsList, length)

numMsgs

## [1] 5051 1400 500 1000 1397

Since the three previous directories are ham and the other two are spam,

isSpam = rep(c(FALSE, FALSE, FALSE, TRUE, TRUE), numMsgs)

msgWordsList = unlist(msgWordsList, recursive = FALSE)

3 Simple Bayes classification

3.1 Test and Training Data

After calculating the total number of spam and ham, the index of the test data is stratified

numEmail = length(isSpam)

numSpam = sum(isSpam)

numHam = numEmail - numSpam

set.seed(418910)

testSpamIdx = sample(numSpam, size = floor(numSpam/3))

testHamIdx = sample(numHam, size = floor(numHam/3))

Split into test and training data and generate variable for spam for each data

testMsgWords = c((msgWordsList[isSpam])[testSpamIdx],

(msgWordsList[!isSpam])[testHamIdx] )

trainMsgWords = c((msgWordsList[isSpam])[ - testSpamIdx],

(msgWordsList[!isSpam])[ - testHamIdx])

testIsSpam = rep(c(TRUE, FALSE),

c(length(testSpamIdx), length(testHamIdx)))

trainIsSpam = rep(c(TRUE, FALSE),

c(numSpam - length(testSpamIdx),

numHam - length(testHamIdx)))

3.2 Probability Estimation Using Training Data

Output number of unique words in training data

bow = unique(unlist(trainMsgWords))

length(bow)

## [1] 79424

Calculate the number of times spam messages are spamWordCounts for each word in bow

spamWordCounts = rep(0, length(bow))

names(spamWordCounts) = bow

tmp = lapply(trainMsgWords[trainIsSpam], unique)

tt = table( unlist(tmp) )

spamWordCounts[ names(tt) ] = tt

Functioning: computeFreqs calculates the number of times spam and ham occur for each word in the list

computeFreqs =

**function**(wordsList, spam, bow = unique(unlist(wordsList)))

{

*# spam, ham, log odd로 이루어진 행렬 생성*

wordTable = matrix(0.5, nrow = 4, ncol = length(bow),

dimnames = list(c("spam", "ham",

"presentLogOdds",

"absentLogOdds"), bow))

*# 각 spam 메시지에 대하여 메시지에서 나오는 단어들의 갯수를 1씩 증가*

counts.spam = table(unlist(lapply(wordsList[spam], unique)))

wordTable["spam", names(counts.spam)] = counts.spam + 1

*# 각 ham 메시지에 대하여 메시지에서 나오는 단어들의 갯수를 1씩 증가*

counts.ham = table(unlist(lapply(wordsList[!spam], unique)))

wordTable["ham", names(counts.ham)] = counts.ham + 1

*# spam과 ham의 갯수*

numSpam = sum(spam)

numHam = length(spam) - numSpam

*# Pr(단어|spam)과 Pr(단어|ham)의 추정값*

wordTable["spam", ] = wordTable["spam", ]/(numSpam + .5)

wordTable["ham", ] = wordTable["ham", ]/(numHam + .5)

*# log odd*

wordTable["presentLogOdds", ] =

log(wordTable["spam",]) - log(wordTable["ham", ])

wordTable["absentLogOdds", ] =

log((1 - wordTable["spam", ])) - log((1 -wordTable["ham", ]))

invisible(wordTable)

}

Apply computeFreqs function to training data

trainTable = computeFreqs(trainMsgWords, trainIsSpam)

trainTable[,1:10]

## doctype html public dtd

## spam 0.055051611 0.5079762 0.08883328 0.053800438

## ham 0.004962779 0.2062790 0.05610098 0.006041644

## presentLogOdds 2.406305227 0.9012051 0.45960806 2.186605415

## absentLogOdds -0.051649833 -0.4782050 -0.03529330 -0.049241810

## transitional en head meta content

## spam 0.047544573 0.10697529 0.29152330 0.17203628 0.3615890

## ham 0.005610098 0.02459812 0.06257417 0.01704607 0.0951559

## presentLogOdds 2.137099431 1.46992774 1.53876734 2.31178587 1.3349916

## absentLogOdds -0.043086075 -0.08823532 -0.28002047 -0.17159292 -0.3487804

## text

## spam 0.4097591

## ham 0.1251483

## presentLogOdds 1.1860698

## absentLogOdds -0.3935237

3.3 Classification of test data

Words may not appear in training data. Log Odds Calculations Excluding New Words. Actual first message is spam and log likelihood ratio is large as positive

newMsg = testMsgWords[[1]]

newMsg = newMsg[!is.na(match(newMsg, colnames(trainTable)))]

present = colnames(trainTable) %in% newMsg

sum(trainTable["presentLogOdds", present]) +

sum(trainTable["absentLogOdds", !present])

## [1] 251.3853

I tried a raw message and found a negative logarithmic value.

newMsg = testMsgWords[[ which(!testIsSpam)[1] ]]

newMsg = newMsg[!is.na(match(newMsg, colnames(trainTable)))]

present = (colnames(trainTable) %in% newMsg)

sum(trainTable["presentLogOdds", present]) +

sum(trainTable["absentLogOdds", !present])

## [1] -130.4715

Functioning: computeMsgLLR asks for log likelihood ratio

computeMsgLLR = **function**(words, freqTable)

{

*# 훈련 데이터에서 나타나지 않는 단어는 제외*

words = words[!is.na(match(words, colnames(freqTable)))]

*# 어떤 단어들이 나타나는지 찾음*

present = colnames(freqTable) %in% words

sum(freqTable["presentLogOdds", present]) +

sum(freqTable["absentLogOdds", !present])

}

Apply computeMsgLLR to the test data and look at the results with figures and boxes

testLLR = sapply(testMsgWords, computeMsgLLR, trainTable)

tapply(testLLR, testIsSpam, summary)

## $`FALSE`

## Min. 1st Qu. Median Mean 3rd Qu. Max.

## -2418.00 -134.00 -108.30 -123.70 -88.65 721.10

##

## $`TRUE`

## Min. 1st Qu. Median Mean 3rd Qu. Max.

## -75.360 -1.567 43.300 135.200 123.800 26190.000

spamLab = c("ham", "spam")[1 + testIsSpam]

boxplot(testLLR ~ spamLab, ylab = "Log Likelihood Ratio",

*# main = "Log Likelihood Ratio for Randomly Chosen Test Messages",*

ylim=c(-500, 500))

Misclassification rates and examples according to the reference value tau

typeIErrorRate =

**function**(tau, llrVals, spam)

{

classify = llrVals > tau

sum(classify & !spam)/sum(!spam)

}

typeIErrorRate(0, testLLR,testIsSpam)

## [1] 0.002157963

typeIErrorRate(-20, testLLR,testIsSpam)

## [1] 0.003452741

Functionalization of one and two errors

typeIErrorRates =

**function**(llrVals, isSpam)

{

o = order(llrVals)

llrVals = llrVals[o]

isSpam = isSpam[o]

idx = which(!isSpam)

N = length(idx)

list(error = (N:1)/N, values = llrVals[idx])

}

typeIIErrorRates = **function**(llrVals, isSpam) {

o = order(llrVals)

llrVals = llrVals[o]

isSpam = isSpam[o]

idx = which(isSpam)

N = length(idx)

list(error = (1:(N))/N, values = llrVals[idx])

}

The 5-fold CV is used to obtain tau so that one type of error is within 1%

k = 5

numTrain = length(trainMsgWords)

partK = sample(numTrain)

tot = k \* floor(numTrain/k)

partK = matrix(partK[1:tot], ncol = k)

testFoldOdds = NULL

**for** (i **in** 1:k) {

foldIdx = partK[ , i]

trainTabFold = computeFreqs(trainMsgWords[-foldIdx], trainIsSpam[-foldIdx])

testFoldOdds = c(testFoldOdds,

sapply(trainMsgWords[ foldIdx ], computeMsgLLR, trainTabFold))

}

testFoldSpam = NULL

**for** (i **in** 1:k) {

foldIdx = partK[ , i]

testFoldSpam = c(testFoldSpam, trainIsSpam[foldIdx])

}

xFoldI = typeIErrorRates(testFoldOdds, testFoldSpam)

xFoldII = typeIIErrorRates(testFoldOdds, testFoldSpam)

tauFoldI = round(min(xFoldI$values[xFoldI$error <= 0.01]))

tauFoldI

## [1] -41

tFold2 = xFoldII$error[ xFoldII$values < tauFoldI ]

max(tFold2)

## [1] 0.07259074