STAT480 Homework3

Chenz Zhang, NetID chenziz2

2/9/2019

Contents

Using code from chapter 3.3, get the sample emails.

```
spamPath = "~/Stat480/RDataScience/SpamAssassinMessages"

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

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)
```

Exercise 1

Get hearlist:

```
splitHeader = function(msg) {
  splitPoint = match("", msg)
  header = msg[1:(splitPoint-1)]
  return(header)
}

sampleHeaderSplit = lapply(sampleEmail, splitHeader)
```

Define my **getSubject** function:

```
getSubject = function(header) {
  subjectIdx = grep("^ *[Ss]ubject:", header, ignore.case = TRUE)
  if(length(subjectIdx) == 0) return(NA)
  gsub("^ *[Ss]ubject: *(.*)", "\\1", header[subjectIdx], ignore.case = TRUE)
}
```

Demonstrate my function on the headers from 15 samples:

```
test1 <- lapply(sampleHeaderSplit,getSubject)
unname(test1)
```

```
## [[1]]
## [1] "Re: New Sequences Window"
##
## [[2]]
## [1] "[zzzzteana] RE: Alexander"
##
## [[3]]
## [1] "[zzzzteana] Moscow bomber"
##
## [[4]]
## [1] "[IRR] Klez: The Virus That  Won't Die"
##
## [[5]]
## [1] "Re: [zzzzteana] Nothing like mama used to make"
##
## [[6]]
## [1] "Re: New Sequences Window"
##
## [[7]]
## [1] "Re: [ILUG] Sun Solaris.."
##
## [[8]]
## [1] "Tiny DNS Swap"
##
## [[9]]
## [1] "Re: Tiny DNS Swap"
##
## [[10]]
## [1] "Re: CVS report"
##
## [[11]]
## [1] "Re: Rambus, Man"
##
## [[12]]
## [1] "Re: CVS report"
##
## [[13]]
## [1] "Re: CVS report"
##
## [[14]]
## [1] "Liberalism in America"
##
## [[15]]
## [1] "Re: ActiveBuddy"
```

Exercise 2

Define my **countLetters** function:

```
countLetters = function(header){
  cleanLetters = tolower(gsub("[[:punct:]0-9[:space:][:blank:]]+", "", getSubject(header)))
  nchar(cleanLetters)
}
```

Demonstrate my function on the headers from 15 samples:

```
test2 <- lapply(sampleHeaderSplit,countLetters)
unname(test2)
```

```
## [[1]]
## [1] 20
##
## [[2]]
## [1] 20
##
## [[3]]
## [1] 21
##
## [[4]]
## [1] 26
##
## [[5]]
## [1] 36
##
## [[6]]
## [1] 20
##
## [[7]]
## [1] 16
##
## [[8]]
## [1] 11
##
## [[9]]
## [1] 13
##
## [[10]]
## [1] 11
##
## [[11]]
## [1] 11
##
## [[12]]
## [1] 11
##
## [[13]]
## [1] 11
##
## [[14]]
## [1] 19
##
## [[15]]
## [1] 13
```

Exercise 3

Use the **splitMessage** function from chapter 3.5.1 and get split email messages:

```
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)
```

Define **getBoundary** and **dropAttach** functions (from chapter 3.5.2).
But here, in case of no boundary string in the header, I add one line.

```
getBoundary = function(header) {
  boundaryIdx = grep("boundary=", header)
  if(length(boundaryIdx) == 0) return(0) # In case of no boundary string
    boundary = gsub('"', "", header[boundaryIdx])
    gsub(".*boundary= *([^;]*);?.*", "\\1", boundary)
}

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) ])
}
```

Define my **countDigits** function:

```
countDigits = function(message){

  msgBoundary = getBoundary(message$header)
  if(msgBoundary == 0) countpart = message$body
  countpart = dropAttach(message$body,msgBoundary)
  countpart = gsub("\\D"," ",countpart)

  digits = unlist(strsplit(countpart,"[[:blank:]]"))
  digits = paste(digits,sep = "",collapse = "")
  nchar(digits)
}
```

Here are my results for 15 sample emails:

```r
results = rep(0,15)
for(i in 1:15){
  results[i] = countDigits(sampleSplit[[i]])
}
results
```

```
##  [1] 116  12  13  27   4  73  11  27  81  42 115  38  38  12  14
```

Exercise 4

```r
DirDigits = function(dirN){
  # read all files in the directory
  fileNames = list.files(dirN, full.names = TRUE)
  # drop files that are not email, i.e., cmds
  notEmail = grep("cmds$", fileNames)
  if ( length(notEmail) > 0) fileNames = fileNames[ - notEmail ]

  messages = lapply(fileNames, readLines, encoding = "latin1")
  # split header and body
  emailSplit = lapply(messages, splitMessage)
  # put body and header in own lists
  bodyList = lapply(emailSplit, function(msg) msg$body)
  headerList = lapply(emailSplit, function(msg) msg$header)
  rm(emailSplit)

  # determine which messages have attachments
  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)

  # find boundary strings for messages with attachments
  boundaries = sapply(headerList[hasAttach], getBoundary)

  # drop attachments from message body
  bodyList[hasAttach] = mapply(dropAttach, bodyList[hasAttach],
                               boundaries, SIMPLIFY = FALSE)

  # new count digits function
  countDigits = function(countpart){
    countpart = gsub("\\D"," ",countpart)

    digits = unlist(strsplit(countpart,"[[:blank:]]"))
    digits = paste(digits,sep = "",collapse = "")
    nchar(digits)
  }
  dirDigits = unlist(lapply(bodyList, countDigits))




  return(dirDigits)
}
```

```
spamPath = "~/Stat480/RDataScience/SpamAssassinMessages"
dirNames = list.files(path = paste(spamPath, "messages",
                                   sep = .Platform$file.sep))
fullDirNames = paste(spamPath, "messages", dirNames,
                     sep = .Platform$file.sep)

countDirDigits = sapply(fullDirNames,DirDigits)
```

```
## Warning in FUN(X[[i]], ...): incomplete final line found on '/home/
## chenziz2/Stat480/RDataScience/SpamAssassinMessages/messages/hard_ham/
## 00228.0eaef7857bbbf3ebf5edbbdae2b30493'

## Warning in FUN(X[[i]], ...): incomplete final line found on '/home/
## chenziz2/Stat480/RDataScience/SpamAssassinMessages/messages/hard_ham/
## 0231.7c6cc716ce3f3bfad7130dd3c8d7b072'

## Warning in FUN(X[[i]], ...): incomplete final line found on '/home/
## chenziz2/Stat480/RDataScience/SpamAssassinMessages/messages/hard_ham/
## 0250.7c6cc716ce3f3bfad7130dd3c8d7b072'
```

```
lapply(countDirDigits,summary)
```

```
## $`~/Stat480/RDataScience/SpamAssassinMessages/messages/easy_ham`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    7.00   20.00   31.92   32.00 1411.00
##
## $`~/Stat480/RDataScience/SpamAssassinMessages/messages/easy_ham_2`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     4.0    16.0    31.3    32.0  2514.0
##
## $`~/Stat480/RDataScience/SpamAssassinMessages/messages/hard_ham`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   74.25  584.00  708.95 1297.00 3980.00
##
## $`~/Stat480/RDataScience/SpamAssassinMessages/messages/spam`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    24.0    56.5   196.4   184.2 18876.0
##
## $`~/Stat480/RDataScience/SpamAssassinMessages/messages/spam_2`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    25.0    63.0   217.9   252.0 13868.0
```

According to mean and median, we can divided messages in these 5 directory into 3 groups:

- easy_ham (dir: easy_ham and easy_ham_2): small median (<=20) and mean (around 31). So, easy-harm tends to have less digits.
- hard_ham (dir: hard_ham): largest median 584.00 and mean 708.95. So, when we roughly seperate ham and spam by one standard for digits, hard_ham trends to be classified as spam. That migtht be

why it is hard.

- spam (dir: spam and spam_2):
  - median (around 60) is larger than easy_ham but much smaller than hard_ham.
  - mean (anoud 200) is larger than easy_ham but much smaller than hard_ham.
  - maximum is much larger than ham's. We can say spam tends to have larger 'variance'.

Therefore, when we classify spam and ham emails, range of number of digits might be a good choice.