Neeraj Asthana
nasthan2
2/12/2016
Stat 480 Homework 4

*Most of the code was taken and modified from code demonstrated in class

Question 8
The alternative mathematical expression I chose to use was:

$$log(\prod_{words\ in\ message} \frac{P(word\ present|spam)}{P(word\ present|ham)}) + log(\prod_{words\ not\ in\ message} \frac{P(word\ absent|spam)}{P(word\ absent|ham)})$$

This new method is defined by the functions “cmoputeFreqs2” and “computeMsgLLR2” in the code. By using this method, the total accuracy dropped from .93967 to .873556. Additionally, using system.time(), the original function (computeMsgLLR) took around 125 seconds, while the new method that I defined (computeMsgLLR2) took around 140 seconds, adding 15 more seconds to the computation time.

Question 9
In the original function, computeFreqs, an error is caused because inside the function, wordTable is defined to be a matrix with the number of columns equal to the number of rows. Therefore, if when we attempt to run the line: “ wordTable["spam", names(counts.spam)] = counts.spam + .5” (and similarly for ham) with our self defined bow, we will receive an error because that word may not exist in our wordTable. In order to fix this problem, I created a new function, computeFreqs3, which instead just adds the counts of the words from the wordList that are defined in the bow. This function can be found in the function labelled computeFreqs3 in the code.

Question 13
I created a new function, isYelling2, defined here and below which will count the number of lines in the message text that have all capital letters. If a message’s body is empty or a message has no text, the function will return NA. All punctuation is removed before counting the number of capital letters. All empty lines are skipped over and not counted at all. The function works by first first checking if the body of a message exists and removes all of the non alphabetical characters from the body. It will then remove all of the empty lines. Lastly, the function will remove all the capital letters from each line and check if there are any lines with no characters left.

```
isYelling2 = function(msg) {
body = msg$body
if (length(body) > 0) {
# if body exists, remove non-alpha characters
lines = gsub("[^[:alpha:]]", "", body)

#removes all empty lines
lines = lines[nchar(lines) > 0]
```

```
#delete all upper case characters
lowercaselines = gsub("[A-Z]", "", lines)

#count number of lines that now have zero characters and return that value
as the result of the function
count = length(lowercaselines[nchar(lowercaselines) < 1])

#adding percentages
percentage = count / length(body)

c(count, percentage)
#percentage

} else
NA
}
```

Question 14

(This is a graduate exercise that I do not have to complete). I defined 2 new functions, isRe2 (which checks for both the “Fwd: Re:” and “Re:” structures in the subject line) and isRe3 (which checks for “Re:” anywhere in the subject line and not just the beginning). These functions are defined below:

```
isRe2 = function(msg) {
"Subject" %in% names(msg$header) && (
length(grep("^[ \t]*Re:", msg$header[["Subject"]])) > 0
|| length(grep("^[ \t]*Fwd: Re:", msg$header[["Subject"]])) > 0 )
}

isRe3 = function(msg) {
"Subject" %in% names(msg$header) &&
length(grep("*Re:", msg$header[["Subject"]])) > 0

}
```

From the results we see that the original isRe and isRe2 functions gives us 3005 emails each while isRe3 gives us 3218 emails.