DATA607_Project_4

RJM

2019-11-19

Calling the appropriate libraries:

```
library(tm); library(e1071); library(dplyr); library(wordcloud);
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
## Loading required package: RColorBrewer
```

```
library(caret); library(readtext); library(TAR); library(R.utils);
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:NLP':
##
##     annotate
```

```
## Loading required package: mvtnorm
```

```
## Loading required package: R.oo
```

```
## Loading required package: R.methodsS3
```

```
## R.methodsS3 v1.7.1 (2016-02-15) successfully loaded. See ?R.methodsS3 for help.
```

```
## R.oo v1.23.0 successfully loaded. See ?R.oo for help.
```

```
##
## Attaching package: 'R.oo'

## The following object is masked from 'package:R.methodsS3':
##
##     throw

## The following objects are masked from 'package:methods':
##
##     getClasses, getMethods

## The following objects are masked from 'package:base':
##
##     attach, detach, load, save

## R.utils v2.9.0 successfully loaded. See ?R.utils for help.

##
## Attaching package: 'R.utils'

## The following object is masked from 'package:utils':
##
##     timestamp

## The following objects are masked from 'package:base':
##
##     cat, commandArgs, getOption, inherits, isOpen, nullfile,
##     parse, warnings
```

```
library(SnowballC)
```

Assigning the right folders and files:

```
# Code to get the email message
spam_file <- "/Users/rmirza/Downloads/20021010_spam.tar.bz2"
ham_file <- "/Users/rmirza/Downloads/20021010_easy_ham.tar.bz2"

spam_file_unzip <- "/Users/rmirza/Downloads/20021010_spam.tar"
ham_file_unzip <- "/Users/rmirza/Downloads/20021010_easy_ham.tar"

bunzip2(spam_file, remove = FALSE, skip = TRUE)
```

```
## [1] "/Users/rmirza/Downloads/20021010_spam.tar"
## attr(,"temporary")
## [1] FALSE
```

```
bunzip2(ham_file, remove = FALSE, skip = TRUE)
```

```
## [1] "/Users/rmirza/Downloads/20021010_easy_ham.tar"
## attr(,"temporary")
## [1] FALSE
```

```
# Creating a combined directory to store the data
spam_ham_dir <- "/Users/rmirza/Documents/DATA607/spam_ham"

untar(spam_file_unzip, exdir = spam_ham_dir)
untar(ham_file_unzip, exdir = spam_ham_dir)

# Segregating the data in separate folders for spam and ham
spam_dir <- "/Users/rmirza/Documents/DATA607/spam_ham/spam"
ham_dir <- "/Users/rmirza/Documents/DATA607/spam_ham/easy_ham"

spam_files <- list.files(spam_dir)
ham_files <- list.files(ham_dir)
```

Preparation of dataframes:

```
# Preparing data to create dataframes
spam_email_content <- NA
for(i in 1:length(spam_files))
{
  file_address_spam <- paste0(spam_dir, "/", spam_files[1])
  spam_lines <- readLines(file_address_spam)
  list_spam <- list(paste(spam_lines, collapse="\n "))
  spam_email_content = c(spam_email_content, list_spam)

}

# Code for the Spam Dataframe
spam_df <- data.frame()
spam_df <- as.data.frame(unlist(spam_email_content), stringsAsFactors = FALSE)
spam_df$type <- "Spam"
colnames(spam_df) <- c("email","type")


ham_email_content <- NA
for(i in 1:length(ham_files))
{
  file_address_ham <- paste0(ham_dir, "/", ham_files[1])
  ham_lines <- readLines(file_address_ham)
  list_ham <- list(paste(ham_lines, collapse="\n"))
  ham_email_content <- c(ham_email_content, list_ham)

}

# Code for the Ham Dataframe

ham_df <- data.frame()
ham_df <- as.data.frame(unlist(ham_email_content), stringsAsFactors = FALSE)
ham_df$type <- "Ham"
```

```r
colnames(ham_df) <- c("email","type")

nrow(spam_df)
```

```
## [1] 502
```

```r
nrow(ham_df)
```

```
## [1] 2552
```

```r
combined_df <- rbind(spam_df, ham_df)

nrow(combined_df)
```

```
## [1] 3054
```

Cleaning the corpus:

```r
original_corpus<- Corpus(VectorSource(combined_df$email))

# To cleand the corpus and create a word cloud to assess
# the type of words used in each type of messages
cleaned_corpus <- suppressWarnings(original_corpus %>%
    # to convert into lower case
    tm_map(content_transformer(tolower)) %>%
    # to remove any punctuation
    tm_map(removePunctuation) %>%
    # to remove numbers
    tm_map(removeNumbers) %>%
    # to remove the stop words
    tm_map(removeWords, stopwords()) %>%
    # to stem the words into root word
    tm_map(stemDocument) %>%
    # to remove the white space
    tm_map(stripWhitespace))
```

```r
# Checking the contents after cleaning
inspect(cleaned_corpus[1:3])
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 3
##
## [1] NA
## [2] mv bfcddbffccadb mv ffbcebefcdceaaa mv cdcbcfaba mv dfdeeaebdceaefa mv eddeaedb mv cbfedfedfbfa r
## [3] mv bfcddbffccadb mv ffbcebefcdceaaa mv cdcbcfaba mv dfdeeaebdceaefa mv eddeaedb mv cbfedfedfbfa r
```

```r
inspect(cleaned_corpus[503:505])
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 3
##
## [1] NA
## [2] exmhworkersadminredhatcom thu aug returnpath exmhworkersadminexamplecom deliveredto zzzzlocalhos
## [3] exmhworkersadminredhatcom thu aug returnpath exmhworkersadminexamplecom deliveredto zzzzlocalhos
```

Inspecting the dataframes created:

```r
combined_df_dtm<- DocumentTermMatrix(cleaned_corpus)
inspect(combined_df_dtm[1:4, 503:505])
```

```
## <<DocumentTermMatrix (documents: 4, terms: 3)>>
## Non-/sparse entries: 0/12
## Sparsity           : 100%
## Maximal term length: 14
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs cant cfd charsetusascii
##    1    0   0              0
##    2    0   0              0
##    3    0   0              0
##    4    0   0              0
```

Creation of word clouds to get an idea of the content of cleaned dataframes:

```r
# Creating a spam word cloud
spam_tags <- which(combined_df$type == "Spam")
suppressWarnings(wordcloud(cleaned_corpus[spam_tags], min.freq=100))
```


```
# Creating a ham word cloud
ham_tags <- which(combined_df$type == "Ham")
suppressWarnings(wordcloud(cleaned_corpus[ham_tags], min.freq=100))
```

Pepraring the data for the Naives Bayes test:

```
# Using 70% of data for the training set and
# 30% for the test set.
train_combined_df <- createDataPartition(combined_df$email, p=0.70, list=FALSE)
```

```
set.seed(123)
combined_df_train <- combined_df[train_combined_df,]
combined_df_test <- combined_df[-train_combined_df,]

cleaned_corpus_train<- cleaned_corpus[train_combined_df]
cleaned_corpus_test<- cleaned_corpus[-train_combined_df]

combined_df_dtm_train <- combined_df_dtm[train_combined_df,]
combined_df_dtm_test <- combined_df_dtm[-train_combined_df,]

train_cleaned_dtm <- DocumentTermMatrix(cleaned_corpus_train)
test_cleaned_dtm <- DocumentTermMatrix(cleaned_corpus_test)

spam <- subset(train_combined_df,cleaned_corpus_train$type == "Spam")
ham <- subset(train_combined_df,cleaned_corpus_train$type == "Ham")

cleaned_count <- function(x) {
  y <- ifelse(x > 0, 1, 0)
  y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
  y
}
```

Running the test:

```
train_cleaned_set <- apply(train_cleaned_dtm, 2, cleaned_count)
test_cleaned_set <- apply(test_cleaned_dtm, 2, cleaned_count)

data_classifier <- naiveBayes(train_cleaned_set, factor(combined_df_train$type))

Test_output <- predict(data_classifier, newdata = test_cleaned_set)

table(Test_output, combined_df_test$type)
```

```
##
## Test_output Ham Spam
##        Ham  765    0
##        Spam   0  150
```

Conclusion:

The results indicate that there are no “false positives” as result of the Naives Bayes test. However, I feel that my model has some issues (probably overfitting) that have caused the model to show each of the spam and ham properly fished out. The model shows that there are 765 hams and 150 spams as of it stands now with none of them erroneously designated.