Data_607_Project_4_Document_Classification

Enid Roman

2022-11-19

It can be useful to be able to classify new "test" documents using already classified "training" documents. A common example is using a corpus of labeled spam and ham (non-spam) e-mails to predict whether or not a new document is spam.

For this project, we start with a spam/ham dataset, then predict the class of new documents (either withheld from the training dataset or from another source such as your own spam folder). One example corpus: https://spamassassin.apache.org/old/publiccorpus/

LOAD REQUIRED LIBRARIES

Install & load necessary libraries.

TM is R's text mining package, which provides several functions for text handling, processing and management. The package uses the concept of a 'corpus' which is a collection of text documents to operate upon. Text can be stored either in-memory in R via a Volatile Corpus or on an external data store such as a database via a Permanent Corpus.

```
#install.packages('tm')
library(tidyverse)
```

Other packages are supplementary packages that are used for reading lines from file, plotting, preparing word clouds, N-Gram generation, etc.

```
## Warning: package 'tidyverse' was built under R version 4.2.2
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidyr)
library(dplyr)
library(purrr)
library(stringr)
library(stringi)
library(ggplot2)
library(readr)
library(stats)
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
##
## The following object is masked from 'package:purrr':
##
##     set_names
##
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(R.utils)
```

```
## Loading required package: R.oo
## Loading required package: R.methodsS3
## R.methodsS3 v1.8.2 (2022-06-13 22:00:14 UTC) successfully loaded. See ?R.methodsS3 for help.
## R.oo v1.25.0 (2022-06-12 02:20:02 UTC) successfully loaded. See ?R.oo for help.
##
## Attaching package: 'R.oo'
##
## The following object is masked from 'package:R.methodsS3':
##
##     throw
##
## The following object is masked from 'package:magrittr':
##
##     equals
##
## The following objects are masked from 'package:methods':
##
##     getClasses, getMethods
##
## The following objects are masked from 'package:base':
##
##     attach, detach, load, save
##
## R.utils v2.12.0 (2022-06-28 03:20:05 UTC) successfully loaded. See ?R.utils for help.
##
## Attaching package: 'R.utils'
##
## The following object is masked from 'package:magrittr':
##
```

```
##     extract
##
## The following object is masked from 'package:tidyr':
##
##     extract
##
## The following object is masked from 'package:utils':
##
##     timestamp
##
## The following objects are masked from 'package:base':
##
##     cat, commandArgs, getOption, isOpen, nullfile, parse, warnings
```

```
#install.packages("kableExtra")
library(kableExtra)
```

```
## Warning: package 'kableExtra' was built under R version 4.2.2
```

```
## Warning in !is.null(rmarkdown::metadata$output) && rmarkdown::metadata$output
## %in% : 'length(x) = 2 > 1' in coercion to 'logical(1)'
```

```
##
## Attaching package: 'kableExtra'
##
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
library(tm)
```

```
## Warning: package 'tm' was built under R version 4.2.2
```

```
## Loading required package: NLP
##
## Attaching package: 'NLP'
##
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(wordcloud)
```

```
## Warning: package 'wordcloud' was built under R version 4.2.2
```

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

```
library(topicmodels)
```

```
## Warning: package 'topicmodels' was built under R version 4.2.2
```

```
library(SnowballC)
library(e1071)
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
#install.packages('quanteda')
library(quanteda)
```

```
## Warning: package 'quanteda' was built under R version 4.2.2
```

```
## Package version: 3.2.3
## Unicode version: 13.0
## ICU version: 69.1
## Parallel computing: 8 of 8 threads used.
## See https://quanteda.io for tutorials and examples.
##
## Attaching package: 'quanteda'
##
## The following object is masked from 'package:tm':
##
##     stopwords
##
## The following objects are masked from 'package:NLP':
##
##     meta, meta<-
```

```
#install.packages('naivebayes')
library(naivebayes)
```

```
## Warning: package 'naivebayes' was built under R version 4.2.2
```

```
## naivebayes 0.9.7 loaded
##
## Attaching package: 'naivebayes'
##
## The following object is masked from 'package:data.table':
##
##     tables
```

DATA COLLECTION

Obtained the spam and ham data from Spam Assassin Appache site: https://spamassassin.apache.org/old/publiccorpus/. Went to Public Corpus folder.

Downloaded spam and ham data to use for the document classification exercise. For spam, I downloaded 20030228_spam.tar.bz2 and for ham folder I downloaded 20030228_easy_ham_2.tar.bz2.

Unzipped both folders using 7-zip file manager.

```
# Load the spam and ham data into R from my folder in my computer. Had trouble creating the corpus when

#url_spam <- "https://spamassassin.apache.org/old/publiccorpus/20030228_spam.tar.bz2"
#url_ham <- "https://spamassassin.apache.org/old/publiccorpus/20030228_easy_ham_2.tar.bz2"

pathName_spam <- "C:\\Users\\enidr\\OneDrive\\Documents\\CUNY SPS DATA 607\\Project 4\\spam"
file_name_spam <- list.files(pathName_spam)

pathName_ham <- "C:\\Users\\enidr\\OneDrive\\Documents\\CUNY SPS DATA 607\\Project 4\\easy_ham_2"
file_name_ham <- list.files(pathName_ham)
```

Opened both files in Notepad ++ to explore spam and ham emails.

```
head(file_name_spam)
```

Got the first number of rows to test if I have the right type of data.

```
## [1] "00001.7848dde101aa985090474a91ec93fcf0"
## [2] "00002.d94f1b97e48ed3b553b3508d116e6a09"
## [3] "00003.2ee33bc6eacdb11f38d052c44819ba6c"
## [4] "00004.eac8de8d759b7e74154f142194282724"
## [5] "00005.57696a39d7d84318ce497886896bf90d"
## [6] "00006.5ab5620d3d7c6c0db76234556a16f6c1"
```

```
head(file_name_ham)
```

```
## [1] "00001.1a31cc283af0060967a233d26548a6ce"
## [2] "00002.5a587ae61666c5aa097c8e866aedcc59"
## [3] "00003.19be8acd739ad589cd00d8425bac7115"
## [4] "00004.b2ed6c3c62bbdfab7683d60e214d1445"
## [5] "00005.07b9d4aa9e6c596440295a5170111392"
## [6] "00006.654c4ec7c059531accf388a807064363"
```

```
length_spam <- length(file_name_spam)
length_spam
```

Got the length 501 for spam emails and 1401 for non spam (ham) emails

```
## [1] 501
```

```
length_ham <- length(file_name_ham)
length_ham
```

```
## [1] 1401
```

```
file_size <- format(object.size(file_name_spam), units = "Kb")
file_num_lines <- length(file_name_spam)
file_words <- sum(stri_count_words(file_name_spam))
cat (" File Size: ", file_size, " Line Size: ", file_num_lines, " File Words: ", file_words)
```

Obtained the file size, line size, and file words for both spam and ham files.

```
##  File Size:  50.9 Kb  Line Size:  501  File Words:  694
```

```
file_size <- format(object.size(file_name_ham), units = "Kb")
file_num_lines <- length(file_name_ham)
file_words <- sum(stri_count_words(file_name_ham))
cat (" File Size: ", file_size, " Line Size: ", file_num_lines, " File Words: ", file_words)
```

```
##  File Size:  142.3 Kb  Line Size:  1401  File Words:  1912
```

```
file_name_spam <- file_name_spam[which(file_name_spam!="cmds")]
file_name_ham <- file_name_ham[which(file_name_ham!="cmds")]
```

Removed the .cmds files from all the files.

```
spam_email = list.files(path = "spam_file", full.names = TRUE)
ham_email = list.files(path = "ham_file",full.names = TRUE)
```

Used 'list.files' on our 'spam_folder' object which produces a character vector of the names of files.

Processing Textual Data - Corpus Creation

```
# spam folder files

spam_corpus <- pathName_spam %>%
  paste(., list.files(.), sep = "/") %>%
  lapply(readLines) %>%
  VectorSource() %>%
  VCorpus()
```

We need to create a collection of documents (technically referred to as a Corpus) in the R environment. This basically involves loading the files created in the TextMining folder into a Corpus object. The tm package provides the Corpus() function to do this.

```
## Warning in FUN(X[[i]], ...): incomplete final line found on 'C:
## \Users\enidr\OneDrive\Documents\CUNY SPS DATA 607\Project 4\spam/
## 00136.faa39d8e816c70f23b4bb8758d8a74f0'
```

```
spam_corpus
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 501
```

```
# ham folder files

ham_corpus <- pathName_ham %>%
  paste(., list.files(.), sep = "/") %>%
  lapply(readLines) %>%
  VectorSource() %>%
  VCorpus()

ham_corpus
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 1401
```

DATA CLEANING

Corpus Cleaning

In terms of cleaning the corpus for each folder we will use the tm package and follow below steps;

Remove the numbers and punctuations

Remove stopwords such as to, from, and, the etc...

Remove blankspaces.

```
# spam emails

spam_corpus <- spam_corpus %>%
  tm_map(removeNumbers) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeWords, stopwords()) %>%
```

```
  tm_map(stripWhitespace) %>%
  tm_map(stemDocument)
spam_corpus
```

Reduce the terms to their stem.

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 501
```

```
# ham emails

ham_corpus <- ham_corpus %>%
  tm_map(removeNumbers) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeWords, stopwords()) %>%
  tm_map(stripWhitespace) %>%
  tm_map(stemDocument)
ham_corpus
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 1401
```

```
spam_or_ham_corpus <- c(spam_corpus, ham_corpus)
```

Combined both spam and non spam (ham) emails.

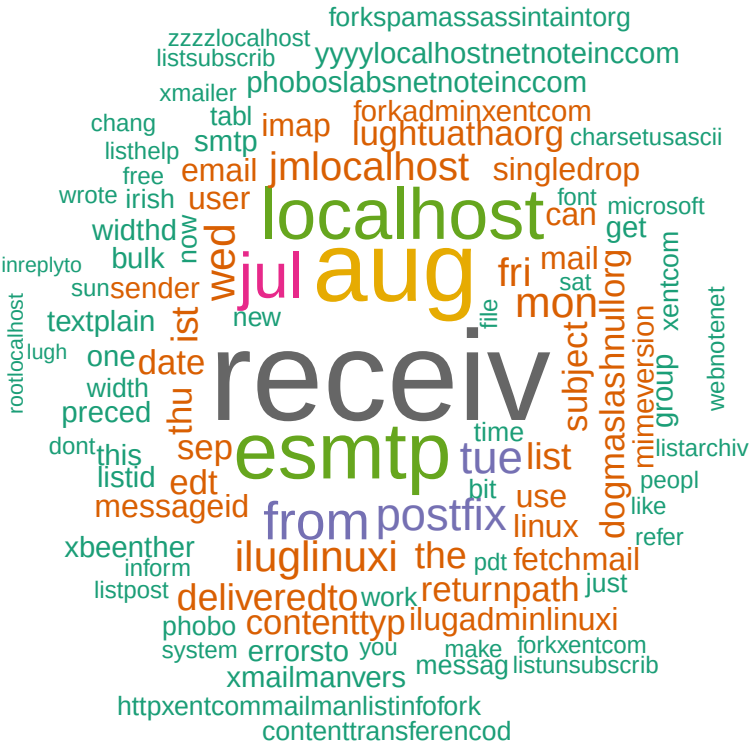
Building a Term Document Matrix

```
tdm <- DocumentTermMatrix(spam_or_ham_corpus)
tdm
```

```
## <<DocumentTermMatrix (documents: 1902, terms: 55131)>>
## Non-/sparse entries: 345402/104513760
## Sparsity           : 100%
## Maximal term length: 461
## Weighting          : term frequency (tf)
```

Creating Word Cloud with Header Text

```
wordcloud(spam_or_ham_corpus, max.words = 100, random.order = FALSE, rot.per=0.15, min.freq=5, colors =
```



MODEL DEVELOPMENT

A classification such as Naive Bayes classifier is being used to find out the presence of certain features (words) in a defined class to predict if the email is spam or ham.

Data Preperation for Model Development

Creating Dataframe

```
df_spam <- as.data.frame(unlist(spam_corpus), stringsAsFactors = FALSE)
df_spam$type <- "ham"
colnames(df_spam)=c("text", "email")

df_ham <- as.data.frame(unlist(ham_corpus), stringsAsFactors = FALSE)
df_ham$type <- "spam"
colnames(df_ham)=c("text", "email")

df_spam_or_ham <- rbind(df_spam, df_ham)

kable(head(df_spam_or_ham))
```

Before we start creating our training and test data sets and process, we need to create a combined dataframe, label the corpus (ham or spam) as part of supervised technique.

text	email
From amailbotwebd Thu Aug	ham
ReturnPath amailbotwebd	ham
DeliveredTo zzzzlocalhostspamassassintaintorg	ham
Receiv localhost localhost	ham
phoboslabsspamassassintaintorg Postfix ESMTP id BC	ham
zzzzlocalhost Thu Aug EDT	ham

Prepare Test and Train Data

Splitting Test and Train Data

```
sample_size <- floor(0.80 * nrow(df_spam_or_ham)) # selecting sample size of 80% of the data for traini

set.seed(123)
train <- sample(seq_len(nrow(df_spam_or_ham)), size = sample_size)

train_spam_or_ham <- df_spam_or_ham[train, ]
test_spam_or_ham <- df_spam_or_ham[-train, ]

head(train_spam_or_ham)
```

We will split 80% of the data as training data and 30% as the test data.

```
##                                       text email
## 182735                                 122  spam
## 188942     httpxentcommailmanlistinfofork  spam
## 134058                  httpthinkgeekcomsf  spam
## 124022                                     spam
## 160997                                     spam
## 226318 mv dcaeadfcaefeebb dcaeadfcaefeebb  spam
```

```
head(test_spam_or_ham)
```

```
##                                                text email
## 2                          ReturnPath amailbotwebd   ham
## 5  phoboslabsspamassassintaintorg Postfix ESMTP id BC   ham
## 6                        zzzzlocalhost Thu Aug EDT   ham
## 14       Receiv rsmtpkoreacom ddit Microsoft SMTPSVC   ham
## 15                                          Sat Aug   ham
## 25                                         HTMLHEAD   ham
```

```
# corpus creation
train_corpus <- Corpus (VectorSource(train_spam_or_ham$text)) # corpus training data
test_corpus <- Corpus(VectorSource(test_spam_or_ham$text)) # corpus test data
```

```
# corpus cleaning
train_corpus <- train_corpus %>%
  tm_map(removeNumbers) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeWords, stopwords()) %>%
  tm_map(stripWhitespace)
```

Create and Clean Corpus and Create Term Document Matrix for Training and Test Data.

```
## Warning in tm_map.SimpleCorpus(., removeNumbers): transformation drops documents

## Warning in tm_map.SimpleCorpus(., removePunctuation): transformation drops
## documents

## Warning in tm_map.SimpleCorpus(., removeWords, stopwords()): transformation
## drops documents

## Warning in tm_map.SimpleCorpus(., stripWhitespace): transformation drops
## documents
```

```
test_corpus <- test_corpus %>%
  tm_map(removeNumbers) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeWords, stopwords()) %>%
  tm_map(stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(., removeNumbers): transformation drops documents

## Warning in tm_map.SimpleCorpus(., removePunctuation): transformation drops
## documents

## Warning in tm_map.SimpleCorpus(., removeWords, stopwords()): transformation
## drops documents

## Warning in tm_map.SimpleCorpus(., stripWhitespace): transformation drops
## documents
```

```
train_tdm <- DocumentTermMatrix(train_corpus)
test_tdm <- DocumentTermMatrix(test_corpus)

train_tdm
```

```
## <<DocumentTermMatrix (documents: 181536, terms: 47685)>>
## Non-/sparse entries: 444327/8656099833
## Sparsity           : 100%
## Maximal term length: 298
## Weighting          : term frequency (tf)
```

```
test_tdm
```

```
## <<DocumentTermMatrix (documents: 45384, terms: 19722)>>
## Non-/sparse entries: 111123/894952125
## Sparsity           : 100%
## Maximal term length: 117
## Weighting          : term frequency (tf)
```

```
train_corpus
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 181536
```

```
test_corpus
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 45384
```

```
spam <- subset(train_spam_or_ham, email == "spam")
ham <- subset(train_spam_or_ham, email == "ham")
```

Separate training data to spam and ham.

```
fifty_times_words<- findFreqTerms(train_tdm, 50)
length(fifty_times_words)
```

If we run all the observation in my data, R doesnt have enough memory to execute it at the moment. So, I am going to narrow down the observations by selecting words that uses at least 50 times in the training document.

```
## [1] 1252
```

```
train_tdm_2<- DocumentTermMatrix(train_corpus, control=list(dictionary = fifty_times_words))

test_tdm_2<- DocumentTermMatrix(test_corpus, control=list(dictionary = fifty_times_words))
```

Create a classifier for each email.

Model Development

```
# This is required in order to set the classifier for naiveBayes

class(train_tdm_2)
```

Create a classifier for each email.

```
## [1] "DocumentTermMatrix"    "simple_triplet_matrix"
```

```
train_tdm_3 <- as.matrix(train_tdm_2)
train_tdm_3 <- as.data.frame(train_tdm_3)
class(train_tdm_3)
```

Train the model.

```
## [1] "data.frame"
```

```
classifier <- naiveBayes(train_tdm_3, factor(train_spam_or_ham$email))
```

```
class(classifier)
```

```
## [1] "naiveBayes"
```

```
class(test_tdm_2)
```

Test the model.

```
## [1] "DocumentTermMatrix"    "simple_triplet_matrix"
```

```
test_tdm_3 <- as.matrix(test_tdm_2)
test_tdm_3 <- as.data.frame(test_tdm_3)
class(test_tdm_3)
```

```
## [1] "data.frame"
```

PREDICTION

```
test_pred <- predict(classifier, newdata=test_tdm_3)
```

```
table(predicted=test_pred,actual=test_tdm_3[,1])
```

We can use the predict function to test the model on new data. " test_pred <- predict(classifier, newdata=test_tdm_3)".

```
##          actual
## predicted     0     1
##      ham  41269   415
##      spam  3700     0
```

CONCLUSION

The Result: We are able to generate prediction of email being ham or spam (using supervised technique -naive Bayes method). We can further test it against the raw data and evaluate model's performance.

I first ran into problem when I loaded the url from the website provided by the professor. Codes were running well until I started to create the Corpus. I then had to load the file from my computer which workded out very well with all the codes. Since most of the codes took longer to run I had to change the class type to make the classifier work.