Project 4

Christian Uriostegui

2022-11-25

Introduction

In my document, I will be detailing the creation and utilization of a Naive Bayes classifier to identify spam and ham emails. The spam and ham datasets utilized can be downloaded from https://spamassassin.apache.org/old/publiccorpus/.

The two files I downloaded are easy_ham (which contains 2,551 email files) and spam (which contains 501 email files). I will be downloading the files, detail the cleaning process, creation of the Corpus, and Document Term Matrix. Once the files are in the correct format, I will then build and test a Naives Bayes classifier to detect spam and ham emails.

Load library

```
library(tidyverse)
library(stringr)
library(knitr)
library(tidytext)
library(tidyr)
library(magrittr)
library(quanteda)
library(tm)
library(e1071)
library(wordcloud)
```

Load Data

Due to the large amount of files, I will be downloading the files directly from the website, load them to my direct drive, unzip the files and assign them to the objects `ham_loc`, and `spam_loc`.

```
# load spam and ham dataset into object
easyham_url <- "https://spamassassin.apache.org/old/publiccorpus/20021010_easy_ham.tar.bz2"

spam_url <- "https://spamassassin.apache.org/old/publiccorpus/20030228_spam.tar.bz2"

# download data into local drive
download.file(easyham_url, destfile = "ham")

download.file(spam_url, destfile = "spam")
```

In the code below, I am extracting the name of every file from my datasets, separating them into the objects `filenames_ham` and `filenames_spam`. This will allow me to later utilize the function `readLines` to read each message contained in the objects `ham_loc`, and `spam_loc`.

```
# read file names from Spam and Ham dataset
filenames_ham <- list.files(ham_loc, pattern = "*.*", full.names = TRUE)

filenames_spam <- list.files(spam_loc, pattern = "*.*", full.names = TRUE)
```

Data Preparation

Create spam and ham dataframes

Prior to creating my dataframes, I created a function that removes HTML strings.

```
# create function that removes HTML tags
htmlclean <- function(htmlString) {
  return(gsub("<.*?>", "", htmlString))
}
```

In the code below, along with creating the `spam` and `ham` dataframes, I perform extensive cleaning and transformation. In the text within the emails, there are headers, senders list, received addresses, and other metadata that will need to be removed. The steps are show below:

1. **Add column names:** We want to make sure that once the messages are extracted, that both the columns that include the name of file and text are named accordingly.
2. **Extract email content:** Using the lapply along with readLines function, we can extract the email content for each file.
3. **Remove HTML strings:** Now we can apply the htmlclean function created earlier to remove HTML strings - content which we are not interest in.
4. **Unnest and paste text column:** The unnest and paste will be applied to expand text by line and eventually combine the strings in the `text` column.
5. **Add additional column label:** This column will be added seperately to the ham and spam dataframe. Depending on which dataframe I am working on, I will add a label of no (this is not spam) or yes (this is spam).

```
ham <- filenames_ham %>%
  as.data.frame() %>%
  set_names("file") %>%
  mutate(text = lapply(filenames_ham,readLines)) %>%
  mutate(text = lapply(text,htmlclean)) %>%
  unnest(c(text)) %>%
  mutate(spam = "no") %>%
  group_by(file) %>%
  mutate(text = paste(text,collapse=" ")) %>%
  ungroup() %>%
  distinct()
```

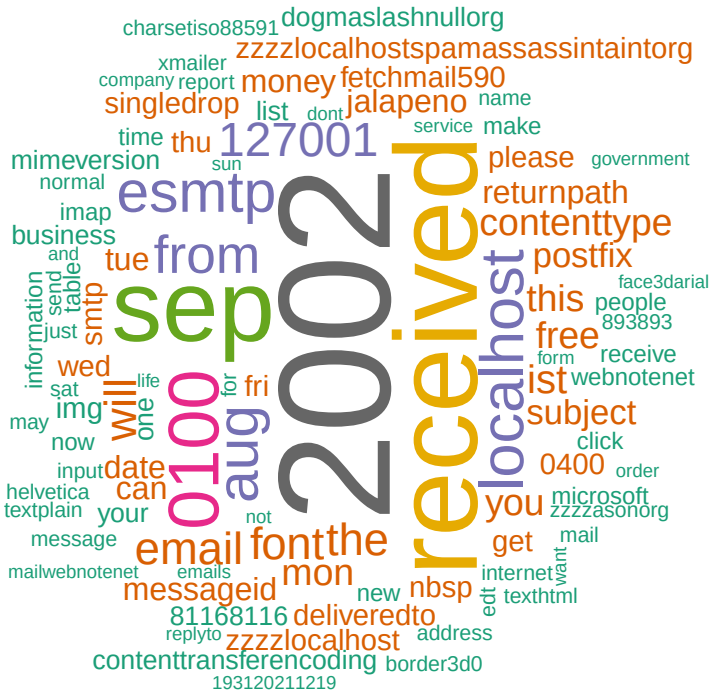
```
spam <- filenames_spam %>%
  as.data.frame() %>%
  set_names("file") %>%
  mutate(text = lapply(filenames_spam,readLines)) %>%
  mutate(text = lapply(text,htmlclean)) %>%
  unnest(c(text)) %>%
  mutate(spam = "yes") %>%
  group_by(file) %>%
  mutate(text = paste(text,collapse=" ")) %>%
  ungroup() %>%
  distinct()
```

Once cleaned and ready, I used the rbind function to join both datasets.

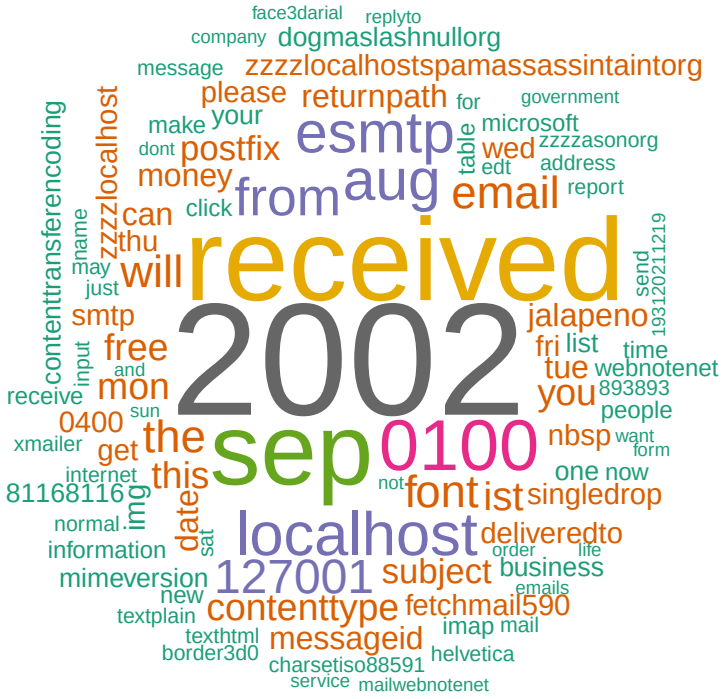
```
spam = read.csv("C:\\Users\\urios\\OneDrive\\Documents\\spam.csv")
ham = read.csv("C:\\Users\\urios\\OneDrive\\Documents\\ham.csv")
combined <- rbind(spam,ham)

set.seed(123)

# shuffle the dataframe by rows
combined= combined[sample(1:nrow(combined)), ]
```

Wordcloud visualization

```
spamword <- subset(combined, spam == "yes")
wordcloud(spamword$text,max.words = 100, scale = c(5, 0.3), random.order = FALSE, rot.per = 0.15, colors
```




```
hamword <- subset(combined, spam == "no")
wordcloud(spamword$text,max.words = 100, scale = c(5, 0.3), random.order = FALSE, rot.per = 0.15, colors
```



Convert spam/ham to factor.

```
combined$spam <- factor(combined$spam)
```

Examine the type variable more carefully

```
table(combined$spam)
```

```
##
##   no  yes
## 2551  501
```

Build a corpus using the text mining (tm) package

```
email_corpus <- Corpus(VectorSource(combined$text))
```

Build a corpus using the text mining (tm) package

```
corpus_clean <- tm_map(email_corpus, tolower)
corpus_clean <- tm_map(corpus_clean, removeNumbers)
corpus_clean <- tm_map(corpus_clean, removeWords, stopwords())
corpus_clean <- tm_map(corpus_clean, removePunctuation)
corpus_clean <- tm_map(corpus_clean, stripWhitespace)
```

Create a document-term sparse matrix

```
email_dtm <- DocumentTermMatrix(corpus_clean)
email_dtm
```

```
## <<DocumentTermMatrix (documents: 3052, terms: 57531)>>
## Non-/sparse entries: 455550/175129062
## Sparsity           : 100%
## Maximal term length: 298
## Weighting          : term frequency (tf)
```

Building and Testing the Naive Bayes Classifier

creating training and test datasets

```
email_raw_train <- combined[1:2000, ]
email_raw_test  <- combined[2000:3052, ]

email_dtm_train <- email_dtm[1:250, ]
email_dtm_test  <- email_dtm[250:301, ]

email_corpus_train <- corpus_clean[1:2000]
email_corpus_test  <- corpus_clean[2000:3052]
```

Check that the proportion of spam is similar

```
prop.table(table(email_raw_train$spam))
```

```
##
##     no    yes
## 0.8345 0.1655
```

```
prop.table(table(email_raw_test$spam))
```

```
##
##        no       yes
## 0.8385565 0.1614435
```

Indicator features for frequent words

```
email_dict <- findFreqTerms(email_dtm_train, 5)
#email_dict <- Dictionary(findFreqTerms(email_dtm_train, 5))
email_train <- DocumentTermMatrix(email_corpus_train, list(dictionary = email_dict))
email_test  <- DocumentTermMatrix(email_corpus_test, list(dictionary = email_dict))
```

Convert counts to a factor

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

Apply() convert_counts() to columns of train/test data

```
email_train <- apply(email_train, MARGIN = 2, convert_counts)
email_test  <- apply(email_test, MARGIN = 2, convert_counts)
```

Training a model on the data

```
email_classifier <- naiveBayes(email_train, email_raw_train$spam)
#email_classifier
```

Evaluating model performance

Evaluate the predition with the actual data using a crosstable from the gmodels package.

```
email_test_pred <- predict(email_classifier, email_test)

library(gmodels)
CrossTable(email_test_pred, email_raw_test$spam,
           prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,
           dnn = c('predicted', 'actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Col Total |
## |-------------------------|
##
##
```

```
## Total Observations in Table:  1053
##
##
##              | actual
##    predicted |        no |       yes | Row Total |
## -------------|-----------|-----------|-----------|
##           no |       875 |         4 |       879 |
##              |     0.991 |     0.024 |           |
## -------------|-----------|-----------|-----------|
##          yes |         8 |       166 |       174 |
##              |     0.009 |     0.976 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       883 |       170 |      1053 |
##              |     0.839 |     0.161 |           |
## -------------|-----------|-----------|-----------|
##
##
```

As shown in the table only 12/1053 messages were classified incorrectly. This means that the algorithm classified the testing set as spam or ham with approx. 98.86% accuracy. That's impressive. To improve the model, one might tamper with the Laplace value, collect more email data, or try splitting the dataset randomly into training and testing.

I suspect the accuracy would increase as the dataset gets bigger. The more data there is to train the algorithm, the more effective it would be in predicting Spam or Ham.

Improving model performance

```
email_classifier2 <- naiveBayes(email_train, email_raw_train$spam, laplace = 1)
email_test_pred2 <- predict(email_classifier2, email_test)
CrossTable(email_test_pred2, email_raw_test$spam,
           prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,
           dnn = c('predicted', 'actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  1053
##
##
##              | actual
##    predicted |        no |       yes | Row Total |
## -------------|-----------|-----------|-----------|
##           no |       857 |         3 |       860 |
##              |     0.971 |     0.018 |           |
## -------------|-----------|-----------|-----------|
```

```
##          yes |        26 |       167 |       193 |
##              |     0.029 |     0.982 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       883 |       170 |      1053 |
##              |     0.839 |     0.161 |           |
## -------------|-----------|-----------|-----------|
##
##
```

As shown in the table only 29/1053 messages were classified incorrectly. This means that the algorithm classified the testing set as spam or ham with approx. 98% accuracy. That's impressive. To improve the model, one might tamper with the Laplace value, collect more email data, or try splitting the dataset randomly into training and testing.

I suspect the accuracy would increase as the dataset gets bigger. The more data there is to train the algorithm, the more effective it would be in predicting spam or ham.