Data607 - MajorAssignment Project4 - Document Classification

Vinayak Kamath

4/26/2020

Document Classification

Introduction 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 (withheld from the same training dataset). The dataset is taken from site

---

```
library(knitr)
library(tidyr)
library(tidytext)
library(stringr)

# Below libraries have been used for the visualizations used in this rmd:
library(ggplot2)
library(wordcloud)

# Below libraries have been used to help specifically with the document classification work, modeling a
library(tm)
library(SnowballC)
library(caret)
library(gbm)
library(e1071)
```

Libraries

```
# Samples files unzipped into local folders and used as below:

ham.dir="spamham\\easy_ham\\"
ham_files = list.files(path = ham.dir,full.names = TRUE)
```

```r
number_ham<-length(list.files(ham.dir, all.files = "FALSE", full.names = "TRUE"))
print(paste("There is a total of",number_ham,"ham emails"))
```

Data

```
## [1] "There is a total of 2500 ham emails"
```

```r
hard.ham.dir="spamham\\hard_ham\\"
hard_ham_files = list.files(path = hard.ham.dir,full.names = TRUE)

number_hard_ham<-length(list.files(hard.ham.dir, all.files = "FALSE", full.names = "TRUE"))
print(paste("There is a total of",number_hard_ham," hard ham emails"))
```

```
## [1] "There is a total of 250  hard ham emails"
```

```r
spam.dir="spamham\\spam_2\\"
spam_files = list.files(path = spam.dir , full.names = TRUE)

number_spam<-length(list.files(spam.dir, all.files = "FALSE", full.names = "TRUE"))
print(paste("There is a total of",number_spam,"spam emails"))
```

```
## [1] "There is a total of 1396 spam emails"
```

Corpus / Data Cleanup A clean corpus is created for each set with removing numbers, removing punctuation symbols, removing stopwords, removing white spaces and converting the data in each file to lower case.

we have used function here to help with repeating the same steps for each three data set we have.

```r
toVector <- function(file.path) {
  corpus <- file.path %>%
    paste(., list.files(.), sep = "/") %>%
    lapply(readLines) %>%
    VectorSource() %>%
    VCorpus()
  return(corpus)
}
CleanData <- function(corpus) {
    corpus <- corpus %>%
    tm_map(removeNumbers) %>%
    tm_map(removePunctuation) %>%
    tm_map(tolower) %>%
    tm_map(PlainTextDocument) %>%
    tm_map(removeWords, stopwords("en")) %>%
    tm_map(stripWhitespace) %>%
    tm_map(stemDocument)
  return(corpus)
}
```

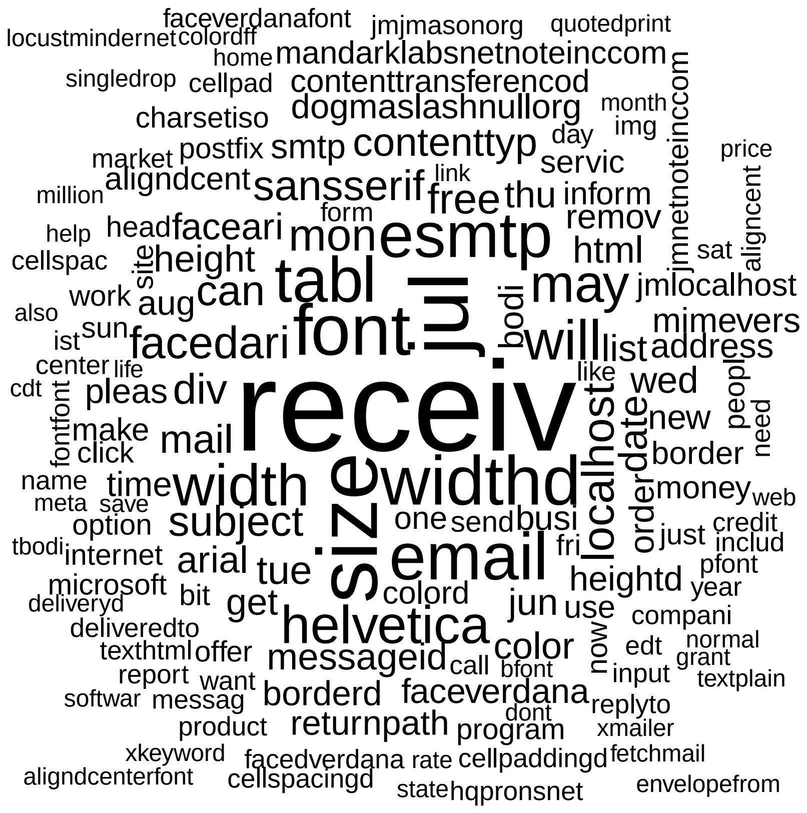
```r
addMetaTag <- function(corpus, tag, value){
  for (i in 1:length(corpus)){
    meta(corpus[[i]], tag) <- value
  }
  return(corpus)
}


# Create ham corpus
hamCorpus <- ham.dir%>%
   toVector %>%
   CleanData  %>%
   addMetaTag(tag = "emails", value = "ham")

# Create hard ham corpus
hardhamCorpus <- hard.ham.dir%>%
   toVector %>%
   CleanData  %>%
   addMetaTag(tag = "emails", value = "hard ham")

# Create spam corpus
spamCorpus <- spam.dir %>%
  toVector %>%
  CleanData %>%
  addMetaTag(tag = "emails", value = "spam")
```

Document Classifier We build a single document classifier using our three data sets. the aim of the classifier to accurately tell the difference between ham, hard ham and spam.

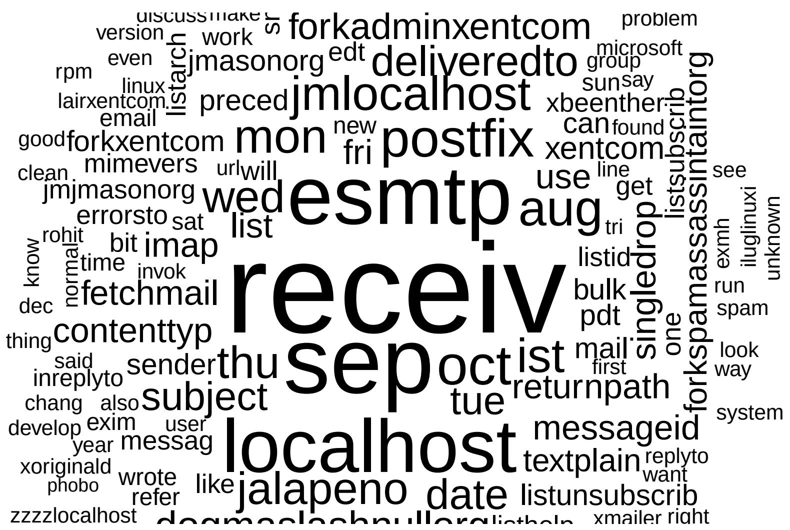
```r
#Ham
hamDF <-as.data.frame(unlist(hamCorpus),stringsAsFactors = FALSE)
hamDF$type <- "ham"
colnames(hamDF) <- c("text","type")

#Hard Ham
hardhamDF <-as.data.frame(unlist(hardhamCorpus),stringsAsFactors = FALSE)
hardhamDF$type <- "hard ham"
colnames(hardhamDF) <- c("text","type")

#Spam
spamDF <-as.data.frame(unlist(spamCorpus),stringsAsFactors = FALSE)
spamDF$type <- "spam"
colnames(spamDF) <- c("text","type")

spam_ham_df <- rbind(hamDF[1:2000,] , hardhamDF[1:2000, ] , spamDF[1:2000,] )
#spam_ham_df <- rbind(hamDF[1:2000,]  , spamDF[1:2000,] )

# Combine all
clean_corpus <- c(spamCorpus, hamCorpus, hardhamCorpus)
#clean_corpus <- c(spamCorpus, hamCorpus)
```

Wordcloud We will use the package and function wordCloud to visualize the words in each and final dataset.

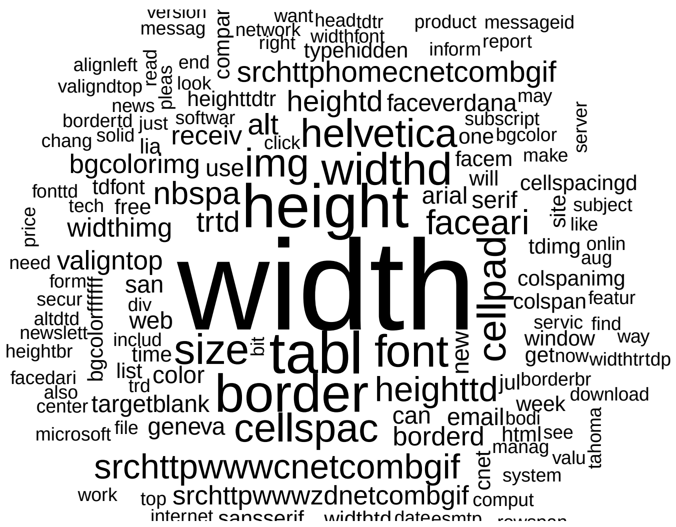
```r
wordcloud(spamCorpus,max.words = 150, random.order = FALSE   )
```



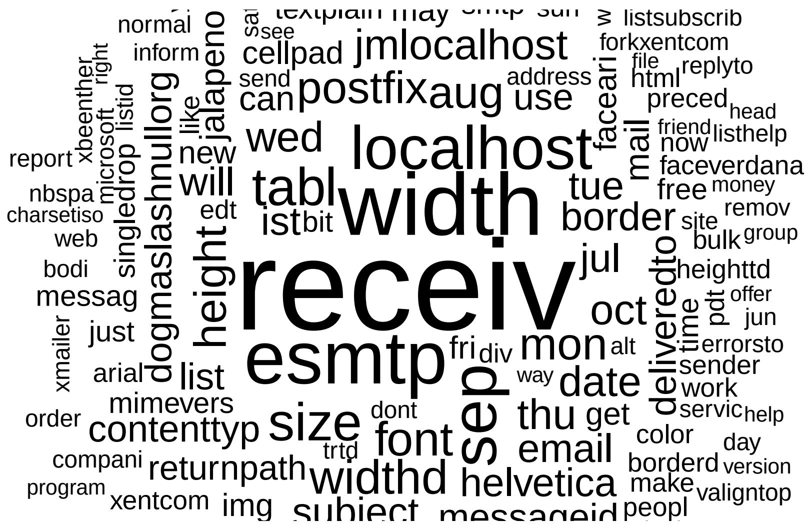
```r
wordcloud(hamCorpus,max.words = 150,   random.order = FALSE )
```



```r
wordcloud(hardhamCorpus,max.words = 150,   random.order = FALSE )
```



```
wordcloud(clean_corpus,max.words = 150,  random.order = FALSE   )
```



Model We will use the below steps to model and determine accuracy of out model:
- Creating DTM (Document Term Matrix) for our final single document classifier.
- Splitting data using caret into 80% as training and 20% as test.
- Creating DTM (Document Term Matrix) for the Training and Testing model set.
- Using NaiveBayes Model.

```
corpus_labels <- unlist(meta(clean_corpus, "emails"))
corpus_dtm <-DocumentTermMatrix(clean_corpus)

set.seed(250)
spam_ham_df$text[spam_ham_df$text==""] <- "NaN"
train_index <- createDataPartition(spam_ham_df$type, p=0.80, list=FALSE)
email_train <- spam_ham_df[train_index,]
email_test <- spam_ham_df[-train_index,]

# Create corpus for training and test data
train_email_corpus <- Corpus(VectorSource(email_train$text))
test_email_corpus <- Corpus(VectorSource(email_test$text))

train_clean_corpus <- tm_map(train_email_corpus ,removeNumbers)
test_clean_corpus <- tm_map(test_email_corpus,removeNumbers)

train_clean_corpus <- tm_map(train_clean_corpus,removePunctuation)
```

```r
test_clean_corpus <- tm_map(test_clean_corpus,removePunctuation)

train_clean_corpus <- tm_map(train_clean_corpus,removeWords,stopwords())
test_clean_corpus  <- tm_map(test_clean_corpus,removeWords, stopwords())

train_clean_corpus<- tm_map(train_clean_corpus,stripWhitespace)
test_clean_corpus<- tm_map(test_clean_corpus,stripWhitespace)

train_email_dtm <- DocumentTermMatrix(train_clean_corpus)
test_email_dtm <- DocumentTermMatrix(test_clean_corpus )

# Here I'm defining input variables 0 and 1 from string to integer
convert_count <- function(x) {
  y <- ifelse(x > 0, 1,0)
  y <- factor(y, levels=c(0,1), labels=c(0,1))
  y
}

train_sms <- apply(train_email_dtm, 2, convert_count)
test_sms <- apply(test_email_dtm, 2, convert_count)

# NaiveBayes Model:
classifier <- naiveBayes(train_sms, factor(email_train$type))
test_pred <- predict(classifier, newdata=test_sms)

table(test_pred, email_test$type)
```

```
##
## test_pred  ham hard ham spam
##   ham      229       43   53
##   hard ham  96      259   86
##   spam      75       98  261
```

Accuracy = hits / total
The hits would be where the email was a Ham and it was predicted as a Ham (229 out of 325); or where the email was a hard Ham and it was predicted as a hard Ham (259 out of 441); or where the email was a spam and it was predicted as a Spam (261 out of 434).

Accuracy = 749/1200 = 0.62

We can see from the above that the accuracy of the overall model is 0.61 ; it does look the model is better predictor for ham then spams