**CUNY MSDS DATA 607 - Project 4 - Text Mining/Classification**

library(tm)

library(RTextTools)

library(knitr)

library(tidyverse)

library(kableExtra)

First we start off with reading in data from the local PC into R. We define two variables, easy\_ham and easy\_spam. A Corpus is a collection of text documents. A VCorpus is a “Volatile” corpus, which means the corpus is stored in memory. DirSource allows us to pull the entire directory into R.

easy\_ham <- VCorpus(DirSource("easy\_ham"))

easy\_spam <- VCorpus(DirSource("easy\_spam"))

We then add meta information to our sets of data: spam, and ham.

meta(easy\_spam, tag = "type") <- "spam"

meta(easy\_ham, tag = "type") <- "ham"

easy\_combined <- c(easy\_spam, easy\_ham)

**Data Cleaning & Tidying:**

The data is filled with inconsistencies - we try to make it easier for the program to understand. We first have to convert all of the characters to a standard format. There were many special characters that were giving issues when trying to convert tolower. We removed numbers, stopwords, punctuation, and white space. All of these aren’t necessary to perform the analysis.

easy\_combined <- tm\_map(easy\_combined, content\_transformer(**function**(x) iconv(x, "UTF-8", sub="byte")))

easy\_combined <- tm\_map(easy\_combined, content\_transformer(tolower))

easy\_combined <- tm\_map(easy\_combined, removeNumbers)

easy\_combined <- tm\_map(easy\_combined, removeWords, stopwords("english"))

easy\_combined <- tm\_map(easy\_combined, removePunctuation)

easy\_combined <- tm\_map(easy\_combined, stripWhitespace)

The data is then arranged into a matrix. According to the text, “Simply put, a term-document matrix is a way to arrange text in matrix form where the rows represent individual terms and columns contain the texts. The cells are filled with counts of how often a particular term appears in a given text.” [1][1]

dtm <- DocumentTermMatrix(easy\_combined)

dtm

## <<DocumentTermMatrix (documents: 3897, terms: 94341)>>

## Non-/sparse entries: 640733/367006144

## Sparsity : 100%

## Maximal term length: 868

## Weighting : term frequency (tf)

The data is then cleaned further by removing sparse words - words that appear infrequently in the dataset (in this case, less than 10 times.).

The primary reason for this operation is computational feasibility. Apart from that, the operation can also be viewed as a safeguard against formatting errors in the data. If a term appears extremely infrequently, it is possible that it contains an error.[1][1]

dtm <- removeSparseTerms(dtm, 1-(10/length(easy\_combined)))

dtm

## <<DocumentTermMatrix (documents: 3897, terms: 6317)>>

## Non-/sparse entries: 484363/24132986

## Sparsity : 98%

## Maximal term length: 73

## Weighting : term frequency (tf)

The data was then analyzed out of courisity, to see what the most frequent terms were. The following kable outputs the results:

dtm2 <- as.matrix(dtm)

frequency <- colSums(dtm2)

frequency <- sort(frequency, decreasing=T)

table\_freq <- head(frequency, 15)

kable(table\_freq, "html", escape = F) %>%

kable\_styling("striped", full\_width = T) %>%

column\_spec(1, bold = T)

|  | **x** |
| --- | --- |
| **received** | 20338 |
| **esmtp** | 11544 |
| **sep** | 9793 |
| **localhost** | 9246 |
| **mon** | 6008 |
| **aug** | 5764 |
| **postfix** | 5627 |
| **oct** | 5256 |
| **jmlocalhost** | 5227 |
| **thu** | 5194 |
| **wed** | 5178 |
| **date** | 4845 |
| **ist** | 4840 |
| **subject** | 4696 |
| **tue** | 4536 |

wf <- data.frame(word=names(frequency), frequency=frequency)

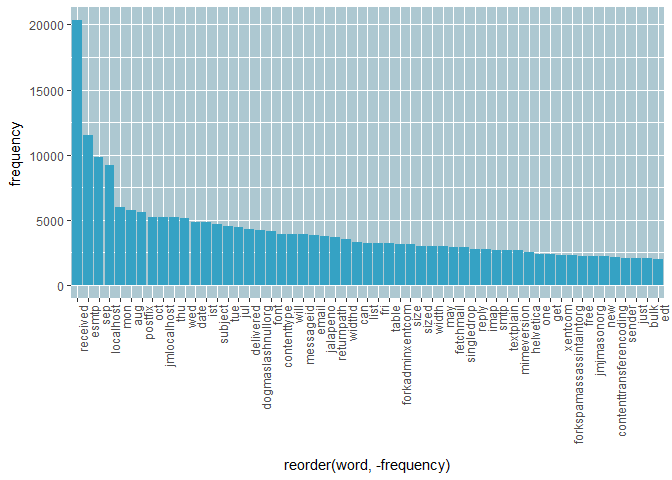
p <- ggplot(subset(wf, frequency>2000), aes(x = reorder(word, -frequency), y = frequency)) +

geom\_bar(stat = "identity", fill='#35a2c4') +

theme(axis.text.x=element\_text(angle=90, hjust=1)) +

theme(panel.background = element\_rect(fill = '#adc8d1'))

p



**Analysis: Models and predictions**

The metadata was then analyzed - we have 2500 emails classified as HAM, and 1397 emails classified as spam.

meta\_type <- as.vector(unlist(meta(easy\_combined)))

meta\_data <- data.frame(type = unlist(meta\_type))

table(meta\_data)

## meta\_data

## ham spam

## 2500 1397

We then created a container using the create\_container() function from RTextTools. In this, we specify which data is going to be part of our train set, and which part is part of the testing set. According to google, a good percentage is 70% train, 30% testing. The virgin attribute = F means that we have labels for all of our documents.

N <- length(meta\_type)

container <- create\_container(dtm,

labels = meta\_type,

trainSize = 1:2727,

testSize = 2728:N,

virgin = F)

The container itself: matrix\_container. It contains a set of objects that are used for the estimation procedures of the supervised learning methods [1][1]:

slotNames(container)

## [1] "training\_matrix" "classification\_matrix" "training\_codes"

## [4] "testing\_codes" "column\_names" "virgin"

We then use the information stored in the container, by using the train\_model() function on the train data:

svm\_model <- train\_model(container, "SVM")

tree\_model <- train\_model(container, "TREE")

maxent\_model <- train\_model(container, "MAXENT")

We then use our model to estimate if an email in our test dataset is spam or ham.

svm\_out <- classify\_model(container, svm\_model)

tree\_out <- classify\_model(container, tree\_model)

maxent\_out <- classify\_model(container, maxent\_model)

Looking at the outcome: all three models were combined into one dataframe where the labels and estimat eof the probability of classification are shown.

test\_out <- data.frame(head(svm\_out), head(tree\_out), head(maxent\_out) )

kable(test\_out, "html", escape = F) %>%

kable\_styling("striped", full\_width = F) %>%

column\_spec(1, bold = T)

| **SVM\_LABEL** | **SVM\_PROB** | **TREE\_LABEL** | **TREE\_PROB** | **MAXENTROPY\_LABEL** | **MAXENTROPY\_PROB** |
| --- | --- | --- | --- | --- | --- |
| **ham** | 0.9893930 | ham | 0.998295 | ham | 0.9999784 |
| **ham** | 0.9962258 | ham | 0.998295 | ham | 0.9999772 |
| **ham** | 0.8823817 | spam | 0.996136 | spam | 0.6678205 |
| **ham** | 0.8183342 | spam | 0.996136 | ham | 0.5153088 |
| **spam** | 0.5064285 | spam | 0.996136 | ham | 0.9602845 |
| **ham** | 0.6308979 | spam | 0.996136 | ham | 0.9539801 |

Since we’re using supervised learning, our models know the correct labels. We can use this to see exactly how correct the algorithm was in correctly classifying the documents.

labels\_out <- data.frame(

correct\_label = meta\_type[2728:N],

svm = as.character(svm\_out[,1]),

tree = as.character(tree\_out[,1]),

maxent = as.character(maxent\_out[,1]),

stringsAsFactors = F)

*#SVM performance*

table(labels\_out[,1] == labels\_out[,2])

##

## FALSE TRUE

## 203 967

prop.table(table(labels\_out[,1] == labels\_out[,2]))

##

## FALSE TRUE

## 0.1735043 0.8264957

*#Random forest performance*

table(labels\_out[,1] == labels\_out[,3])

##

## FALSE TRUE

## 897 273

prop.table(table(labels\_out[,1] == labels\_out[,3]))

##

## FALSE TRUE

## 0.7666667 0.2333333

*#Maximum entropy performance*

table(labels\_out[,1] == labels\_out[,4])

##

## FALSE TRUE

## 93 1077

prop.table(table(labels\_out[,1] == labels\_out[,4]))

##

## FALSE TRUE

## 0.07948718 0.92051282

dfdata <- data.frame(table(labels\_out[,1] == labels\_out[,2]),

table(labels\_out[,1] == labels\_out[,3]),

table(labels\_out[,1] == labels\_out[,4])

)

colnames(dfdata) <- c("SVM","Freq", "Random Forest", "Freq", "Max Entropy", "Freq")

kable(dfdata, "html", escape = F) %>%

kable\_styling("striped", full\_width = F) %>%

column\_spec(1, bold = T)

| **SVM** | **Freq** | **Random Forest** | **Freq** | **Max Entropy** | **Freq** |
| --- | --- | --- | --- | --- | --- |
| **FALSE** | 203 | FALSE | 897 | FALSE | 93 |
| **TRUE** | 967 | TRUE | 273 | TRUE | 1077 |

**Conclusions:**

Looking at the results, we can see that the Maximum Entropy was the best classifier, followed by the SVM. The worst classifier was the Random Forest.

**References:**

[1][1]Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining