DATA 607 - Project 4 - Document Classifier (Using SVM Model)

Project4 Team: Banu Boopalan, Samuel Kigamba, James Mundy, Alain T Kuiete

Project 4: PART 2 CODE (Submitted via this separate link. A separate link using RPUBS via Naive Bayes model will be submitted by our team)

Our Project Team 4 above (Banu Boopalan, Samuel Kigamba, James Mundy, Alain T Kuiete), we will submit 2 separate RPUB documents. The 2nd document link to RPUBS, we have performed data transformations, exploratory data analysis, visualizations using wordclouds, frequency plots on words, and performed SVM model and reported the Confusion Matrix results for the SVM model. We tried to plot the model using plot but we were not successful in representing a way to plot the model, The support vector are high range so we have to dive deeper into how to represent and plot the model through plot or Kernlab pacakge or Kernfit. Within the model we are able to create document term matrix and term document matrix, segment the train and test data and then run the model to report summary model. The SVM reported an accuracy for each of our teammates will be different as we are reading in our own files from the directory. The SVM reported higher accuracy than the Naive Bayes upon first review.

Collaboration via POWERPOINT, GITHUB, GOTO MEETING along with weekly meetings on Tuesday, Friday.

Our Approach

We have utilized SVM model in this project4 code (Our first code that produced uses . Our approach for this project follows:

1. Load required Libraies
2. Get data from spamassassin website
3. Build a Build a Document Corpus
4. Plot Sentiment Analysis and Wordcloud of Corpus
5. Create Document-Term Matrix
6. Clean-up and Normalize Data
7. Create Training Set
8. Build/Train SVM
9. Review Results - Using Confusion Matrix Satistics, Use Radial and Linear type model

```
#loading required Libraries
library(caret)
library(tidyverse)
library(tidyr)
library(dplyr)
library(stringr)
library(tidytext)
library(wordcloud)
library(broom)
library(tm)
```

```
library(e1071)
library(quanteda)
library(ggplot2)
```

Get Data

The data for this project was obtained from:

https://spamassassin.apache.org/old/publiccorpus/

Ham and spam files were extracted and stored in a data folder on a local drive.

Build a Corpus

Next we build the corpus after completing some transforms: convert to plain doucment, remove stopwords, remove punctuation, remove numbers, remove whitespace, etc.

```
create_corpus <- function(dir, label){
  corpus <- VCorpus(DirSource(dir)) %>%
    tm_map(PlainTextDocument)  %>%
    tm_map(content_transformer(tolower)) %>% #
    tm_map(removeWords, stopwords("SMART")) %>%
    tm_map(removePunctuation) %>% #
    tm_map(removeNumbers) %>% #
    tm_map(stripWhitespace) %>% #
    tm_map(stemDocument) #
  meta(corpus, "LABEL") <- label
  return(corpus)
}
corpus<- c(create_corpus("spam_2", "Spam"), create_corpus("easy_ham", "Ham"))
```

Build a Document-Term Matrix

Now we use the corpus to construct a document term matrix and show wordcloud using Bing Lexicon

```
dtm <- DocumentTermMatrix(corpus)
dtm
```

```
## <<DocumentTermMatrix (documents: 5442, terms: 84242)>>
## Non-/sparse entries: 746748/457698216
## Sparsity           : 100%
## Maximal term length: 855
## Weighting          : term frequency (tf)
```

```
dtm_td <- tidy(dtm)
dtm_td
```

```
## # A tibble: 746,748 x 3
##    document     term                count
##    <chr>        <chr>               <dbl>
##  1 character(0) aafcf                   1
##  2 character(0) abandon                 1
##  3 character(0) accept                  1
##  4 character(0) address                 1
##  5 character(0) agre                    2
##  6 character(0) altern                  1
##  7 character(0) altra                   1
##  8 character(0) apolog                  1
##  9 character(0) aug                     8
## 10 character(0) authenticationwarn      1
## # ... with 746,738 more rows
```

```
#slice sentiments of 1000 rows
dtm_sentiments <- slice(dtm_td , 1:5000) %>% inner_join(get_sentiments("bing"), by = c(term = "word"))
dtm_sentiments
```

```
## # A tibble: 267 x 4
##    document     term    count sentiment
##    <chr>        <chr>   <dbl> <chr>
##  1 character(0) betray      1 negative
##  2 character(0) burn        1 negative
##  3 character(0) easier      1 positive
##  4 character(0) error       1 negative
##  5 character(0) fail        1 negative
##  6 character(0) fatal       1 negative
##  7 character(0) flaw        1 negative
##  8 character(0) free        3 positive
##  9 character(0) good        1 positive
## 10 character(0) honest      1 positive
## # ... with 257 more rows
```

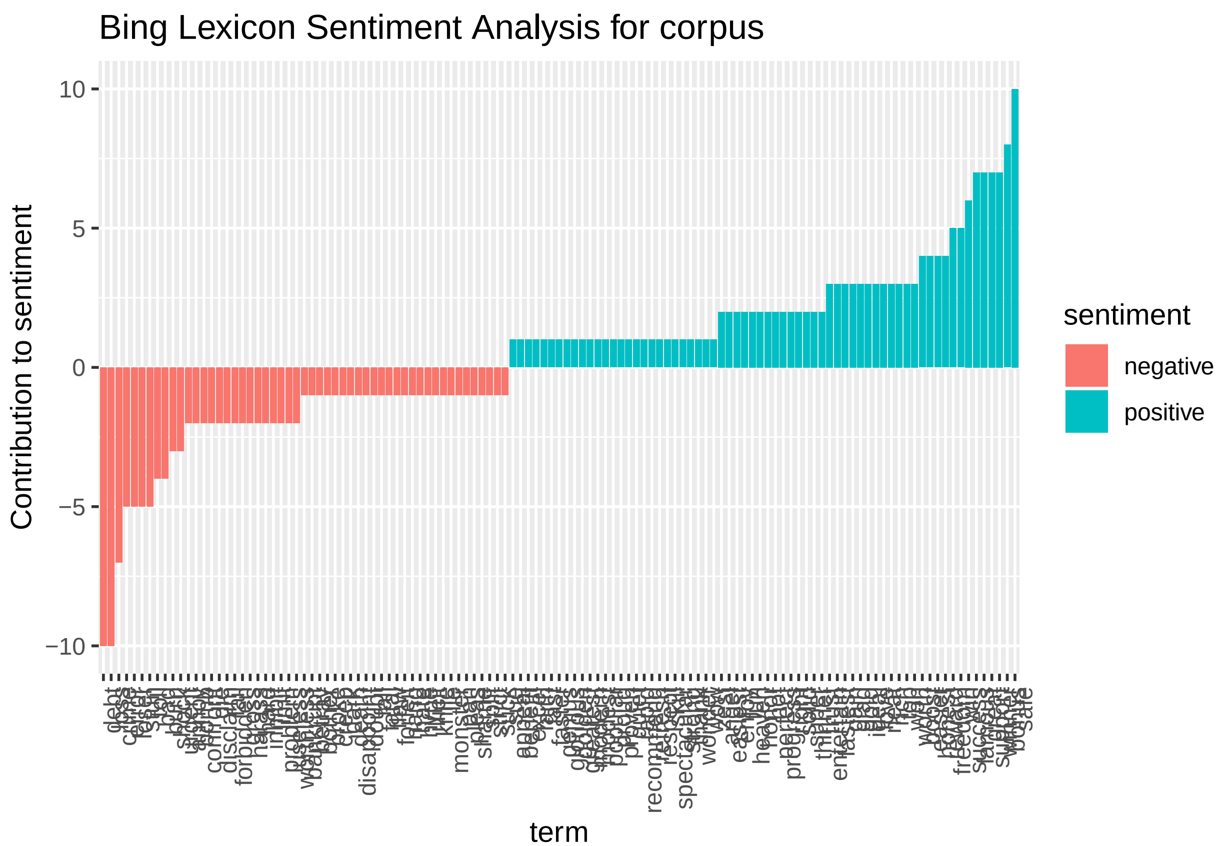
```
#unnext tokens to look at words
slice_words <- tidy(corpus) %>%
  unnest_tokens(word, text)
  slice_words <- slice(slice_words, 1:9000)

  library(broom)
models <- count(slice_words, word) %>% inner_join(get_sentiments("bing"), by = c(word = "word"))

str(models)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    120 obs. of  3 variables:
##  $ word     : chr  "bad" "bankrupt" "benefit" "betray" ...
##  $ n        : int  4 1 1 1 8 4 1 1 2 2 ...
##  $ sentiment: chr  "negative" "negative" "positive" "negative" ...
```

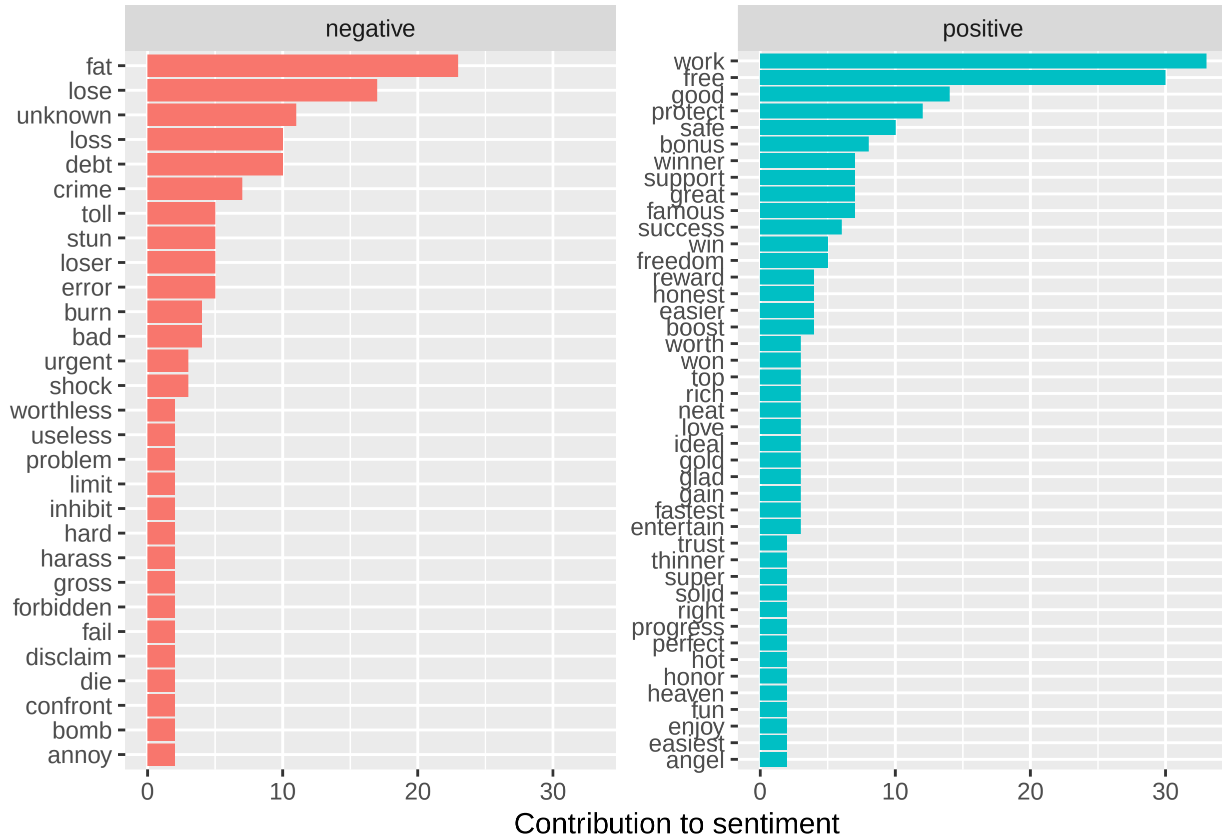
```
dtm_sentiments %>%
  count(document, sentiment, wt = count) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative) %>%
  arrange(sentiment)
```

```
## # A tibble: 1 x 4
##   document     negative positive sentiment
##   <chr>           <dbl>    <dbl>     <dbl>
## 1 character(0)      169      258        89
```

```r
dtm_sentiments %>%
  count(sentiment, term, wt = count) %>%
  filter(n <= 10) %>%
  mutate(n = ifelse(sentiment == "negative", -n, n)) %>%
  mutate(term = reorder(term, n)) %>%
  ggplot(aes(term, n, fill = sentiment)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Contribution to sentiment") + ggtitle("Bing Lexicon Sentiment Analysis for corpus")
```




```r
dtm_sentiments %>%
  count(sentiment, term, wt = count) %>%
  top_n(50) %>%
  ungroup() %>%
  mutate(term = reorder(term, n)) %>%
  ggplot(aes(term, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Contribution to sentiment",
       x = NULL) +
  coord_flip()
```




```
#layout(matrix(c(1, 2), nrow=2), heights=c(1, 4))
#par(mar=rep(0, 4))
plot.new()
text(x=0.5, y=0.5, "Wordcloud using Bing Lexicon for corpus")
```

Wordcloud using Bing Lexicon for corpus

```
wordcloud(words = dtm_sentiments$term, freq = dtm_sentiments$count, min.freq = 1,max.words=200, random.
```

Reduce Sparseness and Normalize

We reduce sparness here by only keeping words that are found more than n times. We tried training the model with diffent values for n but found that 15 produced the best results .

```
#Only Keep Words found in at least 15 documents

min_docs <- 15
dtm <- removeSparseTerms(dtm, 1 - (min_docs / length(corpus)))

model_data <- as.matrix(dtm)
str(model_data)
```

```
##  num [1:5442, 1:4735] 0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ Docs : chr [1:5442] "character(0)" "character(0)" "character(0)" "character(0)" ...
##   ..$ Terms: chr [1:4735] "aaa" "aaf" "aaron" "aaronsw" ...
```

```
words <- rowSums(model_data)
model_data <- model_data / words
model_data <- data.frame(model_data)
model_data <- cbind(meta(corpus), model_data) %>%
  mutate(LABEL = as.factor(LABEL))
```

Create a Training Set

We now divide the data into training and test sets. Seventy-five percent of the data was used for training.

```
set.seed(12345)
in_training_set <- createDataPartition(model_data$LABEL, p = 0.75,  list = FALSE)
training_data <- model_data[in_training_set, ]
testing_data <- model_data[-in_training_set, ]
#head(training_data,n=1)
nrow(testing_data)
```

```
## [1] 1360
```

Build / Train SVM

We use the training data to build a SVM model that predicts if a message is spam or ham.

```
#This outputs Radial kernal type
model <- svm(LABEL ~ ., data = training_data)
model
```

```
##
## Call:
## svm(formula = LABEL ~ ., data = training_data)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  1933
```

Use Kernal Linear type to see results

```
#This outputs linear kernal type
model1 <- svm(LABEL ~ ., data = training_data, kernel = "linear", scale = FALSE)
model1
```

```
##
## Call:
## svm(formula = LABEL ~ ., data = training_data, kernel = "linear",
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  1636
```

Review Results

Finally, we test our model to see how accurate it is. Predictions is Radial type and Predictions1 is Linear type.

```
predictions <- testing_data %>%
  select(-LABEL) %>%
  predict(model, .)

predictions1 <- testing_data %>%
  select(-LABEL) %>%
  predict(model1, .)

#radial
table(Prediction = predictions ,Truth = testing_data$LABEL)
```

```
##           Truth
## Prediction  Ham Spam
##       Ham  1011   29
##       Spam    0  320
```

```
#linear
table(Prediction = predictions1 ,Truth = testing_data$LABEL)
```

```
##           Truth
## Prediction  Ham Spam
##       Ham  1011   42
##       Spam    0  307
```

The confusion matrix below indicates that with n = 15, only 8 emails were misclassified. This equates to approximately 99% accuracy.

```
#install.packages('kableExtra')
library(kableExtra)
table(predictions, testing_data$LABEL) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "responsive"))
```

	Ham	Spam
Ham	1011	29
Spam	0	320

```
#Radial Classification
confMatrix1 <- confusionMatrix(predictions, testing_data$LABEL)
confMatrix1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  Ham Spam
```

```
##       Ham  1011   29
##       Spam    0  320
##
##                Accuracy : 0.9787
##                  95% CI : (0.9695, 0.9857)
##     No Information Rate : 0.7434
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9425
##
##  Mcnemar's Test P-Value : 1.999e-07
##
##             Sensitivity : 1.0000
##             Specificity : 0.9169
##          Pos Pred Value : 0.9721
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7434
##          Detection Rate : 0.7434
##    Detection Prevalence : 0.7647
##       Balanced Accuracy : 0.9585
##
##        'Positive' Class : Ham
##
```

```
#Linear Classification
confMatrix2 <- confusionMatrix(predictions1, testing_data$LABEL)
confMatrix2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  Ham Spam
##       Ham  1011   42
##       Spam    0  307
##
##                Accuracy : 0.9691
##                  95% CI : (0.9585, 0.9777)
##     No Information Rate : 0.7434
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9157
##
##  Mcnemar's Test P-Value : 2.509e-10
##
##             Sensitivity : 1.0000
##             Specificity : 0.8797
##          Pos Pred Value : 0.9601
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7434
##          Detection Rate : 0.7434
##    Detection Prevalence : 0.7743
##       Balanced Accuracy : 0.9398
##
##        'Positive' Class : Ham
```

##