Project #4-Text and Document Classification

Anthony Conrardy

2024-04-14

Initial Library Identification and File Folder Load

For the purposes of this project, we shall be using the Support Vector Machines (SVM) process utilizing the e1071 package. We shall also be using the Text Mining (tm) package to structure and build out a corpus for training and use. First we will load in the spam and ham emails messages in from the GitHub Repo located at https://github.com/Aconrard/DATA607/tree/main/Project_4. The files are loaded locally in my working directory. All data was obtained from https://spamassassin.apache.org/old/publiccorpus/

Due to processing memory limitations, we can only load and process a maximum of three hundred (300) email messages, 150 spam and 150 ham. This is done by extracting the files into the directory and then physically moving 150 messages to each directory. We will also create a list of spam and ham email messages for further processing.

```
# Identify directory holding the spam and ham files
spam_dr <- "C:/Users/para2/Documents/R_Working_Directory/Project #4/20050311_spam_2/spam_2"
ham_dr <-  "C:/Users/para2/Documents/R_Working_Directory/Project #4/20030228_easy_ham_2/easy_ham_2"

# Get list of files in the directory
file_list_spam <- list.files(spam_dr, full.names = TRUE)
file_list_ham <- list.files(ham_dr, full.names = TRUE)
```

Pre-Processing

In this section, we will pre-process the email messages prior to use. The lists of both the ham and spam email message text files will be read into each data frame, and then they will be pre-processed by make all characters lower case and removing numbers, punctuation, stopwords, and white spaces. Initial attempts identified a character coding issue that was stopping the operation, which resulted in use specifying a “latin1” encoding argument.

```
# Read the text into R
spam_text <- sapply(file_list_spam, readLines, simplify = TRUE, encoding = "latin1")
ham_text <- sapply(file_list_ham, readLines, simplify = TRUE, encoding = "latin1")

# Pre-process Spam
spam_text1 <- spam_text |> tolower() |> removeNumbers() |> removePunctuation()
spam_text1 <- removeWords(spam_text1, stopwords("en")) |> stripWhitespace()
# Pre-process Ham
ham_text1 <- ham_text |> tolower() |> removeNumbers() |> removePunctuation()
ham_text1 <- removeWords(ham_text1, stopwords("en")) |> stripWhitespace()
```

Creating the Corpus

In this section we shall combine the files into a single corpus matrix and then placing a label identifying the particular observation as spam. The column identifying whether it is spam or not, is the "spam_rate" variable that is either "1" for spam, or "0" for ham. The matrix will contain a total of 300 observations combining the data from both the spam and ham data sets.

```
corpus_combined <- Corpus((VectorSource(c(spam_text1, ham_text1))))
dtm <- DocumentTermMatrix(corpus_combined)
labels <- as.integer(c(rep("1", length(file_list_spam)), rep("0", length(file_list_ham))))
data_set <- data.frame(as.matrix(dtm), spam_rate=labels)
```

Split the Data

In this section we will randomly select observations for our training and test data sets. We will randomly select 70% of the observations for the training data set, and the remainder will the test data set on which we evaluate the model. This means our training data set will have 210 observations, while the test data set will have 90 observations. We set a seed so as to keep consistency of the data being used from run to run, but it can be removed at any time to identify a new training set by either using the existing developed corpus, or by swapping the ham and spam email text files in the local directories.

```
set.seed(999)
index_sel <- sample(1:nrow(data_set), 0.7 * nrow(data_set))
train_data <- data_set[index_sel, ]
test_data <- data_set[-index_sel, ]
```

Train Support Vector Machine (SVM) Classifier

In this section we will train our classifier to identify spam from ham emails. We will use the Support Vector Machines (SVM) function to carry out a general regression with the spam_rate variable being the dependent variable, and the remaining 16,027 as independent variables. This action did take some time, and was the major contributing factor in limiting the number of text files that could be used at any one time. The resulting model results was over 1GB in size.

```
#train_data$label <- as.factor(train_data$label)
model <- svm(spam_rate~ ., data = train_data, kernel = "linear", cost = 1)
```

Evaluate Classifier Against Original Data

In this section we evaluate the performance of the model learning to identify spam and ham emails. This was a two part process. In the first part, we ran a prediction using the SVM model we trained in the above section with the test data set we created. The results were then placed in a data frame so we could visualize the predicted versus the actual results. We then calculated the mean predicted value (MPV) of the spam emails with confidence interval of 99%. This allows us to identify the lower limit of the interval so as to set a benchmark for the model performance. For this data set, the lower_CI for the MPV was determined to be 0.873638469.

```
# Prediction Model Run
predictions <- predict(model, newdata = test_data)

# Identifying true status of emails
```

```r
true_labels <- test_data$spam_rate

# Matching to True to Predicted Values.
results <- data.frame(model_value = predictions, Actual = true_labels)
tibble(results)
```

```
## # A tibble: 90 x 2
##    model_value Actual
##          <dbl>  <int>
##  1       0.175      1
##  2       0.958      1
##  3       1.22       1
##  4       1.00       1
##  5       0.919      1
##  6       0.919      1
##  7       0.994      1
##  8       1.12       1
##  9       0.998      1
## 10       0.994      1
## # i 80 more rows
```

```r
# Calculate mean predicted value and 99% CI
mpv <- results |> group_by(Actual) |>
  summarise(mean_predicted_value = mean(model_value),
            lower_ci = t.test(model_value, conf.level = 0.99)$conf.int[1],
            upper_ci = t.test(model_value, conf.level = 0.99)$conf.int[2])
tibble(mpv)
```

```
## # A tibble: 2 x 4
##   Actual mean_predicted_value lower_ci upper_ci
##    <int>                <dbl>    <dbl>    <dbl>
## 1      0               0.0436  0.00694   0.0802
## 2      1               0.930   0.874     0.986
```

For second part, we re-labeled the results of the previous run with any prediction having a value of greater than or equal to 0.874 to be identified as spam (1), and anything less than that would be identified as ham. We then calculated the accuracy, precision, recall,and f1_score of the model and reported in a created data frame.

```r
# Re-assign 1 for spam results >= 0.874
results_p <- results |> mutate(Predicted = ifelse(model_value >= 0.874, 1, 0))

# Contingency Table Review
MLmetrics::ConfusionMatrix(results_p$Actual, results_p$Predicted)
```

```
##        y_pred
## y_true  0  1
##      0 47  5
##      1  0 38
```

```r
# Model Performance Metrics
accuracy <- results_p |> summarise(accuracy = sum(Actual == Predicted)/n())
precision <- results_p |> filter(Predicted == 1) |> summarise(precision = sum(Actual == 1)/n())
recall <- results_p |> filter(Actual == 1) |> summarise(recall = sum(Predicted == 1)/ n())
f1_score <- results_p |> summarise(f1_score =(2*precision*recall)/(precision + recall))

# Unlisting data frames
accuracy <- unlist(accuracy)
precision <- unlist(precision)
recall <- unlist(recall)
f1_score <- unlist(f1_score)

#Creating new data frame for reporting.
results <- data.frame(
  Model_Performance_Metrics = c("Accuracy", "Precision", "Recall", "F1-Score"),
  Values = c(accuracy, precision, recall, f1_score))

tibble(results)
```

```
## # A tibble: 4 x 2
##   Model_Performance_Metrics Values
##   <chr>                     <dbl>
## 1 Accuracy                  0.944
## 2 Precision                 1
## 3 Recall                    0.884
## 4 F1-Score                  0.938
```

Harder Dataset Test

In this section we test the classifier against the hard ham data set obtained from the same location as the previously used data. As before, due to processing limitations, we will place 150 spam and 150 harder ham email text files into each folder, for a total of 300 text files.

```r
# Identify directory holding the spam and ham files
spam_dra <- "C:/Users/para2/Documents/R_Working_Directory/Project #4/spam_2a"
ham_dra <-  "C:/Users/para2/Documents/R_Working_Directory/Project #4/Hard Ham/hard_ham"

# Get list of files in the directory
file_list_spama <- list.files(spam_dra, full.names = TRUE)
file_list_hama <- list.files(ham_dra, full.names = TRUE)

# Read the text into R
spam_texta <- sapply(file_list_spama, readLines, simplify = TRUE, encoding = "latin1")
ham_texta <- sapply(file_list_hama, readLines, simplify = TRUE, encoding = "latin1")

# Pre-process New Spam
spam_text1a <- spam_texta |> tolower() |> removeNumbers() |> removePunctuation()
spam_text1a <- removeWords(spam_text1a, stopwords("en")) |> stripWhitespace()

# Pre-process New Harder Ham
ham_text1a <- ham_texta |> tolower() |> removeNumbers() |> removePunctuation()
ham_text1a <- removeWords(ham_text1a, stopwords("en")) |> stripWhitespace()
```

New Corpus and New Datase

In this section, as was previously done, we will create a new data matrix containing 300 email text files, 150 spam and now 150 of the harder ham emails. However, unlike the previous data set, we do not have to train the model as we will be using the previous model to predict response in this new data set. Therefore, we will be asking the model to classify the email text files for 200 randomly selected files from the newly created document term matrix. We will once again set actual identification of the file to be "1" for spam and "0" for ham. The dictionary of terms from the previously trained model will be assigned to this DTM.

```
# Create new corpus and use the terms from the previous trained model as controls.
corpus_combineda <- Corpus((VectorSource(c(spam_text1a, ham_text1a))))
dtma <- DocumentTermMatrix(corpus_combineda, control = list(dictionary = Terms(dtm)))
labelsa <- c(rep("1", length(file_list_spama)), rep("0", length(file_list_hama)))
data_seta <- data.frame(as.matrix(dtma), spam_rate=labelsa)

# Randomly select 200 cases
set.seed(998)
index_sel_rnd <- sample(1:nrow(data_seta), 200)
data_seta_rnd <- data_seta[index_sel_rnd, ]
```

Test Classifier Against New Data Set

As with the previous evaluation data set, we will predict the classification of the emails as either spam or ham. We will calculate the MPV for the predictions and calculate the 99% CI for the results. The lower CI has found to be 0.9149534 and use that as the lower limit threshold to identify an email as spam. Therefore, any prediction greater than or equal to 0.915 will be classified as spam (1), and anything below will be classified as ham (0).

```
predictions_rnd <- predict(model, newdata = data_seta_rnd)
true_labels_rnd <- data_seta_rnd$spam_rate
results_rnd <- data.frame(model_value = predictions_rnd, Actual = true_labels_rnd)

mpv_rnd <- results_rnd |> group_by(Actual) |>
  summarise(mean_predicted_value = mean(model_value),
            lower_ci = t.test(model_value, conf.level = 0.99)$conf.int[1],
            upper_ci = t.test(model_value, conf.level = 0.99)$conf.int[2])
tibble(mpv_rnd)
```

```
## # A tibble: 2 x 4
##   Actual mean_predicted_value lower_ci upper_ci
##   <chr>                 <dbl>    <dbl>    <dbl>
## 1 0                     0.784    0.742    0.826
## 2 1                     0.946    0.915    0.977
```

```
# Re-assign 1 for spam results >= 0.915
results_p_rnd <- results_rnd |> mutate(Predicted = ifelse(model_value >= 0.915, 1, 0))

# Contingency Table Review
MLmetrics::ConfusionMatrix(results_p_rnd$Actual, results_p_rnd$Predicted)
```

```
##       y_pred
## y_true  0  1
```

```
##      0 79 24
##      1 21 76
```

```
accuracy <- results_p_rnd |> summarise(accuracy = sum(Actual == Predicted)/n())
precision <- results_p_rnd |> filter(Predicted==1) |> summarise(precision=sum(Actual==1)/n())
recall <- results_p_rnd |> filter(Actual == 1) |> summarise(recall = sum(Predicted == 1)/ n())
f1_score <- results_p_rnd |> summarise(f1_score =(2*precision*recall)/(precision + recall))

accuracy_rnd <- unlist(accuracy)
precision_rnd <- unlist(precision)
recall_rnd <- unlist(recall)
f1_score_rnd <- unlist(f1_score)

results_rnd <- data.frame(
  Model_Performance_Metrics = c("Accuracy", "Precision", "Recall", "F1-Score"),
  Values = c(accuracy_rnd, precision_rnd, recall_rnd, f1_score_rnd))
tibble(results_rnd)
```

```
## # A tibble: 4 x 2
##   Model_Performance_Metrics Values
##   <chr>                     <dbl>
## 1 Accuracy                  0.775
## 2 Precision                 0.784
## 3 Recall                    0.76
## 4 F1-Score                  0.772
```

Results and Conclusion

While our initially trained model appeared to perform moderately well, using the same trained model with a more difficult email text file source indicated it did not perform as well in comparison. Our initial prediction only misclassified about 6% of the files (n=5), while the second prediction on the harder data set resulted in about 23% of the files being misclassified (n=45). 24 emails were excluded as being spam when they were ham, and 21 emails were identified as ham when they were spam. The performance metrics for the second prediction run against the harder ham email files clearly indicates that there is room for improvement (all below 0.80). While we could adjust the threshold for spam identification above 0.915, it would probably be a better alternative to have a more heterogeneous data set including all levels of ham emails to identify the proper words that could specifically identify an email as spam or ham. Also, we should analyze the observations to see if there are any words that are causing an imbalance in the model.