Project 4 - Document Classification

Nicholas Kunze

2024-04-13

Overview

In this project I will be using the [Naive] [Bayes] [classifier] machine learning algorithm to make predictions on whether an email is spam or not. IBM, the company I work at, has a lot of education around ML algorithms and I leveraged that here in the project as well as other sources. I've also previously worked with a team on something similar (statistical classification) using a TensorFlow model to predict credit card fraud, the public version of which is hosted [here].

First, we'll need to ingest our data and prepare it; preparing it will involve creating the 'metadata' around each email by creating our corpus of terms used in the email, and create our email term frequency matrix. The important part here is that the data we are using is already marked as fraudulent, spam, or not, ham. We then split this data into two groups, one for our model training and the other to test how accurate our trained model is. I usually split these into groups of sized 90/10 or 95/5.

Loading Email Data

I'm going to directly pull the tar files containing our ham and spam emails, parse the contents into data frames, then break them up into our training and testing data.

```
tmpdir <- tempdir()

base_url <- "https://spamassassin.apache.org/old/publiccorpus/"
spam_url <- paste0(base_url,"20021010_spam.tar.bz2")
ham_url <- paste0(base_url,"20021010_easy_ham.tar.bz2")

spamTar <- basename(spam_url)
hamTar <- basename(ham_url)

if(!file.exists("20021010_spam.tar.bz2"))
  download.file(spam_url, spamTar)
if(!file.exists("20021010_easy_ham.tar.bz2"))
  download.file(ham_url, hamTar)

untar(spamTar, exdir = tmpdir)
untar(hamTar, exdir = tmpdir)
```

```
getFileText = function(uri, output){
  text <-readLines(uri)
  # return email text
  return(paste(text, collapse="\n"))
}
```

```
getFrom = function(email, output){
  return(str_extract(email,"(?<=From: ).*?(?=\\n)"))
}

getTo = function(email, output){
  return(str_extract(email,"(?<=To: ).*?(?=\\s)"))
}

getCType = function(email, output){
  return(str_extract(email,"(?<=Content-Type: ).*?(?=\\s)"))
}

# Need to convert body text to UTF-8
getBody = function(email, output){
  return(substr(email,str_locate(email,"\\n\\n")[,2],nchar(iconv(email, from = "", to = "UTF8"))))
}
```

```
hamFiles <- list.files(path = paste0(tmpdir,"/easy_ham/"), full.names = TRUE)
hamText <- apply(array(hamFiles), 1, getFileText)
spamFiles <- list.files(path = paste0(tmpdir,"/spam/"), full.names = TRUE)
spamText <- apply(array(spamFiles), 1, getFileText)
```

```
## Warning in readLines(uri): incomplete final line found on
## 'C:\Users\Nick\AppData\Local\Temp\Rtmp0aSkqy/spam/0143.260a940290dcb61f9327b224a368d4af'
```

```
from <- apply(array(hamText), 1, getFrom)
to <- apply(array(hamText), 1, getTo)
cType <- apply(array(hamText), 1, getCType)
body <- apply(array(hamText), 1, getBody)

ham <- data.frame(from,
                  to,
                  cType,
                  body)

from <- apply(array(spamText), 1, getFrom)
to <- apply(array(spamText), 1, getTo)
cType <- apply(array(spamText), 1, getCType)
body <- apply(array(spamText), 1, getBody)

spam <- data.frame(from,
                  to,
                  cType,
                  body)


unshuffled <- rbind(ham %>% mutate(type = "ham"),
  spam %>% mutate(type = "spam"))

# Shuffle emails to allow us to pull random ham vs spam during test/train separation
emails <- unshuffled[sample(nrow(unshuffled), nrow(unshuffled)),]

head(emails)
```

```
##                                                from
## 467                "John Hall" <johnhall@evergo.net>
## 2058                     fark <rssfeeds@example.com>
## 1381 "" Angles " Puglisi" <angles@aminvestments.com>
## 634              "Bill Stoddard" <bill@wstoddard.com>
## 2554          "Slim Down" <taylor@s3.serveimage.com>
## 376                "Rob Shavell" <rob@mobiusvc.com>
##                                  to            cType
## 467     yyyy@localhost.example.com      text/plain;
## 2058    yyyy@localhost.example.com      text/plain;
## 1381 yyyy@localhost.netnoteinc.com multipart/mixed;
## 634     yyyy@localhost.example.com      text/plain;
## 2554    zzzz@localhost.example.com      text/plain;
## 376  yyyy@localhost.netnoteinc.com      text/plain;
##
## 467
## 2058
## 1381
## 634
## 2554
## 376  \nright Mike,\n\ni will agree to disagree but i take your comments to heart.  my opinion is\nonl
##      type
## 467   ham
## 2058  ham
## 1381  ham
## 634   ham
## 2554 spam
## 376   ham
```

Model Input Prep

Now that we have our emails loaded and tagged as ham or spam, we can build our corpus and use it to build our term frequency matrix (document term matrix). Using this frequency matrix, we can 'train' our Naive Bayes model to predict the ham or spam classification based on specific words that are or are not used. We're able to do this as we already know which emails are ham or spam based on the tag. We can then test our model using our remaining 10% of emails that we did not pass in to our model for training.

```
corpus <- VCorpus(x = VectorSource(emails$body))

# Clean corpuses
corpus <- tm_map(x = corpus, FUN = removeNumbers)
corpus <- tm_map(x = corpus, content_transformer(tolower))
corpus <- tm_map(x = corpus, FUN = removePunctuation)
corpus <- tm_map(x = corpus, FUN = removeWords, stopwords())
corpus <- tm_map(x = corpus, FUN = stripWhitespace)
corpus <- tm_map(x = corpus, FUN = stemDocument)
```

```
# Create our word frequency per doc matrix
docTermMatrix <- DocumentTermMatrix(x = corpus)
```

```
# Split matrix by getting first 90% of matrix for train and last 10% for test
trainData <- docTermMatrix[1:round(nrow(docTermMatrix)*0.9, 0), ]
```

```
testData <- docTermMatrix[(round(nrow(docTermMatrix)*0.9, 0)+1):nrow(docTermMatrix), ]

trainLabels <- emails[1:round(nrow(emails)*0.9, 0), ]$type
testLabels <- emails[(round(nrow(emails)*0.9, 0)+1):nrow(docTermMatrix), ]$type

# Test proportions of
prop.table(table(trainLabels))
```

```
## trainLabels
##      ham     spam
## 0.836913 0.163087
```

```
prop.table(table(testLabels))
```

```
## testLabels
##       ham      spam
## 0.8262295 0.1737705
```

Our ratios are fairly consistent. Good to go.

```
emailFreqWords <- findFreqTerms(docTermMatrix, 10)
str(emailFreqWords)
```

```
##  chr [1:4046] "abandon" "abil" "abl" "ablock" "absolut" "abstract" "abus" ...
```

```
trainDataFreq <- trainData[, emailFreqWords]
testDataFreq <- testData[, emailFreqWords]
```

Now that we have our training and test term matrices, we convert the frequency to a factor of simply whether the doc contains the term or not.

```
convert_counts <- function(x) {
  x <- ifelse(x > 0, "Yes", "No")
}
```

```
trainDataFreq <- apply(trainDataFreq, 2, convert_counts)
testDataFreq <- apply(testDataFreq, 2, convert_counts)
```

Train Model and Test Accuracy

Here we train our model using our training data matrix then test it for accuracy using our test data matrix.

```
classifier = naiveBayes(trainDataFreq, trainLabels)
sample(classifier$tables,5)
```

```
## $wwwosdncom
##             wwwosdncom
## trainLabels          No         Yes
##        ham  0.997390170 0.002609830
```

```
##        spam 0.997767857 0.002232143
##
## $mechan
##            mechan
## trainLabels          No         Yes
##        ham  0.990430622 0.009569378
##        spam 0.997767857 0.002232143
##
## $compar
##            compar
## trainLabels         No        Yes
##        ham  0.98738582 0.01261418
##        spam 0.96875000 0.03125000
##
## $bad
##            bad
## trainLabels         No        Yes
##        ham  0.96085254 0.03914746
##        spam 0.98660714 0.01339286
##
## $green
##            green
## trainLabels          No         Yes
##        ham  0.991735537 0.008264463
##        spam 0.995535714 0.004464286
```

```
testPredictions <- predict(classifier, testDataFreq)
confusionMatrix(data = testPredictions,
 reference = factor(testLabels),
 positive = "spam",
 dnn = c("Predicted", "Observed"))
```

```
## Confusion Matrix and Statistics
##
##          Observed
## Predicted ham spam
##      ham  250   13
##      spam   2   40
##
##                Accuracy : 0.9508
##                  95% CI : (0.9202, 0.9722)
##     No Information Rate : 0.8262
##     P-Value [Acc > NIR] : 4.721e-11
##
##                   Kappa : 0.8134
##
##  Mcnemar's Test P-Value : 0.009823
##
##             Sensitivity : 0.7547
##             Specificity : 0.9921
##          Pos Pred Value : 0.9524
##          Neg Pred Value : 0.9506
##              Prevalence : 0.1738
##          Detection Rate : 0.1311
```

```
##    Detection Prevalence : 0.1377
##       Balanced Accuracy : 0.8734
##
##        'Positive' Class : spam
##
```

Our model has an estimated 96.4% accuracy rating in its ability to determine if an email is ham or spam based on term usage.