HW 2 problem 2

Carter Pearon

August 19, 2015

## 2

Calls the tm library and defines a reader function that will be used later on

library(tm)

## Loading required package: NLP

readerPlain = function(fname){  
 readPlain(elem=list(content=readLines(fname)),   
 id=fname, language='en') }

Training corpus

# Rolls 50 directories together into a single training corpus  
author\_dirs = Sys.glob('./ReutersC50/C50train/\*')  
file\_list = NULL #list of file directories  
train\_labels = NULL #List of author names  
for(author in author\_dirs) {  
 author\_name = substring(author, first=23)  
 files\_to\_add = Sys.glob(paste0(author, '/\*.txt'))  
 file\_list = append(file\_list, files\_to\_add)  
 train\_labels = append(train\_labels, rep(author\_name, length(files\_to\_add)))  
}  
  
# Creates our training corpus from the file\_list using the readerPlain function   
  
all\_docs = lapply(file\_list, readerPlain) #Read in all docs  
names(all\_docs) = file\_list   
names(all\_docs) = sub('.txt', '', names(all\_docs))  
  
train\_corpus = Corpus(VectorSource(all\_docs))   
names(train\_corpus) = file\_list  
  
# Preprocessing steps for our training corpus  
  
# make everything lowercase  
train\_corpus = tm\_map(train\_corpus, content\_transformer(tolower))   
  
# remove numbers  
train\_corpus = tm\_map(train\_corpus, content\_transformer(removeNumbers))   
  
# remove punctuation  
train\_corpus = tm\_map(train\_corpus, content\_transformer(removePunctuation))   
  
# remove excess white-space  
train\_corpus = tm\_map(train\_corpus, content\_transformer(stripWhitespace))   
  
# remove stop words  
train\_corpus = tm\_map(train\_corpus, content\_transformer(removeWords), stopwords("SMART"))  
  
# Creates a Training Document Term Matrix and removes sparse words  
DTM\_train = DocumentTermMatrix(train\_corpus)  
class(DTM\_train)

## [1] "DocumentTermMatrix" "simple\_triplet\_matrix"

DTM\_train = removeSparseTerms(DTM\_train, 0.95)  
DTM\_train\_matrix = as.matrix(DTM\_train)  
# Above I arbitrarily picked .95 as my threshold for sparsity. The interpretation of this number is that a word will not be included in our DTM unless it appears in at least 5% of the documents (1 -.95 = .05 = 5%)

Test corpus

# Rolls 50 directories together into a single test corpus  
author\_dirs = Sys.glob('./ReutersC50/C50test/\*')  
file\_list = NULL  
test\_labels = NULL  
for(author in author\_dirs) {  
 author\_name = substring(author, first=22)  
 files\_to\_add = Sys.glob(paste0(author, '/\*.txt'))  
 file\_list = append(file\_list, files\_to\_add)  
 test\_labels = append(test\_labels, rep(author\_name, length(files\_to\_add)))  
}  
  
# Creates our test corpus from the file\_list using the readerPlain function   
all\_docs = lapply(file\_list, readerPlain)   
names(all\_docs) = file\_list  
names(all\_docs) = sub('.txt', '', names(all\_docs))  
  
#Initialize Testing Corpus  
test\_corpus = Corpus(VectorSource(all\_docs))  
names(test\_corpus) = file\_list  
  
#Preprocessing steps for our test corpus  
test\_corpus = tm\_map(test\_corpus, content\_transformer(tolower))   
test\_corpus = tm\_map(test\_corpus, content\_transformer(removeNumbers))   
test\_corpus = tm\_map(test\_corpus, content\_transformer(removePunctuation))   
test\_corpus = tm\_map(test\_corpus, content\_transformer(stripWhitespace))   
test\_corpus = tm\_map(test\_corpus, content\_transformer(removeWords), stopwords("SMART"))  
  
# create a dictionary of all the words from our training set  
train\_names\_dict = NULL  
train\_names\_dict = dimnames(DTM\_train)[[2]]  
class(train\_names\_dict)

## [1] "character"

# Creates a Test Document Term Matrix using our set of training words and removes sparse words  
DTM\_test = DocumentTermMatrix(test\_corpus, list(dictionary=train\_names\_dict))  
DTM\_test\_matrix = as.matrix(DTM\_test)

Now let’s build a Naive Bayes model

library(e1071)  
library(caret)

## Loading required package: lattice  
## Loading required package: ggplot2  
##   
## Attaching package: 'ggplot2'  
##   
## The following object is masked from 'package:NLP':  
##   
## annotate

library(rpart)  
  
  
# Create our model using the training data  
model\_NB = naiveBayes(x = DTM\_train\_matrix, y = as.factor(train\_labels))  
  
# Making our predictions use the Naive Bayes model and our Test set  
pred\_NB = predict(model\_NB, DTM\_test\_matrix)  
  
# Create a confusion matrix and assess the accuracy of our predictions  
confusion\_matrix = confusionMatrix(table(pred\_NB, train\_labels))  
confusion\_matrix$overall

Using this Naive Bayes model and a sparsity threshold of 5% (.95), the model accurately predicted the author 26.2% of the time. I played around a little bit with this sparsity threshold, and .95 gave me the best accuracy!d to the code chunk to prevent printing of the R code that generated the plot.

Now let’s build a Random Forest model

library(randomForest)

## randomForest 4.6-10  
## Type rfNews() to see new features/changes/bug fixes.

randomforest = randomForest(x= DTM\_train\_matrix, y= as.factor(train\_labels), mtry = 3, ntree=500)  
rfpredict = predict(randomforest, data = DTM\_test\_matrix)  
confusionrf = confusionMatrix(table(rfpredict, test\_labels))  
confusionrf$overall

## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 0.7452000 0.7400000 0.7276380 0.7621859 0.0200000   
## AccuracyPValue McnemarPValue   
## 0.0000000 NaN

Our random forest model gives an accuracy of 74.52%, which is substantially better than the accuracy from our Naive Bayes model!