library(plyr)

library(dplyr)

library(stringr)

library(e1071)

library(tm)

library(sentiment)

library(randomForest)

library(ggplot)

library(Rstem)

library(wordcloud)

library(plotly)

library(cvTools)

#################################################################

#### instructions to set the directory ####

## set the working directory to the data folder of the project ##

#################################################################

setwd("C:/Users/DELL/Desktop/project data")

#load up word polarity list and format it

afinn\_list <- read.delim(file='AFINN-111.txt', header=FALSE, stringsAsFactors=FALSE)

names(afinn\_list) <- c('word', 'score')

afinn\_list$word <- tolower(afinn\_list$word)

#categorize words as very negative to very positive and add some movie-specific words

vNegTerms <- afinn\_list$word[afinn\_list$score==-5 | afinn\_list$score==-4]

negTerms <- c(afinn\_list$word[afinn\_list$score==-3 | afinn\_list$score==-2 | afinn\_list$score==-1], "second-rate", "moronic", "third-rate", "flawed", "juvenile", "boring", "distasteful", "ordinary", "disgusting", "senseless", "static", "brutal", "confused", "disappointing", "bloody", "silly", "tired", "predictable", "stupid", "uninteresting", "trite", "uneven", "outdated", "dreadful", "bland")

posTerms <- c(afinn\_list$word[afinn\_list$score==3 | afinn\_list$score==2 | afinn\_list$score==1], "first-rate", "insightful", "clever", "charming", "comical", "charismatic", "enjoyable", "absorbing", "sensitive", "intriguing", "powerful", "pleasant", "surprising", "thought-provoking", "imaginative", "unpretentious")

vPosTerms <- c(afinn\_list$word[afinn\_list$score==5 | afinn\_list$score==4], "uproarious", "riveting", "fascinating", "dazzling", "legendary")

#load up positive and negative sentences and format

posText <- read.delim(file='rt-polarity-pos.txt', header=FALSE, stringsAsFactors=FALSE, quote ="")

posText <- posText$V1

posText <- unlist(lapply(posText, function(x) { str\_split(x, "\n") }))

negText <- read.delim(file='rt-polarity-neg.txt', header=FALSE, stringsAsFactors=FALSE, quote="")

negText <- negText$V1

negText <- unlist(lapply(negText, function(x) { str\_split(x, "\n") }))

#function to calculate number of words in each category within a sentence

sentimentScore <- function(sentences, vNegTerms, negTerms, posTerms, vPosTerms){

final\_scores <- matrix('', 0, 5)

scores <- laply(sentences, function(sentence, vNegTerms, negTerms, posTerms, vPosTerms){

initial\_sentence <- sentence

#remove unnecessary characters and split up by word

sentence <- gsub('[[:punct:]]', '', sentence)

sentence <- gsub('[[:cntrl:]]', '', sentence)

sentence <- gsub('[0-9]', '', sentence)

sentence <- gsub('\\d+', '', sentence)

sentence <- tolower(sentence)

wordList <- str\_split(sentence, '\\s+')

words <- unlist(wordList)

#build vector with matches between sentence and each category

vPosMatches <- match(words, vPosTerms)

posMatches <- match(words, posTerms)

vNegMatches <- match(words, vNegTerms)

negMatches <- match(words, negTerms)

#sum up number of words in each category

vPosMatches <- sum(!is.na(vPosMatches))

posMatches <- sum(!is.na(posMatches))

vNegMatches <- sum(!is.na(vNegMatches))

negMatches <- sum(!is.na(negMatches))

score <- c(vNegMatches, negMatches, posMatches, vPosMatches)

#add row to scores table

newrow <- c(initial\_sentence, score)

final\_scores <- rbind(final\_scores, newrow)

return(final\_scores)

}, vNegTerms, negTerms, posTerms, vPosTerms)

return(scores)

}

#build tables of positive and negative sentences with scores

posResult <- as.data.frame(sentimentScore(posText, vNegTerms, negTerms, posTerms, vPosTerms))

negResult <- as.data.frame(sentimentScore(negText, vNegTerms, negTerms, posTerms, vPosTerms))

posResult <- cbind(posResult, 'positive')

colnames(posResult) <- c('sentence', 'vNeg', 'neg', 'pos', 'vPos', 'sentiment')

negResult <- cbind(negResult, 'negative')

colnames(negResult) <- c('sentence', 'vNeg', 'neg', 'pos', 'vPos', 'sentiment')

#combine the positive and negative tables

results <- rbind(posResult, negResult)

#run the naive bayes algorithm using all four categories

classifier <- naiveBayes(results[,2:5], results[,6])

class\_pol = classify\_polarity(results[,1], algorithm="bayes")

polarity = class\_pol[,4]

class\_emo = classify\_emotion(results, algorithm="bayes", prior=1.0)

emotion = class\_emo[,7]

emotion[is.na(emotion)] = "unknown"

for (i in 1:10662){

if(class\_pol[i,4]=='neutral'){

class\_pol[i,4] = 'positive'

}

}

#display the confusion table for the classification ran on the same data

confTable <- table(predict(classifier, results), results[,6], dnn=list('predicted','actual'))

confTable

#run a binomial test for confidence interval of results

binom.test(confTable[1,1] + confTable[2,2], nrow(results), p=0.5)

binom.test(confTable1[1,1] + confTable1[2,2], nrow(results), p=0.5)

#######################################

### Classification with Naive Bayes ###

#######################################

set.seed(1234) # set seed for reproducibility

confTable <- matrix(0,2,2)

k <- 5 # number of folds

folds <- cvFolds(nrow(results), K = k, type = "interleaved")

for(i in 1:k){

xtest <- subset(results, folds$which == i)[, 2:6]

x <- subset(results, folds$which != i)[, 2:6]

classifier <- naiveBayes(x[,1:4], x[,5])

#display the confusion table for the classification ran on the same data

confTable <- confTable + table(predict(classifier, xtest), xtest[,5], dnn=list('predicted','actual'))

confTable

}

confTable

#################################

## Classification with SVM ######

#################################

set.seed(1234) # set seed for reproducibility

confTable <- matrix(0,2,2)

svdat <- as.data.frame(results[,2:6])

k <- 10 # number of folds

folds <- cvFolds(nrow(results), K = k, type = "interleaved")

for(i in 1:k){

traindata <- subset(svdat, folds$which == i)

testdata <- subset(svdat, folds$which != i)

model1 <- svm(traindata$sentiment~. , data = traindata)

#display the confusion table for the classification ran on the same data

confTable <- confTable + table(predict(model1, testdata), testdata[ ,5], dnn=list('predicted','actual'))

}

confTable

###########################################

## Classification with Random Forest ######

###########################################

set.seed(1234) # set seed for reproducibility

confTable <- matrix(0,2,2)

svdat <- as.data.frame(results[,2:6])

k <- 10 # number of folds

folds <- cvFolds(nrow(results), K = k, type = "interleaved")

for(i in 1:k){

traindata <- subset(svdat, folds$which == i)

testdata <- subset(svdat, folds$which != i)

model2 <- randomForest(traindata$sentiment~. , data = traindata, importance = TRUE, proximity=TRUE)

#display the confusion table for the classification ran on the same data

confTable <- confTable + table(predict(model2, testdata), testdata[ ,5], dnn=list('predicted','actual'))

}

confTable

# to visualize the polarity

plot\_ly(results, x=~polarity, type="histogram",

marker = list(color = c('magenta', 'gold',

'lightblue'))) %>%

layout(yaxis = list(title='Count'), title="Sentiment Analysis: Polarity")

# to visualize the emotions

plot\_ly(results, x=~emotion,type="histogram",

marker = list(color = c('grey', 'red',

'orange', 'navy',

'yellow'))) %>%

layout(yaxis = list(title='Count'), title="Sentiment Analysis: Emotions")

# combine the emotion and polairy

sent\_df = data.frame(text=results, emotion=emotion,polarity=polarity, stringsAsFactors=FALSE)

class(sent\_df)

head(sent\_df)

sent\_df = within(sent\_df,emotion <- factor(emotion, levels=names(sort(table(emotion), decreasing=TRUE))))

head(sent\_df)

# to visulaize the emotion

ggplot(sent\_df, aes(x=emotion)) +

geom\_bar(aes(y=..count.., fill=emotion)) +

scale\_fill\_brewer(palette="Dark2") +

labs(x="emotion categories", y="density")

# to visalize the ploarity

ggplot(sent\_df, aes(x=polarity)) +

geom\_bar(aes(y=..count.., fill=polarity)) +

scale\_fill\_brewer(palette="RdGy") +

labs(x="polarity categories", y="")

emos = levels(factor(sent\_df$emotion))

nemo = length(emos)

emo.docs = rep("", nemo)

for (i in 1:nemo)

{

tmp = results[emotion == emos[i]]

emo.docs[i] = paste(tmp, collapse=" ")

}

emo.docs = removeWords(emo.docs, stopwords("english"))

corpus = Corpus(VectorSource(emo.docs))

tdm = TermDocumentMatrix(corpus)

tdm = as.matrix(tdm)

colnames(tdm) = emos

comparison.cloud(tdm, colors = brewer.pal(nemo, "Dark2"),

scale = c(3,.5), random.order = FALSE,

title.size = 1.5)