Project Sentiment Analysis

This Sentiment Analysis approach uses a generalized linear model with logistical regression as the family function. This is used because there is two types for the Sentiment class namely Negative, and Neutral.

Load Required Libraries

```
library(stringi)
library(qdapDictionaries)
library(tm)
```

```
## Loading required package: NLP
```

```
library(caTools)
library(tidytext)
library(SparseM)
```

```
##
## Attaching package: 'SparseM'
```

```
## The following object is masked from 'package:base':
##
##     backsolve
```

Load Dataset

```
My_data<-read.csv(file="C:/Users/Jayan/Documents/Ryerson - Big Data, Analytics, Predictive Analytics/CKME 136 - Capstone Pro
ject/Programming Part/My Data.csv", header=T, sep=",", na.strings=c("","NA"))
```

Clean the Dataset

```
My_data<-My_data[complete.cases(My_data),]
is.word  <- function(x) x %in% GradyAugmented
My_data$Response<-tolower(My_data$Response)
split_word<-stri_extract_all_words(My_data$Response, simplify=TRUE)
split_word[split_word==""]<-NA
nonengrownums<-which(apply(split_word, 1, function(x) sum(is.word(x)/sum(!is.na(x))))<0.75)
My_data<-My_data[-nonengrownums,]
```

Randomly shuffle the dataset and convert the Sentiment class to factor

```
SentiData<-My_data[sample(nrow(My_data)),]
SentiData$Sentiment<-as.factor(SentiData$Sentiment)
```

Create folds for cross validation and create matrix for accuracy which is used later

```
folds <- cut(seq(1,nrow(SentiData)),breaks=10,labels=FALSE)

accuracy<-matrix(nrow=10, ncol=3, dimnames=list(c(),c("Accuracy", "Negative Discovery Error", "Neutral Predicted Error")))
rownames(accuracy)<-c("Fold 1", "Fold 2", "Fold 3", "Fold 4", "Fold 5", "Fold 6", "Fold 7", "Fold 8", "Fold 9", "Fold 10")
```

Cross Validation and Anlaysis

```
for(i in 1:10){
  #Segementing the data by fold using the which() function
  testIndexes <- which(folds==i,arr.ind=TRUE)
  testData <- SentiData[testIndexes, ]
  trainData <- SentiData[-testIndexes, ]
  testData<-testData[-c(2:3)]
  trainData<-trainData[-c(2:3)]

  #Creating the Corpus and cleaning the data
  corpus <- Corpus(VectorSource(c(trainData$Response, testData$Response)))
  corpus <- tm_map(corpus, content_transformer(tolower))
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, removeWords, stopwords("english"))
  corpus <- tm_map(corpus, stripWhitespace)
  corpus <- tm_map(corpus, stemDocument)

  #Putting the corpus into a document matrix
  dtm<-DocumentTermMatrix(corpus)

  #sparse words, words that appear in atleast 1% of the Responses are taken
  sparse<-removeSparseTerms(dtm,0.99)

  #Those sparse words are put into a data frame called important words
  important_words_df <- as.data.frame(as.matrix(sparse))
  colnames(important_words_df) <- make.names(colnames(important_words_df))

  # split into train and test
  important_words_train_df <- head(important_words_df, nrow(trainData))
  important_words_test_df <- tail(important_words_df, nrow(testData))

  # Add to original dataframes
  train_data_words_df <- cbind(trainData, important_words_train_df)
  test_data_words_df <- cbind(testData, important_words_test_df)

  train_data_words_df$Response <- NULL
  test_data_words_df$Response <- NULL

  #Building the linear model from the train data
  log_model <- glm(Sentiment~., data=train_data_words_df, family=binomial)

  #Using the model to predict the test data
  log_pred <- predict(log_model, newdata=test_data_words_df, type="response")

  #Create the Results table and calculate the accuracy
  Results<-table(test_data_words_df$Sentiment, log_pred>.5)
  accuracy[i,1]<-(Results[1,1]+Results[2,2])/nrow(test_data_words_df)
  accuracy[i,2]<-Results[1,2]/(Results[1,1]+Results[1,2])
  accuracy[i,3]<-Results[2,1]/(Results[2,1]+Results[2,2])

}
```

```
## Warning in simple_triplet_matrix(i, j, v, nrow = length(terms), ncol =
## length(corpus), : bytecode version mismatch; using eval
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Summary Statistics and Analysis of results

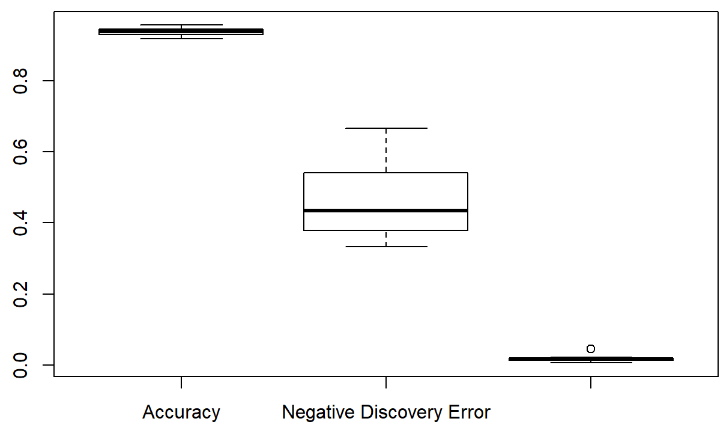
```
#This matrix shows the accuracy of the trial, Negative error, and Neutral Error or each cross validation segment
accuracy
```

```
##          Accuracy Negative Discovery Error Neutral Predicted Error
## Fold 1  0.9292035                0.5405405             0.013245033
## Fold 2  0.9380531                0.4166667             0.019801980
## Fold 3  0.9467456                0.3793103             0.022653722
## Fold 4  0.9174041                0.4516129             0.045454545
## Fold 5  0.9439528                0.3658537             0.013422819
## Fold 6  0.9378698                0.5625000             0.009803922
## Fold 7  0.9380531                0.4166667             0.019801980
## Fold 8  0.9230769                0.6666667             0.006622517
## Fold 9  0.9321534                0.4864865             0.016556291
## Fold 10 0.9557522                0.3333333             0.019230769
```

```
#Create a summary and boxplot of the accuracy of each trial data
summary(accuracy)
```

```
##     Accuracy      Negative Discovery Error Neutral Predicted Error
##  Min.   :0.9174   Min.   :0.3333           Min.   :0.006623
##  1st Qu.:0.9299   1st Qu.:0.3886           1st Qu.:0.013289
##  Median :0.9380   Median :0.4341           Median :0.017894
##  Mean   :0.9362   Mean   :0.4620           Mean   :0.018659
##  3rd Qu.:0.9425   3rd Qu.:0.5270           3rd Qu.:0.019802
##  Max.   :0.9558   Max.   :0.6667           Max.   :0.045455
```

```
boxplot(accuracy)
```



This model uses logistic regression to determine negative and neutral sentiments by taking into consideration the words that appear in at least 1% of the responses. This is done using the removeSparseTerms(dtm, 0.99) command where terms that are more sparse then 0.99% are removed. By using this command, terms that are frequently utilized will be considered in the analysis. The average accuracy of this model is 93.56% with average negative error of 45% and neutral error of 2%. It worked well with predicting the Neutral but not the Negative sentiments. The dataset is unbalanced between the neutral and negative classes; there are 342 negative records, and 3045 neutral records. Due to this large discrepancy, determining the negative classes proved to be more difficult.