Mobile App Mentions & Classification - Team 3

Description

For this analysis we leverage supervised learning techniques in order to create a model that will accurately classify future documents. We leverage NLP and tm packages in R, to create Text-Document Matrices. Then we combine our matrices with our existing data, in order to build out predictive models. Our model is applied to a training and test data set, as final test to verify performance.

Corpus

Our body of text/corpus are reviews and ratings for mobile apps in the Apple and Google Play stores. The code below ingests our text files, and then creates Document-term matrices, based on the open ended responses provided in our description field (MentionsTable$Description).

```r
#Corpus Import & Syntax
#fileencoding needs to be set to "latin1"
MentionsURL <-"https://raw.githubusercontent.com/Misterresearch/CUNY-Projects/master/App%20Mentions.csv

MentionsTable <- read.csv(file = MentionsURL, header = TRUE, sep = ",", strip.white = TRUE, na.strings =
MentionsTable$description = as.character(MentionsTable$description)

#Create data frame
mentiondesc <- data.frame(MentionsTable$description)

#makes each row in data frame a document, required for subsequent statistical analysis.
mentiondesc <- Corpus(DataframeSource(mentiondesc))

#Corpus Loading, filtering and stemming code. See "Basic Text Mining" source in end notes.
mentiondesc <- tm_map(mentiondesc, removePunctuation)
#for(j in seq(mentiondesc))
#{
  #mentiondesc[[j]] <- gsub("/", " ", mentiondesc[[j]])
  #mentiondesc[[j]] <- gsub("@", " ", mentiondesc[[j]])
  #mentiondesc[[j]] <- gsub("\\|", " ", mentiondesc[[j]])
#}
mentiondesc <- tm_map(mentiondesc, removeNumbers)
mentiondesc <- tm_map(mentiondesc, tolower)
mentiondesc <- tm_map(mentiondesc, removeWords, stopwords("english"))
mentiondesc <- tm_map(mentiondesc, removeWords, c("none","the", "and", "or" , "http\\w*"))
mentiondesc <- tm_map(mentiondesc, stemDocument)
mentiondesc <- tm_map(mentiondesc, stripWhitespace)
mentiondesc <- tm_map(mentiondesc, PlainTextDocument)

#Single Term Matrices
mdtm <- DocumentTermMatrix(mentiondesc)
mtdm <- TermDocumentMatrix(mentiondesc)
mdtm
```

```
## <<DocumentTermMatrix (documents: 4832, terms: 7152)>>
## Non-/sparse entries: 41757/34516707
## Sparsity           : 100%
## Maximal term length: 218
## Weighting          : term frequency (tf)
```

```
mtdm
```

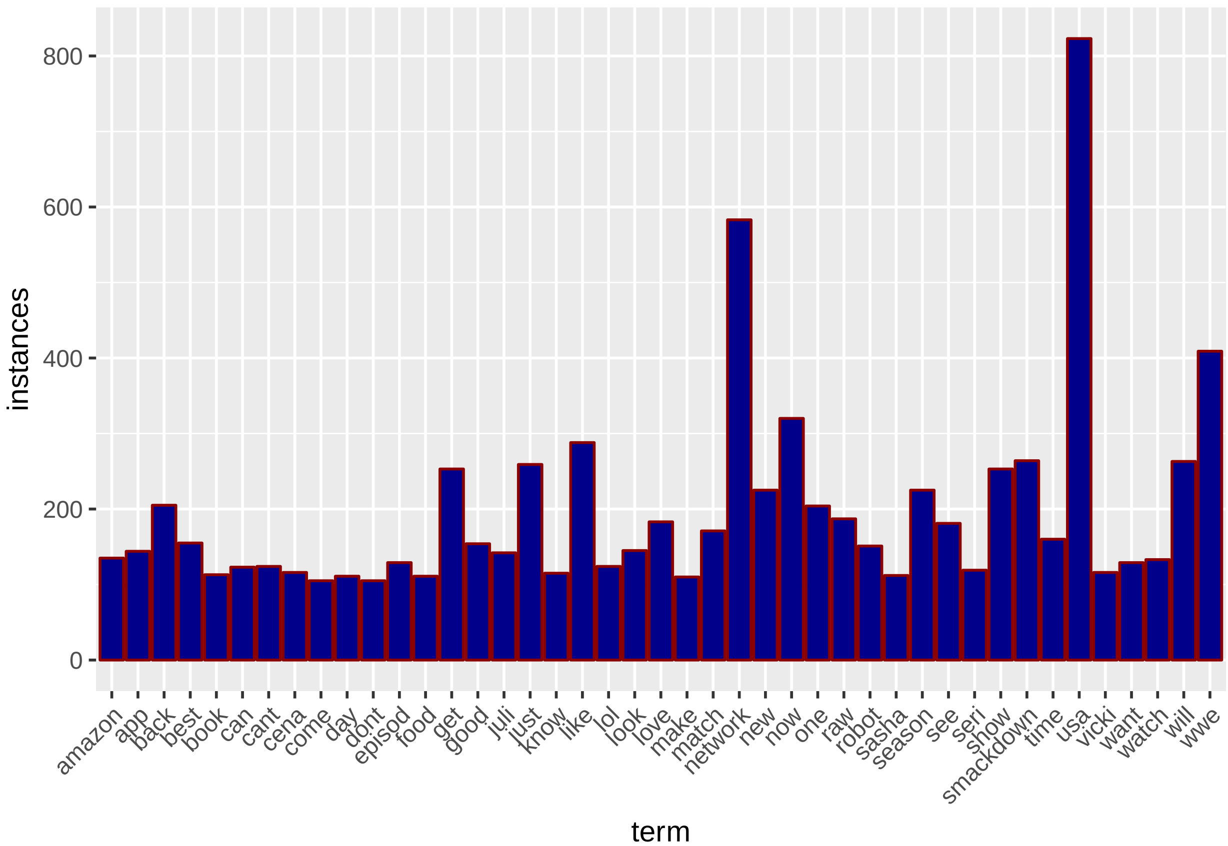
```
## <<TermDocumentMatrix (terms: 7152, documents: 4832)>>
## Non-/sparse entries: 41757/34516707
## Sparsity           : 100%
## Maximal term length: 218
## Weighting          : term frequency (tf)
```

```
#Sparcity settng adjustments
mentionstdm2 <- removeSparseTerms(mtdm, .97)
mentionsdtm2 <- removeSparseTerms(mdtm, .97)
mentionsfreq <- rowSums(as.matrix(mtdm))
#findFreqTerms(mentionstdm2, lowfreq = 1)
```

Word Frequency & Visualization

Below is a series of frequency charts on single words and bi grams to help us understand what words and terms are used most, and cluster together.

```
#Single Term Frequency Charts
tf <- data.frame(term = names(mentionsfreq), instances=mentionsfreq)
subset(tf, mentionsfreq>100) %>%
  ggplot(aes(term,instances)) +
  geom_bar(stat="identity", fill="darkblue", colour="darkred") +
  theme(axis.text.x=element_text(angle = 45, hjust = 1))
```

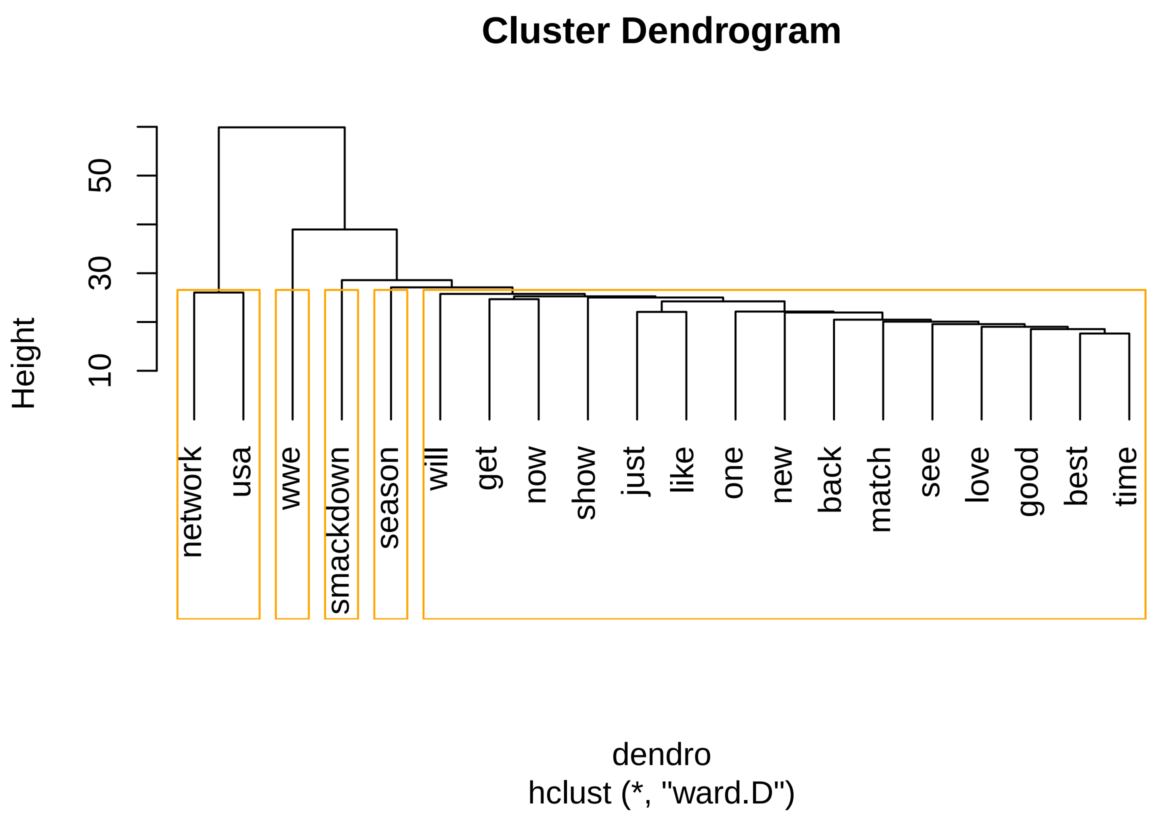



```
#Single Term Word Clouds
wordcloud(names(mentionsfreq), mentionsfreq, min.freq = 100, scale=c(5, .1), colors=brewer.pal(6, "Dark
```

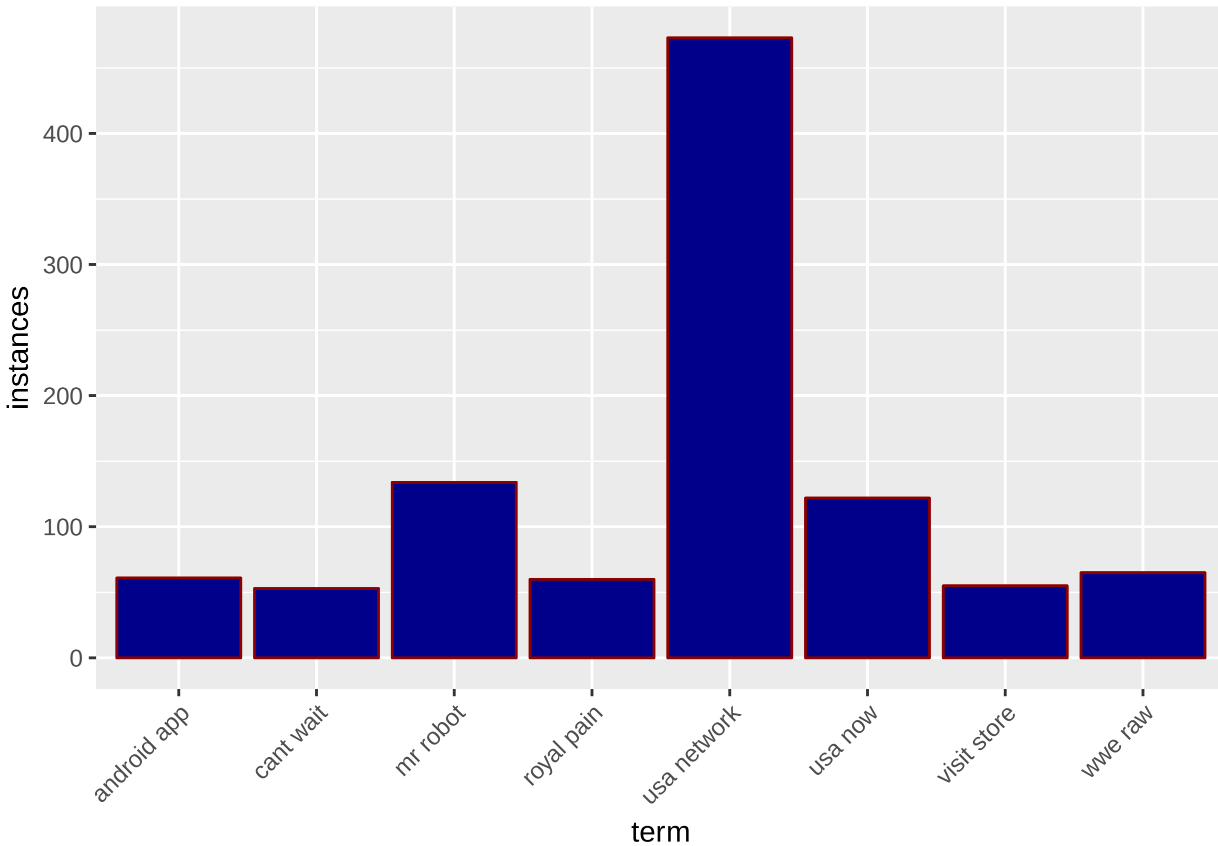

```
#Single Term Correlation Analysis
findAssocs(mtdm, c("Twitter", "Amazon", "chromecast"), corlimit = .2)
```

```
## $Twitter
## numeric(0)
##
## $Amazon
## numeric(0)
##
## $chromecast
## numeric(0)
```

```
#Single Term Cluster Analysis, requires Document-Text Matrix

dendro <- dist(t(mentionsdtm2), method="euclidean")
cluster <- hclust(d=dendro, method="ward.D")
plot(cluster, hang=-1)
rect.hclust(cluster, k=5, border="orange")
```

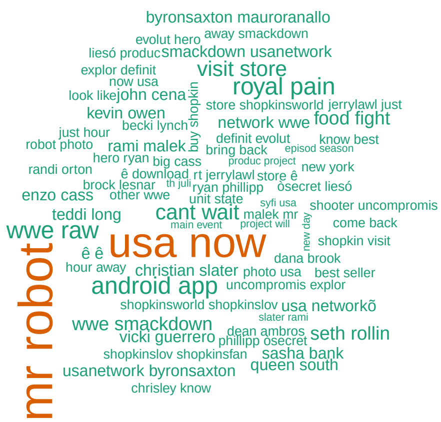



```r
#Bigram Corpus, see source for "Bigram Text-Document Matrices" in endnotes
BigramTokenizer <-
     function(x)
        unlist(lapply(ngrams(words(x), 2), paste, collapse = " "), use.names = FALSE)
mtdm2 <- TermDocumentMatrix(mentiondesc, control = list(tokenize = BigramTokenizer))
mdtm2 <- DocumentTermMatrix(mentiondesc, control = list(tokenize = BigramTokenizer))
mentionsfreq2 <- rowSums(as.matrix(mtdm2))


#Bigram Frequency Chart
tf <- data.frame(term = names(mentionsfreq2), instances=mentionsfreq2)
subset(tf, mentionsfreq2>50) %>%
  ggplot(aes(term,instances)) +
  geom_bar(stat="identity", fill="darkblue", colour="darkred") +
  theme(axis.text.x=element_text(angle = 45, hjust = 1))
```

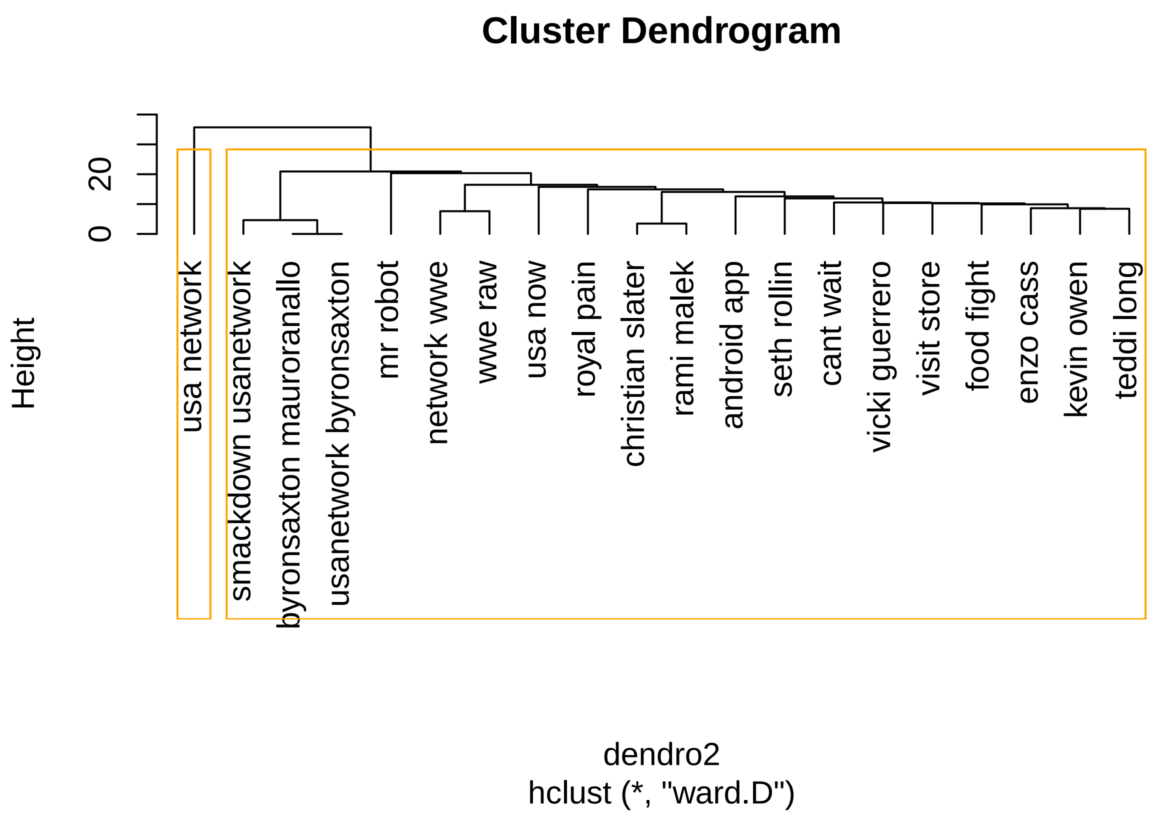



```r
#Bigram Word Cloud
wordcloud(names(mentionsfreq2), mentionsfreq2, min.freq = 25, scale=c(5, .1), colors=brewer.pal(6, "Dar
```

```
## Warning in wordcloud(names(mentionsfreq2), mentionsfreq2, min.freq = 25, :
## usa network could not be fit on page. It will not be plotted.
```




```r
#Bigram Dendrogram
mdtm2a <- removeSparseTerms(mdtm2, .993)
dendro2 <- dist(t(mdtm2a), method="euclidean")
cluster <- hclust(d=dendro2, method="ward.D")
plot(cluster, hang=-1)
rect.hclust(cluster, k=2, border="orange")
```



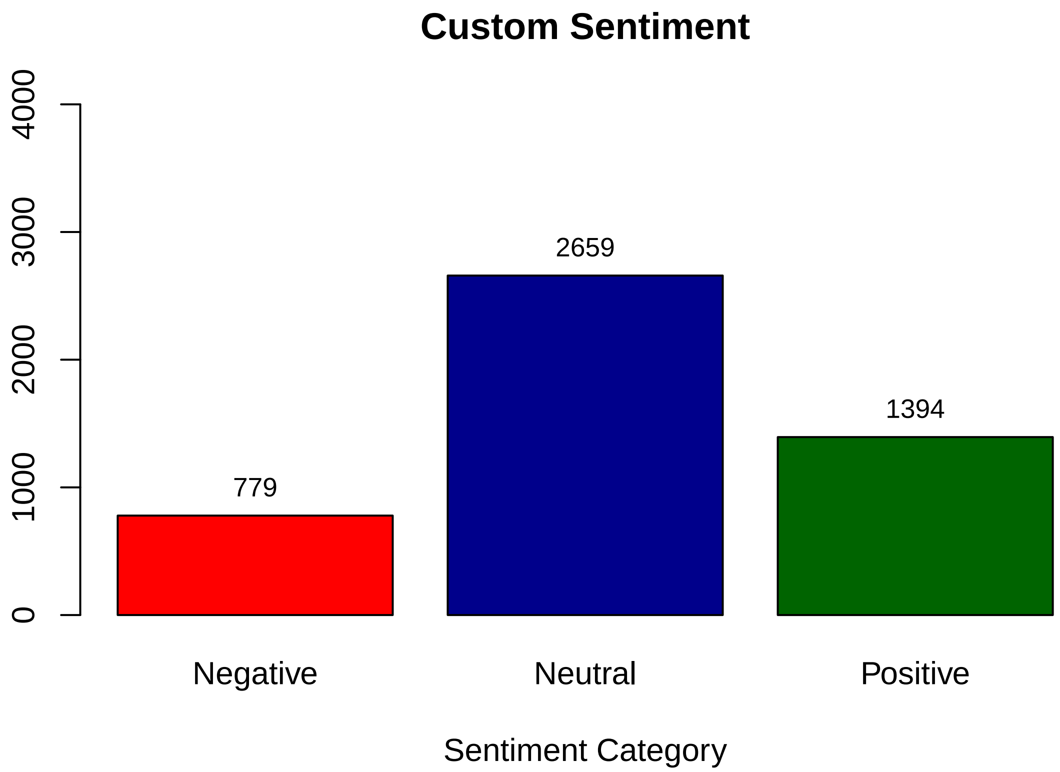
Mentions Classification & Analysis

Leveraging the sentiment corpus for negative and positive terms provided by Hu * Liu, and the tm_term_score function to allocate an integer value for each document. Once we've derived a score, then we create two variables "pos" and "neg", that become added to our data frame and as an independent variable in predictive model. For this analysis, we've also derived a custom variable called "sentiment", based on the ratio of "pos" to "neg" comments.

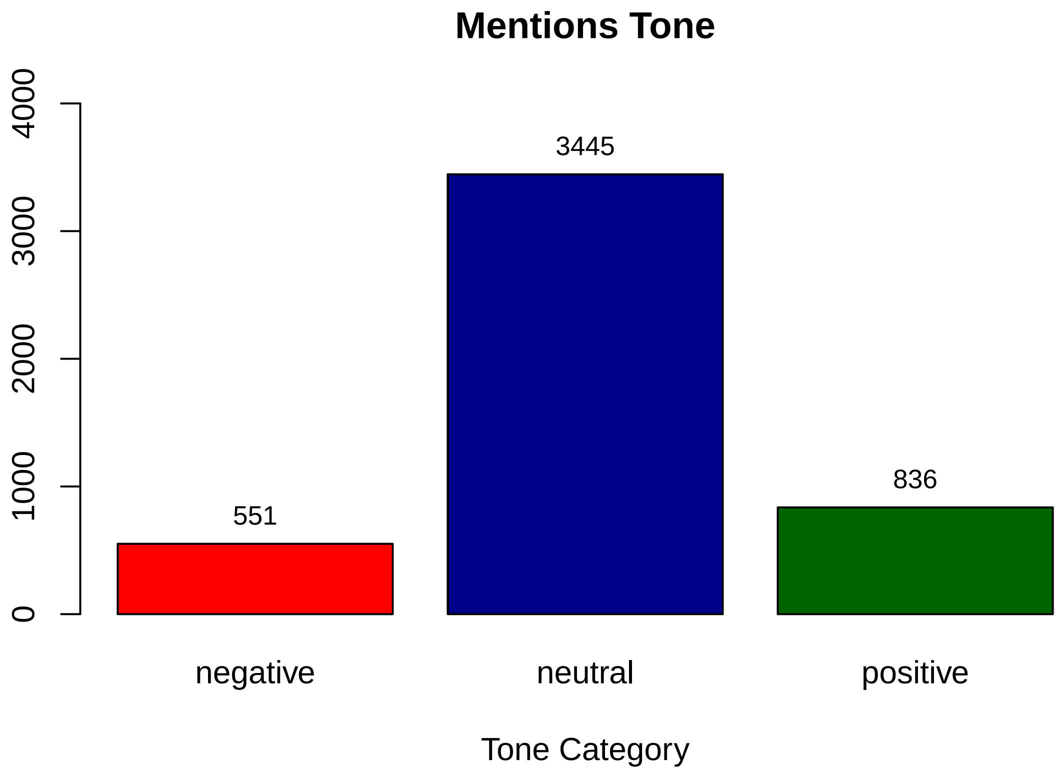
```
#pos and neg sentiment sourced from Hu and Liu
#header notes removed from source file, adjust file path
pos_words = read.table("/Users/digitalmarketer1977/Desktop/positive-words.txt", header = F, stringsAsFa
neg_words = read.table("/Users/digitalmarketer1977/Desktop/negative-words.txt", header = F, stringsAsFa
```

```
MentionsTable$neg = sapply(mentiondesc, tm_term_score, neg_words)
MentionsTable$pos = sapply(mentiondesc, tm_term_score, pos_words)

#Raw Sum of Neg vs. Pos Comments
sumpos <- sum(MentionsTable$pos)
sumneg <- sum(MentionsTable$neg)
#sumcomments <- barplot(c(sumneg,sumpos),col = c("red","darkgreen"), ylim = c(0,3500))
#text(x = sumcomments, y = c(sumneg,sumpos),labels = c(sumneg,sumpos), pos = 3 , cex = .8, col = "black

#Sentiment Classification & Analysis
MentionsTable$sentiment = MentionsTable$pos/MentionsTable$neg
MentionsTable$sentiment[MentionsTable$sentiment > 1.50] <- "Positive"
MentionsTable$sentiment[(MentionsTable$sentiment <= 1.50) & (MentionsTable$sentiment >= .50)] <- "Neutra
MentionsTable$sentiment[MentionsTable$sentiment == "NaN"] <- "Neutral"
MentionsTable$sentiment[MentionsTable$sentiment < .50] <- "Negative"
sentiment <- table(MentionsTable$sentiment)
```

```
myValues <- barplot(sentiment, main="Custom Sentiment", xlab = "Sentiment Category", col=c("red","darkbl
text(x = myValues, y = sentiment, labels = sentiment, pos = 3, cex = .8, col = "black")
```




```
tone <- table(MentionsTable$tone)
myValuesB <- barplot(tone, main="Mentions Tone", xlab = "Tone Category", col=c("red","darkblue","darkgre
text(x = myValuesB, y = tone, labels = tone, pos = 3, cex = .8, col = "black")
```

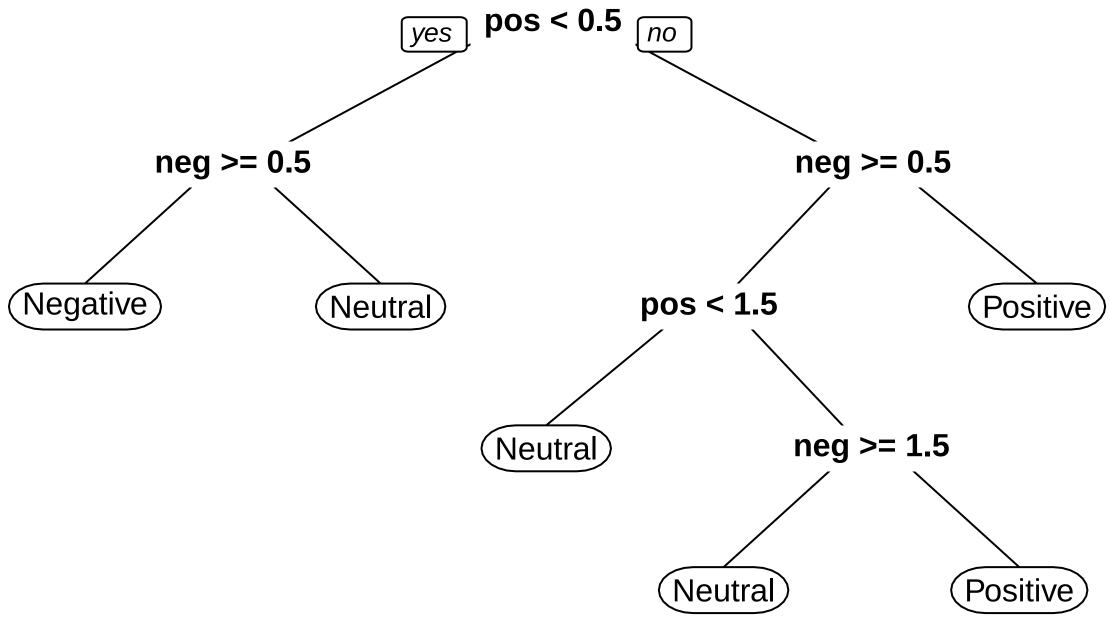


Predictive Model

By appending Document Term Matrices to our existing data frame, essentially as dummy variables. We're now able to combine the most of frequent terms that we've found in our corpus from app feedback, along with the "pos" & "neg" scores derived from the generic sentiment corpus from Hu & Liu, to build out a more robust model.

The model we used for our tree and our final predictive model was "sentiment ~ neg + pos + wwe + usa + smackdown".

```
#append to data frame

MentionsTable = cbind(MentionsTable, as.matrix(mentionsdtm2))
MentionsTable$tone = as.factor(MentionsTable$tone)

id_train <- sample(nrow(MentionsTable),nrow(MentionsTable)*0.80)
MentionsTable.train = MentionsTable[id_train,]
MentionsTable.test = MentionsTable[-id_train,]

MentionsTable.tree = rpart(sentiment ~ neg + pos + wwe + usa + smackdown,  method = "class", data = Ment
prp(MentionsTable.tree)
```



Conclusion & Insights

Splitting our data between train and test (80/20), we're able to apply the model we selected (manually) above for our decision tree to both splits of the data. The purpose of applying our model to our test data set, is to compare our predictions to the training (historical) data - as proxy for how accurately future data would be predicted. We can see from the predict values on our test data below, that the output matches what we're seeing on the training data - indicating that we have a good fit for our predictive model.

```
##           Pred
## Obs         Negative Neutral Positive
##   Negative       153       5        0
##   Neutral          0     538        0
##   Positive         0       0      271
```

Code Source: Basic Text Mining in R

Code Source: Bigram Text-Document Matrices

Reference: Automated Data Collection with R, Wiley (2015)

Code Source: Predictive Modeling

Data Source: Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews."