Twitter Sentiment Analysis

The search has been done on a specific hashtag #donaldtrump and 3000 tweets retrieved

Step 1: Establishing the connection with Twitter App :-> #donaldtrump

Step 2: Preparing positive and negative words dictionary.

Step 3: Preprocessing of the tweets

Several steps performed like:

- Removed whitespace
- Replaced apostrophes
- Removed emojis and other Unicode characters
- Removed additional Unicode parts that may have remained
- Removed orphaned full-stops
- Reduced double spaces to single spaces
- Removed URL from tweet
- Replaced any line breaks with -
- Fixed ampersand
- Removed trailing whitespaces
- Removed the digits
- Replaced orphaned fullstops with space
- Removed leading whitespaces
- Removed trailing whitespaces
- Removed pesky Unicodes like <U+A>
- Removed emojis/dodgy Unicode

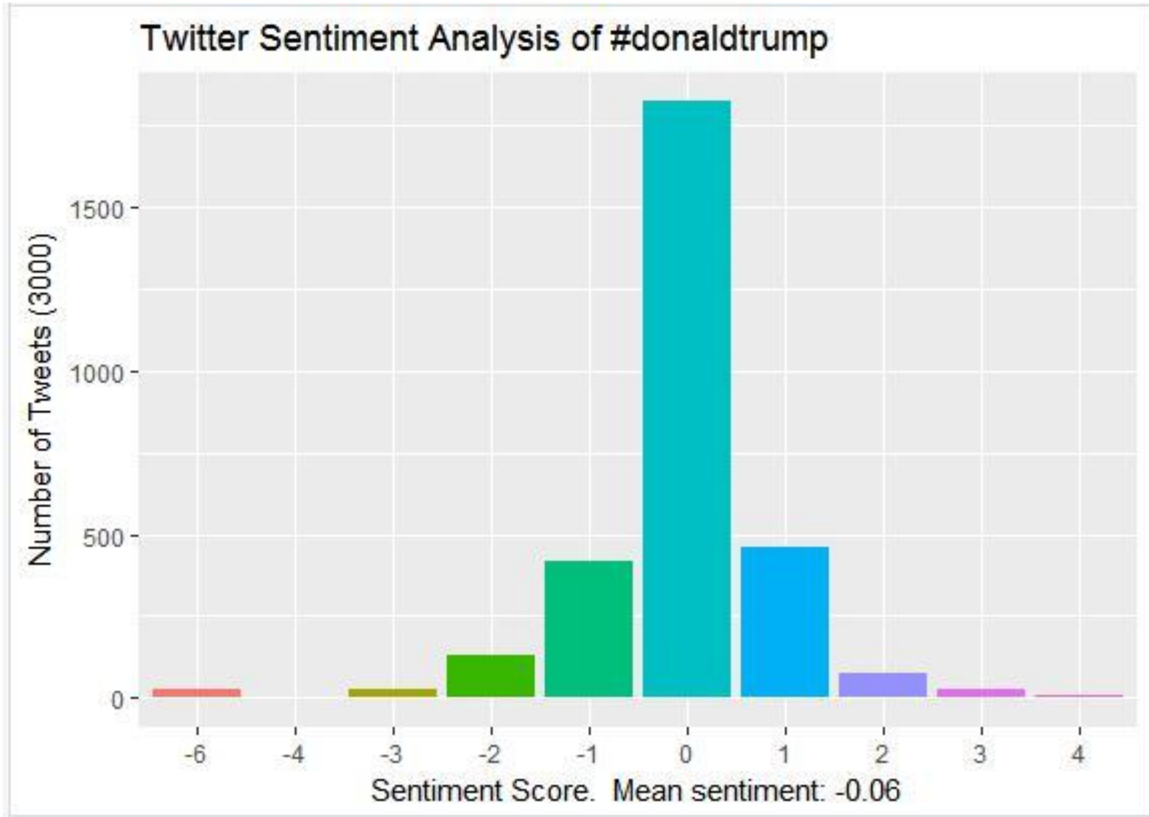
STEP 4: Performing the Sentiment Score Analysis:

Below things has been done

To Calculate Score

- ✓ Ran through the tweets and has extracted text.
- ✓ After that I have splitted the words into list.
- ✓ Moreover reduced list levels.
- ✓ Performed the matching of positive and negative words.

Below is the Plot made for the Sentiment score: **Graph Sentiment**



- More than 1500 sentiments has the score of 0 (Neutral)
- Tweets with score -1 and 1 almost has the same count.
- Mean Sentiment is -0.06 [Sentiment of Tweets are more negative than positive.]

STEP 5:- Analyzing twitter feeds

In Step 5. 1: Here I am creating the tweet corpus and adding a new column having **sentiment class**
Column name is **class_sentiment.** This new column will help me in getting sentiment
Class as per sentiment score. This has been done for performing the Naïve Bayes prediction

```
##Step 5.1
###Here i am creating the tweet corpus and adding a new column having sentiment class
## column name is class_sentiment.This new column will help me in getting sentiment
##class as per sentiment score.
Tweet_corpus$class_sentiment <- ifelse( Tweet_corpus$sentimentScore >0, 'pos', 'neg')
head(Tweet_corpus,5)
```

Output:

```
  sentimentScore class_sentiment
1             -2             neg
2             -2             neg
3              0             neg
4              0             neg
5              0             neg
>
```

In Step 5.2 I am preparing the corpus for positive and negative counts.

In STEP 5.3:

I am counting +ve and -ve words in the tweets.

Here in this step i am comparing the twitter text feeds with the **word dictionaries** and retrieving the matching words.

To do this, i have first defined a function to count the number of positive and negative words that are matching with **my dictionary.**

Function pos_score is being made for counting the positive matching words

Result: p_count ## 491 positive tweets as per the prepared word dictionary

```
> p_count   ##491  positive tweets as per the prepared word dictionary
[1] 491
>
```

Function neg_score is being made for counting the negative matching words

Result: 726 negative tweets as per the prepared word dictionary

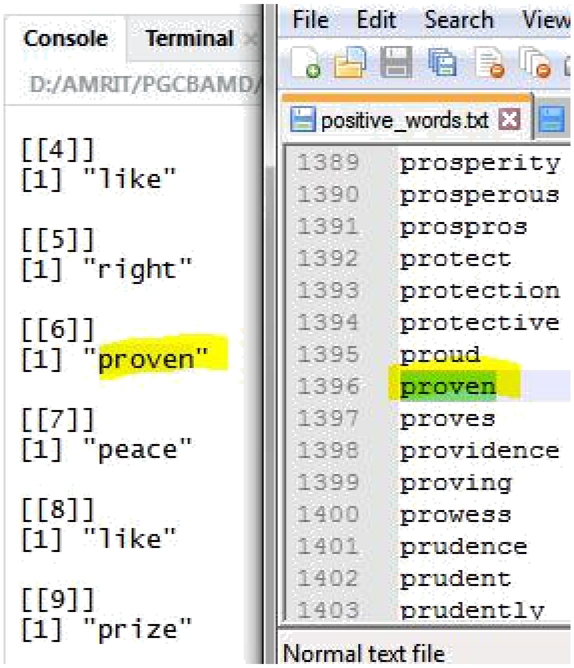
```
> ###Negative score
> ##neg_score for counting the negative matching words
> neg_score=function(tweet) {
+   neg.match=match(tweet,negative_words)
+   neg.match=!is.na(neg.match)
+   neg.match=sum(neg.match)
+   return(neg.match)
+ }
> negative_score=lapply(Twt_corpus,function(x) neg_score(x))
> ####to count the total number of negative words present in the tweets as per our dictionary
> n_count=0
> for (i in 1:length(negative_score)) {
+   n_count=n_count+negative_score[[i]]
+ }
> n_count   ##1425  ##negative tweets as per the prepared word dictionary
[1] 726
>
```

Step 5.4:-

In this step I am calculating/finding the positive and negative matching words as per dictionary.
i.e Finding the Positive Match:-> between twitter tweets and word dictionary

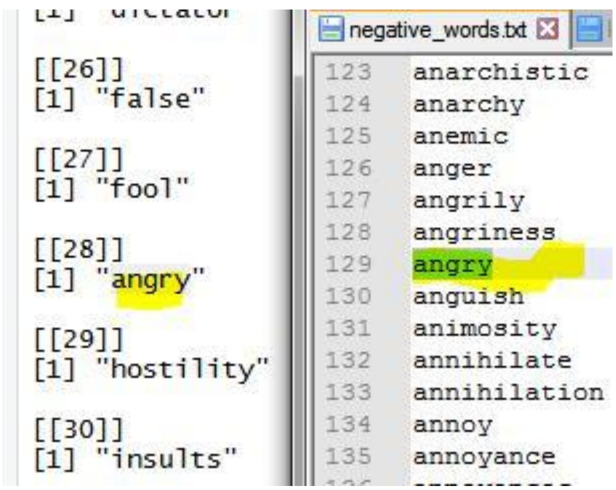
poswords: function is used to find the Positive Match.

Below is the screenshot which compares between my dictionary and the tweets



==Negwords:== function is used to find the negative Match.

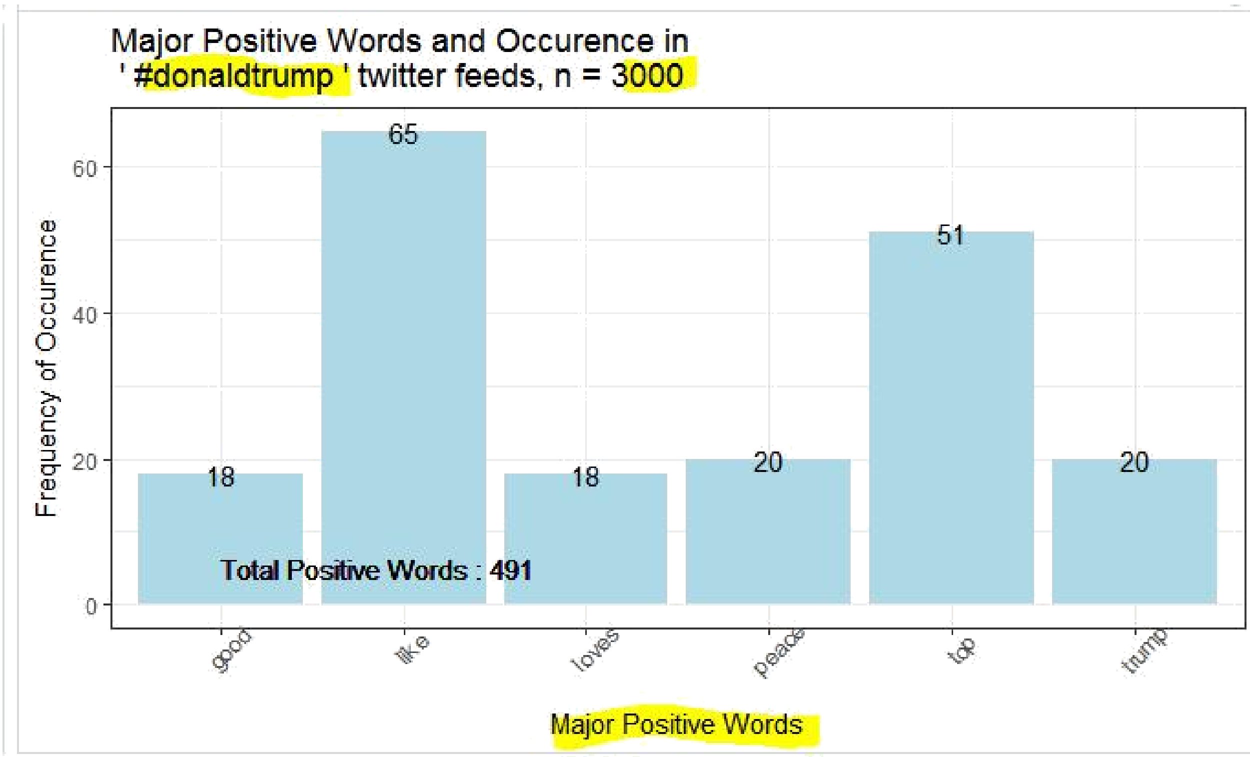
Below is the screenshot which compares between my dictionary and the tweets.



STEP 6:

6. Plotting high frequency negative and positive

words High Frequency Positive words:

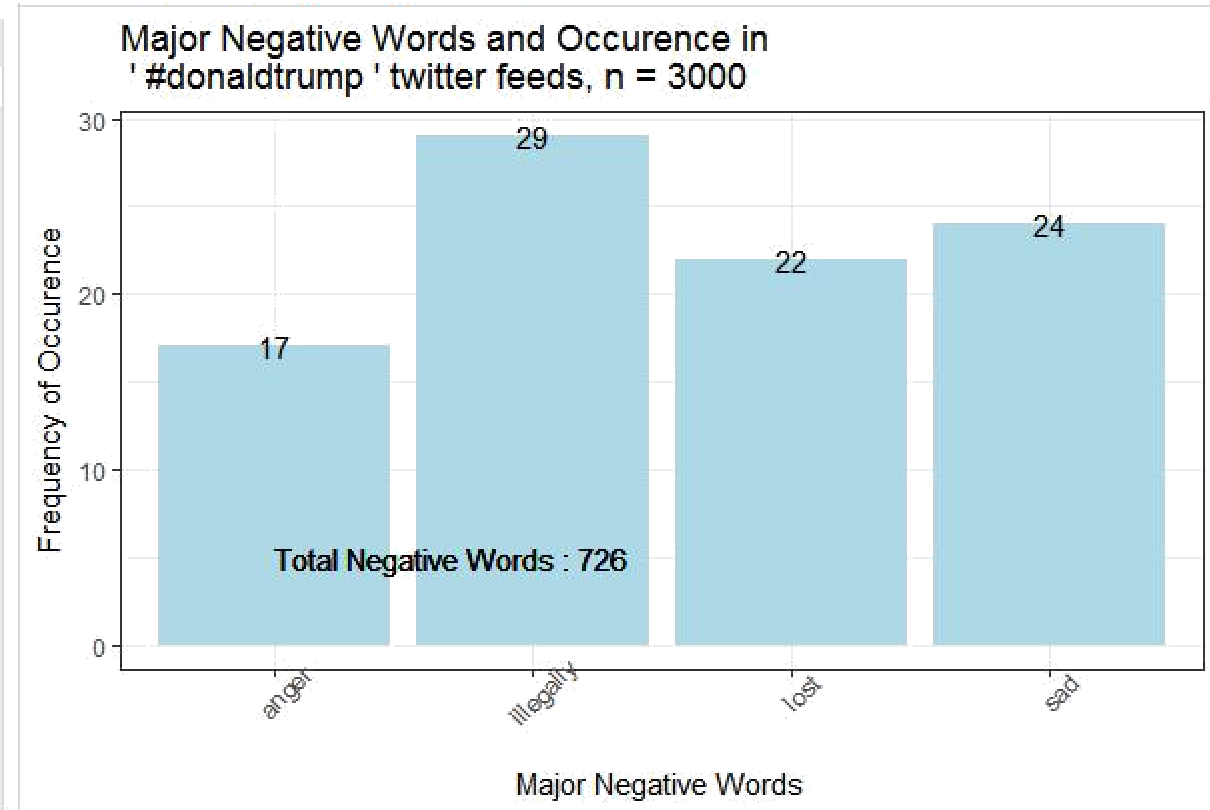


In this screenshot I have using ggplot2 package of R , I have displayed Major positive words in Donald trump tweets.

Total positive words= 491

LIKE is the word **which is maximum in the tweet** of **Donald Trump**. Followed by **TOP** and **PEACE.**

High Frequency negative words:



Total Negative words:726

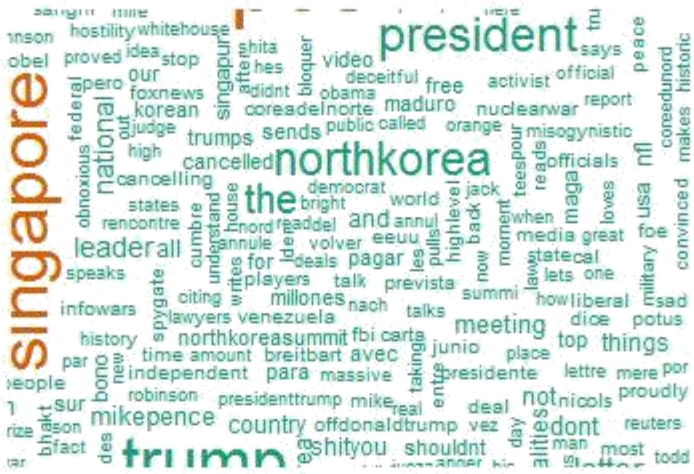
Among all ILLEGALLY is the negative word being used by Donald Trump. Followed by SAD.

STEP 7:

Twitter Analysis to calculate sentiments using sentiment package.

1. **Creating the corpus using vectorsource**
2. **Using tm_map to clean the corpus**
3. **Removing the stop words**
4. **Removing the custom words**
5. **Creating a Word Cloud of tweets using the wordcloud package.**

Word cloud:



STEP 8: Analyzing and plotting high frequency words 8

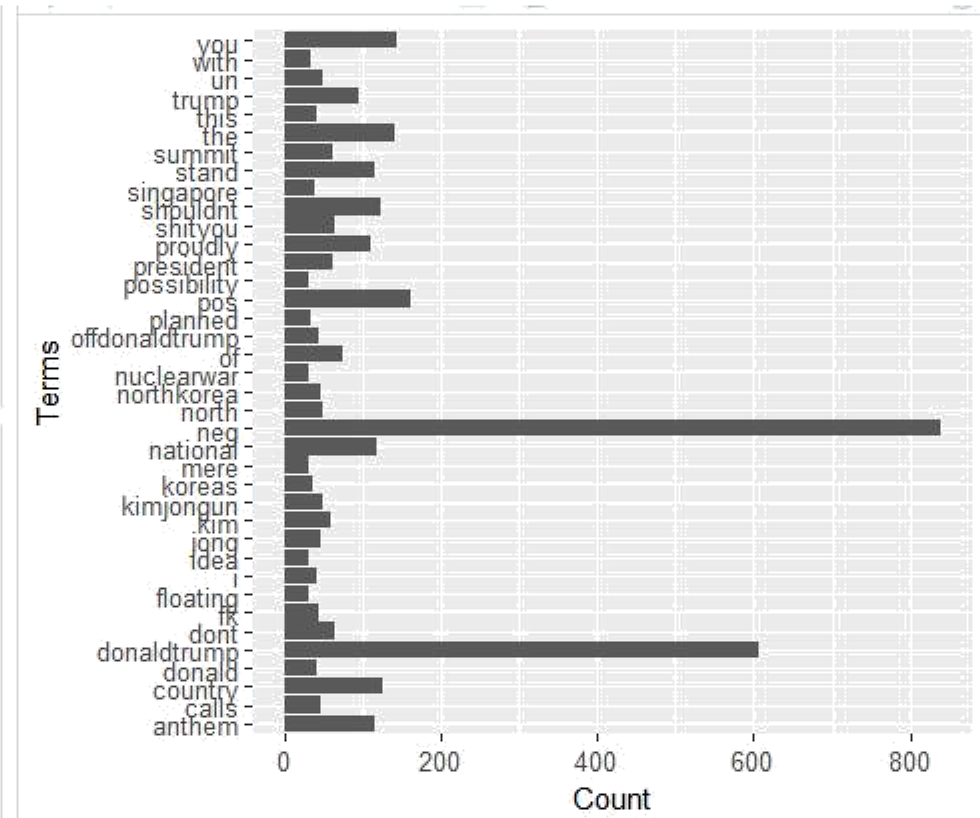
.Analyzing and plotting high frequency words

In this step, I am converting the word corpus into a document matrix using the function DocumentTermMatrix. The Document matrix is being used to examine most frequently occurring words.

Methods/Steps:

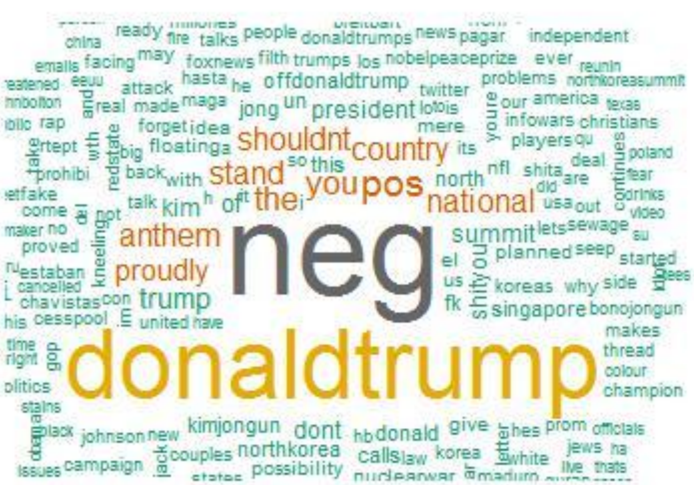
1. **Converting the word corpus into a Term Document Matrix using the function TermDocumentMatrix.**
2. **Calculating the term frequency**
3. **Selecting the tweets which has appeared over 30 times.**

Below is the screenshot:



WordClouds

➔ **Words appearing at least 6 times.**



STEP 9: Topic modeling

This is about what does Donald trump tweet about on a higher level.
Here I am using topic modeling to discover commonly words
And grouping them into 5 buckets.

```
                                                    Topic 1
           "donaldtrump, singapore, trump, to, kim"
                                                    Topic 2
      "donaldtrump, summit, north, jong, kimjongun"
                                                    Topic 3
    "donaldtrump, summit, northkorea, i, president"
                                                    Topic 4
"donaldtrump, kimjongun, trump, northkorea, donald"
                                                    Topic 5
         "donaldtrump, summit, donald, trump, north"
>
```

Topic 1: Donald Trump Singapore visit and meeting with KIM.
Topic 2: It is about summit where Donald trump and North Korean president.
Topic 3: Kimjon and trump and North Korea
Topic 4: Donald trump North Korea Kim jong
Topic 5: Summit Donald trump and North Korea in a summit

STEP 10: Sentiment Analysis using Sentiment package

Here I am displaying What is the overall attitude of donaldtrump
Here i am using sentiment package. It classifies every tweet as
#either “negative”, “neutral”, or “positive” based on the amount of positive/negative words.

```
> table(sentiment_analysis_package$polarity)

 neutral positive
       1        1
>
```

This count is as per sentiment package.

STEP 11: Sentiment Classification using NAIVE BAYES.

The prediction accuracy of a classification model is given by the proportion of the total number of correct predictions.

- Creating DocumentTermMatrix.
- Partitioning the or splitting the data

```
> dim(dtm.train)
[1]  700 2282
> ##Feature Selection
> dtm.train
<<DocumentTermMatrix (documents: 700, terms: 2282)>>
Non-/sparse entries: 385/1597015
Sparsity           : 100%
Maximal term length: 83
Weighting          : term frequency (tf)
> dtm.test
<<DocumentTermMatrix (documents: 300, terms: 2282)>>
Non-/sparse entries: 179/684421
Sparsity           : 100%
Maximal term length: 83
Weighting          : term frequency (tf)
>
```

```
> dim(dtm.train.nb)
[1] 700 181
> dtm.train.nb
<<DocumentTermMatrix (documents: 700, terms: 181)>>
Non-/sparse entries: 385/126315
Sparsity           : 100%
Maximal term length: 23
Weighting          : term frequency (tf)
>
```

```
> dtm.train.nb <- DocumentTermMatrix(corpus.clean.train, control=list(dictionary = one_freq))
> dim(dtm.train.nb)
[1] 700 181
> dtm.train.nb
<<DocumentTermMatrix (documents: 700, terms: 181)>>
Non-/sparse entries: 385/126315
Sparsity           : 100%
Maximal term length: 23
Weighting          : term frequency (tf)
> dtm.test.nb <- DocumentTermMatrix(corpus.clean.test, control=list(dictionary = one_freq))
> dim(dtm.train.nb)
[1] 700 181
>
```

TRAINING THE CLASSIFIER:

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = trainNB, y = df.train$class_sentiment,
    laplace = 1)

A-priori probabilities:
df.train$class_sentiment
      neg       pos
0.8585714 0.1414286

Conditional probabilities:
                        the
df.train$class_sentiment        neg        pos
                     neg 0.98175788 0.01824212
                     pos 0.98019802 0.01980198

                        democratic
df.train$class_sentiment         neg          pos
                     neg 0.991708126 0.008291874
                     pos 0.970297030 0.029702970
```

- **Using the NB classifier to build or to make predictions on the test set.**

```
# Use the NB classifier to built to make predictions on the test set.
pred <- predict(classifier, newdata=testNB)
head(pred)
pred
# Creating a truth table by tabulating the predicted class labels with the actual clas
table("Predictions"= pred,  "Actual" = df.test$class_sentiment )

length(df.test)
head(df.test$class_sentiment)

confusion_matrix <- confusionMatrix(pred, df.test$class)
```