Sentiment Analysis with R

Felipe Solares

28/12/2019

About

This is a Twitter Sentiment Analysis project using 'R' developed by Felipe Solares da Silva and is part of his *professional portfolio*.If you want to see more of my projects, go and check my portfolio at https://github.com/fsolares/professional-portfolio.

Contact: solares.fs@gmail.com

Acknowledgment

Thank you **Jaques D'Erasmo** (https://github.com/Jaquesd) and **Eduardo Passos** (https://github.com/eduardosteps), old friends and also Data Science students (12/19/2019) for your immeasurable contribution on this project, all your feedback, code sujection and support during my path gave me the strength to overcome this challenge. Congratualation for us, that was an ammazing and real team work.

Versions

- Windows OS release 10
- machine AMD64
- processor Intel64 Family 6 Model 58 Stepping 9, GenuineIntel
- platform x86_64-w64-mingw32
- arch x86_64
- os mingw32
- system x86_64, mingw32
- svn rev 76782
- language R
- version.string R version 3.6.1 (2019-07-05)

Sentiment Analysis using sentimentr and Naive Bayes Classifier

Project Purpose

Perform a sentiment analysis using a Naive Bayes Classifer and the sentimentr package to identify the sentiment from tweets related to a specific Netflix content (series, movies and documentaries, for example).

Step 1 - Importing Essential Packages and Modules

- 1 - rtweet - Allows you to connect to twitter and fetch the data.
- 2 - caret - Provides tools for machine learning activities.
- 3 - tm - A framework for text mining applications within R.
- 4 - sentimentr - Designed to quickly calculate text polarity sentiment at the sentence level.
- 5 - wordcloud - Designed to create a word cloud.
- 6 - e1071 - Package that bring the NaiveBayes classifier.
- 7 - ggplot2 - Create Elegant Data Visualisations Using the Grammar of Graphics.

- 8 - dplyr - Provides tools for manipulating datasets.

If you don't have any of these packages already installed in your rstudio please, run the code below!

```
chooseCRANmirror(graphics=FALSE, ind=1) # No need to run this part,
                                        #i'ts a R Markdown correction.

packs <- c('rtweet', 'caret', 'stringr', 'tm', 'sentimentr',
           'wordcloud','e1071','ggplot2','dplyr')
for(p in packs){
  install.packages(p)
}
```

If you have some of the packages, please run the code below giving the name of the package that are missing for you!

```
chooseCRANmirror(graphics=FALSE, ind=1) # No need to run this part,
                                        # i'ts a R Markdown correction.

install.packages('your missing package')
```

Loading Essential Packages

After the packages are intalled, make sure to run the code below to import the modules and make the required connections.

```
chooseCRANmirror(graphics=FALSE, ind=1) # No need to run this part,
                                        # i'ts a R Markdown correction.

lapply(packs, require, character.only = T)
```

Step 2 - Saving Twitter credentials and attempting connection

Storing your Keys

In order to protect your keys of being used by others and to optimize your access to the API the script below will save your keys into a token, using the *create_token()* function.

It should automatically save your token as an environment variable, so next time you start an R session (on the same machine), rtweet should automatically find your token using another function called *get_token()*. To make sure it works, restart your R session, run the following code, and again check to make sure the app name and api_key match.

First, you must gather your authorization keys and your app name.

- app = 'Your app name'
- consumer_key <- 'Your consumer key'
- consumer_secret <- 'Your consumer key'
- access_token <- 'Your access token'
- access_secret <- 'Your access secret'

Second, run the script below.

```
token <- create_token(app = 'Your app name',
                      consumer_key <- 'Your consumer key',
                      consumer_secret <- 'Your consumer key',
                      access_token <- 'Your access token',
                      access_secret <- 'Your access secret')
```

Attempting connection

Once the token was created, we run the *rtweet* function *get_token()* to search and establish connection. Next time that we run this script (on the same machine), this function should automatically find your token.

```r
tryCatch(
  expr = {
    get_token()
    message("Access Granted!")
  },
  error = function(e){
    message('Caught an error!')
    print(e)
  },
  warning = function(w){
    message('Caught an warning!')
    print(w)
  }
)
```

```
## Access Granted!
```

Step 3 - Fetching Tweets

The code below invoke the Twitter app and extracted data about a netflix's show called The Witcher. This code wil return a data frame with tweets text (excluding re-tweets) and all metadata provided by Twitter API.

```r
tweets <- search_tweets('"The witcher netflix"', n = 1000,
                        include_rts = F, lang = 'en',retryonratelimit = TRUE)

# We will now see what format we have got and what steps do we need to clean the data.

View(tweets)

# The field 'tweets$text' contains the tweet part, hashtags, and URLs.
# We need to remove hashtags and URLs from the text field so that we are
# left only with the main tweet # to run our sentiment analysis with sentimentr.
```

IMPORTANT!

Twitter API documentation recommends limiting searches to 10 keywords and operators. Complex queries may also produce API errors preventing recovery of information related to the query. It should also be noted Twitter's search API does not consist of an index of all Tweets. At the time of searching, the search API index includes between only 6-9 days of Tweets.

Number of tweets returned will often be less than what was specified by the user. This can happen because (a) the search query did not return many results (the search pool is already thinned out from the population of tweets to begin with), (b) because user hitting rate limit for a given token, or (c) of recent activity (either more tweets, which affect pagination in returned results or deletion of tweets).

Preprocessing Tweets

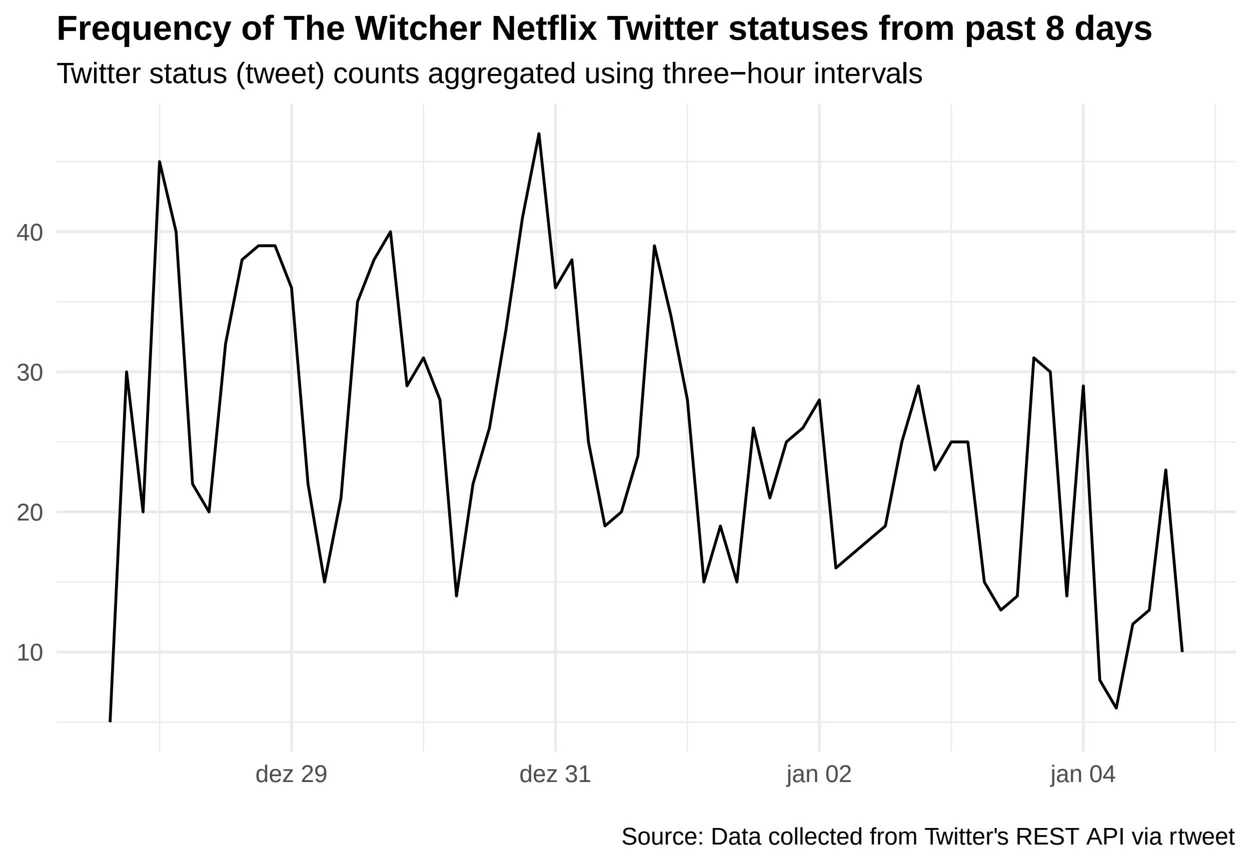
The preprocess step will use few R packages and tools to work with strings as detailed bellow:

```r
tweets.df <- gsub("https.*","", tweets$text) # Remove URLs.
tweets.df <- removePunctuation(tweets.df)    # Remove Punctuation (@#,.'´!?...).
tweets.df <- removeNumbers(tweets.df)         # Remove Numbers.
tweets.df <- stripWhitespace(tweets.df)       # Remove white spaces(except between words).
tweets.df <- replace_emoji(tweets.df)         # Replace emoji.
tweets.df <- replace_emoticon(tweets.df)      # Replace emoticons.

# Revome the pattern left by the replace functions.
tweets.df <- gsub("<[a-z|0-9][0-9|a-z]>","", tweets.df)

# Convert to lower case
tweets.df <- tolower(tweets.df)
```

Frequency Analysyis

Let's quickly visualize the frequency of tweets over time about the query selected for the search.

```
tweets %>%
  ts_plot("3 hours") +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold")) +
  labs(x = NULL, y = NULL,
       title = "Frequency of The Witcher Netflix Twitter statuses from past 8 days",
       subtitle = "Twitter status (tweet) counts aggregated using three-hour intervals",
       caption = "\nSource: Data collected from Twitter's REST API via rtweet")
```



Step 4 - Sentiment Analysis with sentimentr

Why Sentimentr?

The Sentimentr package for R is immensely helpful when it comes to analyzing text for psychological or sociological studies. Its first big advantage is that it makes sentiment analysis simple and achievable within a few lines of code. Its second big advantage is that it corrects for inversions, meaning that while a more basic sentiment analysis would judge "I am not good" as positive due to the adjective good, sentimentr recognizes the inversion of good and classifies it as negative.

Getting sentiment scores by sentences

This function allows the user to easily extract the polarity score (positive, negative or neutral) of each tweet in a data set. The score is represented by either a positive or a negative number.

```r
sentences <- get_sentences(tweets.df)
sentiments <- sentiment(sentences)

# Integrating your sentiment scores into the original dataset
sentiment.df <- cbind(tweets.df,sentiments)
```

Getting emotion scores by sentences

We will first try to get the emotion score for each of the tweets. *sentimentr* breaks the emotion into 8 different emotions – anger, anticipation, disgust, fear, joy, sadness, surprise, trust.

```r
emotion.df <- emotion(sentences)
```

Getting the most positive and most negative tweets

```r
# most positive
most.positive <- sentiment.df[sentiment.df$sentiment == max(sentiment.df$sentiment)]
most.positive$tweets.df

# most negative
most.negative <- sentiment.df[sentiment.df$sentiment == min(sentiment.df$sentiment)]
most.negative$tweets.df
```

Preparing sentiment.df and emotion.df for futher analysis

Let's filter the dataframes, change labels and column names to ease our analysis.

```r
sentiment.df <- sentiment.df %>%
                mutate(label = ifelse(sentiment.df$sentiment > 0, 'pos',
                                      ifelse(sentiment.df$sentiment < 0,
                                             'neg', 'neu'))) %>%
                distinct() # Cleaning possible duplicates

emotion.df <- emotion.df %>%
  filter(emotion_count != 0) %>%
  filter(emotion_type %in% c("anger", "anticipation", "disgust",
                             "fear", "joy", "sadness", "surprise", "trust")) %>%
  select(emotion_type, emotion_count) %>%
  group_by(emotion_type) %>%
  summarise(tot = n())
```

Counting positive, negative and neutral tweets

```r
table(sentiment.df$label)
```

```
##
## neg neu pos
## 358 662 641
```

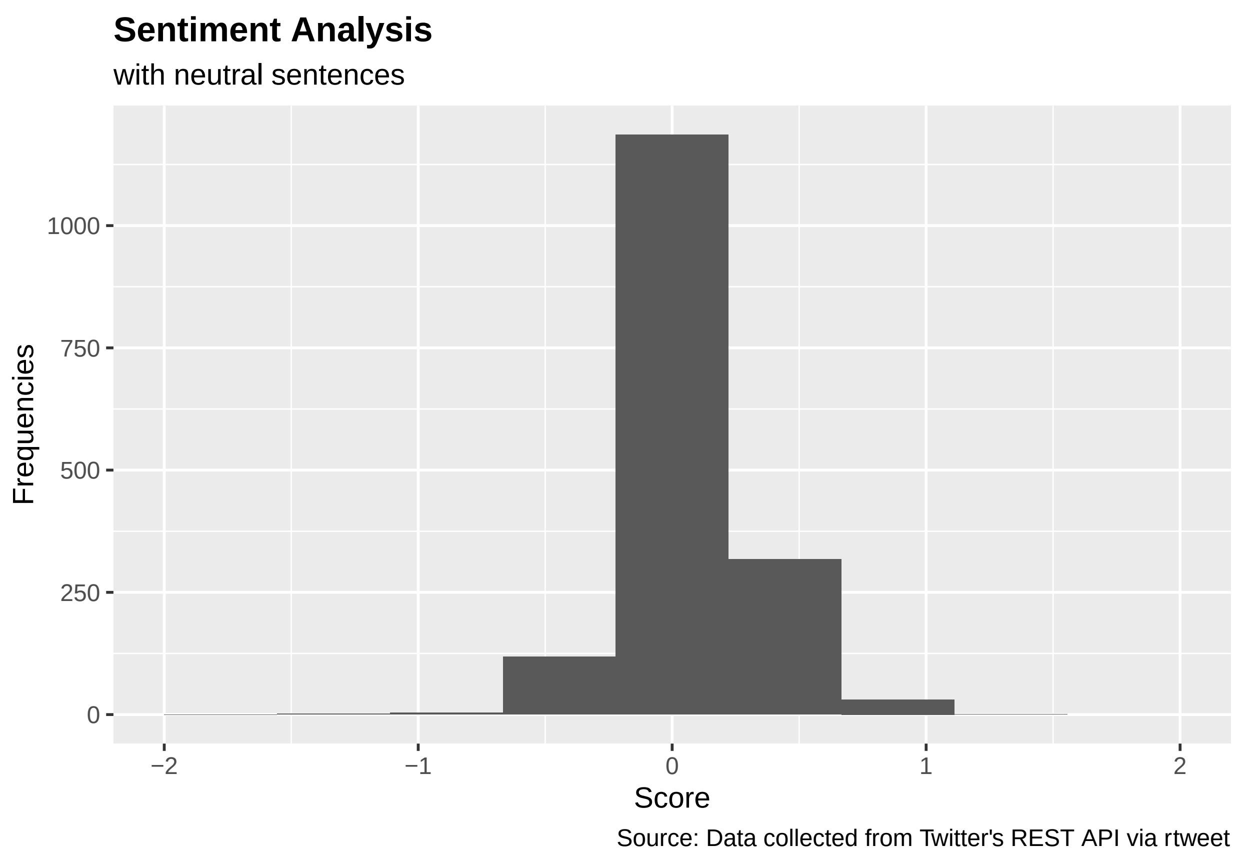
Step 5 - Analysing

In this step, we're going to try different approachs in order to conclude if the netflix's show had a good acceptance from the viewers. All used techniques will be listed below: - Graphical Analysis - Numerical Analysis - Wordcloud Analysis

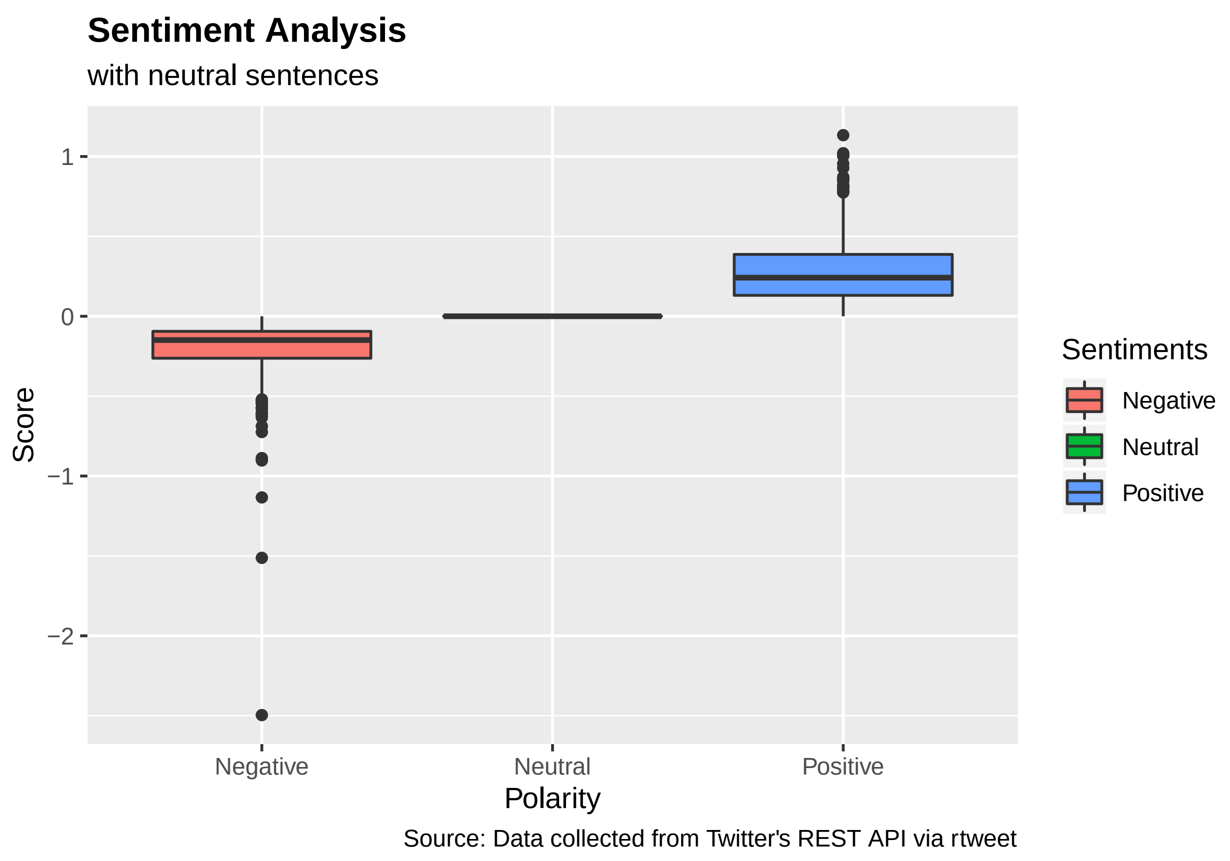
Graphical Analysis

Plotting Histograms, Boxplots and Barplots considering all tweets.

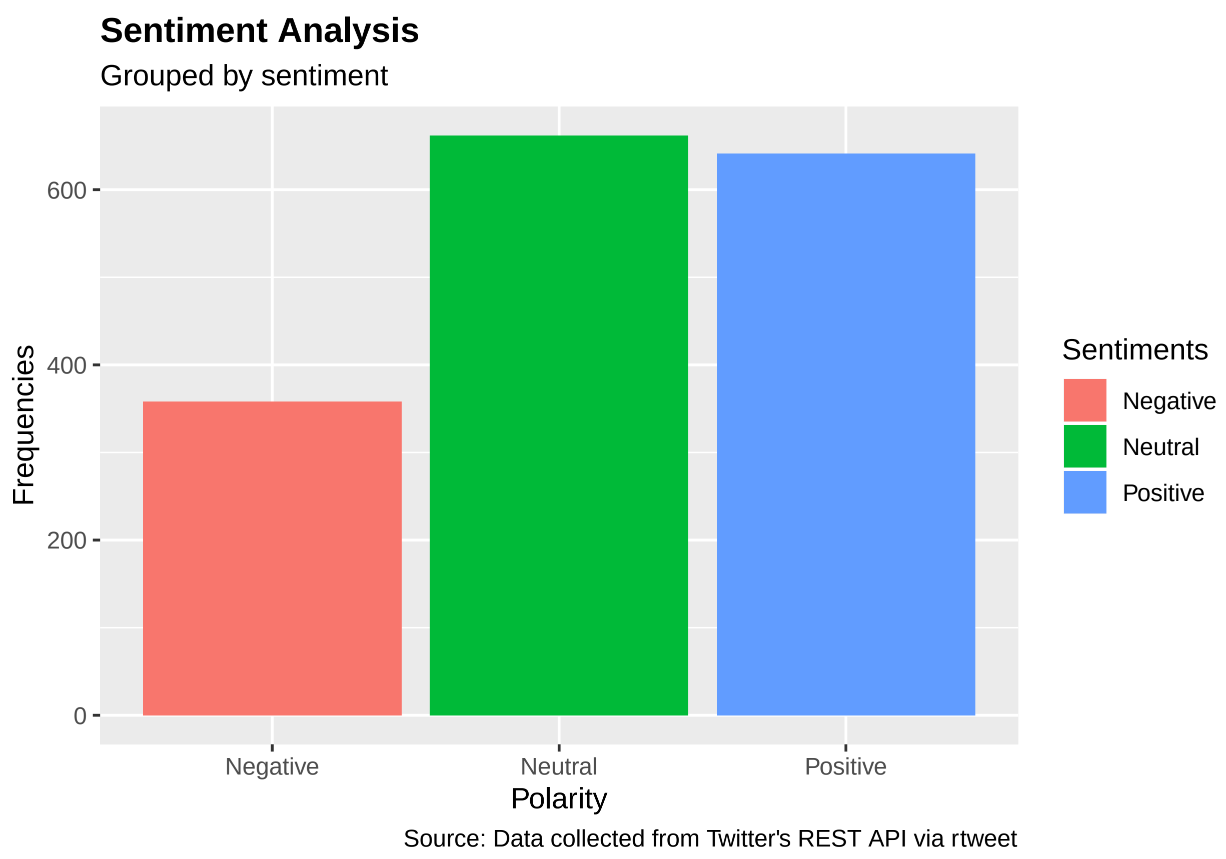
```
# Histogram with neutral positioning
sentiment.df %>%
  ggplot(aes(x = sentiment)) +
  geom_histogram(bins = 10) +
  theme(plot.title = element_text(face = "bold")) +
  xlim(-2,2) +
  labs(title="Sentiment Analysis", subtitle="with neutral sentences",
       caption = 'Source: Data collected from Twitter\'s REST API via rtweet',
       y="Frequencies", x="Score")
```



```
# Boxplot with neutral positioning
sentiment.df %>%
  ggplot(aes(x = label, y = sentiment)) +
  geom_boxplot(aes(fill = label)) +
  theme(plot.title = element_text(face = "bold")) +
  labs(title="Sentiment Analysis", subtitle="with neutral sentences",
       caption = 'Source: Data collected from Twitter\'s REST API via rtweet',
       x="Polarity", y ="Score") +
  scale_x_discrete(breaks = c("neg", "neu", "pos"),
                   labels = c("Negative", "Neutral", "Positive")) +
  scale_fill_discrete(name = "Sentiments", labels = c("Negative", "Neutral", "Positive"))
```

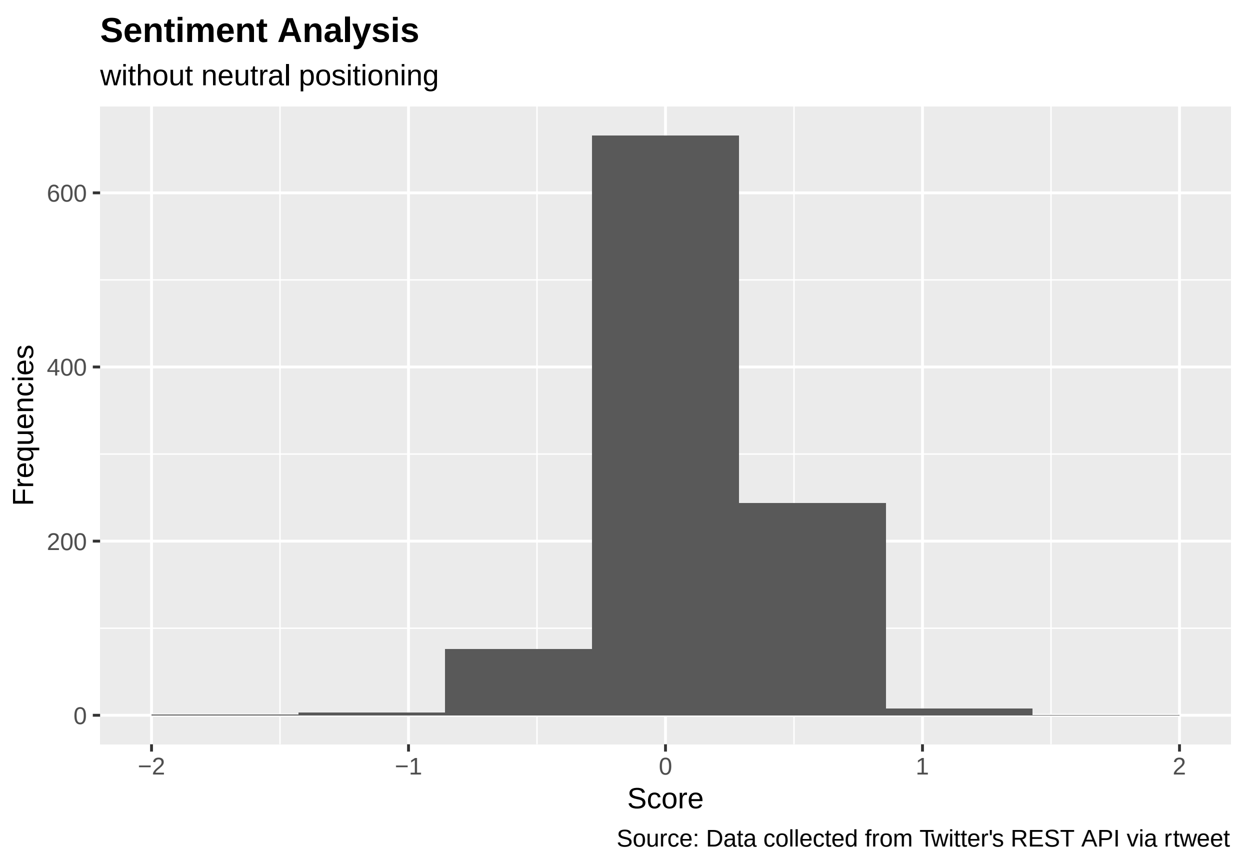


```
# Barplot with neutral positioning

sentiment.df %>%
  ggplot(aes(x = label, fill= label)) +
  geom_bar()+
  theme(plot.title = element_text(face = "bold")) +
  labs(title="Sentiment Analysis", subtitle="Grouped by sentiment",
       caption = 'Source: Data collected from Twitter\'s REST API via rtweet',
       y="Frequencies", x="Polarity") +
  scale_x_discrete(breaks = c("neg", "neu", "pos"),
                   labels = c("Negative", "Neutral", "Positive")) +
  scale_fill_discrete(name = "Sentiments", labels = c("Negative", "Neutral", "Positive"))
```

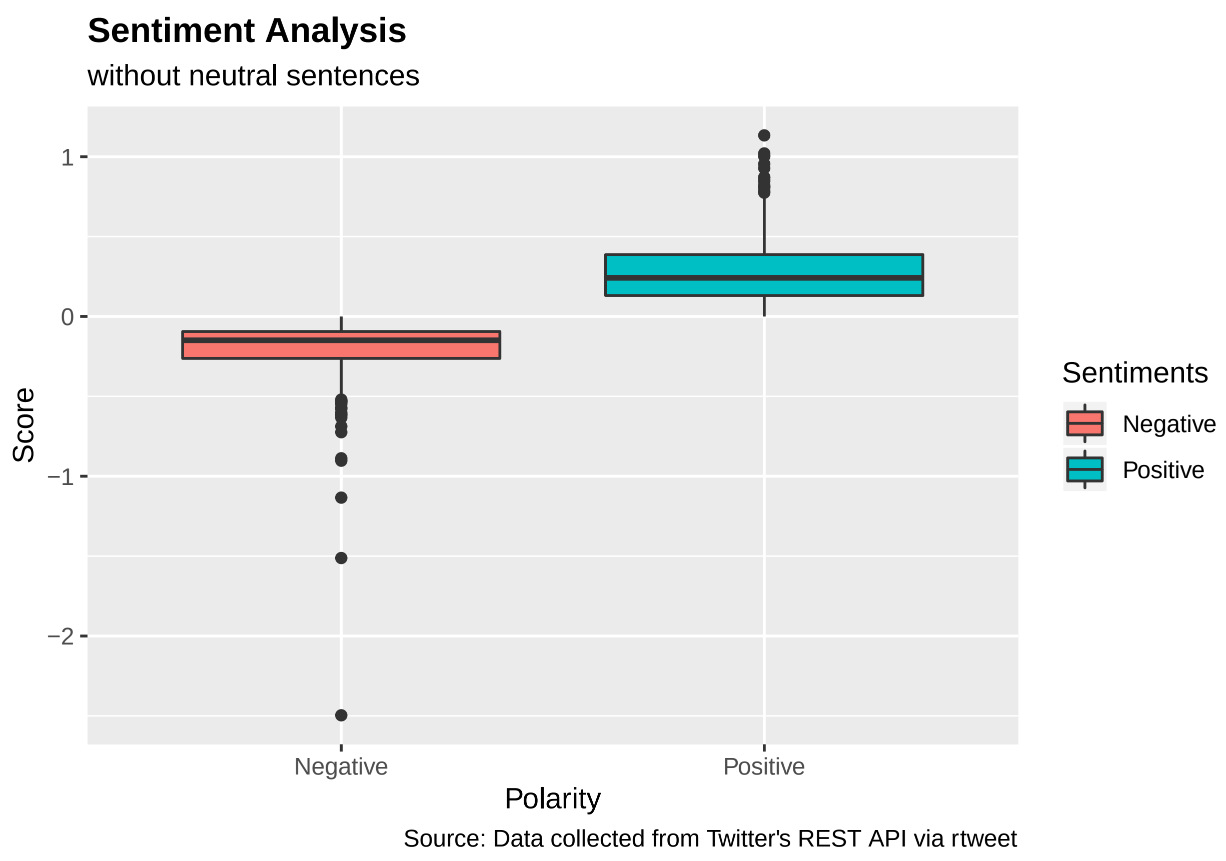


According to the data, plotted in the graphs above, we noticed a overwhelming impact of neutral positioning in our results. So, in order to obtain more accurate results we decided to re-run the analysis without neutral tweets.

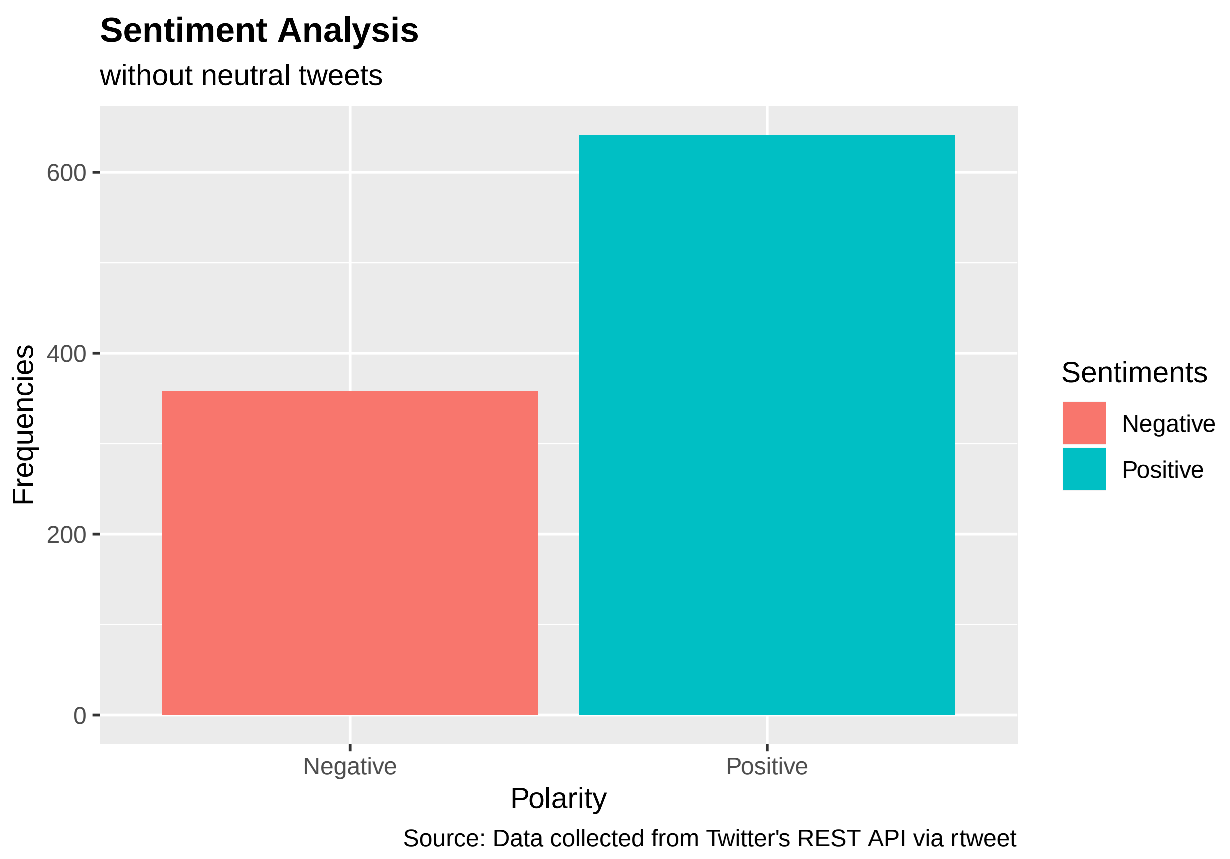
```
# Histogram without neutral positioning

sentiment.df %>%
  filter(sentiment != 0) %>%
  ggplot(aes(x = sentiment)) +
  geom_histogram(bins = 8) +
  xlim(-2,2) +
  theme(plot.title = element_text(face = "bold")) +
  labs(title="Sentiment Analysis", subtitle="without neutral positioning",
       caption = 'Source: Data collected from Twitter\'s REST API via rtweet',
       y="Frequencies", x="Score")
```



```r
# Boxplot without neutral positioning

sentiment.df %>%
  filter(sentiment != 0) %>%
  ggplot(aes(x = label, y = sentiment)) +
  geom_boxplot(aes(fill = label)) +
  theme(plot.title = element_text(face = "bold")) +
  labs(title="Sentiment Analysis", subtitle="without neutral sentences",
       caption = 'Source: Data collected from Twitter\'s REST API via rtweet',
       x="Polarity", y ="Score") +
  scale_x_discrete(breaks = c("neg", "pos"),
                   labels = c("Negative", "Positive")) +
  scale_fill_discrete(name = "Sentiments", labels = c("Negative", "Positive"))
```



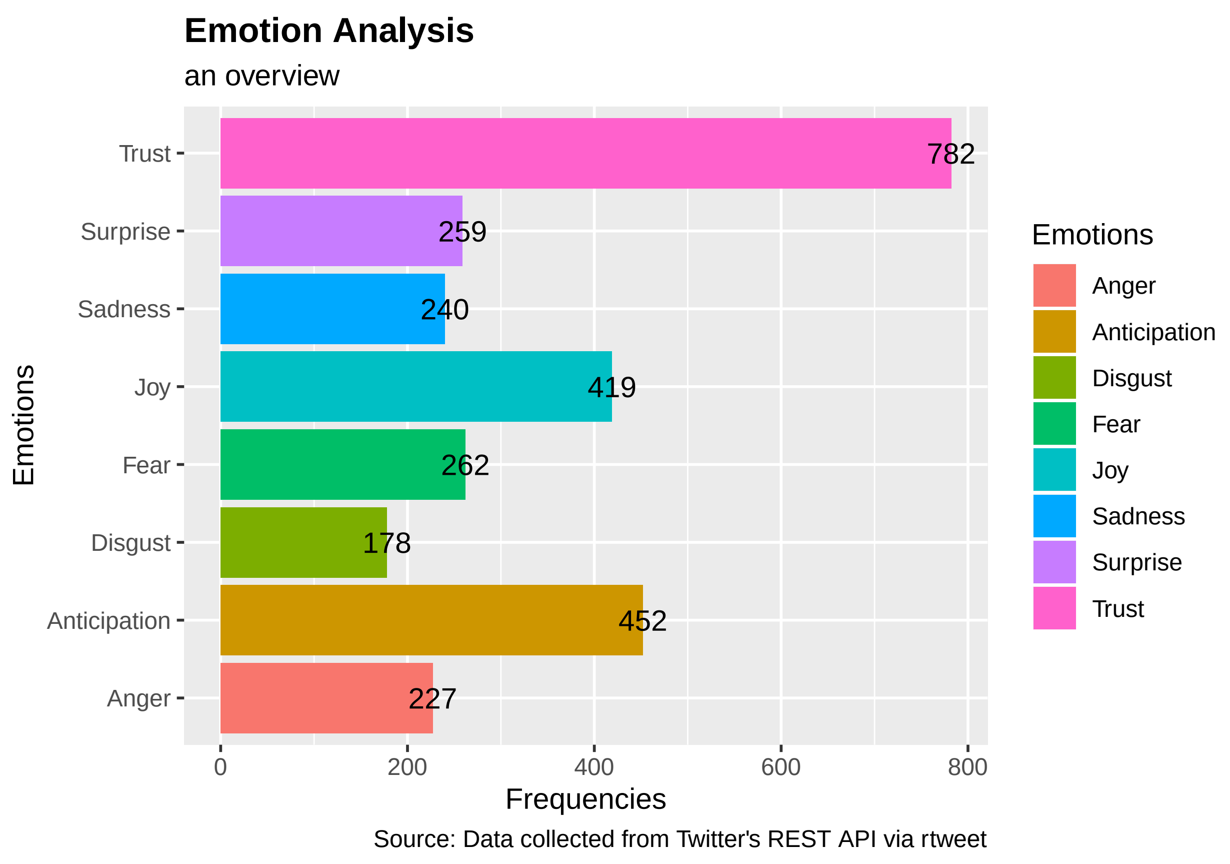
```r
# Barplot without neutral positioning

sentiment.df %>%
  filter(label != 'neu') %>%
  ggplot(aes(x = label, fill= label)) +
  geom_bar() +
  theme(plot.title = element_text(face = "bold")) +
  labs(title="Sentiment Analysis", subtitle="without neutral tweets",
       caption = 'Source: Data collected from Twitter\'s REST API via rtweet',
       y="Frequencies", x="Polarity") +
  scale_x_discrete(breaks = c("neg", "pos"),
                   labels = c("Negative", "Positive")) +
  scale_fill_discrete(name = "Sentiments", labels = c("Negative", "Positive"))
```



Removing the neutral tweets from our analysis, it's possible to observe a very clear dominance of positive tweets over the negatives ones in this scenario.

Next, let's plot the emotions extracted from all fetched tweets.

```
emotion.df %>%
  ggplot(aes(x = emotion_type, y = tot, fill= emotion_type)) +
  geom_bar(stat = 'identity') +
  geom_text(aes(label = tot), position = position_dodge(0.9))+
  coord_flip()+
  theme(plot.title = element_text(face = "bold")) +
  labs(title="Emotion Analysis", subtitle="an overview",
       caption = 'Source: Data collected from Twitter\'s REST API via rtweet',
       fill = 'Emotions',
       y="Frequencies", x="Emotions") +
  scale_fill_discrete(labels = c("Anger", "Anticipation", "Disgust",
                                 "Fear", "Joy", "Sadness", "Surprise", "Trust")) +
  scale_x_discrete(breaks = c("anger", "anticipation", "disgust",
                   "fear", "joy", "sadness", "surprise", "trust"),
                   labels = c("Anger", "Anticipation", "Disgust",
                              "Fear", "Joy", "Sadness", "Surprise", "Trust"))
```



Numerical Analysis

Trying a different approach to ratify the results obtained above. Let's find out the percentage of positive, negative and neutral tweets.

```
nppos <- (table(sentiment.df$label)['pos'] / sum(table(sentiment.df$label))) * 100
npneg <- (table(sentiment.df$label)['neg'] / sum(table(sentiment.df$label))) * 100
pneut <- (table(sentiment.df$label)['neu'] / sum(table(sentiment.df$label))) * 100

cat(sprintf('%.2f%% of the analyzed tweets were positive',nppos),
    sprintf('%.2f%% of the analyzed tweets were negative',npneg),
    sprintf('%.2f%% of the analyzed tweets were neutral',pneut),fill = 45)
```

```
## 38.59% of the analyzed tweets were positive
## 21.55% of the analyzed tweets were negative
## 39.86% of the analyzed tweets were neutral
```

Now, we're going to acquire the percentage only of the positive and negative tweets.

```
noneutral <- sentiment.df %>%
  filter(label != 'neu')

ppos <- (table(noneutral$label)['pos'] / sum(table(noneutral$label))) * 100
pneg <- (table(noneutral$label)['neg'] / sum(table(noneutral$label))) * 100



cat(sprintf('%.2f%% of the analyzed tweets were positive',ppos),
```

```r
    sprintf('%.2f%% of the analyzed tweets were negative',pneg),fill = 45)
```

```
## 64.16% of the analyzed tweets were positive
## 35.84% of the analyzed tweets were negative
```

Wordcloud Analysis

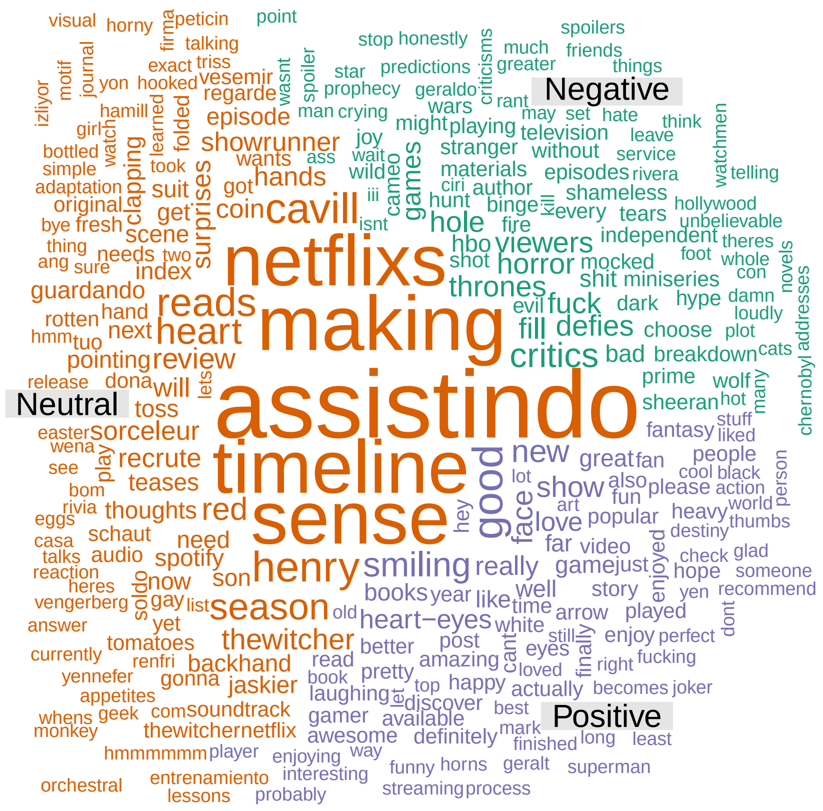
At this point, it's possible to conclude our analysis about the acceptance of the netflix's show. Now we're going to perform a deeper analysis, at word level, to check if *sentimenr* perfomed a good segregation between positive and negative words.

First, we're going to tranform the data to a format required by *wordcloud* functions.

```r
# transforming
sentiment.df <- sentiment.df %>%
  group_by(label) %>%
  summarise(pasted = paste(tweets.df,collapse = ' '))

sentiment.df$pasted <- removeWords(sentiment.df$pasted,stopwords(kind = 'en'))
sentiment.df$pasted <- stripWhitespace(sentiment.df$pasted)
toremove <- c('witcher','netflix','series','watching')
sentiment.df$pasted <- removeWords(sentiment.df$pasted,toremove)

# Creating Corpus
tweetcorpus <- Corpus(VectorSource(sentiment.df$pasted))
tdm = TermDocumentMatrix(tweetcorpus)
tdm = as.matrix(tdm)
colnames(tdm) <-  c("Negative", "Neutral", 'Positive')
```

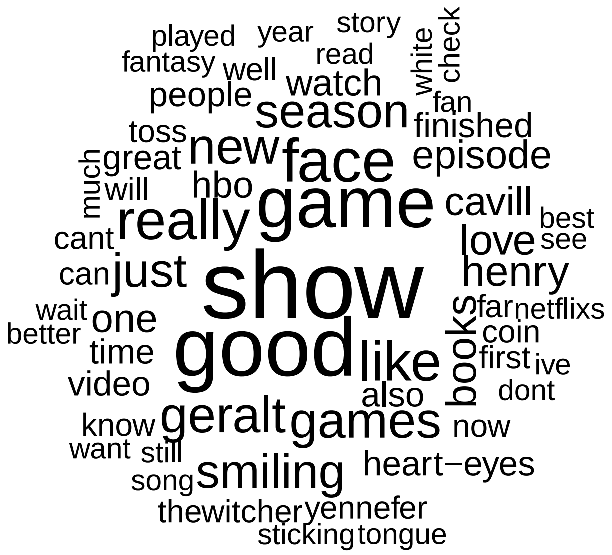
Making the comparison cloud.

```r
comparison.cloud(tdm, colors = brewer.pal(3, 'Dark2'),
                 scale = c(3,.5), random.order = FALSE, title.size = 1)
```



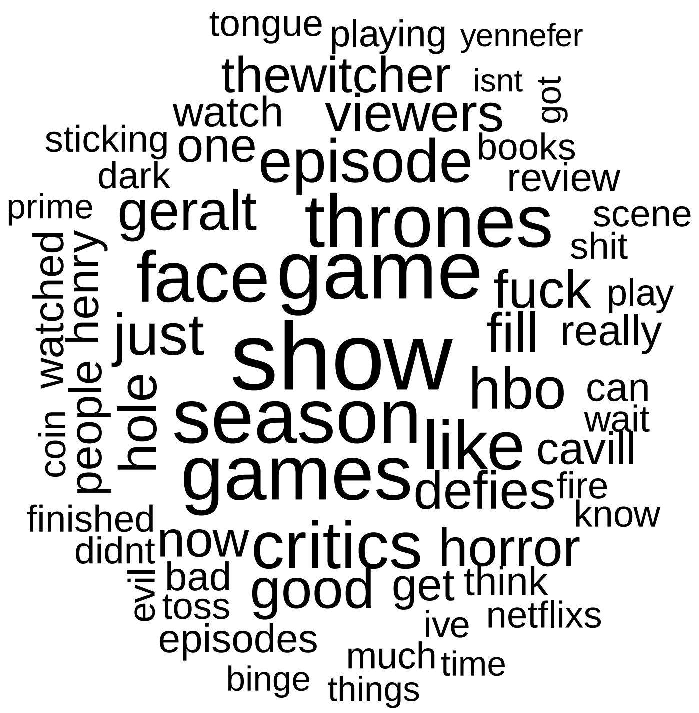
Making the Positive cloud.

```
wordcloud(tweetcorpus[3],
          min.freq = 2,
          scale = c(3,0.5),
          random.color = F,
          max.word = 60,
          random.order = F)
```



Making the Negative cloud.

```
wordcloud(tweetcorpus[1],
          min.freq = 2,
          scale = c(3,0.5),
          random.color = F,
          max.word = 60,
          random.order = F)
```



RESULTS!

The charts shown us a clear dominance of positive tweets over negative ones. In the emotion field, it is possible to conclude that emotions such as joy, trust and surprise (positive emotions) surpasses the negatives sentiments. With all that was said before it is rigth to conclude that the netflix's show was a success. Among 1000 tweets the positive ones stands out.