2018

GAMES OF THRONES TWITTER ANALYTICS

SOCIAL MEDIA INTELLIGENCE

MUTHU GOVINDHAN, KANIKA SHARMA, BOMIN XIE, JING ZHANG

Contents

1. Introduction

In our project, we use Twitter this social media tool to analysis Game of Thrones. In this part, we will introduce the Twitter and Game of Thrones page.

Twitter is a popular microblogging site and it is online news and social networking service where users post and interact with messages, known as "tweets". Begging of the Twitter there is a limit to use the text 140 characters but on November 7, 2017, it was double to 280 characters for all languages expect Japanese, Korean and Chinese. The Twitter is website interface, Short message Service (SMS) and mobile device application software (App).

Tweets are frequently used to express a Twitter's emotion on a particular subject. Nowadays the Twitter is very famous and interactive and a lot of people using Twitter like student, youngster, politicians, celebrities and also educators.

The Twitter now often be used to promoting new Games and music. we are planning to analysis Twitter sentiment analysis for famous tweets page. and we select interesting and interactive Twitter page it is a Game of Thrones.

The Game of Thrones is an American fantasy drama television series, which is adaption song of Ice and Fire. It is based on George R.R Martin's series of fantasy novel. The first part they released 2011 for United Kingdom, Canada, Malta, Morocco, Spain and the united states. Then the HBO channel in United States planned to telecast on April 17, 2011, and Game of thrones now finished 7th season. The 8th season starting by 2019.

Twitter Game of Thrones is very famous and you can see how much tweets, following, followers, Likes.

Tweets – 60K Following – 40.5K Followers – 7.04M Likes – 8,480

It is interesting we are working this page because very huge data for this page to analysis. And we are trying to analysis text mining word cloud, sentiment analysis, Emotion out of words.

2.Fetching Twitter feed using Twitter API

In the first part, we need API access. The flowing step will help you set up your twitter account and be able to use the Twitter API.

If you have already twitter account just log in your account and click "Creat New App" in the right-top side. If you don't have twitter account, the first step you should do is creating an account.

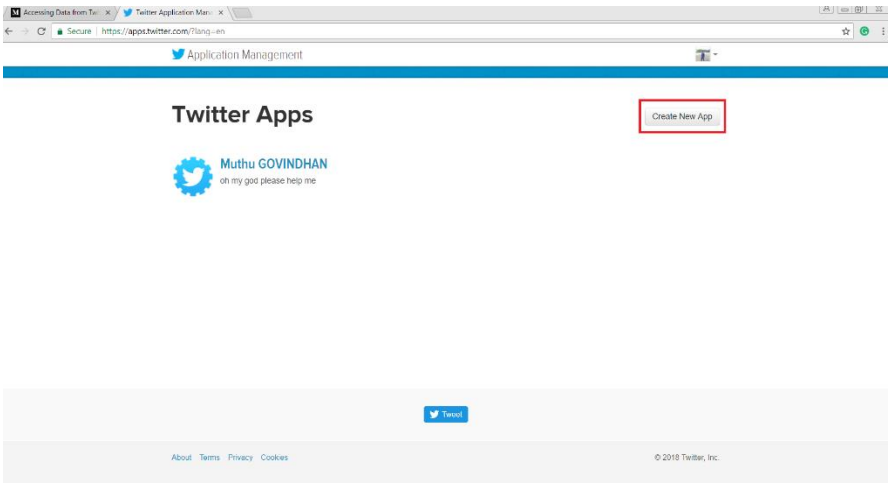

Figure 1

Now you get an application page like I attached blow and you can fill in the application details.

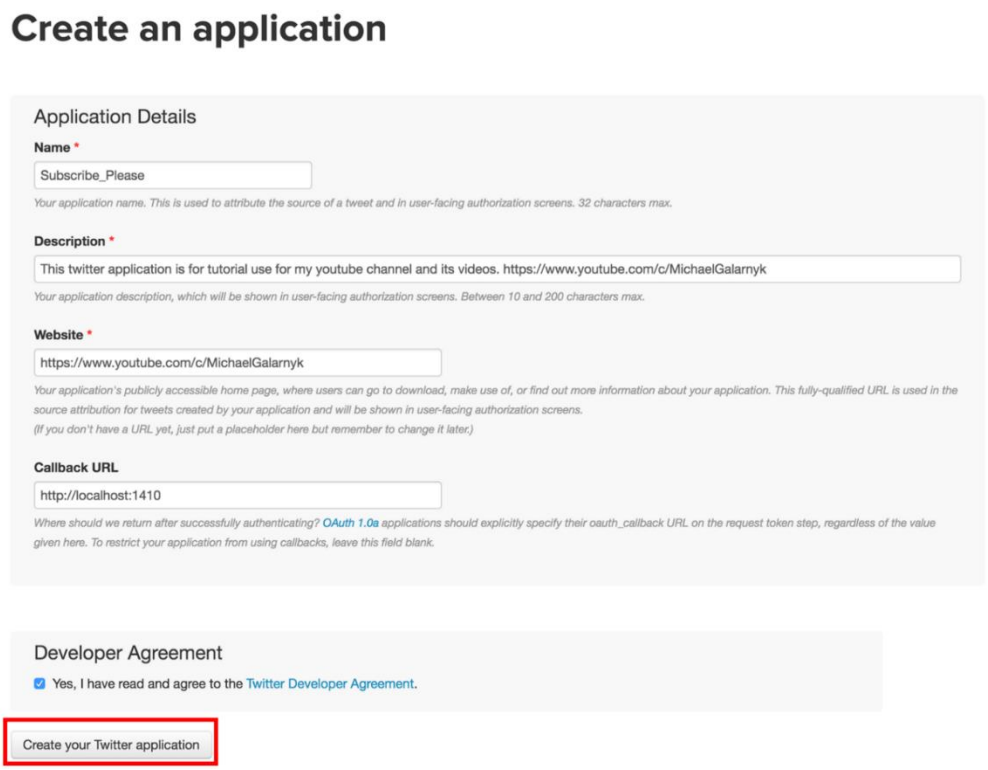

Figure 2

Now you get your app page, in the figure 3, it is important to copy and save some details, using it as the basis. And also the application setting you can see many option is there, like Callback URL, Sign in with twitter, App-only authentication, Request token URL, Authorize URL and Access token URL, these all you can use in your R code to connected twitter.

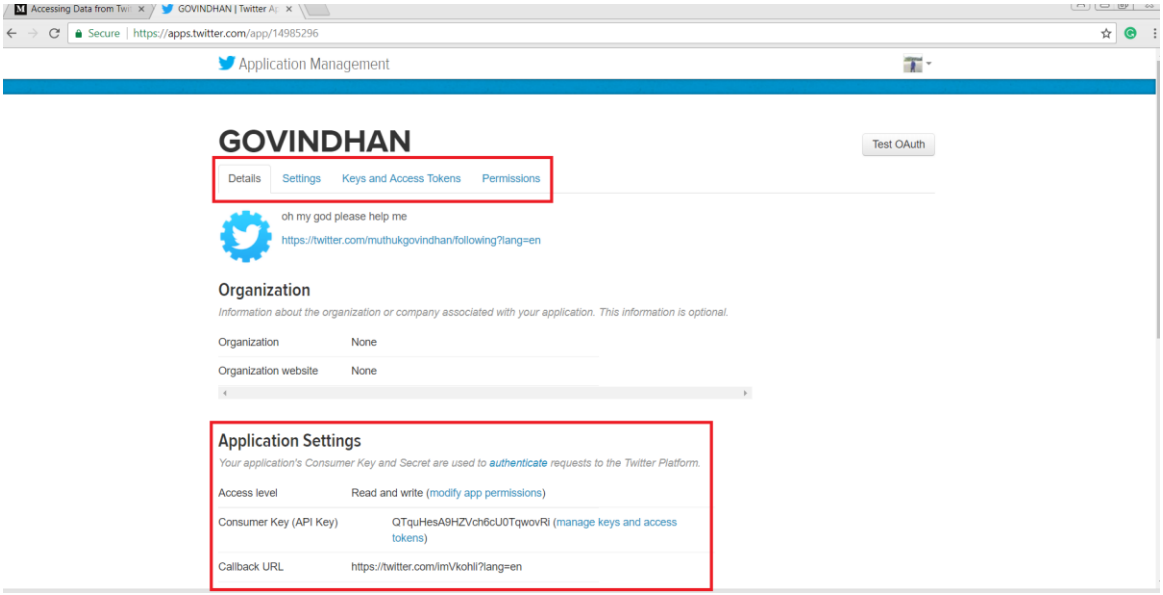

Figure 3

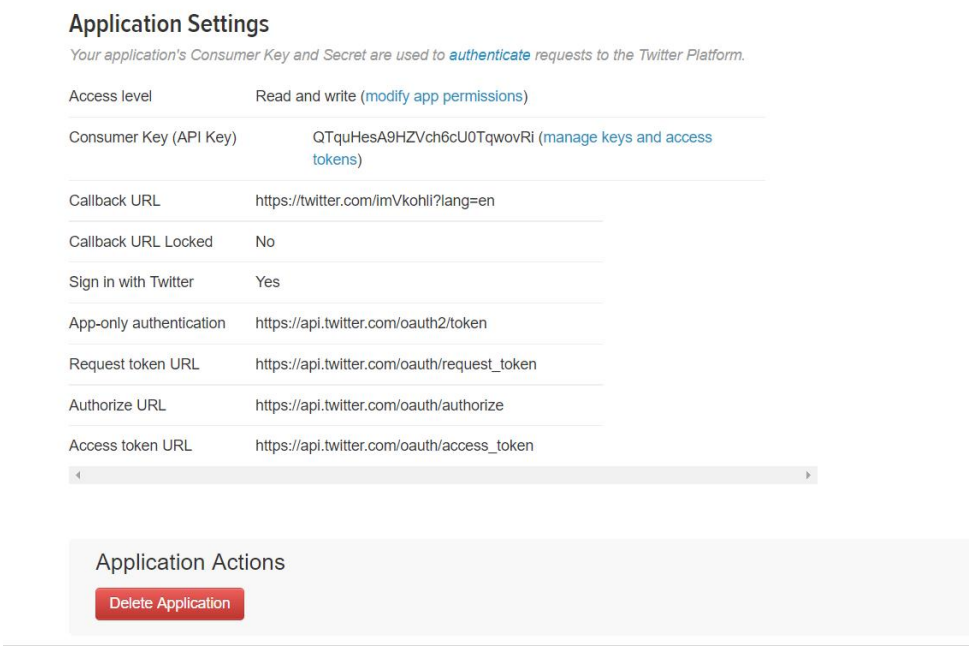

Figure 4

Then you click details button, and then there will be your information. You click Keys and access token button is showing the access keys.

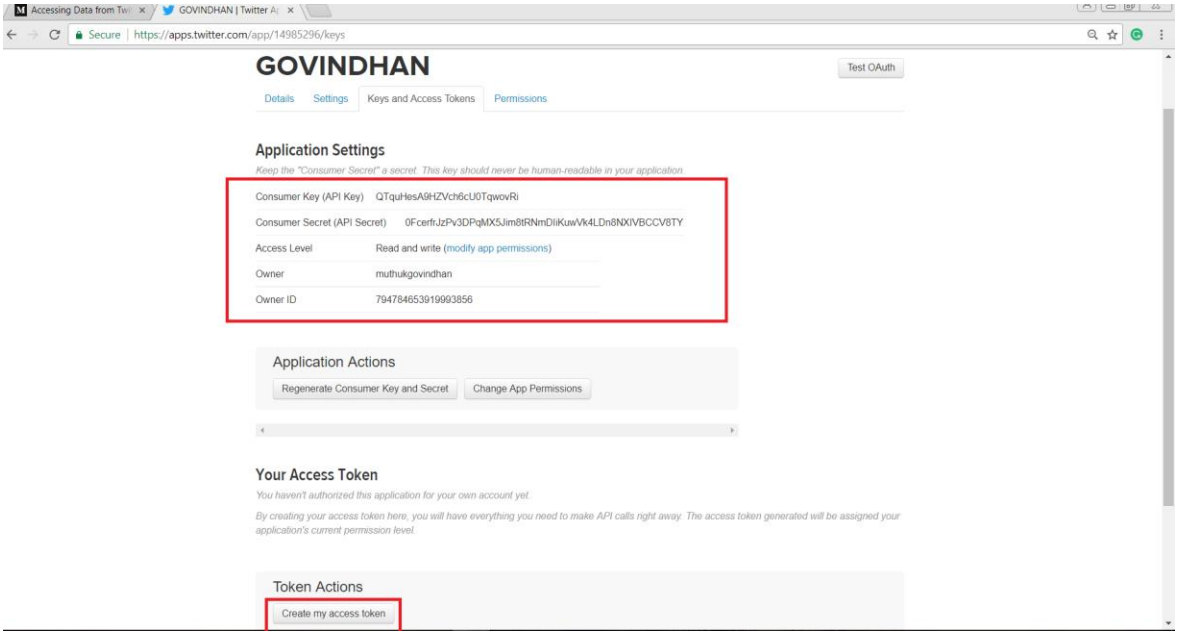

Figure 5

You can get your access token like Access token, Access Token Secret code, which can be pasted in your R code.

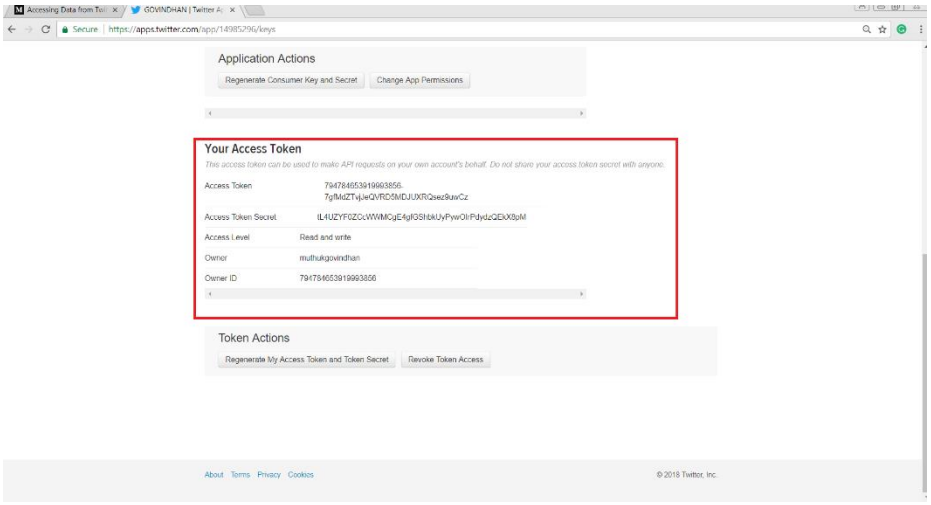

Figure 6

R Studio Set Up

R uses the **twitteR** library, an R based Twitter client that handles communication with the Twitter API.

```
library(twitteR)
library(reshape)
library(wordcloud)
library(igraph)
library("base64enc")
library("ROAuth")
library("devtools")
library("memoise")
library("whisker")
library("rstudioapi")
library("git2r")
library("withr")
library("rjson")
library("bit64")
library ("httr")
library ("httpuv")
library(qdapRegex)
library(RTextTools)
library(tm)
library(tidyr)
library(dplyr)
```

Figure 7

3.AUTHENTICATION

Twitter uses Open Authentication (OAuth) to grant access to the information. Open Authentication is a token-based authentication method. There are two steps connect the Twitter API.

Step 1

```
consumerKey <- "VPUGAVX7KmgQWTjCFHskef7p0"
consumerSecret <- "JuveJmxzPrAXhmjmlRDBddpOFgRkq3FKEP1SnnWoGhQqgGJwVq"
accessToken <- "79478465391999385 6-9svGGq2M3t2VuIHSHlchTcAJ7S1tSlH"
accessTokenSecret <- "8OZw8tPmG0X1pDRRkn1CpjdwDg5P2pcpGFeeyuVuT6Ihk"
setup_twitter_oauth(consumerKey,consumerSecret,accessToken,accessTokenSecret)
origop <- options("httr_oauth_cache")
options(httr_oauth_cache = TRUE)
```

Figure 8

Step 2

We use the **setup_twitter_oauth** function to set up our authentication. The **setup_twitter_oauth()** function takes in the four twitter credentials that we generated from the API set up above.

```
setup_twitter_oauth(consumerKey,consumerSecret,accessToken,accessTokenSecret)
origop <- options("httr_oauth_cache")
options(httr_oauth_cache = TRUE)
```

Figure 9

If you run the above code you get you using authentication, you can see below line.

```
> setup_twitter_oauth(consumerKey,consumerSecret,accessToken,accessTokenSecret)
[1] "Using direct authentication"
>
```

Figure 10

2. Data Cleaning

We used this **dplyr** library to traverse through this data frame. And we managed to store the results into a data frame on our computer. Now analysis this data to find out whether it has the highest Retweets and give them a shutout straight from our script.

```
cleanTweet = gsub("rt|RT", "", GameOfThronesText) # remove Retweet
cleanTweet = gsub("http\\w+", "", cleanTweet)  # remove links http
cleanTweet = gsub("<.*?>", "", cleanTweet) # remove html tags
cleanTweet = gsub("@\\w+", "", cleanTweet) # remove at(@)
cleanTweet = gsub("[[:punct:]]", "", cleanTweet) # remove punctuation
cleanTweet  = gsub("\r?\n|\r", " ", cleanTweet) # remove /n
cleanTweet = gsub("[[:digit:]]", "", cleanTweet) # remove numbers/Digits
cleanTweet = gsub("???|???|???|???|???|???|???|???|???|???", "", cleanTweet) #  asian letters
cleanTweet = gsub("[ |\t]{2,}", "", cleanTweet) # remove tabs
cleanTweet = gsub("^ ", "", cleanTweet)  # remove blank spaces at the beginning
cleanTweet = gsub(" $", "", cleanTweet) # remove blank spaces at the end
```

Figure 11

We are removed (Retweet, links http, html tags, at (@), punctuation, /n, numbers/Digits, tabs, blank spaces at the beginning and the blank space at the end). Without this action, we will face the problem word cloud.

3. Sentiment Analysis

Here, we use sentiment analysis, which is based on the Word-Emotion Association of Saif Mohammad and Peter Turney. A dictionary is used that associates the words to eight different emotions and a negative/Positive sentiment.

4.1 Sentiment Evaluation

Here, we use **syuzhet** this library so that "get_nrc_sentiment" this function can be used. From the picture below, we can find some interesting resluts.

Firstly, from the first line, we can guess that the original comment in twitter is "I hate 'game of thrones', a incredible and deceiving movie." Similarly, we can guess something happened about the fourth line result, which maybe was "'game of thrones' is interesting and nice movie".

Second, It can be seen from the results that people's emotions are very complex or even contradictory. In general, different people have different understandings and views on the same thing, but even the same person can produce a variety of emotions.

```
library(syuzhet)
GameOfThronesSentiment = get_nrc_sentiment(cleanTweet)
head(GameOfThronesSentiment,5)
```

Figure 12

```
> head(GameOfThronesSentiment,5)
  anger anticipation disgust fear joy sadness surprise trust negative positive
1     1            0       1    1   0       1        0     0        1        0
2     0            0       0    1   1       1        0     0        1        1
3     0            1       0    0   0       0        0     0        0        0
4     0            0       0    0   1       0        0     0        0        1
5     0            1       0    0   0       0        0     0        1        1
```

Figure 13

4.2 Sentiment Visualization

In this part, we are trying to do sentiment visualization.

```
GameOfThronesFinalData = cbind(GameOfThrones,GameOfThronesSentiment)

plotData1 =gather(GameOfThronesFinalData,"sentiment","values",17:24)  %>%
  group_by( sentiment) %>%
  summarise(Total = sum(values))
library(ggplot2)
ggplot(data = plotData1, aes(x = plotData1$sentiment, y = plotData1$Total)) +
  geom_bar(aes(fill = sentiment), stat = "identity") +
  theme(legend.position = "none") +
  xlab("Emotions") + ylab("Total") + ggtitle("Emotion for Search GameOfThrones")+
  geom_text(aes(label =   plotData1$Total), position = position_dodge(width=0.75), vjust = -0.25)
plotData2 =gather(GameOfThronesFinalData,"Polarity","values",25:26)  %>%
  group_by( Polarity) %>%
  summarise(Total = sum(values))

ggplot(data = plotData2, aes(x = plotData2$Polarity, y = plotData2$Total)) +
  geom_bar(aes(fill = plotData2$Polarity), stat = "identity") +
  theme(legend.position = "none") +
  xlab("Sentiment") + ylab("Total") + ggtitle("Sentiment for Search GameOfThrones")+
  geom_text(aes(label =   plotData2$Total), position = position_dodge(width=0.75), vjust = -0.25)
```

Figure 14

The most people showed the feeling of trust, followed by anticipation. The least people show disgust, which is easy to understand because it is a fantastique tv drama.

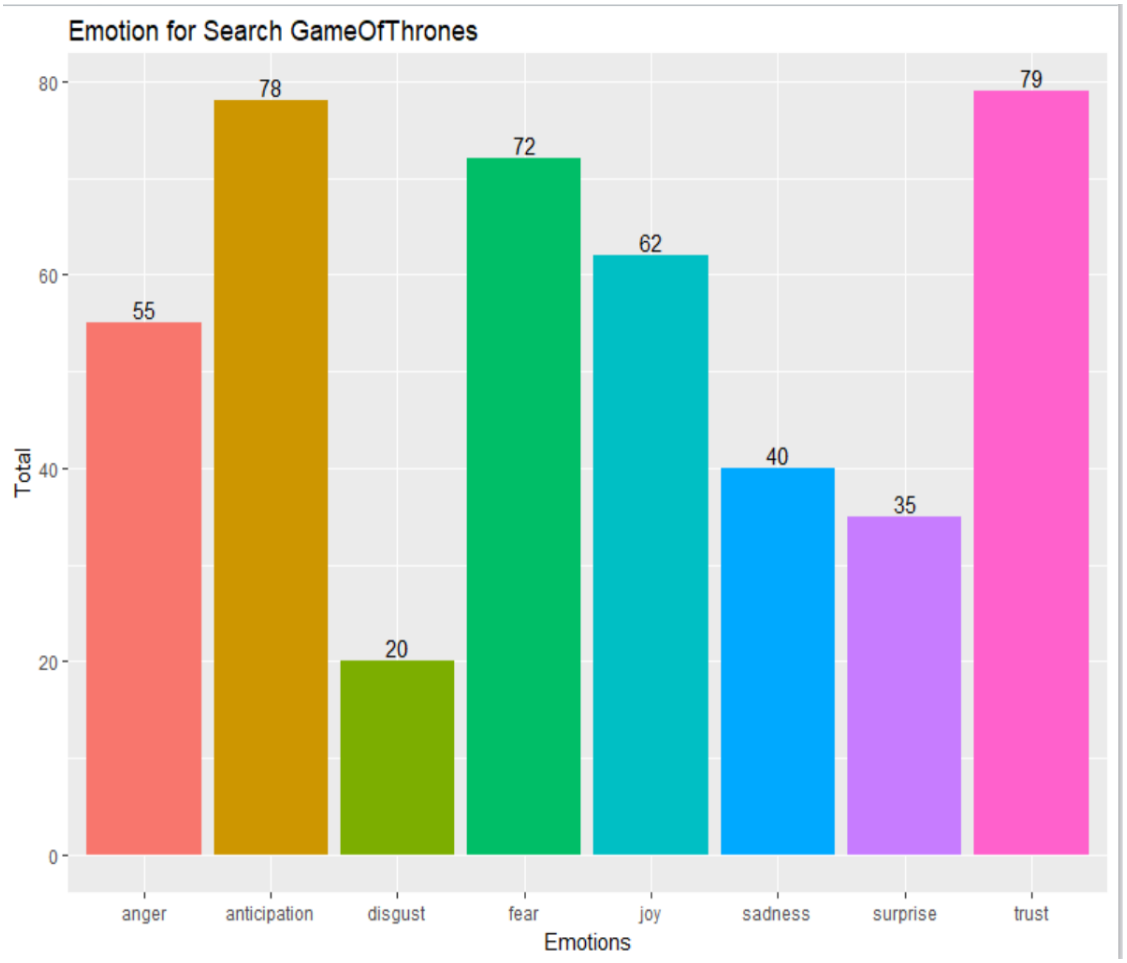

Figure 15

The above distribution is relatively dense, so we can't intuitively see whether people who show positive or negative emotions. So here, we use the charts to show it directly.

From the chart below, we can clearly see that although there are many positive and negative emotions. By comparison, more people show positive emotions. In other words, more people like the film. Popularly speaking, if we judge a TV play only from the audience's preferences, it is successful.

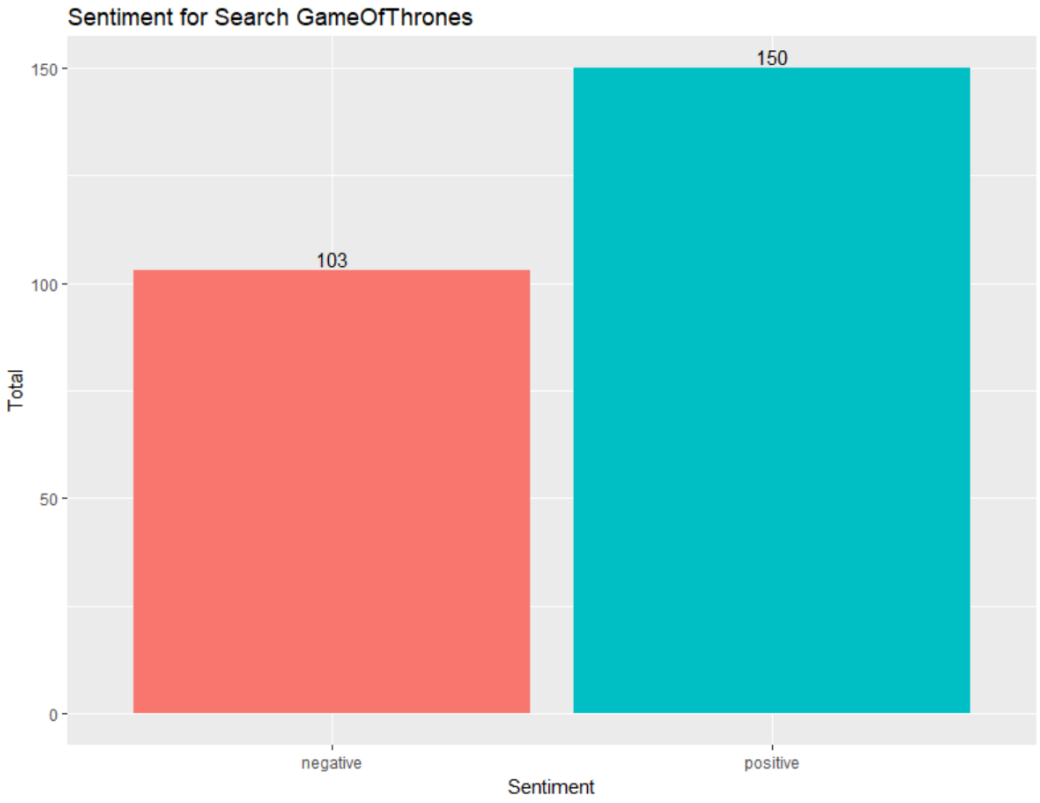

Figure 16

5 Classification and Accuracy

5.1 Necessary R packages

- E1071

However, the naive bayes method is not included into RTextTools. The e1071 package did a good job of implementing the naive bayes method. e1071 is a course of the Department of Statistics (e1071), TU Wien. Its primary developer is David Meyer. It is still necessary to learn more about text analysis.

- DTM

Transforming texts to documentterm matrix (dtm). The most important part of text analysis is to get the feature vectors for each document. The word feature is the most important one. Of course, you can also extend the unigram word features to bigram and trigram, and so on to n-grams. However, here for our case, we stick to the unigram word features.

5.2 Data reduction

After we complete the sentimental analysis, we can see that the 10th and 11th columns of the matrix are about negative and positive. Here we extract these two columns separately from the matrix , used to do Bayesian classification. However, there is a problem. In the first part, when we complete the semantic analysis, negative and positive represent the extent to which this comment expresses positive feelings and negative feelings. So its value is greater than or equal to zero. However, what we need to do at this moment is classification. We only need to judge whether this comment is positive or not positive, so its value should be 0 or 1. To sum up, we further simplify the data. When the value is greater than 0, it becomes 1. It means that this is a positive comment.

```
#if "positive" is equal to 2 or 3, it will be 1(means positive),otherwise it will be 0(means negative).

GameOfThronesFinalData[,3]<-ifelse(GameOfThronesFinalData[,3]>0,1,0)

GameOfThronesFinalData[,3]
```

Figure 17

```
> GameOfThronesFinalData[,3]<-ifelse(GameOfThronesFinalData[,3]>0,1,0)
> GameOfThronesFinalData[,3]
  [1] 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0
 [78] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0
 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[155] 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0
[232] 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0
```

Figure 18

5.3 NaiveBayes classifation and accuracy

Then we can build the document-term matrix:

```
################################################ Naivebayes  classification    ######################################
# library(naivebayes)
library(e1071)
# build dtm
matrix= create_matrix(GOTFinalData[,1], language="english",removeStopwords=FALSE, removeNumbers=TRUE, stemWords=FALSE)
matrix
mat = as.matrix(matrix)
```

Figure 19

Now, we can train the naive Bayes model with the training set. Note that, e1071 asks the response variable to be numeric or factor. Thus, we convert characters to factors here. This is a little trick.

```
# train the model
mat = as.matrix(matrix)

#naiveBayes
classifier = naive_bayes(mat[1:950,], as.factor(GameOfThronesFinalData[1:950,2]) )
classifier
```

Figure 20

Now we can step further to test the accuracy.

```
recall_accuracy(GameOfThronesFinalData[951:1000, 2], predicted)
```

Figure 21

recall_accuracy= 0.4

The value is too low, so we try other methods.

5.4 The other models

RTextTools is a machine learning package for automatic text classification that makes it simple for novice users to get started with machine learning, while allowing experienced users to easily experiment with different settings and algorithm combinations. The package includes nine algorithms for ensemble classification (svm, slda, boosting, bagging, random forests, glmnet, decision trees, neural networks, maximum entropy), comprehensive analytics, and thorough documentation.

```
# build models

models = train_models(container, algorithms=c("MAXENT" , "SVM", "RF", "BAGGING", "TREE"))
results = classify_models(container, models)
results
```

Figure 22

SVM

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are

then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

```
> cross_validate(container,N,"SVM")
Fold 1 Out of Sample Accuracy = 0.8257261
Fold 2 Out of Sample Accuracy = 0.8082707
Fold 3 Out of Sample Accuracy = 0.8429752
Fold 4 Out of Sample Accuracy = 0.8286853
[[1]]
[1] 0.8257261 0.8082707 0.8429752 0.8286853

$meanAccuracy
[1] 0.8264143
```

Figure 23

Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.[1][2] Random decision forests correct for decision trees' habit of overfitting to their training set.[3]:

```
> cross_validate(container,N,"RF")
Fold 1 Out of Sample Accuracy = 0.8059072
Fold 2 Out of Sample Accuracy = 0.8434164
Fold 3 Out of Sample Accuracy = 0.8486056
Fold 4 Out of Sample Accuracy = 0.8658009
[[1]]
[1] 0.8059072 0.8434164 0.8486056 0.8658009

$meanAccuracy
[1] 0.8409325
```

Figure 24

Decision tree

Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modelling approaches used in statistics, data miningand machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.

```
> cross_validate(container,N,"TREE")
Fold 1 Out of Sample Accuracy = 0.766129
Fold 2 Out of Sample Accuracy = 0.7782101
Fold 3 Out of Sample Accuracy = 0.7764228
Fold 4 Out of Sample Accuracy = 0.7309237
[[1]]
[1] 0.7661290 0.7782101 0.7764228 0.7309237

$meanAccuracy
[1] 0.7629214
```

Figure 25

Bagging

Bootstrap aggregating, also called bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting. Although it is usually applied to decision tree methods, it can be used with any type of method. Bagging is a special case of the model averaging approach.

```
> recall_accuracy(as.numeric(as.factor(GameOfThronesFinalData[951:1000, 2])), results[,"BAGGING_LABEL"])
[1] 0.8
```

Figure 26

Maximum entropy

The principle of maximum entropy states that the probability distribution which best represents the current state of knowledge is the one with largest entropy, in the context of precisely stated prior data (such as a proposition that expresses testable information). Another way of stating this: Take precisely stated prior data or testable information about a probability distribution function. Consider the set of all trial probability distributions that would encode the prior data. According to this principle, the distribution with maximal information entropy is the best choice.

```
> cross_validate(container,N,"MAXENT")
Fold 1 Out of Sample Accuracy = 0.972973
Fold 2 Out of Sample Accuracy = 0.9874477
Fold 3 Out of Sample Accuracy = 0.9847328
Fold 4 Out of Sample Accuracy = 0.9875
[[1]]
[1] 0.9729730 0.9874477 0.9847328 0.9875000

$meanAccuracy
[1] 0.9831634
```

Figure 27

6. Word Tokenization and segmentation

6.1 Word Tokenization

Work Tokenization is the process of breaking up the sentence or the tweets extracted or any sequence of the words into words. The goal of tokenization is the exploration of the words in a sentence. Before we do any kind on analysis on the text using a language processor, we need to normalize the words. When we do quantitative analysis on the text we consider it a bag of words. and extract the key words, frequency of occurrence, and the importance of each word in the text. Tokenizing provides various kind of information about text, like the number of words or tokens in a text, the vocabulary or the type of words. First thing is to create the Term Document Matrix. We can do it using Term Document Matrix function in R and access the various attributes like Docid, term and docs. This Matrix can help us to evaluate the text mathematically.

Let's analyze out TDM below.

```
> tdm
<<TermDocumentMatrix (terms: 2912, documents: 1000)>>
Non-/sparse entries: 8965/2903035
Sparsity           : 100%
Maximal term length: 25
Weighting          : term frequency (tf)
>
```

Figure 28

The size of our DTM is 1000 X 2912. It says 100% of the rows are zero, that is, most of words will appear in a few documents. We can reduce the sparsity of the document for computational efficiency. Term-document matrices tend to get very big already for normal sized data sets. Therefore, we provide a method to remove sparse terms, i.e., terms occurring only in very few documents. Normally, this reduces the matrix dramatically without losing significant relations inherent to the matrix.

Frequent Terms Can be found using findFreqTerms functions in R and result can be seen in below format. In this result, we are looking for the terms occurring at least 20 times.

```
> tdm = TermDocumentMatrix(Corpus)
> (freq.terms <- findFreqTerms(tdm, lowfreq = 20))
 [1] "best"          "calling"       "decide"        "episodes"
 [5] "ever"          "fans"          "greatest"      "producing"
 [9] "show"          "want"          "game"          "gameofthrones"
[13] "got"           "thrones"       "first"         "last"
[17] "jonsnow"       "episode"       "fight"         "love"
[21] "queen"         "season"        "going"         "just"
[25] "can"           "one"           "like"          "cersei"
[29] "daenerys"      "will"          "get"           "throne"
[33] "today"         "lannister"     "made"          "hard"
```

Figure 29

Moreover, even the associations among the words can be found too using findAssocs function.

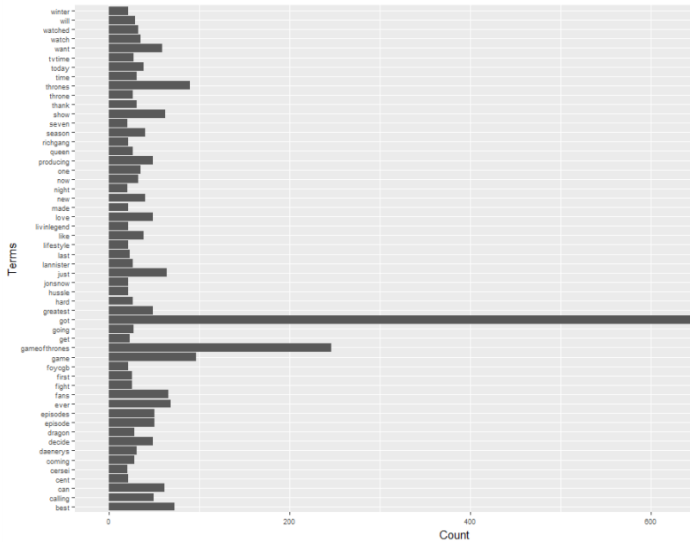

Figure 30

6.2 CORPUS

These documents are maintained using CORPUS function in tm library. One the corpus is ready, we need to remove sparse terms from the matrix too. Using tm_map function, we need to remove convert the text to lower case, remove whitespaces, remove stop words and in a way, we need to standardize the collection of the documents.

6.3 Word Cloud

One can create a word cloud, also referred as text cloud or tag cloud, which is a visual representation of text data. The procedure of creating word clouds is very simple in R if you know

the different steps to execute. The word cloud generator package (*word cloud*) is available in R for helping us to analyze texts and to quickly visualize the keywords as a word cloud.

Figure 31

Word Clouds add simplicity, clarity, visibility and engagement. Researchers, Marketers, Educators are using word clouds more and more now a day. This word cloud shows that “got”, “games of thrones”, ‘greatest”, “decide” are more popular or retweeted words.

7. Term Network

7.1 term

When people tweet, they may get retweeted by other people, repeating the message for their followers to view. Each retweet is a one-way flow of information that links the first person to each person who retweeted them (forwarded the original tweet into their own network)

```
           Terms
Terms       ever fans greatest producing show want
  ever        68   54       47        48   50   48
  fans        54   65       47        48   48   48
  greatest    47   47       48        47   47   47
  producing   48   48       47        48   48   48
  show        50   48       47        48   62   48
  want        48   48       47        48   48   57
> |
```

Figure 32

To create an graph object from an edge-list data frame we can use the graph_from_data_frame() function, which is a bit more straight forward than network(). There are three arguments in the graph_from_data_frame() function: d, vertices, and directed. Here, d refers to the edge list, vertices to the node list, and directed can be either TRUE or FALSE depending on whether the data is directed or undirected.

The htmlwidgets set of packages makes it possible to use R to create interactive JavaScript visualizations. Here, I will show how to make graphs with the visNetwork and networkD3 packages. These two packages use different JavaScript libraries to create their graphs. visNetwork uses vis.js, while networkD3 uses the popular d3 visualization library to make its graphs. One difficulty in working with both visNetwork and networkD3 is that they expect edge lists and node lists to use specific nomenclature. The above data manipulation conforms to the basic structure for visNetwork, but some work will need to be done for networkD3.

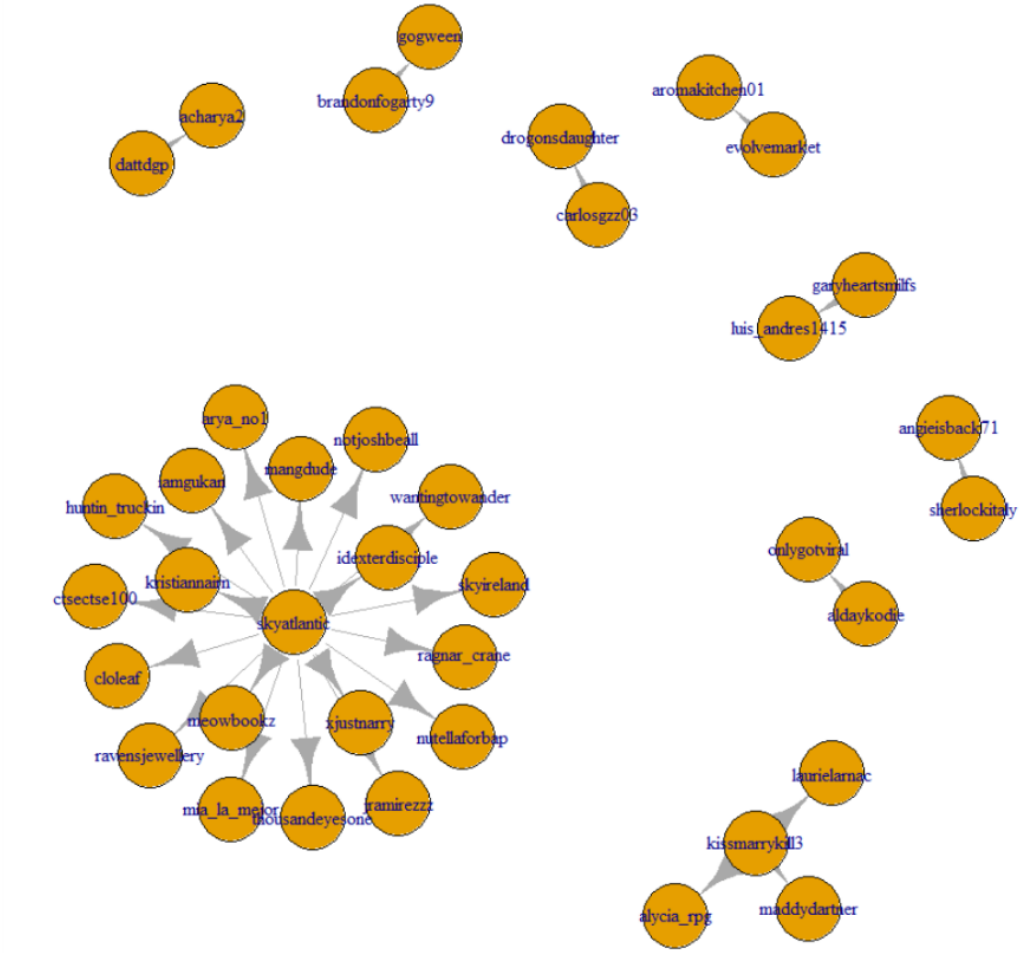

Figure 33

This graph needs to be converted to adjacency matrix . An adjacency matrix is a square matrix in which the column and row names are the nodes of the network. Within the matrix a 1 indicates that there is a connection between the nodes, and a 0 indicates no connection. The nodes in the data are identified by unique IDs. If the distinction between source and target is meaningful, the network is directed. If the distinction is not meaningful, the network is undirected

8 Conclusion

Analysis of tweets from Twitter can be useful from business perspective for the Producers of Games of thrones to get reviews about their new episodes or existing ones from their customer base. This analysis when coupled with visualizations becomes that much more powerful. The tweets will only give us clues about the show's performance. We have to "read between the lines" (of tweets). We have to put on our detective hat and search for the real answers. Without the metrics however, it is near impossible to get this process underway.

Reference

Ho, Tin Kam (1995). Random Decision Forests(PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.

Ho TK (1998). "The Random Subspace Method for Constructing Decision Forests" (PDF). IEEE Transactions on Pattern Analysis and Machine Intelligence. 20 (8): 832–844. doi:10.1109/34.709601.

Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). The Elements of Statistical Learning (2nd ed.). Springer. ISBN 0-387-95284-5.

Kleinberg E (1990). "Stochastic Discrimination" (PDF). Annals of Mathematics and Artificial Intelligence. 1 (1-4): 207–239. doi:10.1007/BF01531079.

http://blog.sciencenet.cn/blog-397960-666113.html

https://cran.r-project.org/web/packages/RTextTools/RTextTools.pdf

P. Bickel and K. Doksum. Mathematical Statistics. Prentice Hall, 2000.

D. Blei and M. Jordan. Modeling annotated data. In SIGIR, pages 127–134. ACM Press, 2003.

D. Blei and J. McAuliffe. Supervised topic models. In preparation, 2007.

Breiman, Leo. Bagging predictors. Machine Learning. 1996, 24 (2): 123–140. doi:10.1007/BF00058655. CiteSeerX: 10.1.1.32.9399.

Alfaro, E., Gámez, M. and García, N. adabag: An R package for classification with AdaBoost.M1, AdaBoost-SAMME and Bagging. 2012.