Intelligent Social Media Analytics

Niranjan Gawli

Ashwin Dinoriya

Tanmay Ingle

Nayyar Unda

Index

Topic	Page Number
Introduction	**3**
Twitter Data Analysis using R	**4**
Sentimental Analysis using R	**13**
Product Sentimental Analysis	**18**
Python Implementation	**19**
Twitter Sentimental Analysis using Amazon EC2	**25**
Microsoft Azure Machine Learning for Twitter Sentimental Analysis	**41**
Data Visualization	**47**
Power of Sentiments	**60**
Future Scope	**62**
References	**63**

Introduction

Let's consider a hypothetical situation where we are a Marketing Analytics company. We have a client who opened a restaurant recently near Northeastern University and sells burrito. He approached us to give him analytics using social media.

We demonstrate this process by using R and real time twitter data.

Twitter Data Analysis using R

Following are three steps to perform this task:

1) **Data Extraction**
    a. Connecting R with Twitter to get real time data
    b. Preprocessing the data
    c. Visualizing the data
    d. Clustering and topic modelling
2) **Find the top 10 and top 50 influential users**
    a. Extract user details
    b. Extract number of their followers
    c. Get top 50 influential user
3) **Sentimental Analysis**
    a. Find opinion of customers staying in the vicinity about the product
    b. Find opinion of customers about the competitors in the area.
    c. Visualizing the results for better insight

To perform above analysis, we have used the concept of Text Mining. We have installed 3 libraries in R.

1) ***'tm'*** to use all the functions for text mining.
2) ***'twitteR'*** to scrape tweets and interact with twitter.
3) ***'ROAuth'*** to create a handshake between twitter and R.

Twitter is a great source for sentiment data and social media mining. Furthermore it is quite easy to get significant amounts of data to be able to scrape data from Twitter. We need a standard Twitter account and then update it to a developer account to be able to use twitter API.

```
#Connection between RStudio and twitter

key <- "4aN0Mvc78BJvHG0EC7BM80su5"
secret <- "rHcFMOmCx6vh8jP7ZydovrtXjrehlQtd50FC0QR96va4ESoNxM"
secrettk <- "DzdYlNviNPg7VhZczj7l34luWAvkosZEzAgn1gWDI5ISC"
mytoken <- "3146945200-fD12d1UEfBAK5wx0IjRPIiJjPQ5RZ6nj9FxpBei"

setup_twitter_oauth(key,secret,mytoken,secrettk)
```

We scrape tweets from twitter in real time which contains the word 'burrito' in it. We restrict the search to area near Northeastern University using 'geocode' argument. We restrict it to English language.

```
19
20  #scraping data from twitter.
21  burritotweets = searchTwitter("burrito",n = 1000, lang = "en",geocode='40.712940,-73.987920,5mi')
22
```

We get data in the form of list. Then we extract the text part of it. Next step is to clean the data.

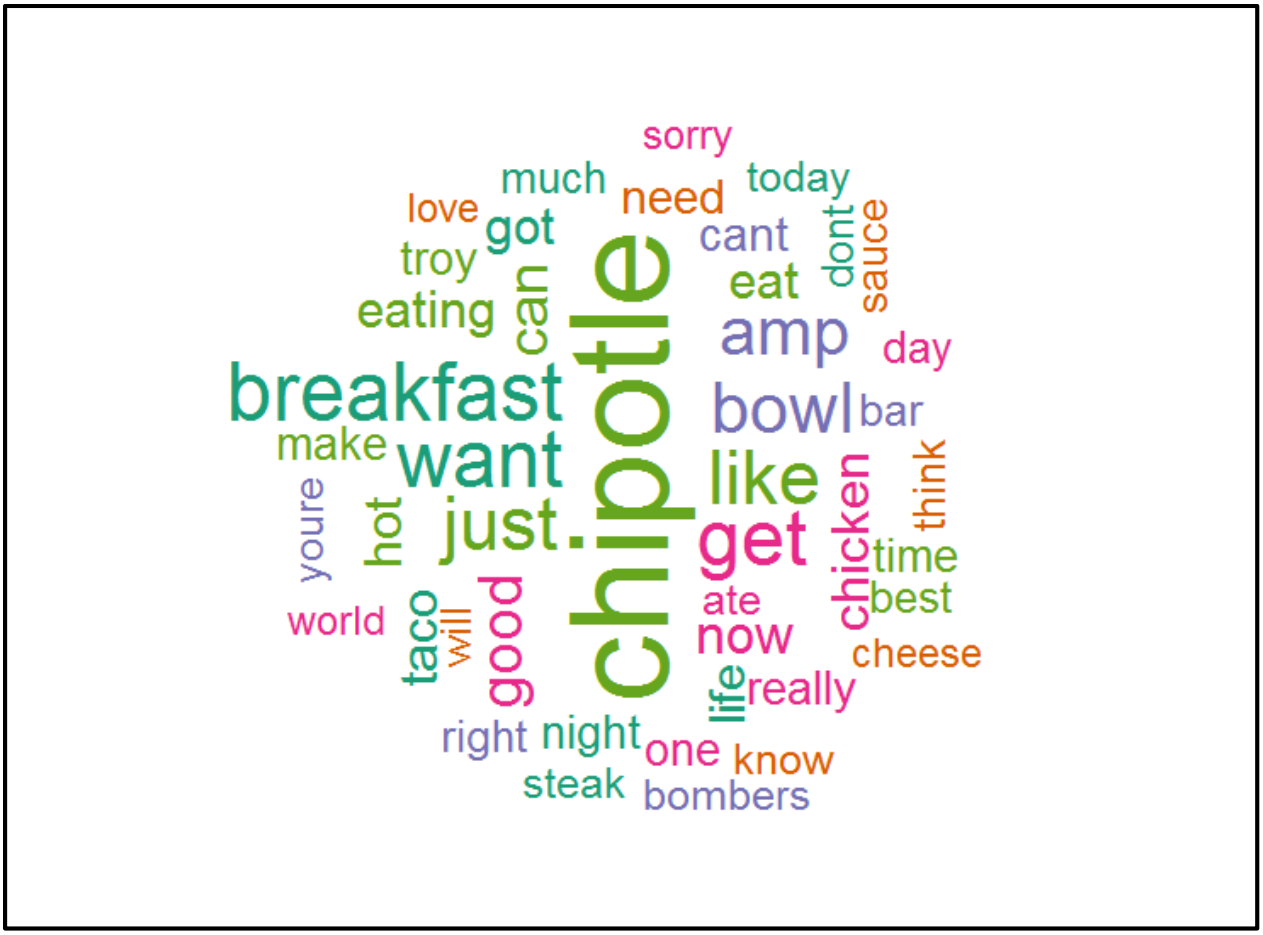
a) A corpus is the text body consisting of all the text including the meta info. Corpus is needed to use 'tm_map' function.
b) Remove emoticons – emoticons are present in latin format and thus are of no use for analysis.
c) Remove Numbers, Remove Punctuations
d) Tolower – we convert all the alphabets to lowercase for consistency.
e) Remove stopwords – These are the words like just, do, ask etc which won't contribute to the analysis. We remove the word burrito to prevent it from taking a huge place in wordcloud since it occurs most number of times.

```
#using function getText to extract text part of tweets
burritolist <- sapply(burritotweets,function(x) x$getText())


# converting all the latin1 values to ASCII. These are basically the emoticons
burritolist <- sapply(burritolist,function(row) iconv(row, "latin1", "ASCII", sub = ""))




#Converting to corpus which is a text body consisting of all text along with the meta data.
burritocorpus <- Corpus(VectorSource(burritolist))



#Preprosessing the text data

burritocorpus <- tm_map(burritocorpus, removeNumbers)  #to remove numbers
burritocorpus <- tm_map(burritocorpus, removePunctuation) # to remove punctuations
burritocorpus <- tm_map(burritocorpus, tolower)   #converting everything to lowercase
burritocorpus <- tm_map(burritocorpus,function(x)removeWords(x,c("burrito",stopwords())))
#removing meaningless words like I, you ,just etc.
#removing the word burrito to eliminate it from wordcloud
```

Corpus has to be converted to Plain Text Document to make a wordcloud. We use RColourBrewer package to make the wordcloud colorful.

```
#converting to plain text document which is the input for making word cloud
burritocorpus <- tm_map(burritocorpus,PlainTextDocument)


#Making the wordcloud from the above plain text document.
col <- brewer.pal(5,"Dark2")

wordcloud(burritocorpus, min.freq = 5,rot.per = 0.7, scale = c(5,1), random.color = T,
          max.word = 45, random.order = F, colors=col)
```

Following is the *WordCloud* where frequency of any word is represented by size of the word.



We convert our data to a TermDocumentMatrix which is a matrix with all the terms in the document as columns and all tweets as rows. It stores the number of times a term has been used in a tweet.

```
#Making a TermDocument Matrix
burritotdm <- TermDocumentMatrix(burritocorpus)

#to Find top 10 frequent terms used in the tweets
findFreqTerms(burritotdm, lowfreq = 10)


#to find terms which have correlation with the term chicken having correlation score more than 0.3
findAssocs(burritotdm,'chicken', 0.3)
```

We find the frequesntly used words which occurred minimum 10 times.

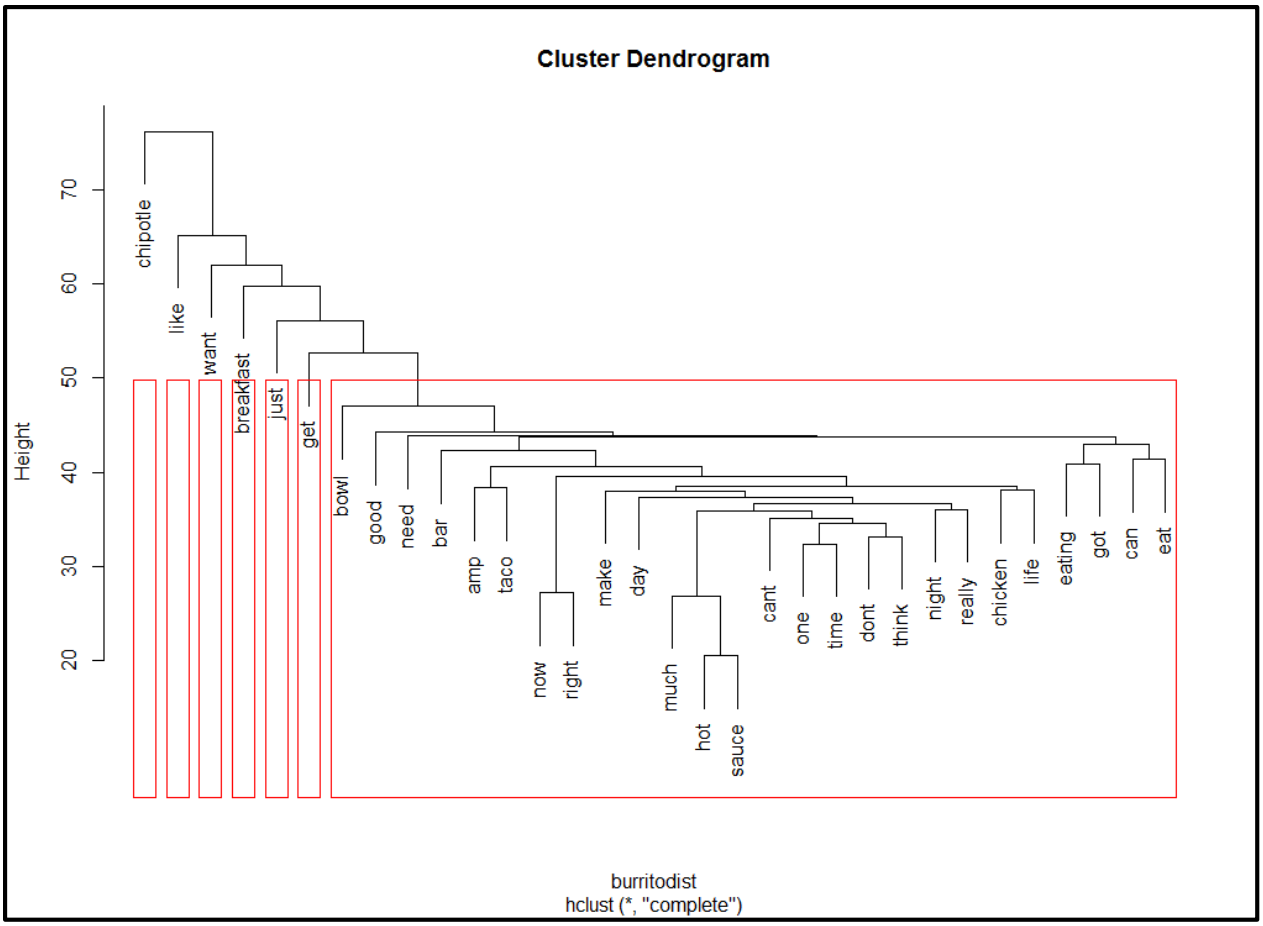
```
> findFreqTerms(burritotdm, lowfreq = 10)
  [1] "amp"          "ass"          "ate"              "ave"          "back"
  [6] "bacon"        "bad"          "bar"              "bean"         "bed"
 [11] "bell"         "best"         "better"           "big"          "blanket"
 [16] "bombers"      "bowl"         "breakfast"        "bring"        "burritos"
 [21] "buy"          "came"         "can"              "cant"         "cheese"
 [26] "chicken"      "chipotle"     "chipotletweets"   "come"         "day"
 [31] "delivery"     "dinner"       "dont"             "eat"          "eating"
 [36] "ever"         "food"         "free"             "freebirds"    "freebirdswb"
 [41] "fries"        "get"          "getting"          "girl"         "going"
 [46] "good"         "got"          "grill"            "guy"          "head"
 [51] "home"         "hot"          "httptcokmzbjpfr"  "hungry"       "ill"
 [56] "ive"          "just"         "know"             "large"        "last"
 [61] "life"         "like"         "live"             "lol"          "look"
 [66] "love"         "lunch"        "make"             "mexican"      "minutes"
 [71] "much"         "need"         "new"              "night"        "nikestreetz"
 [76] "nothing"      "now"          "one"              "people"       "pizza"
 [81] "put"          "really"       "rice"             "right"        "sauce"
 [86] "shit"         "someone"      "sonic"            "sorry"        "start"
 [91] "steak"        "still"        "taco"             "take"         "thing"
 [96] "think"        "time"         "today"            "troy"         "vegetarian"
[101] "want"         "way"          "well"             "will"         "work"
```

We can find which all words are associated with the word chicken with correlation more than 0.3.

```
> findAssocs(burritotdm,'chicken', 0.3)
                chicken
extra              0.46
fingers            0.40
cheese             0.39
kumpalicious       0.37
shredded           0.37
hunterallee        0.35
bacon              0.34
fries              0.34
pasta              0.34
fuzzys             0.33
httpstconakmzgoh   0.33
imagine            0.33
nachos             0.33
>
```

Sparsity basically is the ratio of number of zeroes to all the numbers in the matrix. High sparsity signifies less association between words. Thus we can limit the number of variables by reducing the sparsity. Earlier we had total of 2765 terms and a sparsity of 99%. We reduce the sparsity to get more densed data and only 32 variables. We follow the procedure for hierarchical clustering by scaling and taking euclidean distance. We plot dendogram of the data to find groups/clusters.

```
74  #To create a dendogram for visualization.
75
76  burritotdm2 <- removeSparseTerms(burritotdm, sparse = 0.975)
77  #remove sparse (infrequentyl used) terms from the term Document matrix
78
79
80
81
82  burritotdm2scale <- scale(burritotdm2)  #to scale the data
83
84  burritodist <- dist(burritotdm2scale, method = 'euclidean') # to create a distance matrix
85
86  burritofit <- hclust(burritodist) #Hierarchical clustering
87
88  plot(burritofit) # Visualization the results
89
90  cutree(burritofit, k = 7) # to calculate certain number of groups
91
92  rect.hclust(burritofit, k=7, border = "red") #Colour the groups with border for better visialization.
93
94
```



Targeting Influential customers for better marketing.

We extracted usernames from all the tweets by using getScreenName() on scraped tweet data.

```
100 #-----------Targeting Influencers-------------------------------------------------------------------
101
102
103 #using function to get usernames of of owner of the tweets
104 burritolistnames <- sapply(burritotweets,function(x) x$getScreenName())
105
```

Thus we get 1000 usernames from 1000 tweets.

```
> burritolistnames
   [1] "Hiserote43"       "jordan_jude"      "RagingBurrito"    "OatzTheGreat"     "YoungJaneJones"
   [6] "EverettThomson"   "Gunjan_Marwah"    "revdavidhuber"    "allyourdata"      "ESchardt"
  [11] "chriswong13"      "larryfisherboy"   "adboone1990"      "paige_criswell"   "SexyShida"
  [16] "youryouthpastor"  "christiee813"     "RRFleming"        "RikterSr"         "leximackenzie3"
  [21] "lusciousraen"     "_Bryan_Martinez"  "locaines"         "lrasmussenjr"     "GITNB_EATS"
  [26] "generousbabble"   "ckerfin"          "ArtisticBulldog"  "lyssylala"        "_b_ay__"
  [31] "WFMportlandME"    "Shruggsy"         "solvalero98"      "angel_burrito"    "ShortCakee_xoxo"
  [36] "T3DO"             "RalphRenna"       "burnt_burrito"    "taylorsaylorrr"   "RalphRenna"
  [41] "RalphRenna"       "RalphRenna"       "RalphRenna"       "lamboghiniMIRcy"  "mortonfox"
  [46] "meeeowkat"        "AdvoATX"          "jgcarrington"     "RalphRenna"       "kellirhiannon"
  [51] "WesleyRaczka"     "AutumnFish13"     "RagingBurrito"    "ethanr04"         "HotHeadBurrito"
  [56] "HotHeadBurrito"   "jennaerb"         "AlexisGuilbertt"  "virgoprophecy"    "MoriahBryn"
```

Next we get all the details of those users from twitter by using lookupuser() function

```
106
107 #Scraping user details
108 users <- lookupUsers(burritolistnames)
109
110
```

We are interested in followers count feature of the list ‘users’. Thus we convert this list to dataframe and sort it on followers count.

```
111
112 #converting this list to dataframe
113 userprofiledf <- twListToDF(users)
114
115
116
117 #sorting data on the variable 'followersCount'
118 sortedusers <- arrange(userprofiledf, desc(followersCount))
119
```

We then get top 10 Influential customers and top 50 influential customers. Our client can give various perquisites to these customers and increase their outreach.

```
123
124
125 #getting top 10 influencers
126 x <- sortedusers[1:10,]
127 top10influencers <- x$screenName
128 top10influencers
129
130
131
132 #getting top 50 influencers
133 y <- sortedusers[1:50,]
134 top50influencers <- y$screenName
135 top50influencers
136
137
138
```

Sentimental Analysis using R

Following analysis is just to give a preview of sentimental analysis before going into depth. Consider following four statements. First sentence contains 1 negative word (bad) and thus its sentimental score is -1. However second statement contains one positive word (good) and thus it is +1. Third sentence is neutral and thus is 0.

```
> example = c("it is bad", "It is good", "I am batman", "It's expensive and useless")
> examplesentiment = score.sentiment(example,pos,neg)
> examplesentiment
                        text score
1                  it is bad    -1
2                 It is good     1
3                I am batman     0
4 It's expensive and useless    -2
```

Sentiment analysis is used to see if a text is neutral, positive or negative. Emotion analysis is used to see which emotion a text has (happy, fear, anger). Both are using similar codes but the comparison lexicon is different.

Example: What is the sentiment towards my company?

Twitter data is useful for that type of analysis because:

a) It has high volumes (500 mill/day) of data

b) It consists of short messages like sms - 140 characters

c) It contains special strings (hashtags)

Thus twitter is full of sentiments.

Sentiment Lexicon is a list of words which we can use to compare any scraped text. Hu Liu Lexicon made a standard of sentiment analysis by manually creating a list of positive and negative words. Combined it consists of approximately 6800 words.

We have downloaded the lexicons and imported into our R environment.

```
147  #importing positive and negative lexicon
148  pos = readLines("positive_words.txt")
149  neg = readLines("negative_words.txt")
150
```

Following is the main function which we will use for generating score for sentimental analysis. This function takes arguments as a text matrix, and positive and negative lexicons, and control for progress bar.

It uses laply from (plyr package) and creates a function to clean the data the way it is discussed above. It splits all the words of individual tweets by using str_split from stringr package. It matches these words with lexicon by match function which returns position of that word or else return NA. However we just want to know if it exists regardless of the position. So we use (!is.na)function to get number of matches. And finally we calculate the total by subtracting number of negative matches from positive matches.

Since we used laply, this function gets score for all the tweets individually. It is returned as a dataframe with sentimental scores.

```r
#Function definition
score.sentiment = function(sentences, pos.words, neg.words, .progress='none')
{
  #Parameters
  #Sentences : Vector of text to score
  #pos.words: Vector of words of positive sentiments
  #neg.words: vector of words of negative statement
  #.progress:control of progress bar


  scores = laply(sentences,
                 function(sentence, pos.words, neg.words)
                 {
                   # remove punctuation - using global substitute
                   sentence = gsub("[[:punct:]]", "", sentence)
                   # remove control characters
                   sentence = gsub("[[:cntrl:]]", "", sentence)
                   # remove digits
                   sentence = gsub('\\d+', '', sentence)
                   # define error handling function when trying tolower
                   tryTolower = function(x)
                   {
                     # create missing value
                     y = NA
                     # tryCatch error
                     try_error = tryCatch(tolower(x), error=function(e) e)
                     # if not an error
                     if (!inherits(try_error, "error"))
                       y = tolower(x)
                     # result
                     return(y)
                   }
                   # use tryTolower with sapply
                   sentence = sapply(sentence, tryTolower)
                   # split sentence into words with str_split (stringr package)
                   word.list = str_split(sentence, "\\s+")
                   words = unlist(word.list)
                   # compare words to the dictionaries of positive & negative terms
                   pos.matches = match(words, pos.words)
                   neg.matches = match(words, neg.words)
                   # get the position of the matched term or NA
                   # we just want a TRUE/FALSE
                   pos.matches = !is.na(pos.matches)
                   neg.matches = !is.na(neg.matches)
                   # final score
                   score = sum(pos.matches) - sum(neg.matches)
                   return(score)
                 }, pos.words, neg.words, .progress=.progress )
  # data frame with scores for each sentence
  scores.df = data.frame(text=sentences, score=scores)
  return(scores.df)
}
```

Major competitors around Northeastern University which sell burrito are Chipotle, Qdoba and Boloco. Thus we can understand what people think about the competitors by using sentimental analysis on them.

We scrape tweets about these three restaurants and extract text out of them by using getText(). We also convert latin characters to ASCII for better analysis.

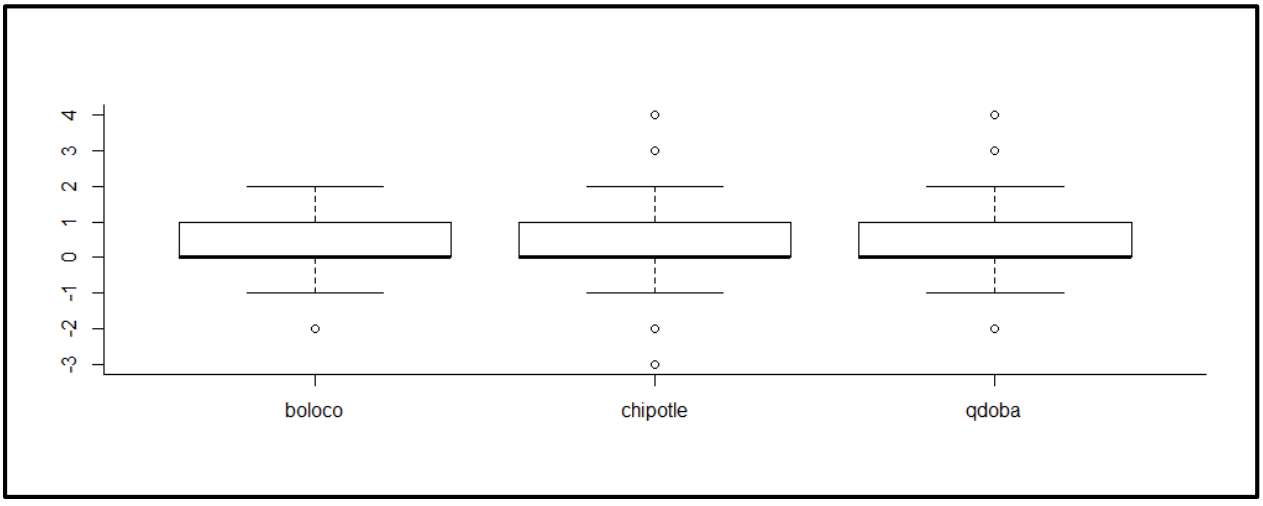
```
214
215 #getting tweets for three big burrito restaurants
216 qdobatweets = searchTwitter("qdoba",n = 1000, lang = "en",geocode='40.712940,-73.987920,5mi')
217 bolocotweets = searchTwitter("boloco",n = 1000, lang = "en",geocode='40.712940,-73.987920,5mi')
218 chipotletweets = searchTwitter("chipotle",n = 1000, lang = "en",geocode='40.712940,-73.987920,5mi')
219
220
221
222 #Extracting text out of tweets by using getText function
223 qdoba_txt = sapply(qdobatweets, function(x) x$getText())
224 boloco_txt = sapply(bolocotweets, function(x) x$getText())
225 chipotle_txt = sapply(chipotletweets, function(x) x$getText())
226
227
228 #converting latin1 characters to ASCII.
229 qdoba_txt <- sapply(qdoba_txt,function(row) iconv(row, "latin1", "ASCII", sub = ""))
230 boloco_txt <- sapply(boloco_txt,function(row) iconv(row, "latin1", "ASCII", sub = ""))
231 chipotle_txt <- sapply(chipotle_txt,function(row) iconv(row, "latin1", "ASCII", sub = ""))
232
```

We combine the three matrices and add another column called burritocompany to categorize tweets. We calculate sentiment score on this matrix and store it in scores data frame.

```
238
239 #join texts
240 burritocompany = c(qdoba_txt, boloco_txt, chipotle_txt)
241
242
243
244 #applying function score.sentiment
245 scores = score.sentiment(burritocompany, pos, neg, .progress='text')
246
247
248
249 #add variable burritocompany ti the data frame
250 scores$burritocompany = factor(rep(c("qdoba", "boloco", "chipotle"), nooftweets))
251
```

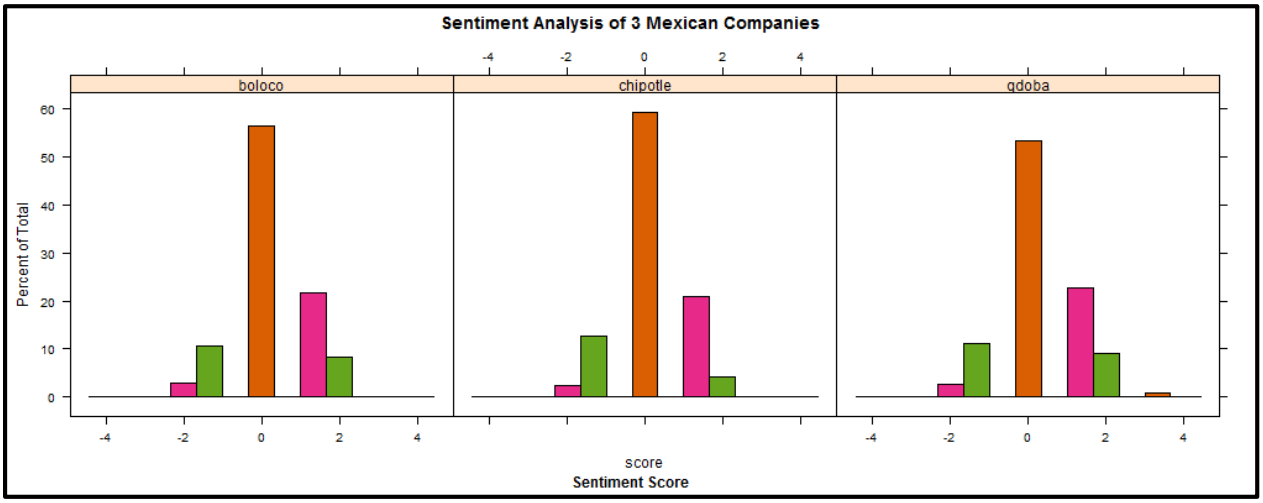
We created a boxplot of sentiment scores.

```
70 boxplot(score~burritocompany, data=scores) #making a boxplot of sentiments
```



However we can analyze the data and get more insight by creating a histogram.

```
72
73  #creating a histogram of sentiments. We need lattice library for this
74  histogram(data=scores, ~score|burritocompany, main="Sentiment Analysis of 3 Mexican Companies",
75            col = col, sub="Sentiment Score")
76
```



Thus we can infer from the above histogram that most of the tweets are neutral (0) (green). Also qdoba has highest percent of scores which are positive. Thus reputation of qdoba is better.

Product Sentimental Analysis

We scraped burrito data

```
19
20 #scraping data from twitter.
21 burritotweets = searchTwitter("burrito",n = 1000, lang = "en",geocode='40.712940,-73.987920,5mi')
22
23
24 #using function getText to extract text part of tweets
25 burritolist <- sapply(burritotweets,function(x) x$getText())
26
27
28 # converting all the latin1 values to ASCII. These are basically the emoticons
29 burritolist <- sapply(burritolist,function(row) iconv(row, "latin1", "ASCII", sub = ""))
30
31
```

We applied score.sentiment function on it to get sentiment scores. Then we checked how many are very positive and very negative and finally took out a percentage score.

```
12
13 #applying function score.sentiment
14 scores = score.sentiment(burritolist, pos, neg, .progress='text')
15
16
17 #to calculate global sentiment score
18 scores$very.pos = as.numeric(scores$score >= 2) #adding varablevery.pos
19 scores$very.neg = as.numeric(scores$score <= -2)# adding variable very.neg
20
21 #to calculate numberof very positives and number of very negatives
22 numpos = sum(scores$very.pos)
23 numneg = sum(scores$very.neg)
24
25
26 #to calculate global score
27 global_score = round( 100 * numpos / (numpos + numneg) )
28
29
```

Thus the score of 60% depicts that customers near Northeastern University have positive opinion about burritos.

```
> global_score
[1] 60
>
```

Python Implementation

In this section we would implement sentiment analysis using Python which would help us in measuring the Top 10 influencer more efficiently. Here we would stream the tweets from Twitter API using the developer's settings for Twitter account. We would require consumer key and consumer secret from the developer account on twitter. Also you need is the access token and a secret token to be created on the dashboard of Twitter developer account. Once we have that we would need Tweepy that is a Python module for Streaming Twitter Tweets. Tweepy API archive file can be gettable from github. Download it and extract it to a particular location. Use the below command in command prompt to install this library.

```
python setup.py install
```

Similarly we would also need elasticsearch library to use elasticseach's REST API. Elasticsearch allows you to index the stream of text data and find attributes in data quickly and efficiently. The main functions of this implementation would be to get the stream of data from Twitter for a particular keyword and then measure sentiments of each Tweets. Sentiments can be then aggregated and provided for finding the influential people.

The following is the code for implementing the job:

```
import json
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
from textblob import TextBlob
from elasticsearch import Elasticsearch

#consumer key, consumer secret, access token, access secret.
consumer_key="4aN0Mvc78BJvG0EC7BM80su5"
consumer_secret="rHcFMOmCx6v8jP7ZydovrtXjrehlQtd50FC0QR96va4ESoNxM"
```

```
access_token="3146945200-fD12d1UEfBAK5wx0IjRPIiJjPQ5RZ6nj9FxpBei"
access_token_secret="DzdYlNvNPg7VhZczj7l34luWAvkosZEzAgn1gWDI5ISC

# create instance of elasticsearch
es = Elasticsearch()

class TweetStreamListener(StreamListener):

    # on success
    def on_data(self, data):

        # decode json
        dict_data = json.loads(data)

        # pass tweet into TextBlob
        tweet = TextBlob(dict_data["text"])

        # output sentiment polarity
        print(tweet.sentiment.polarity)


        # determine if sentiment is positive, negative, or neutral
        if tweet.sentiment.polarity < 0:
            sentiment = "negative"
        elif tweet.sentiment.polarity == 0:
            sentiment = "neutral"
```

```
        else:
            sentiment = "positive"

        # output sentiment
        print(sentiment)

        # Details sentiment
        tweet = dict_data["text"]
        username = dict_data["user"]["screen_name"]

        # output sentiment
        print(tweet,username)

# add text and sentiment info to elasticsearch
        es.index(index="sentiment",
                 doc_type="test-type",
                 body={"author": dict_data["user"]["screen_name"],
                       "date": dict_data["created_at"],
                       "message": dict_data["text"],
                       "polarity": tweet.sentiment.polarity,
                       "subjectivity": tweet.sentiment.subjectivity,
                       "sentiment": sentiment})
        return True

    # on failure
    def on_error(self, status):
```

```
        print(status)


if __name__ == '__main__':
    # create instance of the tweepy tweet stream listener
    listener = TweetStreamListener()

    # set twitter keys/tokens
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)

    # create instance of the tweepy stream
    stream = Stream(auth, listener)

    # search twitter for "burrito" keyword
    stream.filter(track=['burrito'])
```

Code Explanation: The following are the steps we included in the code:

- Import all the required libraries including the Tweepy API and Elastic Search API
- Get the access token, secret token, access key, secret key and store it in variables
- Inside class `TweetStreamListener` define two functions:
    - `on_data`: It gets implemented on receiving the data
    - `on_error`: It gets implemented if there is any error in processing data
- Load data from JSON objects into the `dict_data`
- `TextBlob` would load the text field from the `dict_data` which are nothing but the tweets
- Print the twitter sentiments score using the `polarity` `variable` of the sentiments.

- Sentiments in this section is measured in the score from -1 to 1. -1 to 0 is negative decreasing as it tends towards 0. O is neutral whereas 0 to 1 is a positive score increasing towards 1.
- Print the sentiments on the console
- `On_error` function prints the error on the console if it encounters any problems.
- Elastic search would add an index to the tweets data extracting the important field.

We would also write some configuration to get connected to twitter API. We define the

```
listener = TweetStreamListener()
```

This listener would constantly monitor for the new tweets in real time.

- For handshaking our application with twitter, keys and token are authenticated in the below function.

```
auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
```

- Get the streams in stream object and for the Keyword.
- Run the application and you get the below output for minimum input.

Output:

```
>>>
0.0
neutral
I'm at California Burrito Co. in Medellin, Antioquia https://t.co/idOd7z4V6G Paundreca
-0.6
negative
RT @grizzlyshibear: She loves being wrapped like a burrito on cold mornings http://t.co/CB24sIAHvq IzzatieAmran
0.0
neutral
RT @camirecamu: #TomorrowlandBrasil2015 @Book_Raver nos fuimos en burrito sabanero Book_Raver
0.2857142857142857
positive
i want a burrito right now 94voguehes
0.0
neutral
RT @embenson3: 1,000 rts and I'll take a burrito from Chipotle to prom dominiqruth
0.0
neutral
```

First line is the sentiment score while the second classifies it. The third we can see the tweets and the name of the user. Using mechanical turk we would aggregate and find mean of the score.

We can even store this data in database just by adding the below code:

```
import MySQLdb

import time


conn=MySQLdb.connect("localhost","root","password","twitterschema")


# obtain the cursor

c = conn.cursor()


c.execute("INSERT INTO taula (time, username, tweet, score) VALUES (%s,%s,%s)",

            (time.time(), username, tweet, score))


conn.commit()
```

Twitter Sentimental Analysis using Amazon EC2 platform

Sentiment analysis refers to various methods of examining and processing data in order to identify a subjective response, usually a general mood or a group's opinions about a specific topic. For example, sentiment analysis can be used to gauge the overall positivity toward a blog or a document, or to capture constituent attitudes toward a political candidate. Sentiment data is often derived from social media services and similar user-generated content, such as reviews, comments, and discussion groups. The data sets thus tend to grow large enough to be considered "big data."

Amazon EMR integrates open-source data processing frameworks with the full suite of service from AWS. The resulting architecture is scalable, efficient, and ideal for analyzing large-scale sentiment data, such as tweets over a given time period.

We'll launch an **AWS Cloud Formation stack** that provides a script for collecting tweets. You'll store the tweets in **Amazon S3** and customize a **mapper file** for use with **Amazon EMR**. Then you'll create an Amazon EMR cluster that uses a Python **natural language toolkit**, implemented with a **Hadoop** streaming job, to classify the data. Finally, you'll examine the output files and evaluate the aggregate sentiment of the tweets.

The purpose of this task would be to determine the sentimental score of Twitter data based on the keyword.

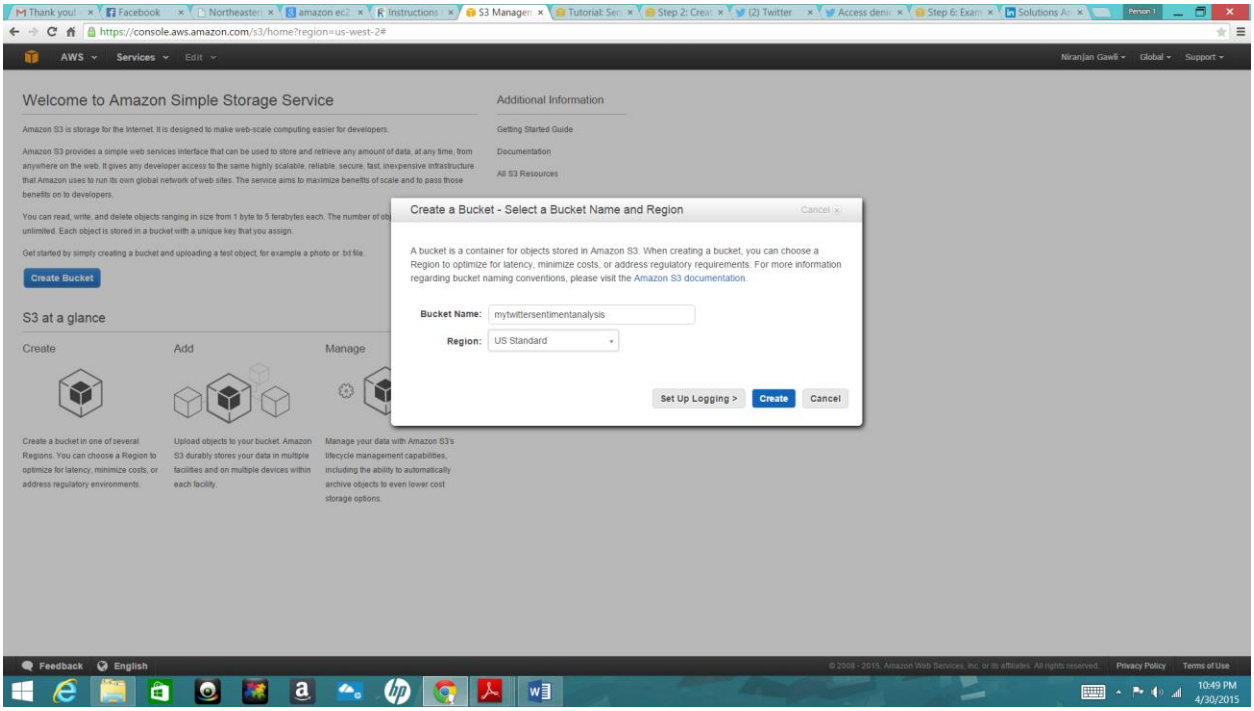
The following are the task to setup the implementation

- Step 1: Create a Twitter Developer Account
- Step 2: Create an Amazon S3 Bucket
- Step 3: Collect and Store the Sentiment Data
- Step 4: Customize the Mapper
- Step 5: Create an Amazon EMR Cluster
- Step 6: Examine the Sentiment Analysis Output

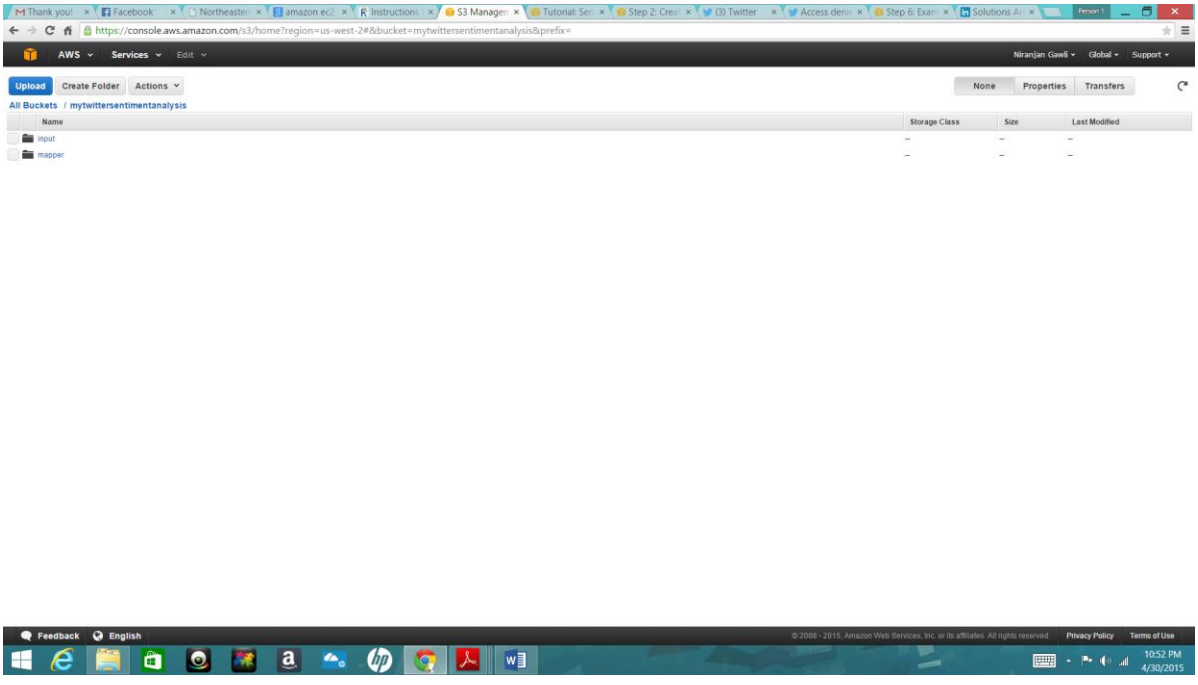
We would see how each step would work with the screenshots provided below:

Step 1: Create a Twitter Developer Account

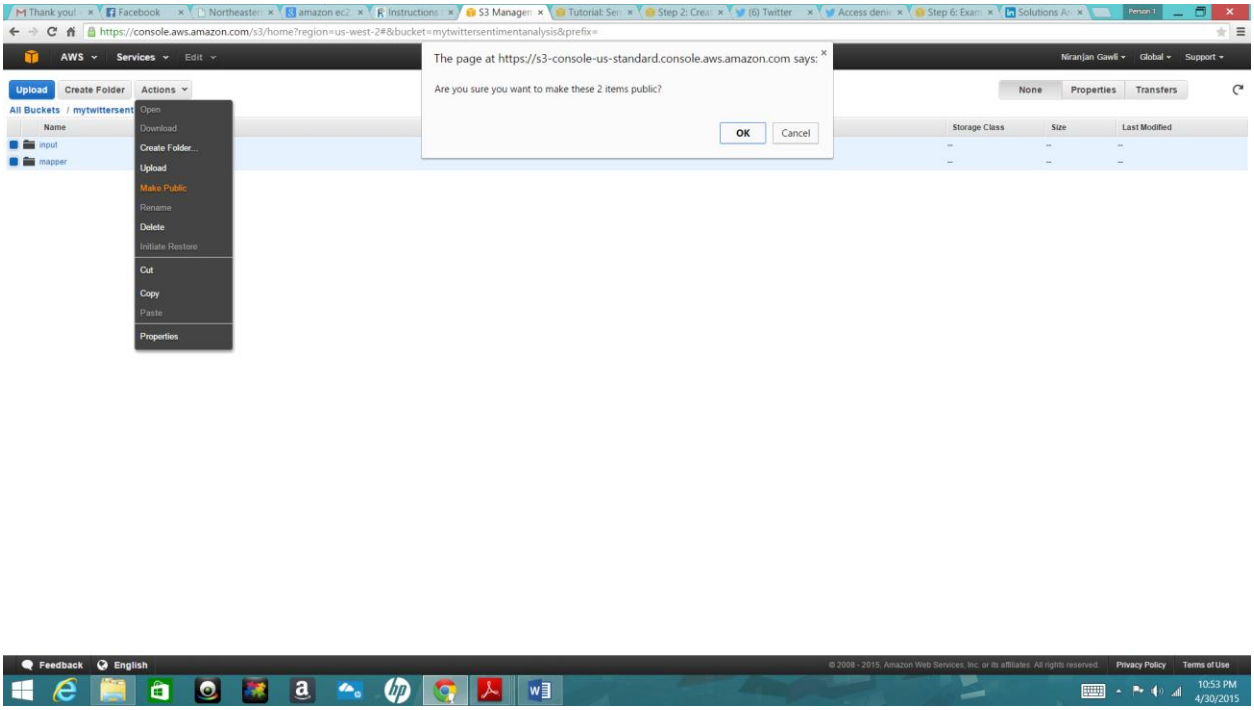
Step 2: Create an Amazon S3 Bucket



Create a folder as “input” and “mapper”:



Make this folders as “public”:



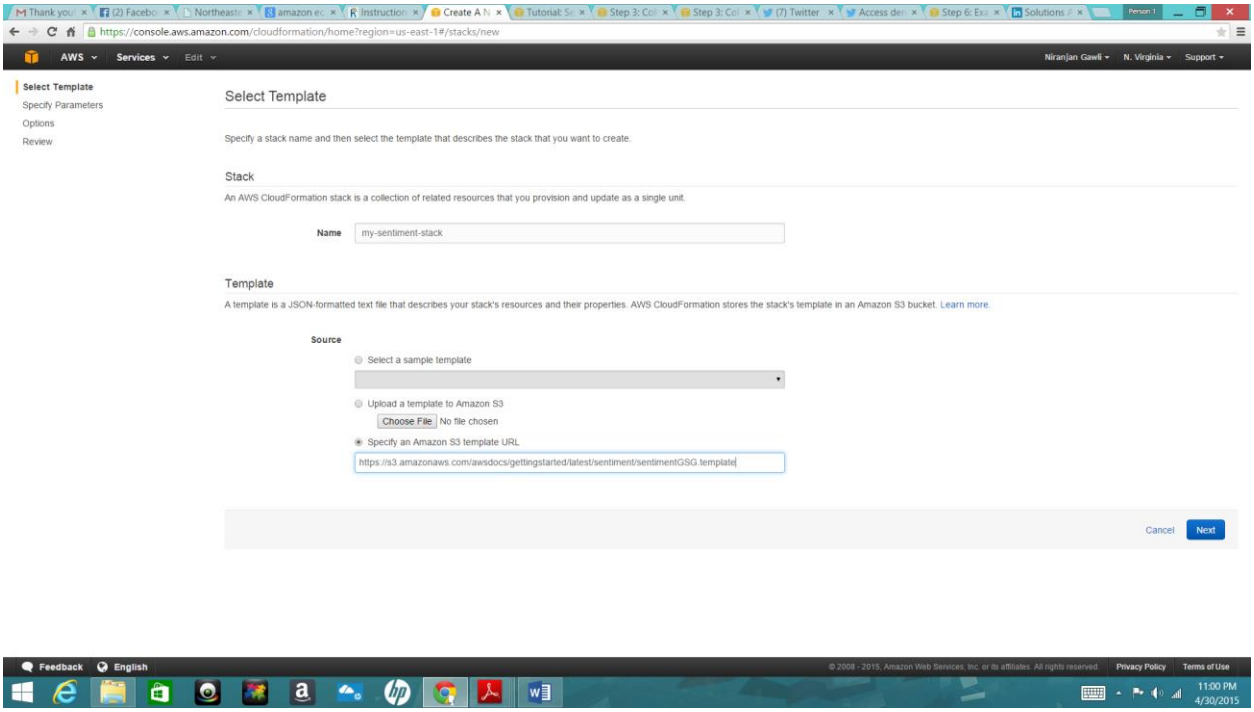
Step 3: Collect and Store the Sentiment Data

In this step, you'll use an AWS Cloud Formation template to launch an instance, and then use a command-line tool on the instance to collect the Twitter data. You'll also use a command-line tool (PuTTy) on the instance to store the data in the Amazon S3 bucket that you created.

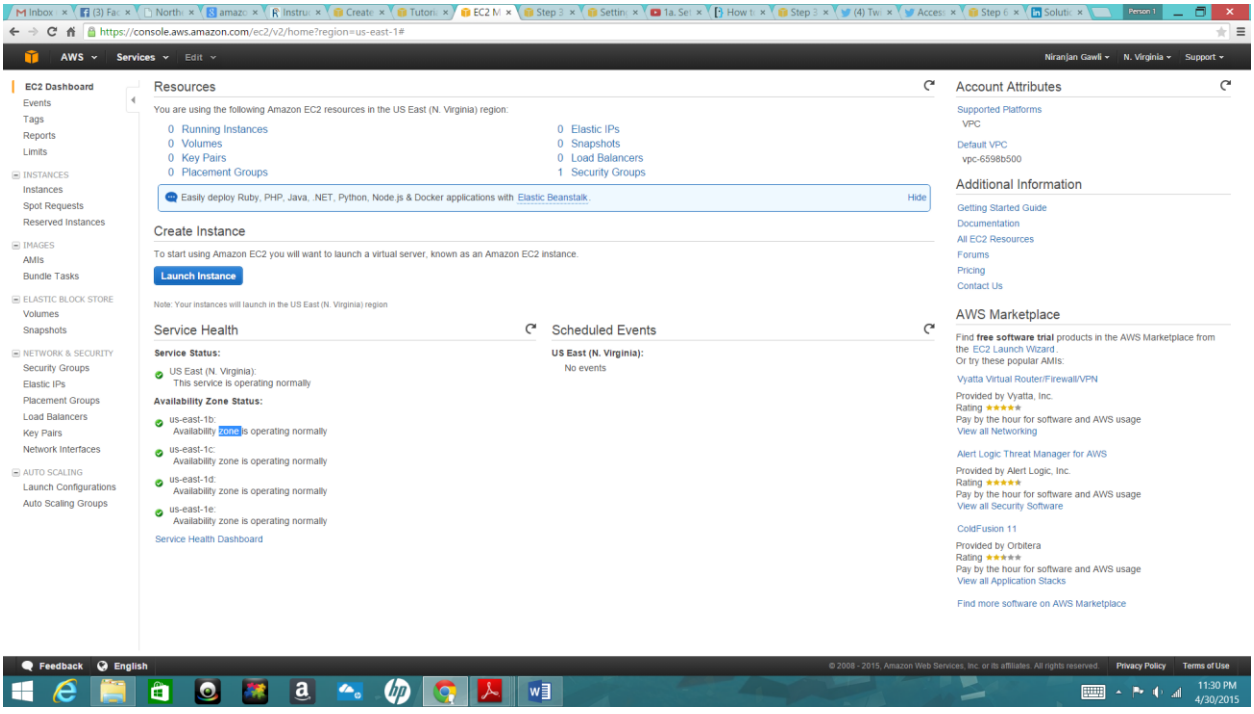
Tasks

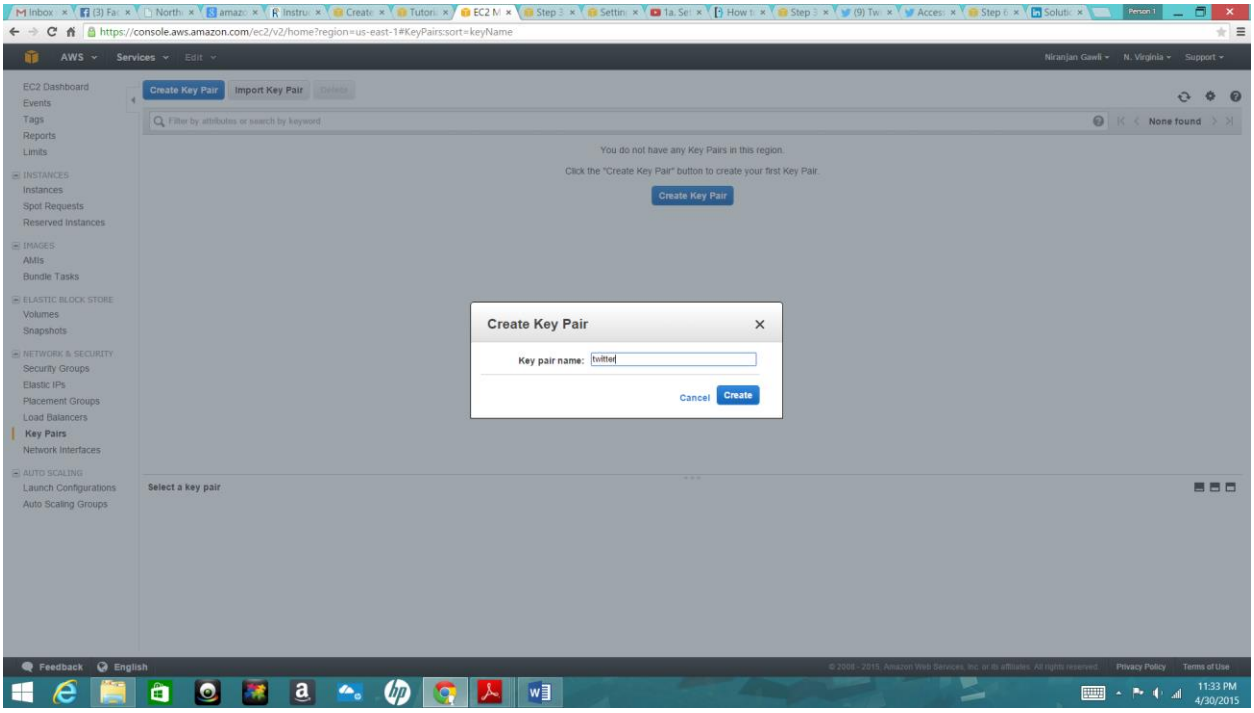
- Launch an Instance Using AWS CloudFormation

Create a new stack on AWS CloudFormation console.

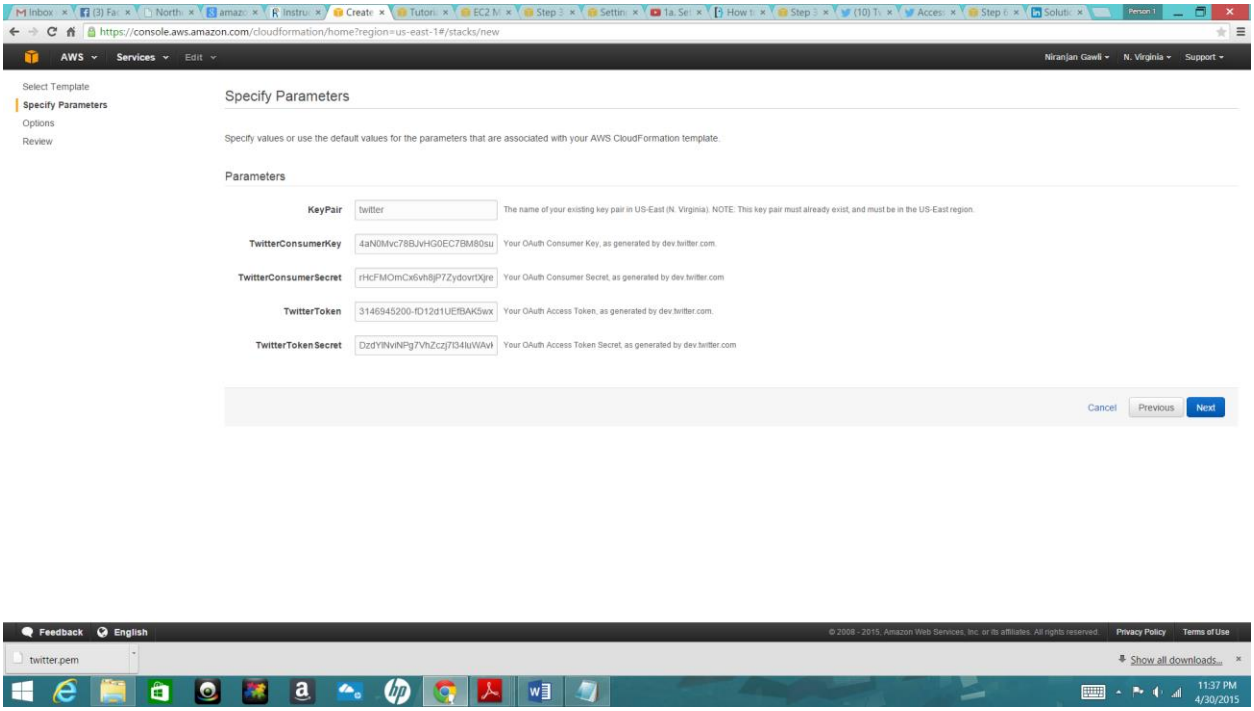


Create a Key Pair for running the instances

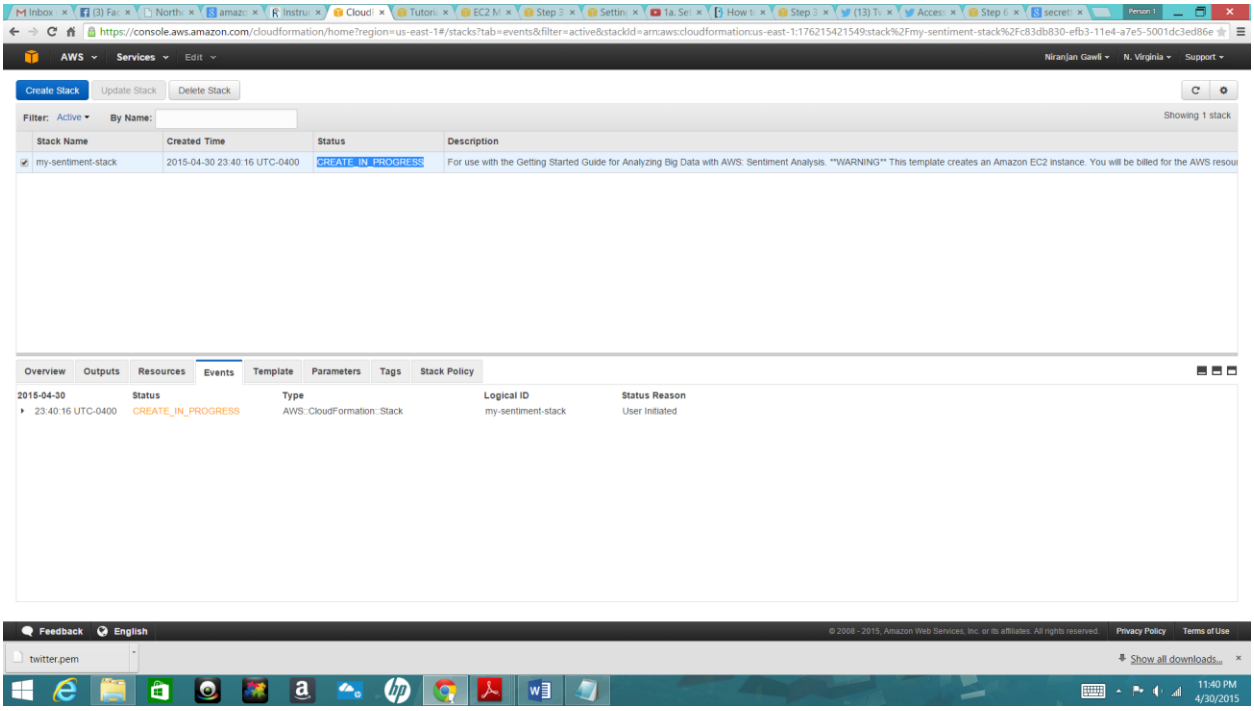




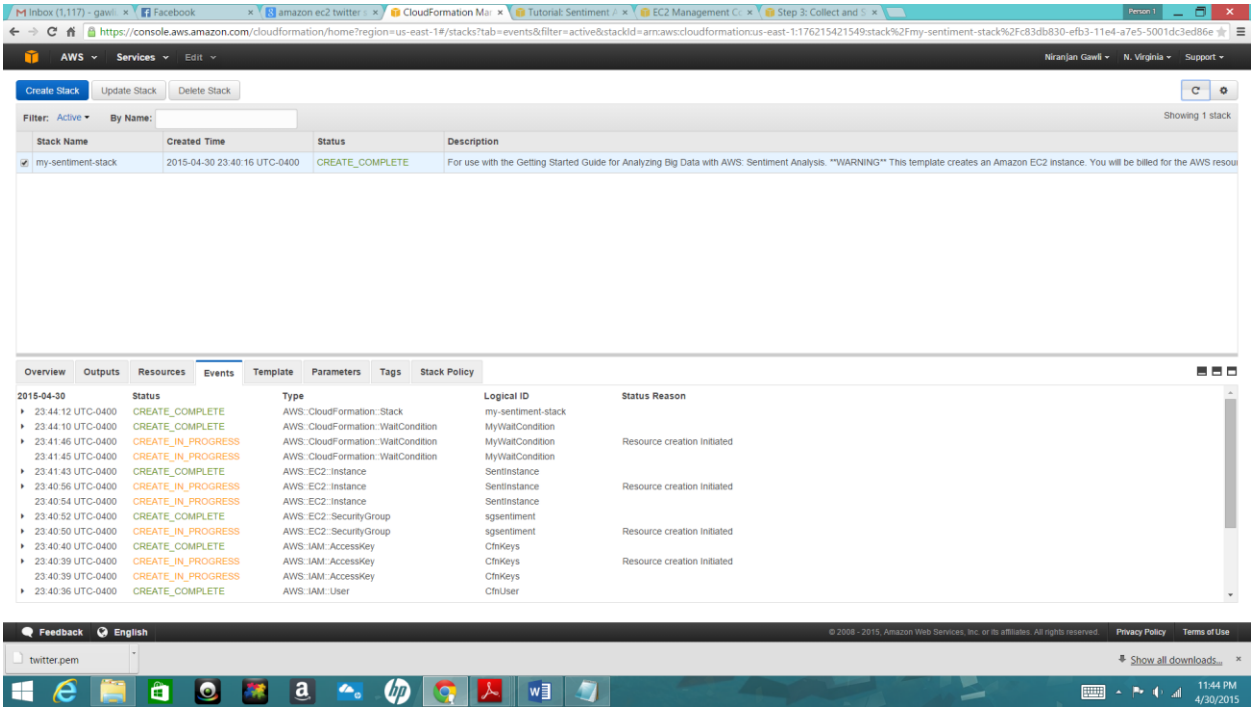
Connect to Twitter using access key, access token. Secret key and secret token



Wait till the stack is created, It would take some minutes:



Once it is complete, proceed to next step:

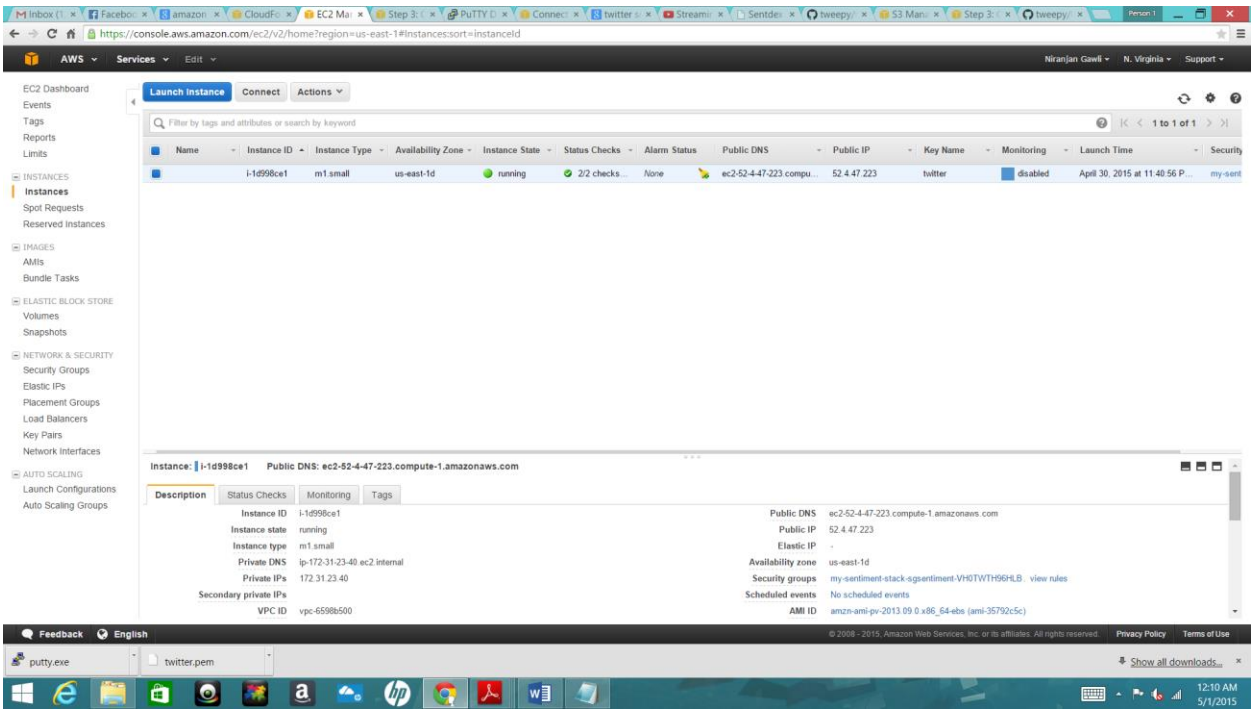


- Collect Tweets Using the Instance

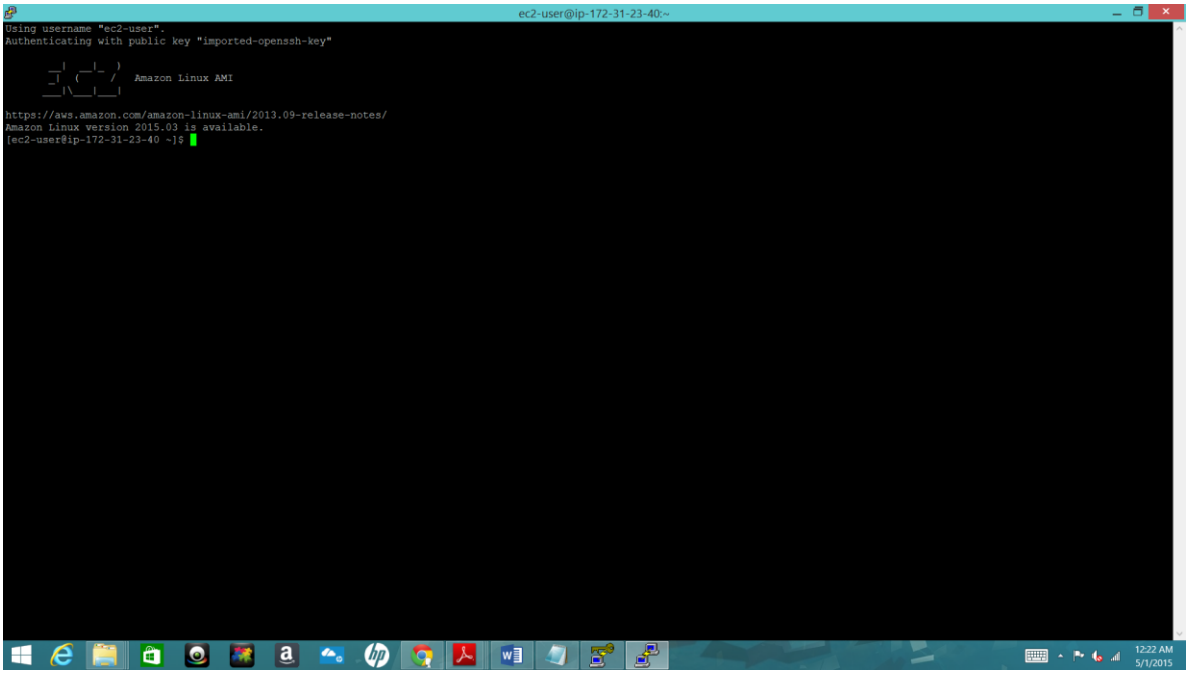
Collect tweets using tweepy an open source package for the use of open source twitter API

First we would connect to instance using putty.

Get the instance and public DNS name from EC2 console to connect:

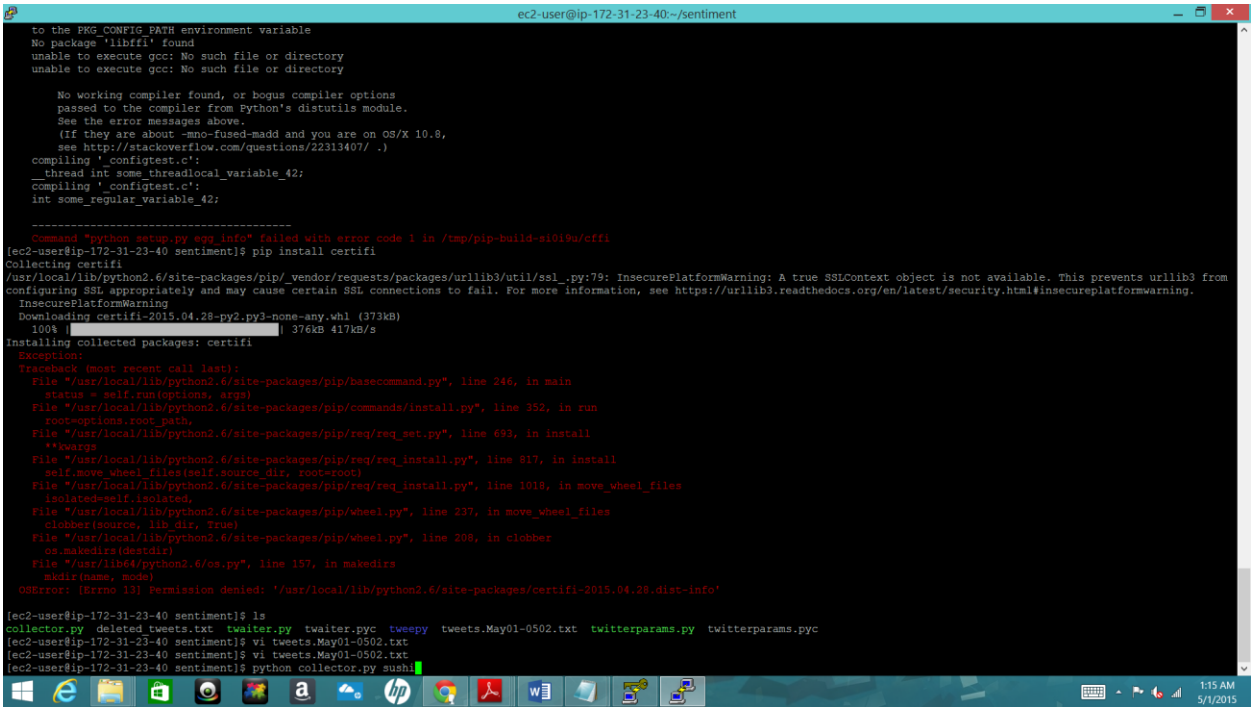


Connect using Putty to your EC2 instance:

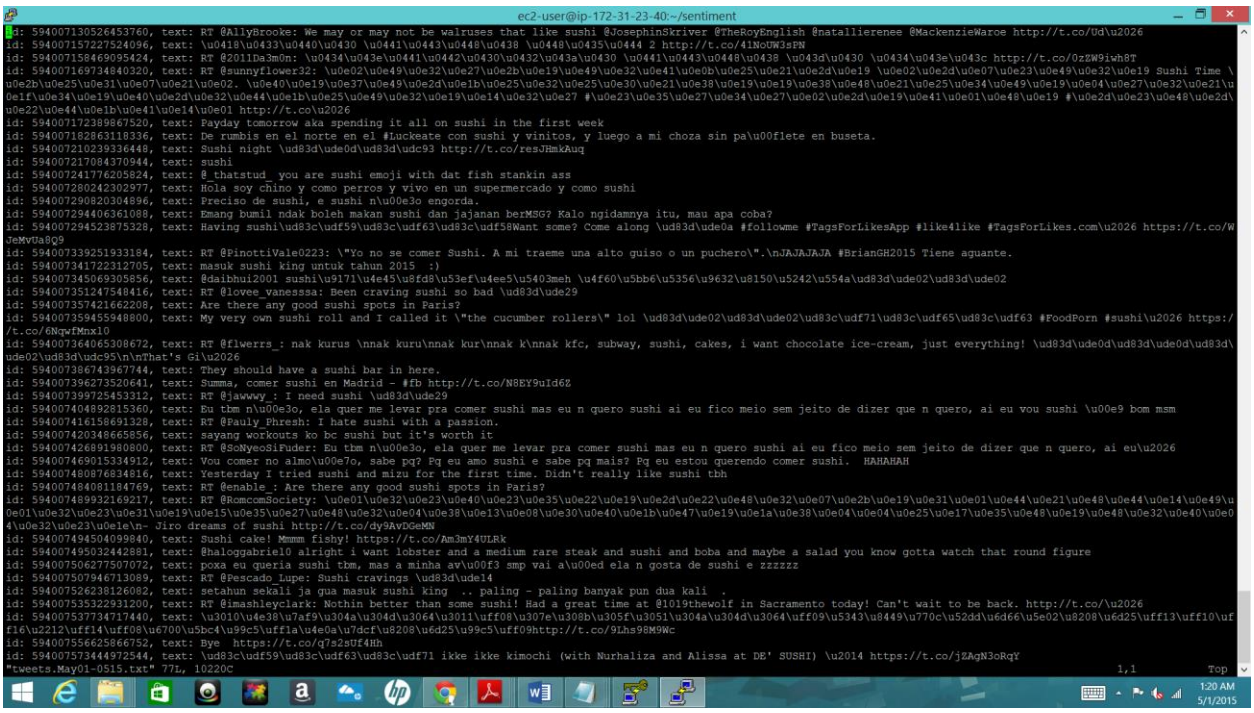


Use the command as shown below to collect tweets

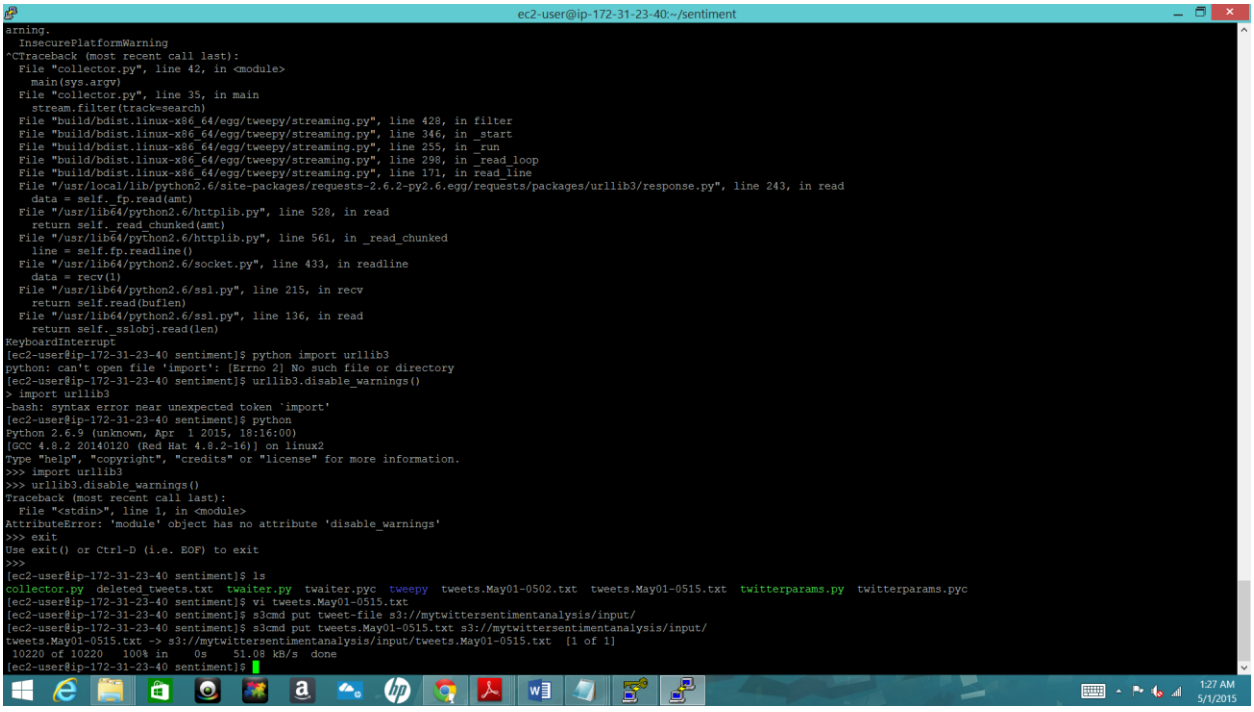
Python collector.py sushi



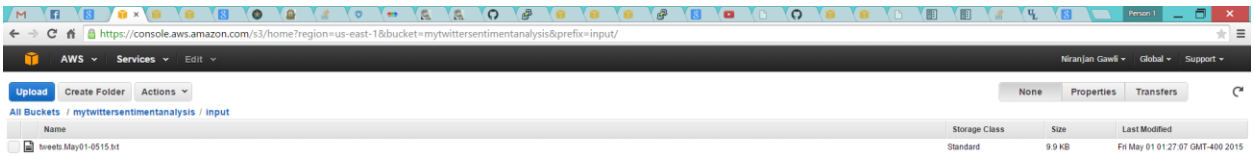
Below are the tweets:



- Store and copy the Tweets in Amazon S3



Text file is present in the following folder

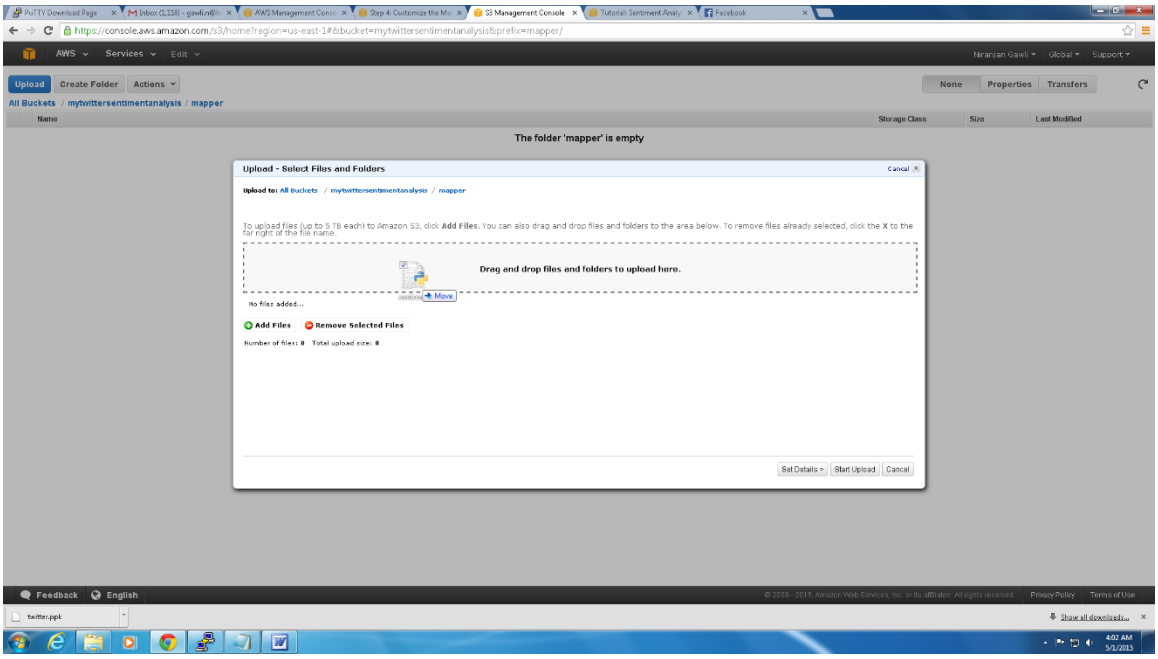


Step 4: Customize the Mapper

Write your own mapper in python: This mapper would get the tweets and convert into text. It would remove the punctuations and the stop words which makes it available for the sentimental analysis. The classifier would classify the tweets as positive, negative, neutral and sums the count of the no of tweets.

```
#!/usr/bin/python

import cPickle as pickle
import nltk.classify.util
from nltk.classify import NaiveBayesClassifier
from nltk.tokenize import word_tokenize
import sys

sys.stderr.write("Started mapper.\n");


def word_feats(words):
    return dict([(word, True) for word in words])


def subj(subjLine):
    subjgen = subjLine.lower()
    # Replace term1 with your subject term
    subj1 = "term1"
    if subjgen.find(subj1) != -1:
        subject = subj1
        return subject
    else:
        subject = "No match"
        return subject


def main(argv):
    classifier = pickle.load(open("classifier.p", "rb"))
    for line in sys.stdin:
        tolk_posset = word_tokenize(line.rstrip())
        d = word_feats(tolk_posset)
        subjectFull = subj(line)
        if subjectFull == "No match":
            print "LongValueSum:" + " " + subjectFull + ": " + "\t" + "1"
        else:
            print "LongValueSum:" + " " + subjectFull + ": " + classifier.classify(d) + "\t" + "1"

if __name__ == "__main__":
```

In mapper folder of S3 console on Amazon cloud upload your sentiment.py file

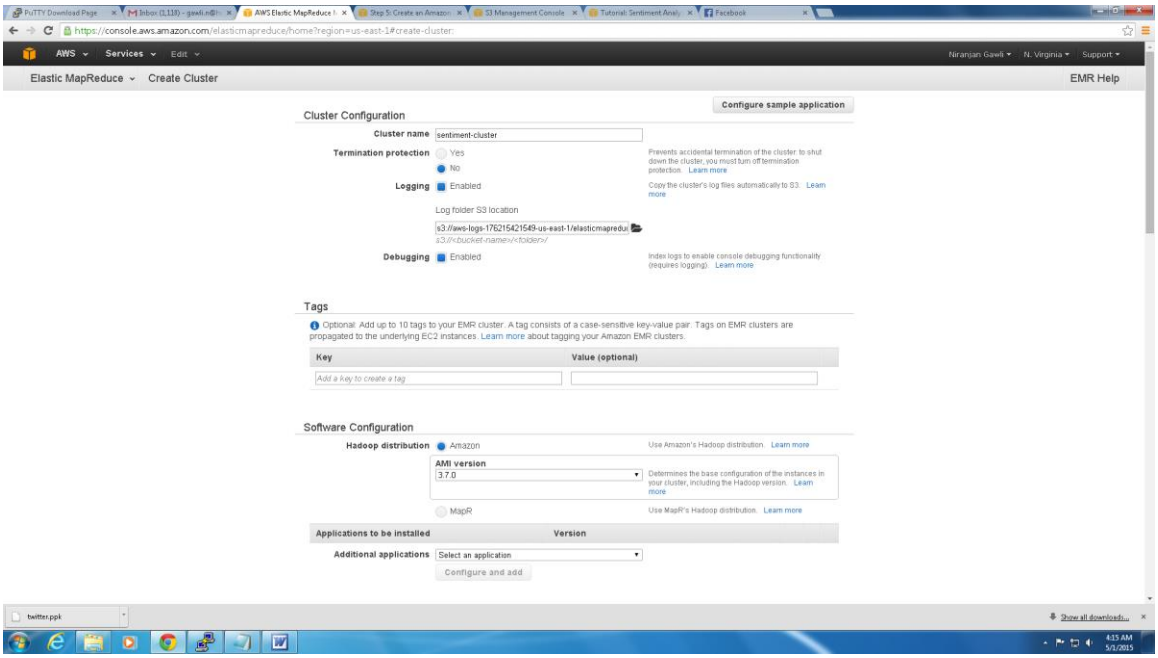


Step 5: Create an Amazon EMR Cluster

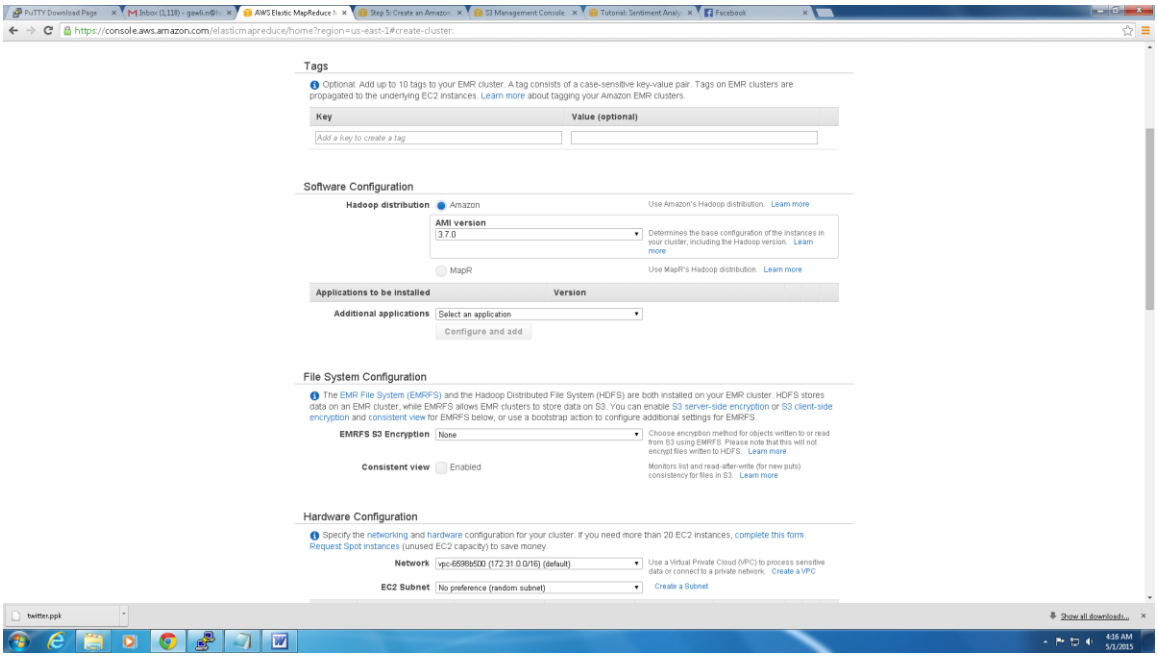
When you configure a cluster with a **Hadoop** streaming program in **Amazon EMR**, you specify a mapper and a reducer, as well as any supporting files. The following list provides a summary of the files you'll use for this tutorial.

- For the **mapper**, you'll use the file you customized in the preceding step.
- For the **reducer** method, you'll use the predefined Hadoop package **aggregate**.

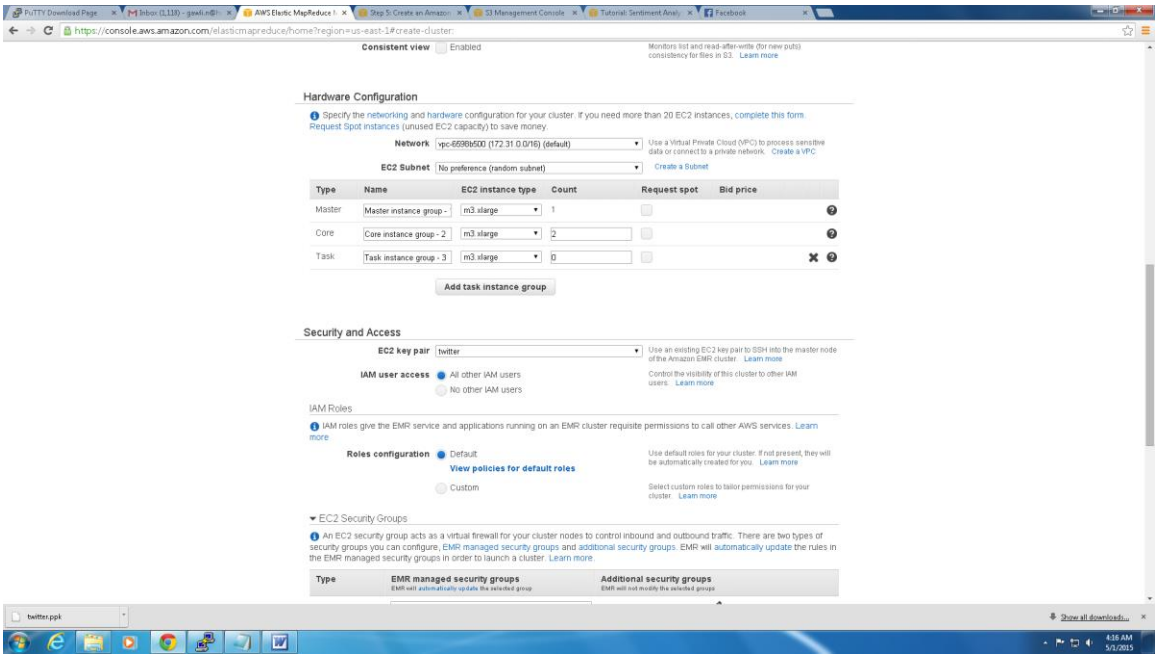
For Natural Language processing we would use Natural Language Toolkit. Amazon EMR bootstrap action would install NLT python module and loads this custom software onto one of the instances.



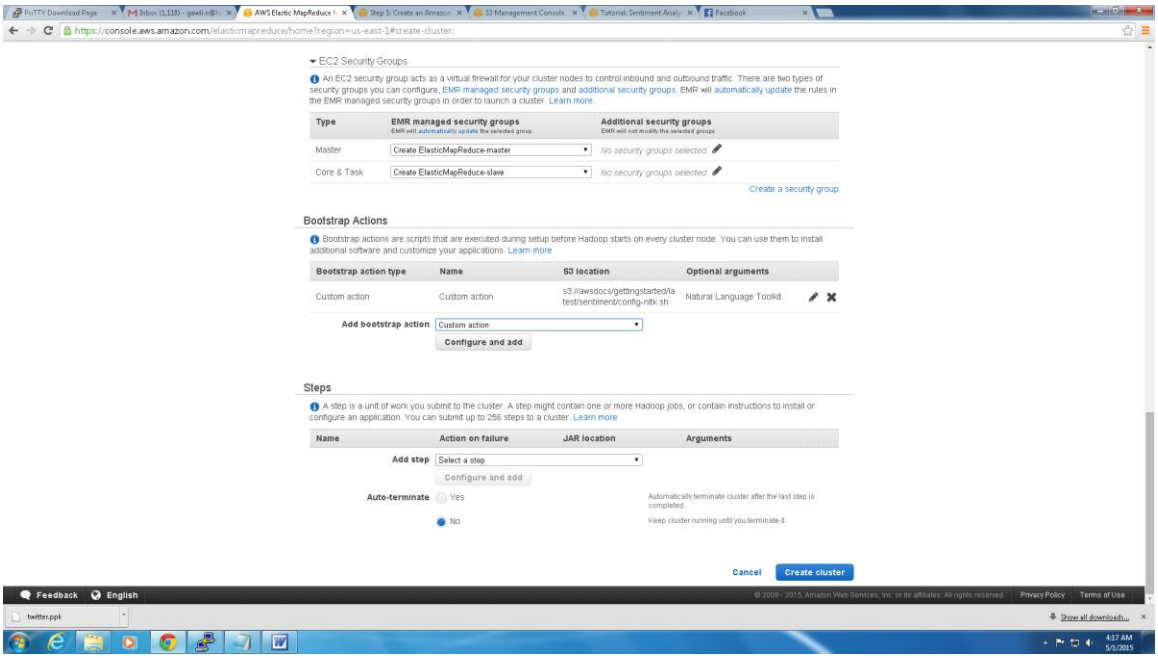
Keep the configuration and file system as default.



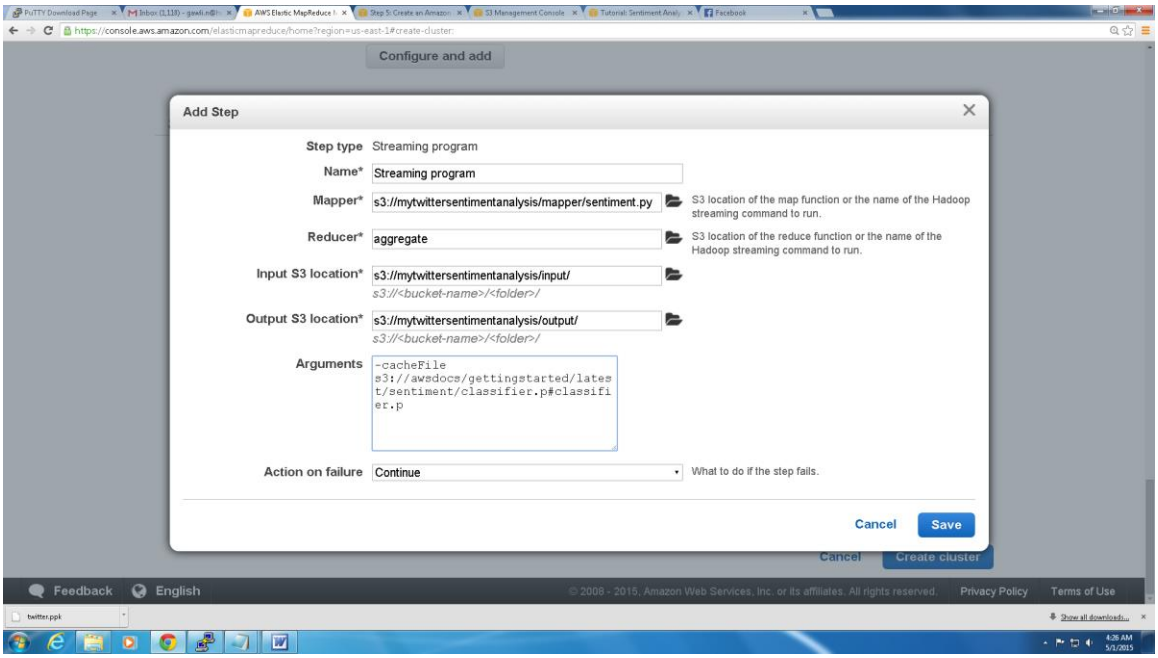
Keep the hardware configuration as it is:



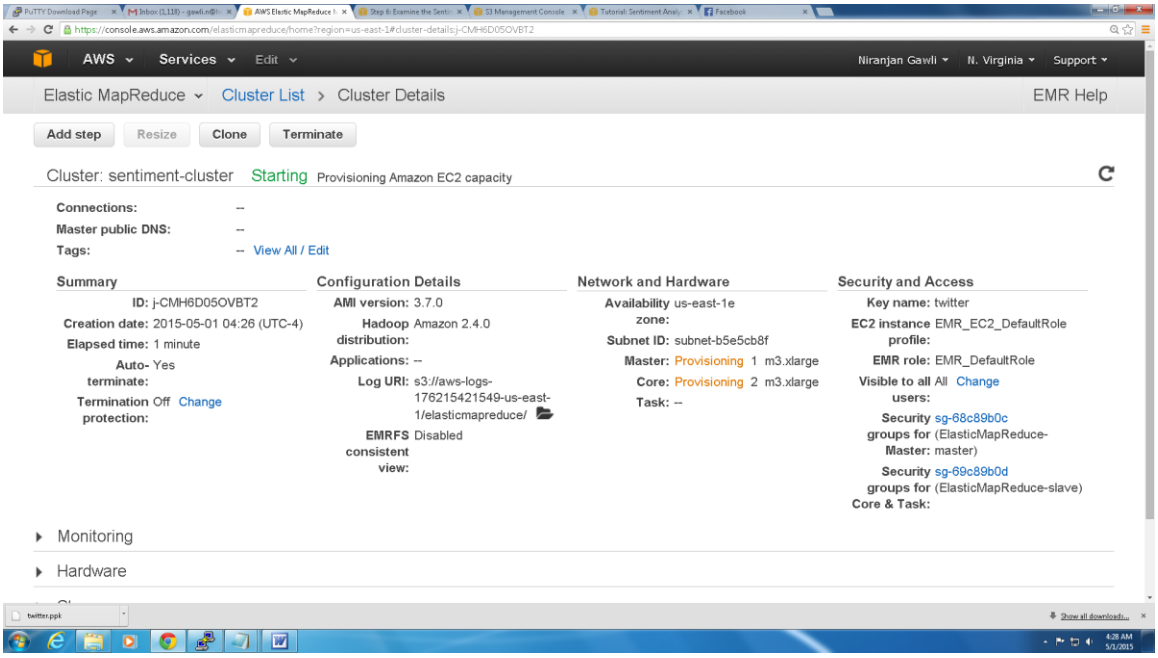
Add custom steps for Bootstrap steps:



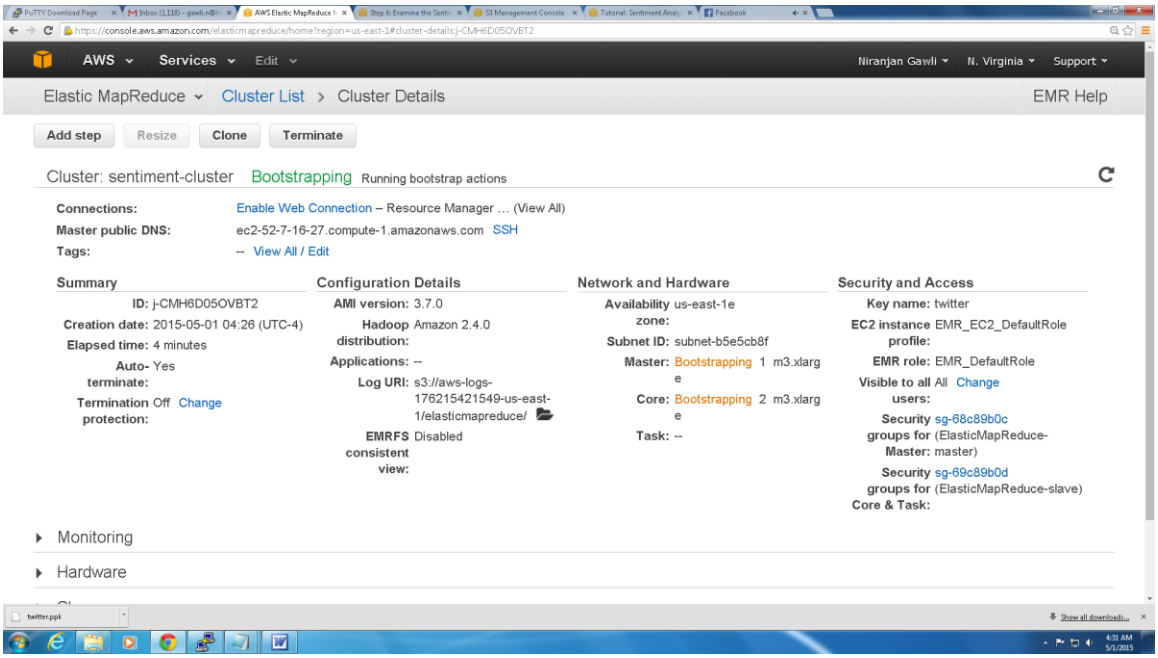
Add the following configurations to Steps: Define the mapper and reducer. Also provide the location of the input and the output files.



And then create a cluster for doing the Analysis.



Wait for some time till the analysis is done.

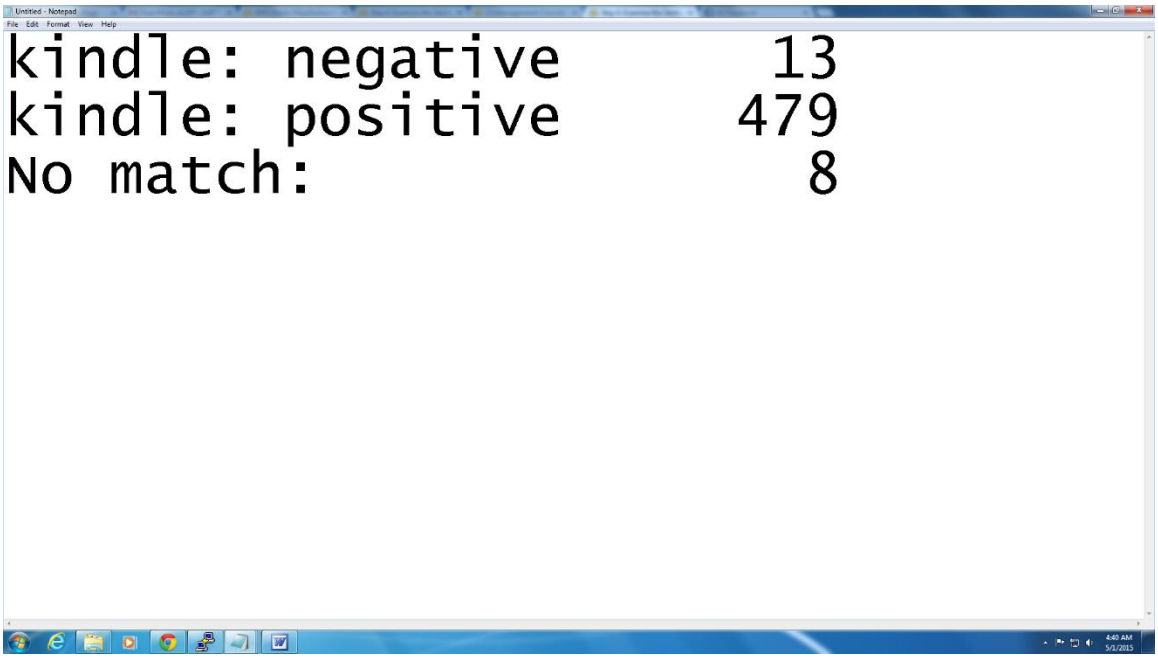


When your cluster's status in the Amazon EMR console is **Waiting: Waiting after step completed**, you can examine the results.

Once done locate the output file in your S3 bucket.

Download and save it.

Open the file in Text Editor.



Dashboard would provide with all the details of the process execution. You can monitor the log and check the status of the machine.

Microsoft Azure Machine Learning for Twitter Sentimental Analysis

Summary

This experiment demonstrates the use of the Execute R Script, Feature Selection, Feature Hashing modules to train a text sentiment classification engine.

Description

Sentiment analysis is a special case of text mining that is increasingly important in business intelligence and social media analysis. For example, sentiment analysis of user reviews and tweets can help companies monitor public sentiment about their brands, or help consumers who want to identify opinion polarity before purchasing a product. This experiment demonstrates the use of the Feature Hashing, Execute R Script and Filter-Based Feature Selection modules to train a sentiment analysis engine. We use a data-driven machine learning approach instead of a lexicon-based approach, as the latter is known to have high precision but low coverage compared to an approach that learns from a corpus of annotated tweets. The hashing features are used to train a model using the Two-Class Support Vector Machine (SVM), and the trained model is used to predict the opinion polarity of unseen tweets. The output predictions can be aggregated over all the tweets containing a certain keywords, such as brand, celebrity, product, book names, etc in order to find out the overall sentiment around that keyword.

Procedure:

The main steps of the experiment are:

Step 1: Collect data

Step 2: Preprocessing text data in R

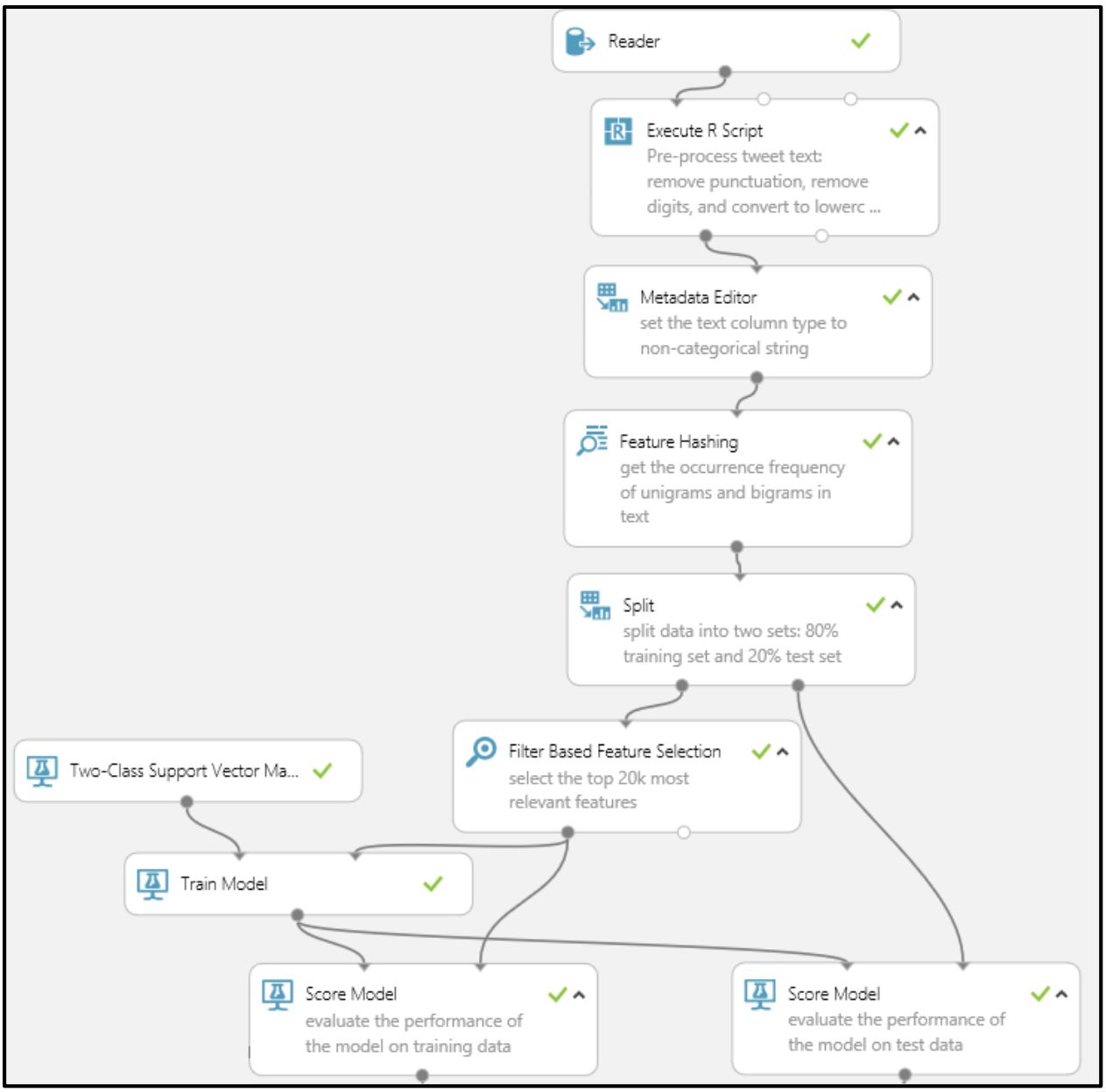
Step 3: Feature hashing and Filtered based feature selection

Step 4: Split the data into train and test

Step 5: Train model for prediction

Step 6: Evaluate model performance

Workflow:



Explanation:

Step 1: Collect data

We are collecting data from Public Cloud data source of Twitter.

Step 2: Preprocessing text data in R

Tweets are nothing but unstructured text and usually requires some preprocessing before it can be analyzed. We used the R code to remove punctuation marks, special character and digits, and then performed case normalization

After the text was cleaned, we used the *Metadata Editor module* to change the metadata of the text column as follows.

1. marked the text column as non-categorical column.
2. marked the text column as a non-feature.

The reason is that we want the learner to ignore the source text and not use it as a feature when training the model, but rather to use the extracted features that we build in the next step.

Step 3: Feature hashing and Filtered based feature selection

Feature hashing

The Feature Hashing module can be used to represent variable-length text documents as equal-length numeric feature vectors. An added benefit of using feature hashing is that it reduces the dimensionality of the data, and makes lookup of feature weights faster by replacing string comparison with hash value comparison.

Feature selection

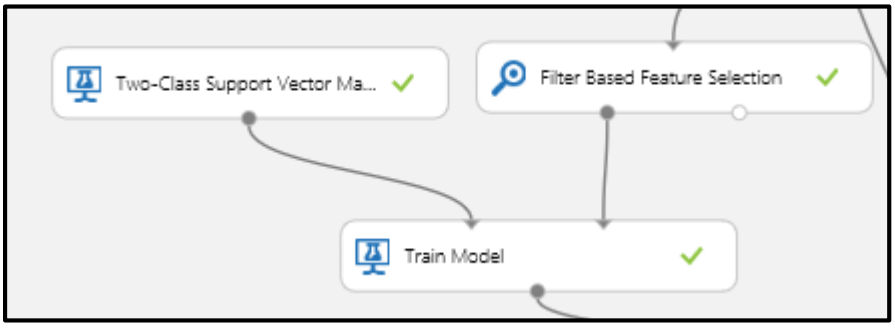
The classification complexity of a linear model is linear with respect to the number of features. However, even with feature hashing, a text classification model can have too many features for a good solution. Therefore, we used the Filter Based Feature Selection module to select a compact feature subset from the exhaustive list of extracted hashing features. The aim is to reduce the computational complexity without affecting classification accuracy.

Step 4: Split the data into train and test

The Split module in Azure ML is used to split the data into train and test sets where the split is stratified. The stratification will maintain the class ratios into the two output groups. We use the first 80% of the Sentiment140 sample tweets for training and the remaining 20% for testing the performance of the trained model.

Step 5: Train model for prediction

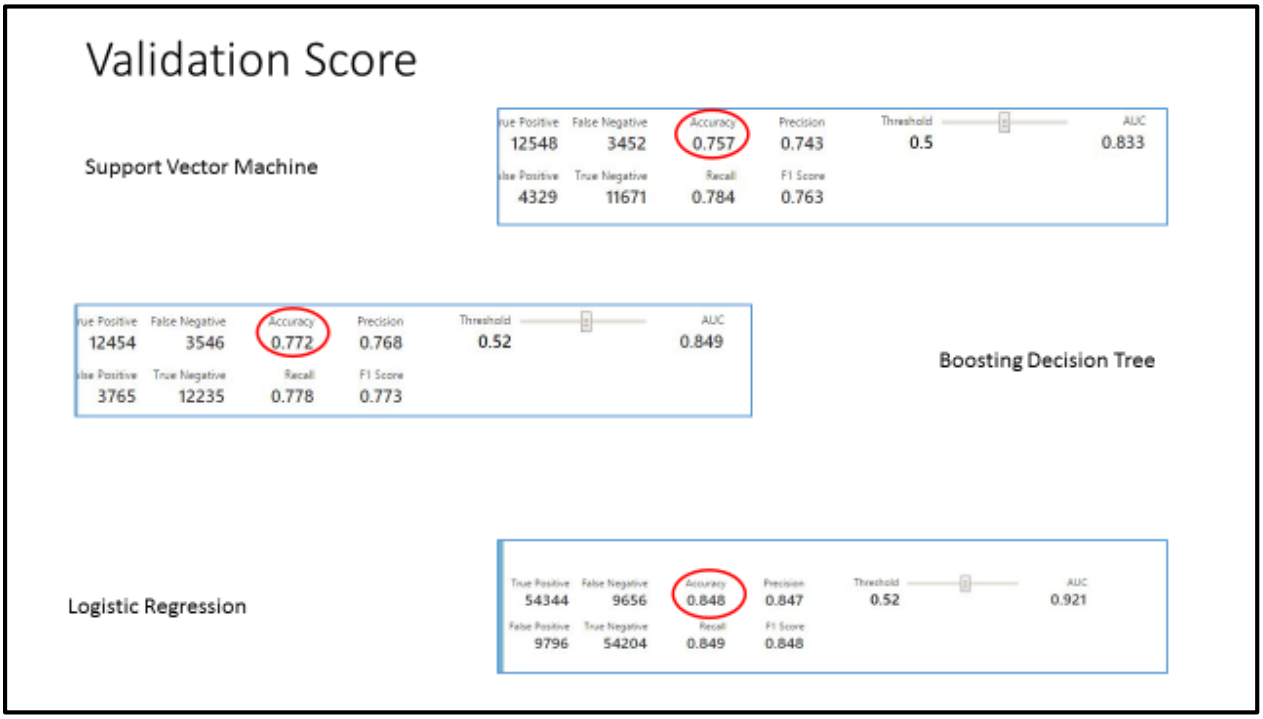
To train the model, we connected the text features created in the previous steps (the training data) to the *Train Modelmodule. Microsoft Azure Machine Learning Studio supports a number of learning algorithms but we select SVM and Boosting Decision Tree for illustration.



Step 6: Evaluate trained model performance

In order to evaluate the generalization ability of the trained Support Vector Machine model on unseen data, the output model and the test data set are connected to the Score Model module in order to score the tweets of the test set. Then connect the out predictions to the Evaluate Model module in order to get a number of performance evaluation metrics as shown below. Finally, we added the Evaluate Model module, to get the evaluation metrics (ROC, precision/recall, and lift).

Validation Score Comparison:



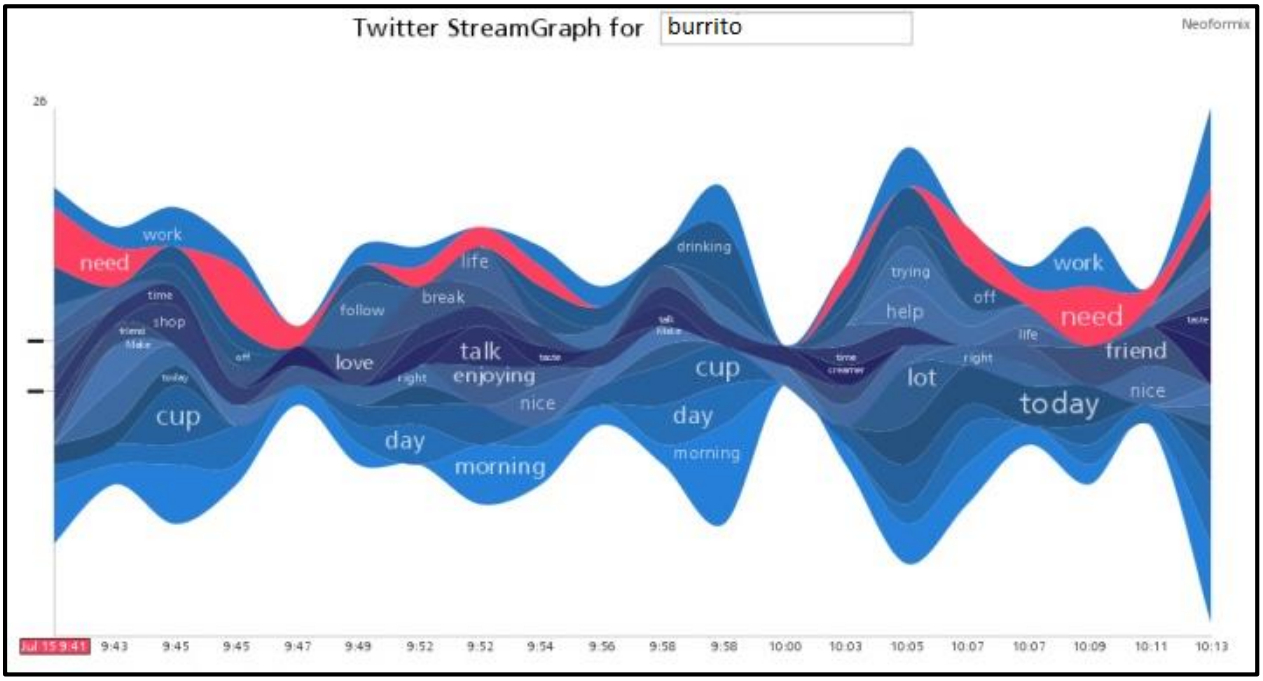
We have used three different classification algorithm for performing sentimental analysis. The algorithm we used are SVM, Boosting Decision Tree and Logistic Regression.

From the results we obtained, it is very clear that Logistic regression is best fit for current data set with highest accuracy of 84.8%. SVM and Boosting Decision Tree has accuracy of 75.7% and 77.2% respectively.

Conclusion:

Using the scoring module and with the saved model attached, we can use this model to analyze new data. Also, a key feature of Azure Machine Learning is the ability to easily publish models as web services on Windows Azure. In order to publish the trained sentiment prediction model, first we must save the trained model.

Data Visualization



Twitter Streamgraph Sample

Twitter is a very powerful platform of analytics especially for marketing. It has developed its APIs using which data analysts can extract useful information to make more sense of social media data. The main purpose of creating visualization is to communicate important information to business or non-technical person. Using Twitter APIs, we worked hard on creating simple and beautiful visualizations for better decision making. From the data we processed using twitter APIs, various attributes can be used for making more sense in data. Some of the most important *attributes* are:

1. Tweeted Text
2. Follower Count
3. Friends Count
4. Favorites Count
5. Statuses Count
6. Screen Name
7. Retweeted Count
8. Created Date

There are various tools that can be utilized for data visualization purpose. But, every tool has its pros and cons. We have considered most relevant and useful feature of such tools and finally selected following *tools*:

1. *Tableau*
2. *TIBCO Spotfire*
3. *R Libraries*
4. *Microsoft Azure Machine Learning Library*

Data Visualization for Twitter Analysis:

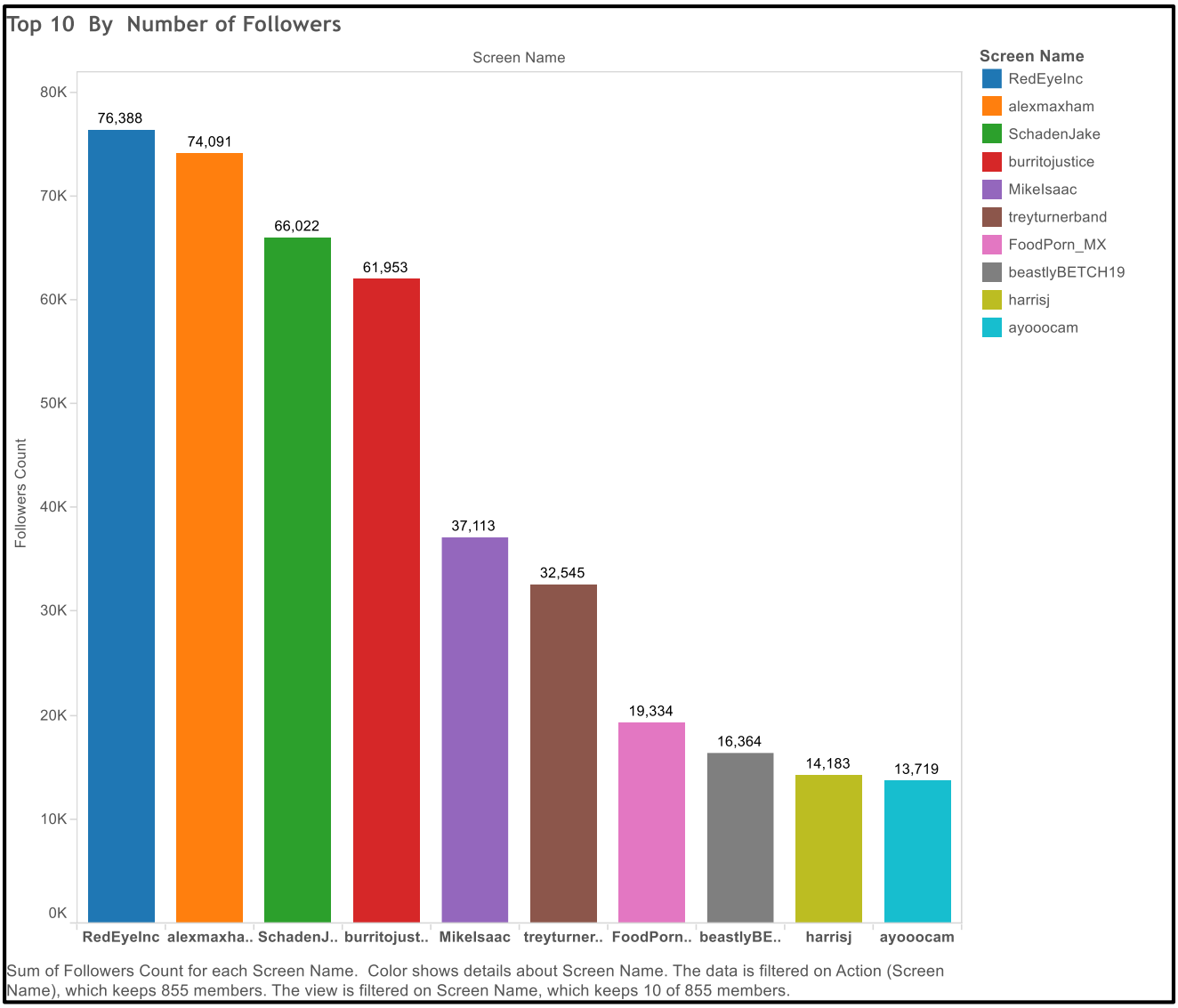
Some of the most important visualizations created for better understanding of twitter data related to Influential User, Hashtags and topics are represented as follows.

TABLEAU Visualization:

A. Top 10 Influential Users on Twitter:

The bar chart is created for getting top 10 Twitter Influential users. Such users can be used for in person targeted marketing purpose. These users can boost twitter marketing strategy by attracting more followers of that user. Following bar chart represents this information.

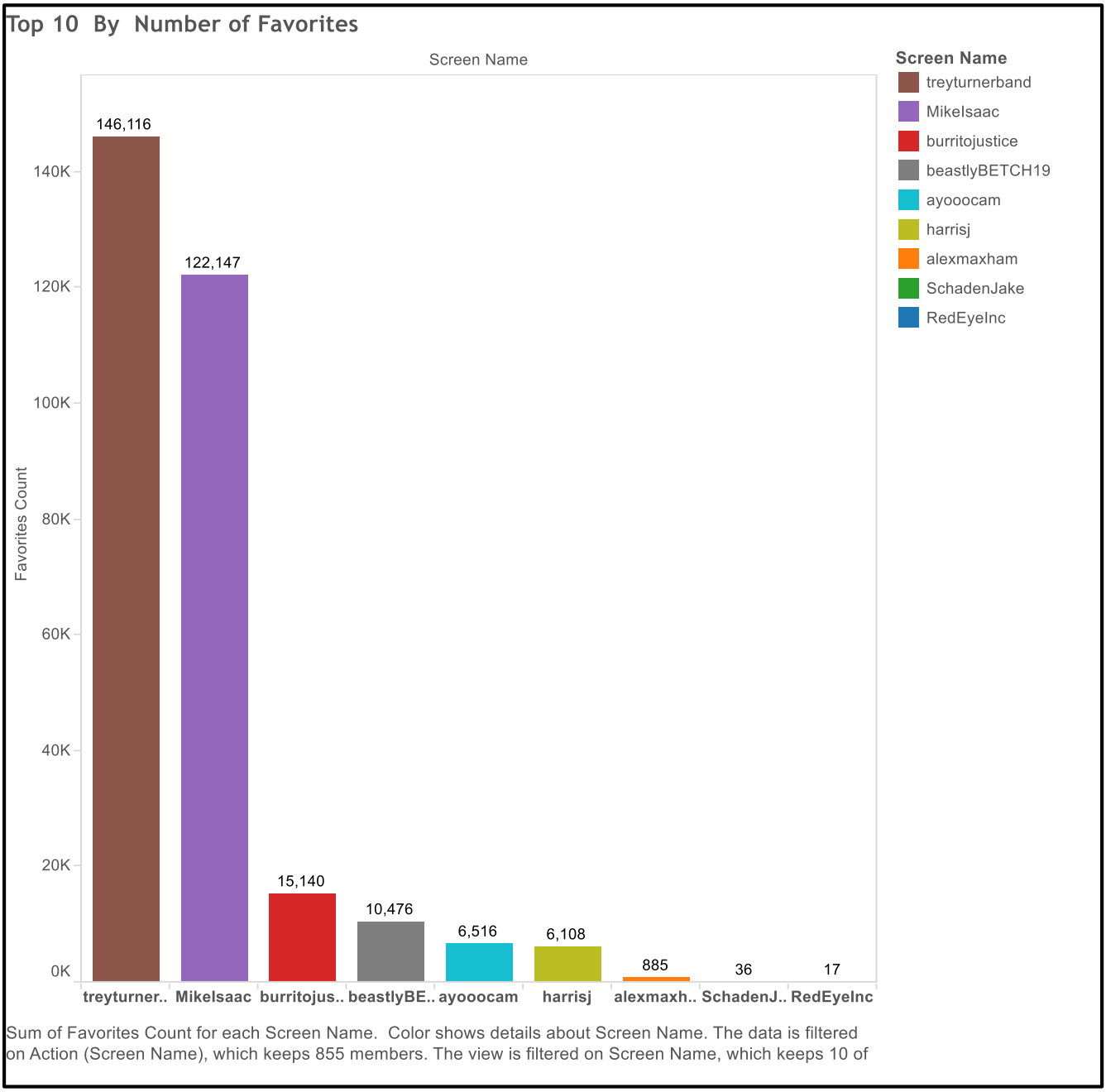
This graph represents total number of followers of a particular user. For the purpose of finding most influential users on Twitter, we have considered followers count seriously. According to twitter analysts, followers count most significantly represents popularity of that user on social media (here, *Twitter*). As the data we used was real time, the count is not that high compared to followers of some celebrity. We are not targeting celebrities for finding influential users. Our aim is to find such people from general audience.



Sum of Followers Count for each Screen Name. Color shows details about Screen Name. The data is filtered on Action (Screen Name), which keeps 855 members. The view is filtered on Screen Name, which keeps 10 of 855 members.

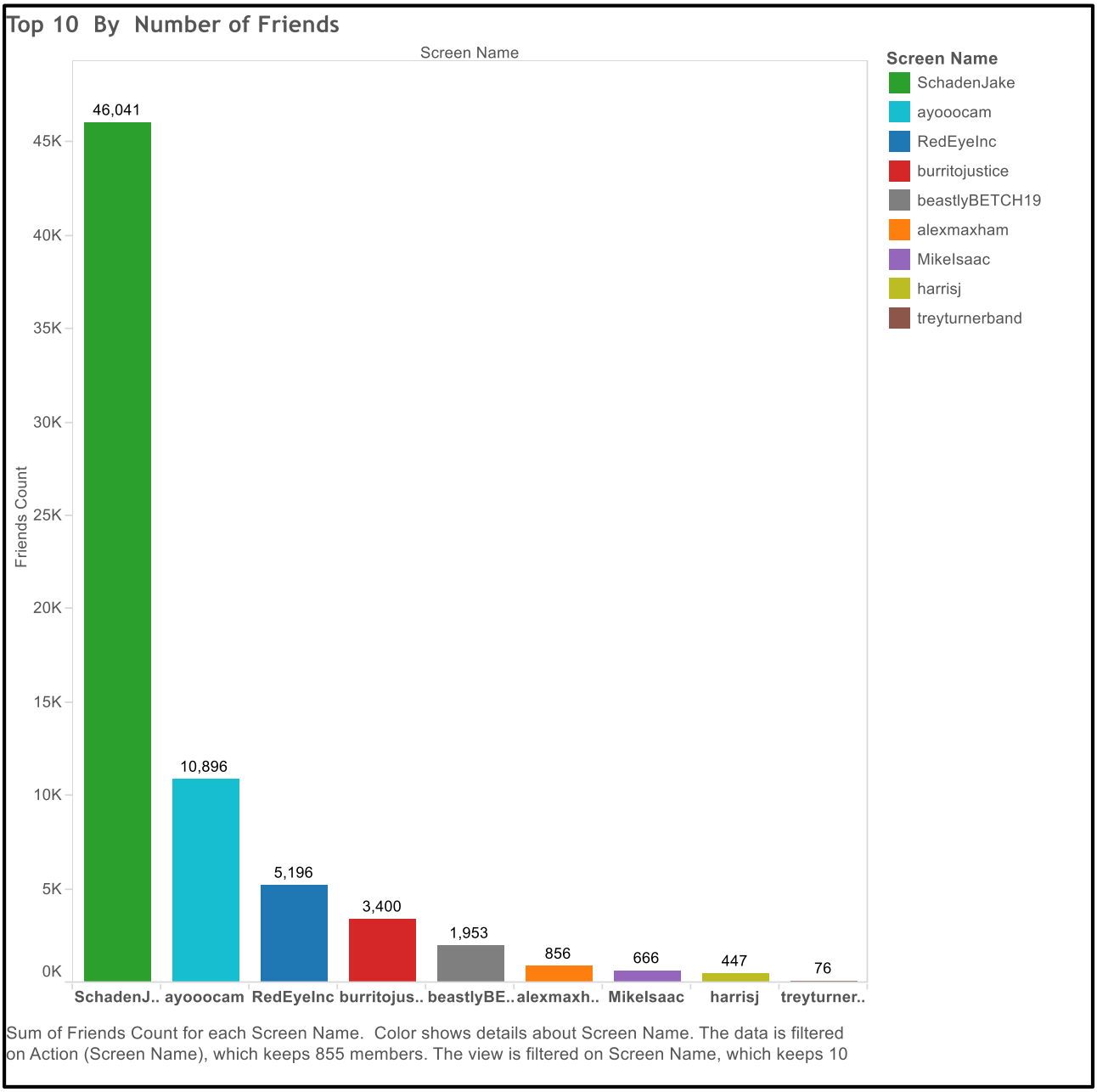
B. Top 10 Favorites:

In this graph, we focused on analyzing, maximum number of favorite count of top 10 users found in previous analysis. Favorites count can be used to interpret how actively that user is participating in social media activities. Based on this report, we can focus on response given to this users by its followers.



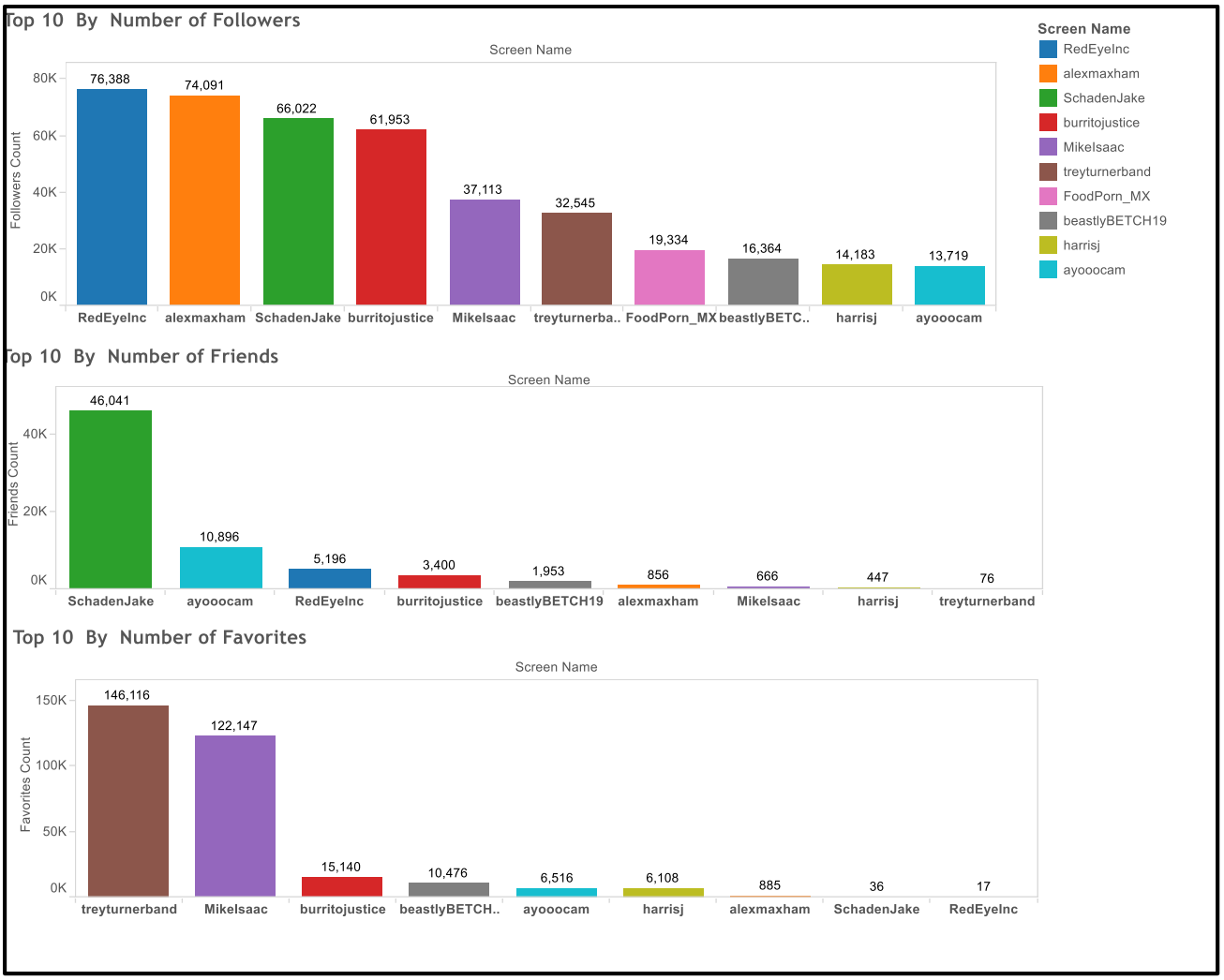
C. Top 10 Influential Users By Number of Friends:

This information can be used for deriving relationship between user and its social contacts. Considering Twitters privacy terms and conditions, it has become more important to consider number of friends of particular user. Private marketing can be enhanced by targeting users having maximum number of friends. For example, royal customers can be filtered based on this algorithm and perks can be offered to them for marketing in their network only.



D. Comparison Graph:

We have developed this comparison graph for better understanding of relationship, network and impact of top influential people. It can be seen from graph that there is no direct relationship between number of followers, their friends and favorites. All the three attributes are highly independent of each other. That's why, it can be used for different type of marketing on social media. It gives more flexibility for selecting and targeting potential customers to business people.

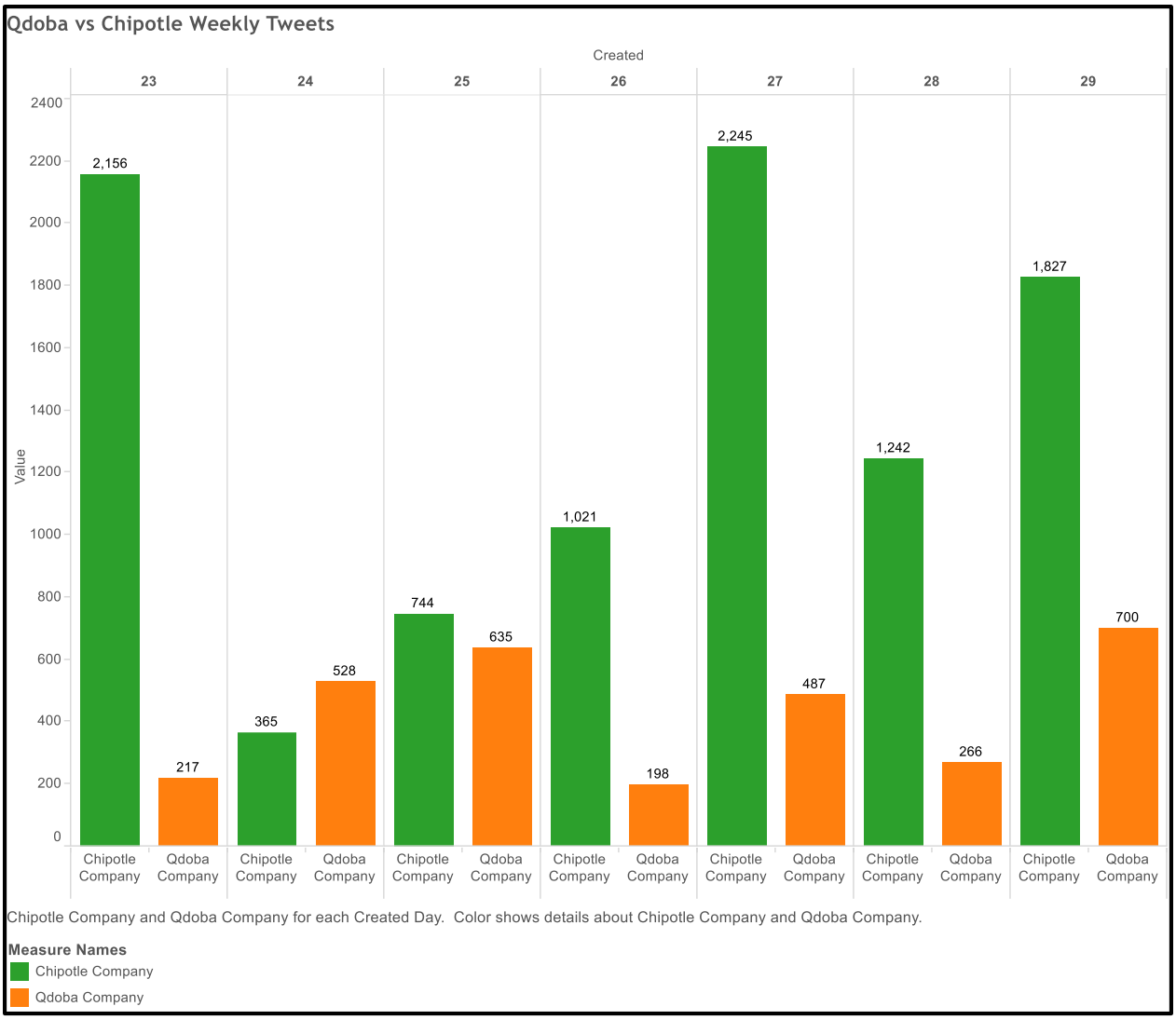


E. Brand Comparison:

Considering the basic assumption of this analysis, the purpose is to find most influential users for starting new business. One more thing that can help client to understand is which brand or company is trending in that category of business. We have considered Burrito Business plan by adapting franchise of either Qdoba or Chipotle company. That's why we have performed this analysis to measure popularity of these brands among social media customers.

We have analyzed tweets related to these brands individually. The number of tweets represents not only popularity of product/brand but also using sentimental analysis, we have analyzed opinion/review of customers. It is one of the most popular and trending analysis on Twitter platform.

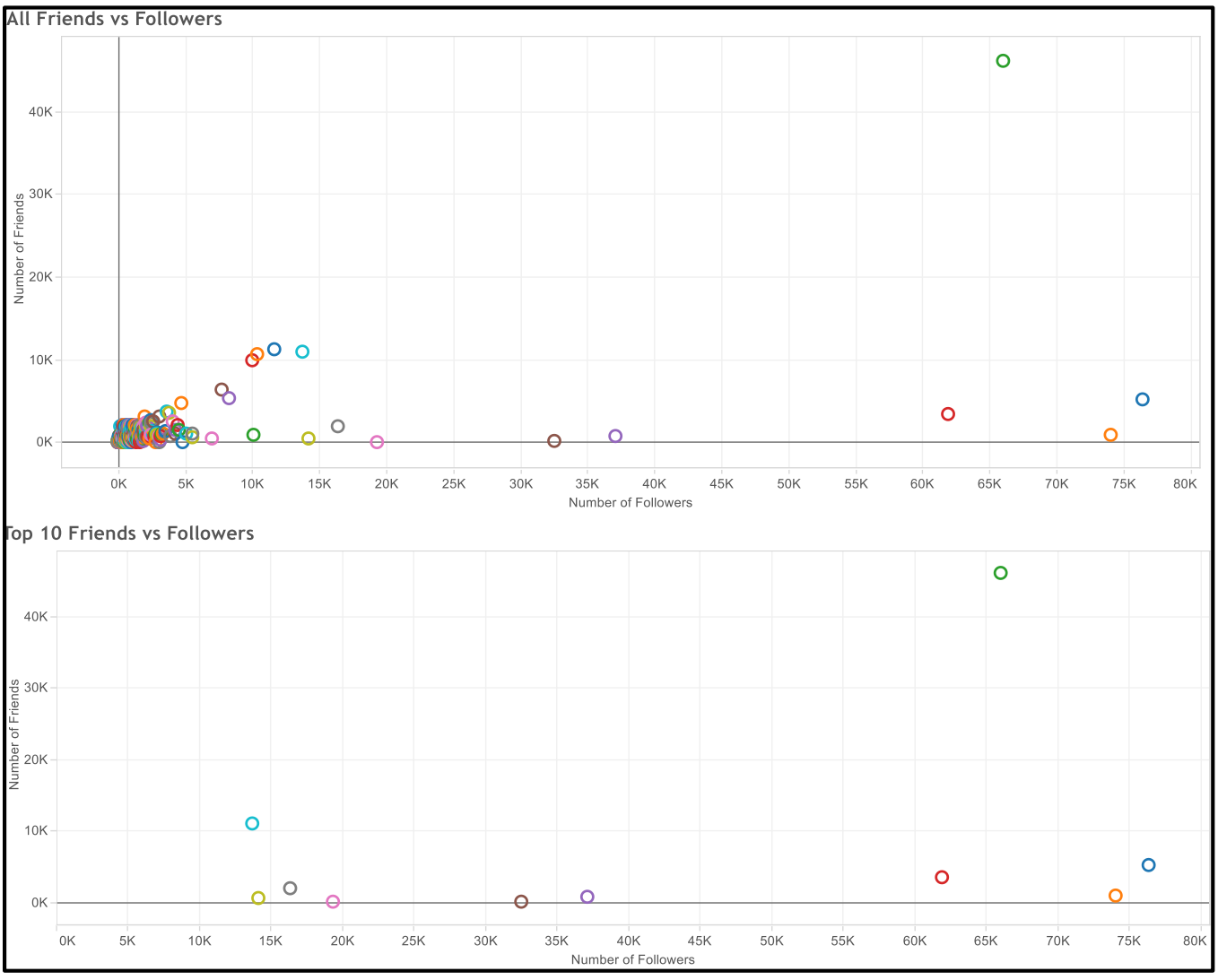
Here, we have analyzed tweets of Qdoba and Chipotle over one week period. It is very clear from this graph that Chipotle tweets are more than Qdoba tweets.


Qdoba vs Chipotle Weekly Tweets
Created
23
24
25
26
27
28
29
Value
2400
2200
2000
1800
1600
1400
1200
1000
800
600
400
200
0
2,156
217
365
528
744
635
1,021
198
2,245
487
1,242
266
1,827
700
Chipotle Company
Qdoba Company
Chipotle Company and Qdoba Company for each Created Day. Color shows details about Chipotle Company and Qdoba Company.
Measure Names
Chipotle Company
Qdoba Company

F. Followers and Friends Analysis:

In this graph, a scatter plot has been drawn for analyzing relationship between followers of influential users and their friends against all general users. The first graph represents follower vs friends for all users who tweeted about the topic. As most of the users are situated in left bottom of the graph because they have less number of followers and friends too. On the other hand few users have significantly more number of followers compared to number of friends.

The second graph in the bottom is drawn specially to show friends and followers of influential users we selected in previous stage. It is very clear that number of followers are very large compared to friends they have.

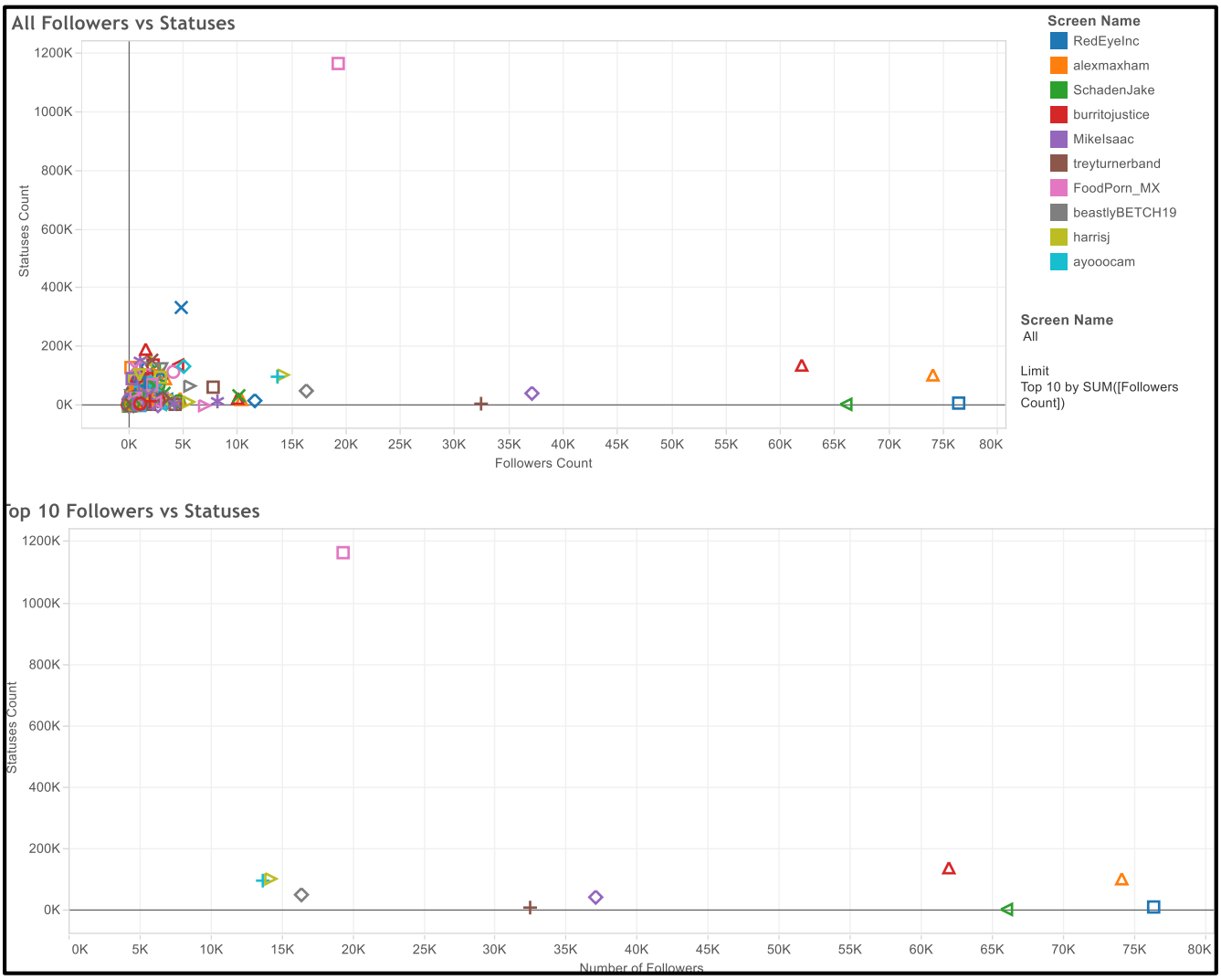


G. Followers and Status Count Analysis:

This graph represents number of followers of influential users and count of their statuses. So basically, it can be seen that most people have constant ratio between followers and status count.

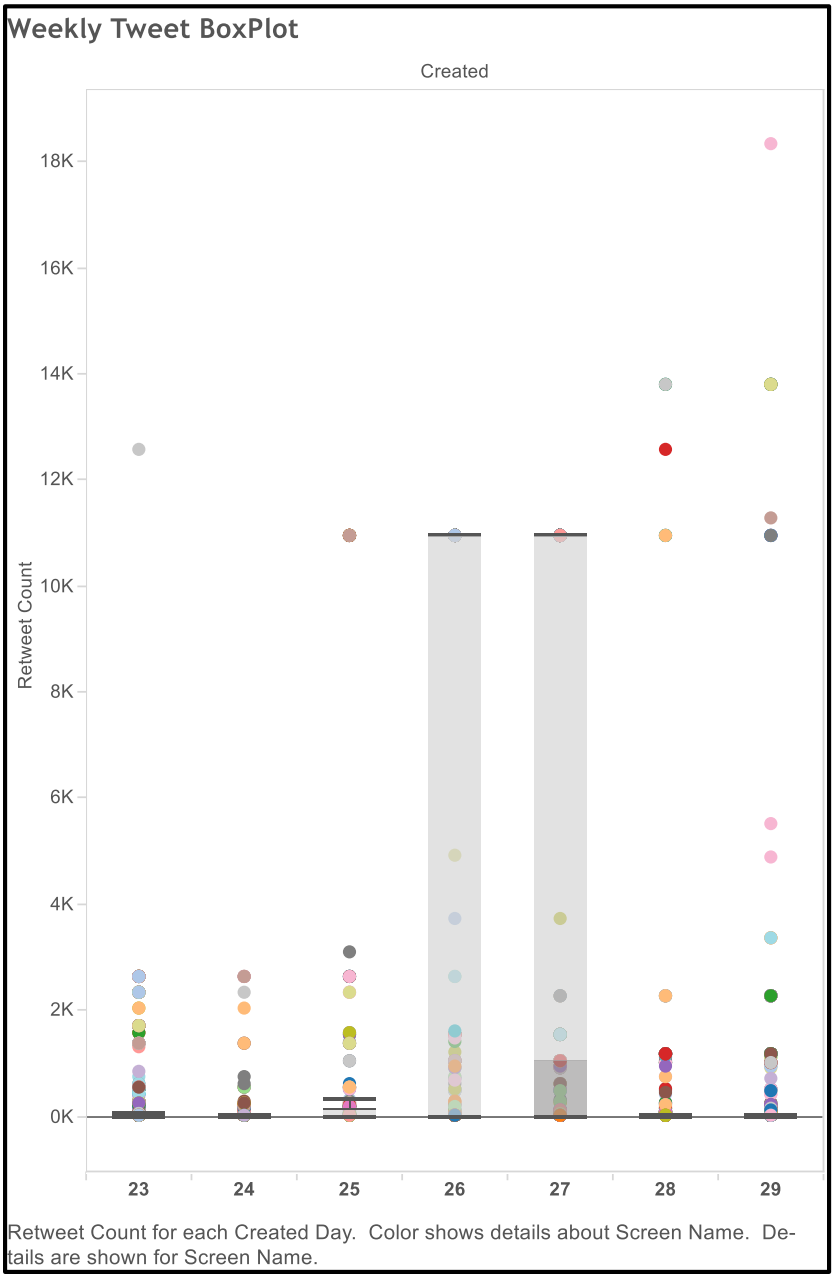
But from the second graph, we can state that influential people have comparatively more number of statuses than other users.

Different shapes and colors are used to represent different users in both graph.



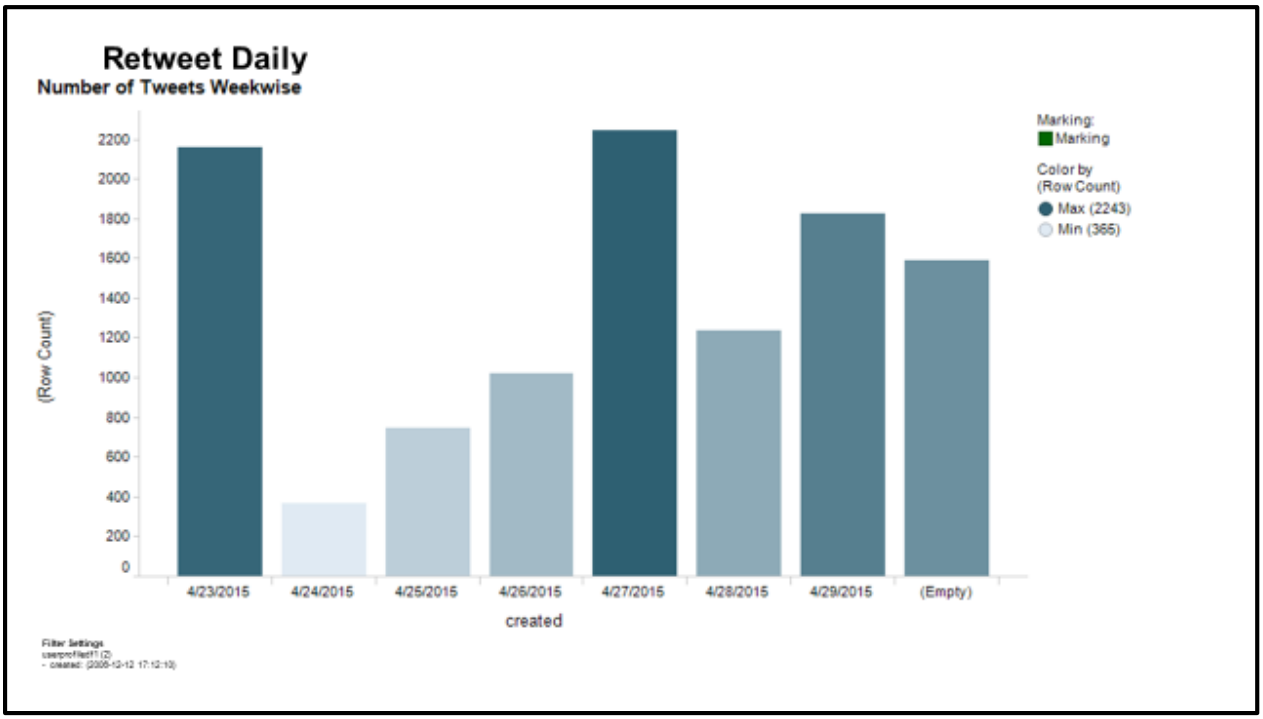
H. Weekly Numbers of Retweet Graph:

This boxplot represents retweet analysis of one week data collected from Twitter. In this graph we can observe that all colored outliers are nothing but Top influential users of twitter. Because there tweets are retweeted by most of their followers and friends, their retweet count is very large compared to average of other users. Here, outliers share most important and useful information.



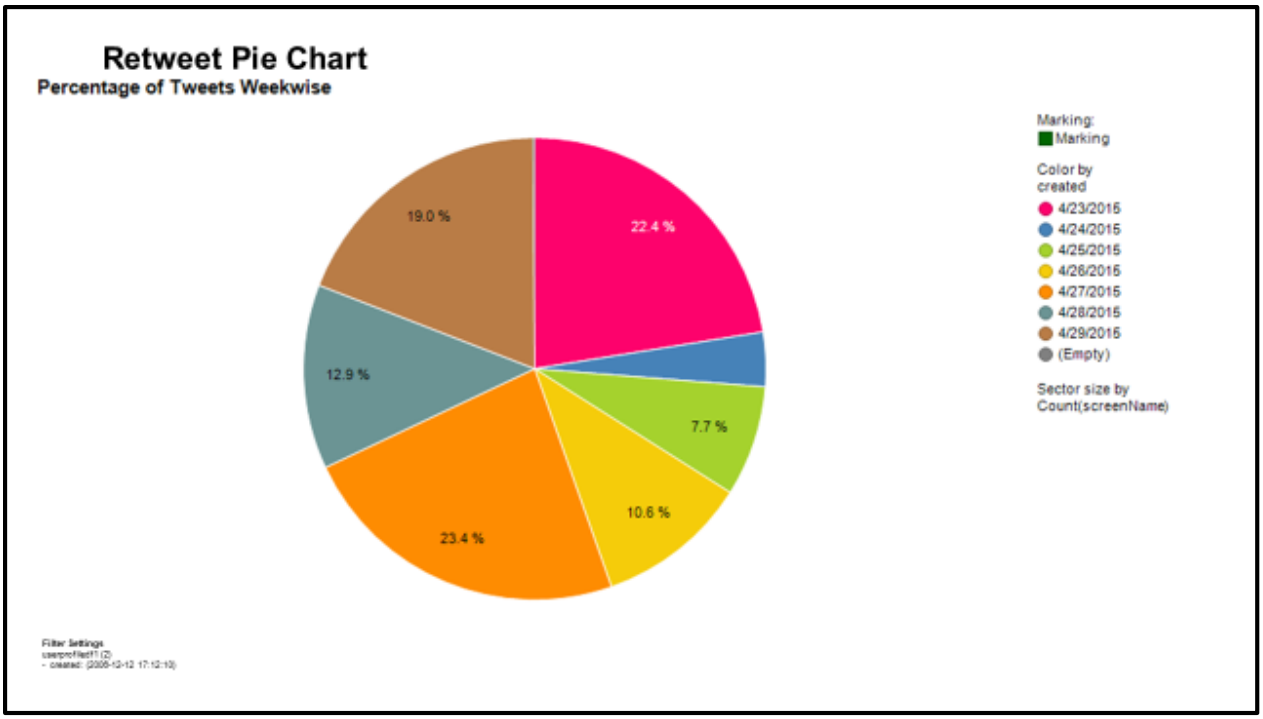
TIBCO Spotfire VISUALIZATION:

A. Retweet Analysis Weekly:



This bargraph represents retweet analysis of one week data collected from Twitter. In this graph we can observe that on 23rd, 27th and 29th of Week (23-29) we had maximum number of retweets. Because there tweets are retweeted by most of their followers and friends, their retweet count is very large compared to average of other users. Here, outliers share most important and useful information.

B. Retweet Pie Chart Percentagewise:



This pie chart can be used to make comparison on the basis of percentage shared by retweet on day to day basis. It can be observed that 23.9% of retweets are made on 27th followed by 22.4% retweets on 23rd. Also, 19.0% retweets are posted on 29th of that week.

Power of Sentiments

We performed sentimental analysis on four cellphone companies with recent 1000 tweets and immediately checked and compared their stock prices on yahoo finance. Our Analysis is as follows:-

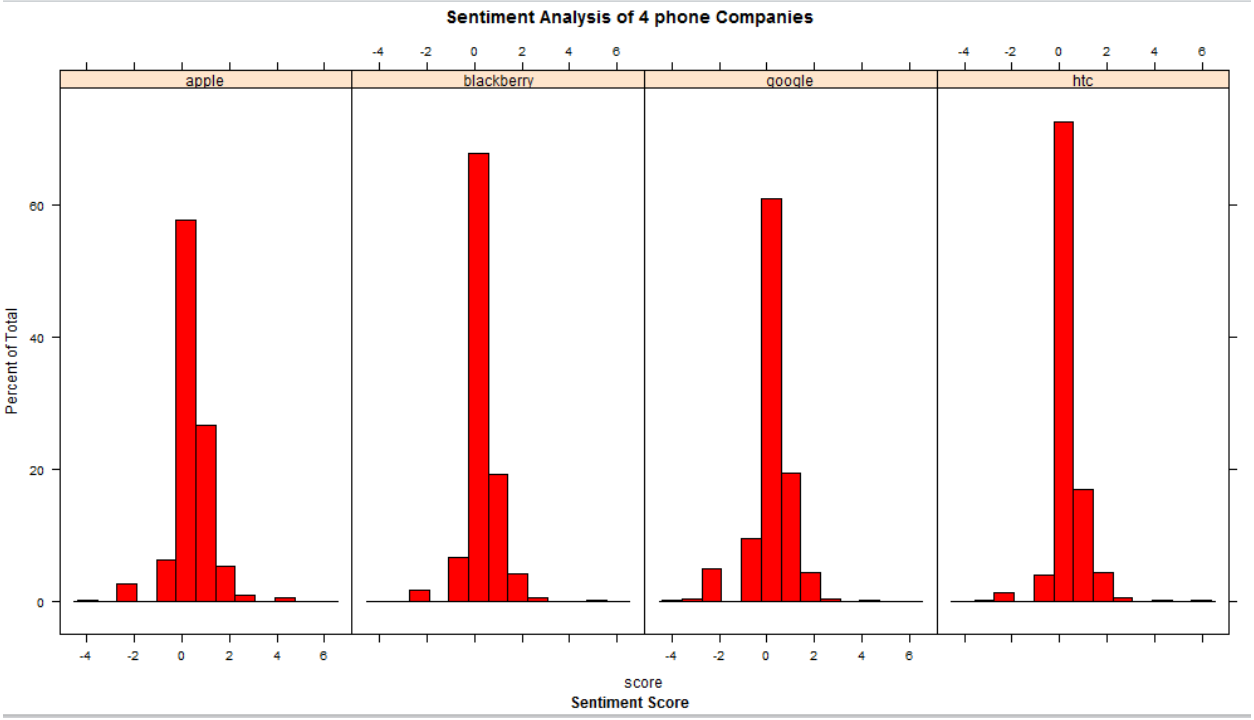
The tallest bar which is seen represents neutral sentiment and thus should be neglected. As it can be seen in the below histogram Apple has most number of tweets on the positive side. Thus we rank apple most positive.

HTC has lowest positive tweets thus we rank it fourth position.

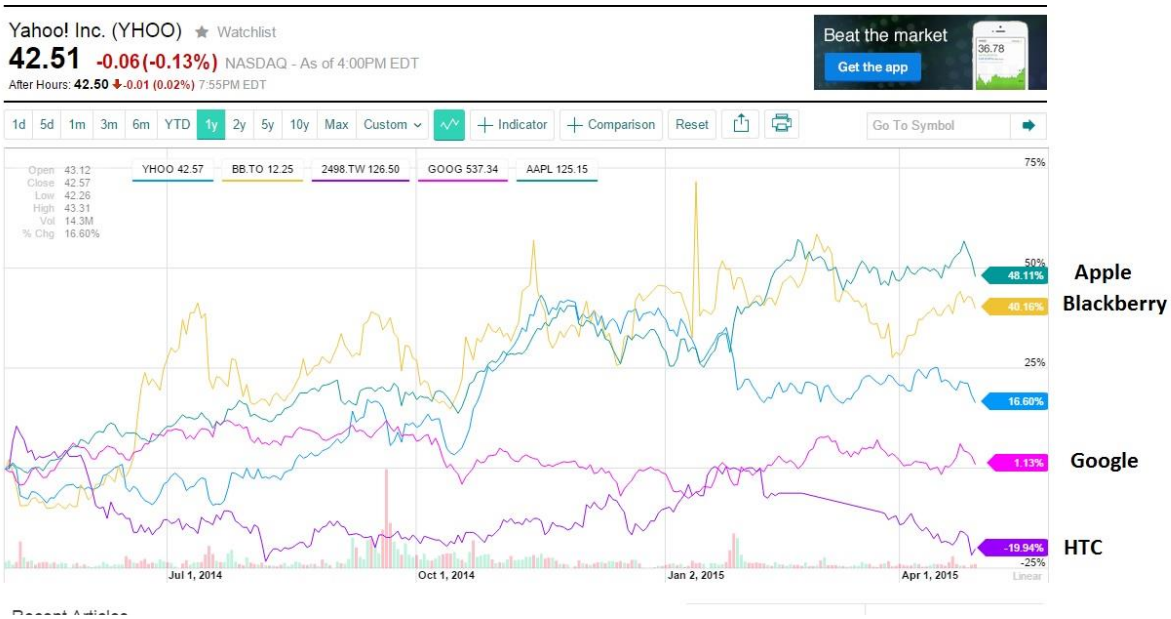
Between blackberry and google, their number of positive tweets looks almost same however google has high negative tweets. Thus we rank blackberry as second and google as third. To solve this confusion we also calculated their global score by following formula:

```
> numpos = sum(scoresgoogle$very.pos)
> numneg = sum(scoresgoogle$very.neg)
> global_score_google = round( 100 * numpos / (numpos + numneg) )
> global_score_google
[1] 62
> numpos = sum(scoresblackberry$very.pos)
> numneg = sum(scoresblackberry$very.neg)
> global_score_blackberry = round( 100 * numpos / (numpos + numneg) )
> global_score_blackberry
[1] 74
>
```

Thus we can see that blackberry score is higher than google.



Ranks which we gave four companies via sentimental analysis are reflected in their stock prices as calculated on yahoo finance immediately after scraping twitter data.



Future Scope

- This project can be extended to use different social media platforms like facebook, google+ etc.
- Emoticons can be used from sentimental analysis.

References

gallery.azureml.net

http://cran.r-project.org/web/packages/twitteR/twitteR.pdf

http://cran.r-project.org/web/views/NaturalLanguageProcessing.html