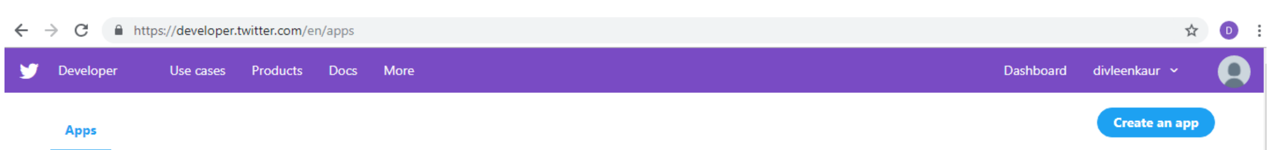
Implementation Prerequisites:

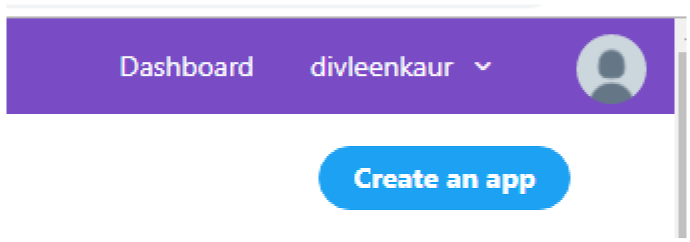
Here I have referred the following steps to create an app in Twitter and generate Consumer Key (API Key), Consumer Secret (API Secret), Access Token, and Access Token Secret. These codes have allowed to access to API in Twitter through R. API keys support a single connection on one IP. I could further use multiple application keys and/or IP addresses to besiege this limitation of Twitter applications.

Steps for accessing all tokens and access keys through twitter account:

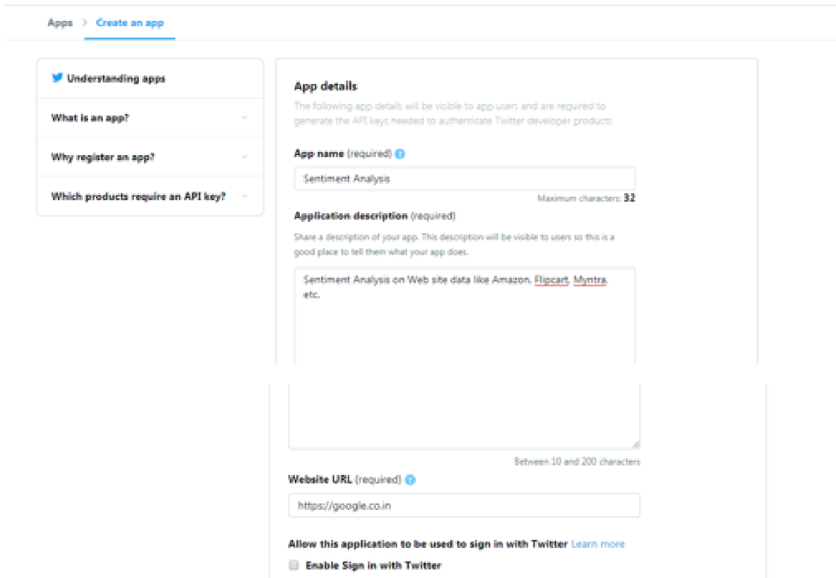
- Visit **https://apps.twitter.com/**

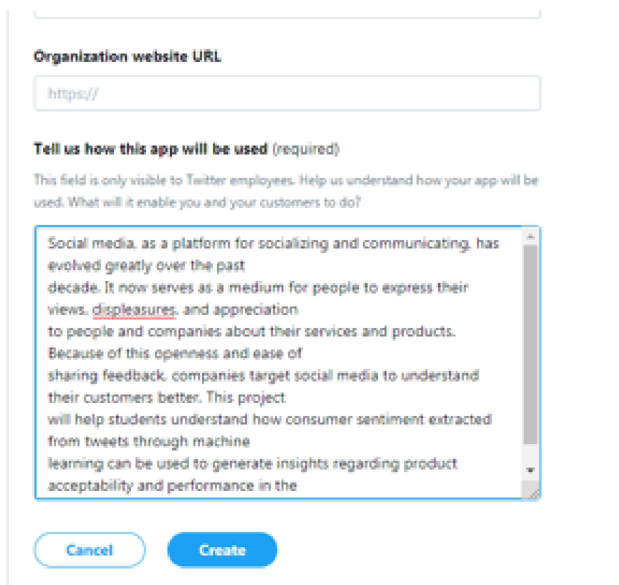
- Login using twitter account credential.



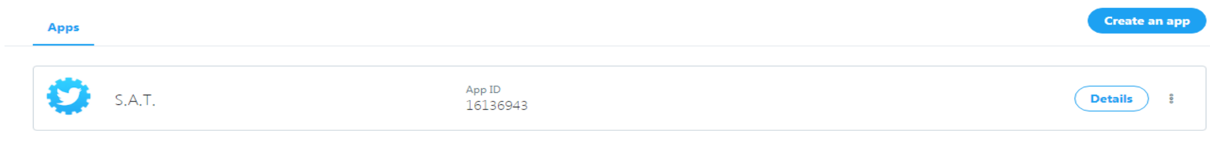


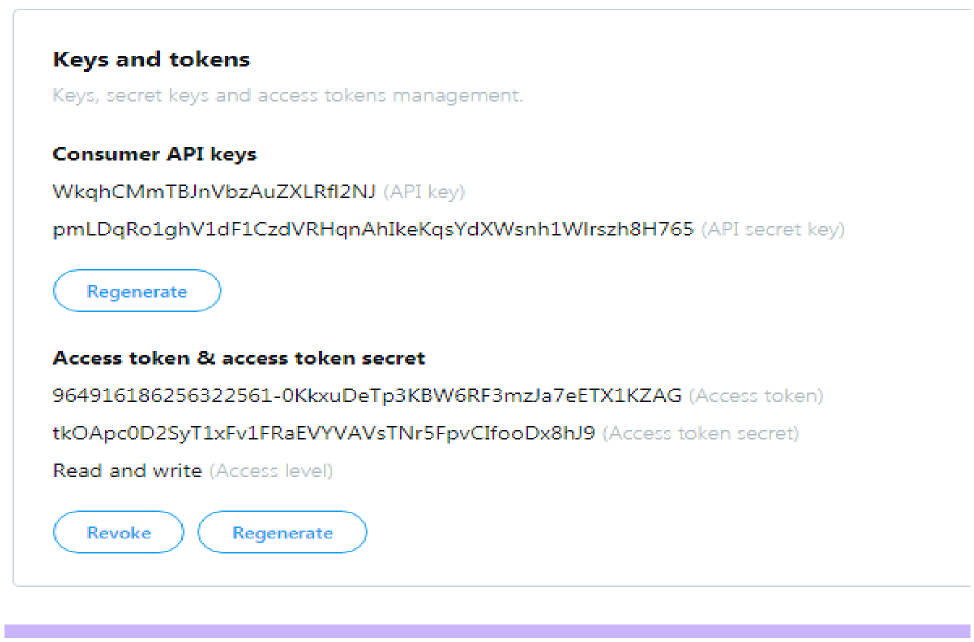
- Click on the Create an App.





- Click on Create button, then go to Keys and Tokens Tab.





- Now, copy the keys and tokens as you can see above. Paste the keys in your code for further analysis. Twitter will generate a pin after valid authentication.

Downloaded positive and negative words files in the project folder. Neutral words are created later in the R script.

final_coding_twitter analysis_divleenkaur	3/12/2019 8:38 PM	R File	9 KB
sentiment analysis Negative word list	2/28/2019 11:26 PM	Microsoft Excel C...	49 KB
sentiment analysis positive word list	2/28/2019 11:23 PM	Microsoft Excel C...	21 KB

PART I:

1. Extracting and Analyzing Tweets:

We can extract tweets containing a given # 'hashtag' or @ 'address' words or terms from a user's account or public tweets. Follow the codes below for creating the API keys:

<u>i. Setting the Authorization for Extracting Tweets:</u>

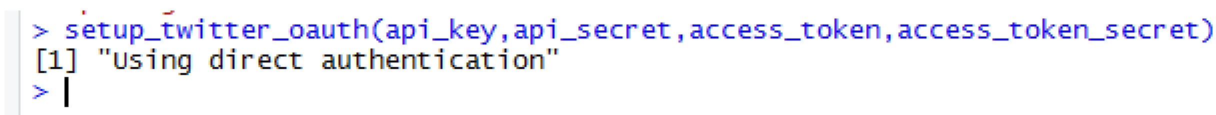
a. Run the code in R-Studio to set the authorization for extracting tweets:

```
> api_key<-"WkqhCMmTBJnVbzAuZXLRfl2NJ"
>
secret_key<-"pmLDqRo1ghV1dF1CzdVRHqnAhIkeKqsYdXWsnh1Wlrszh8
H765"
>
access_token<-"964916186256322561-0KkxuDeTp3KBW6RF3mzJa7eET
X1KZAG"
```

> access_token_secret="tkOApc0D2SyT1xFv1FRaEVYVAVsTNr5FpvCIfooDx8hJ9"

b. Set up connection between the Twitter app and R:

> setup_twitter_oauth(api_key,api_secret,access_token,access_token_secret)



ii. Required Libraries:

```
> install.packages("twitteR")
> install.packages("plyr")
> install.packages("ROAuth")
> install.packages("stringr")
> install.packages("ggplot2")
> install.packages("RTextTools")
> install.packages("e1071")
> install.packages("tm")
> install.packages("dplyr")
> install.packages("caret")
```

```
> library("twitteR")
> library("plyr")
> library("ROAuth")
> library("stringr")
> library("ggplot2")
> library("RTextTools")
> library("e1071")
> library("tm")
> library("dplyr")
> library("caret")
```

iii. Importing files:

We have to now import files containing the dictionary of positive and negative words. We already have two files, one for positive and another for negative sentiments. And this can be imported using the below code:

> positiveText<-read.csv("E:/Project Twiiter Analysis/sentiment analysis positive word list.csv",header=FALSE,stringsAsFactors=FALSE)
> str(positiveText)

```
> str(positiveText)
'data.frame':   2006 obs. of  1 variable:
 $ V1: chr  "a+" "abound" "abounds" "abundance" ...
> |
```

> positiveText<-positiveText$V1
> positiveText<-unlist(lapply(positiveText, function(x){ str_split(x,"\n")}))

> negativeText=read.csv("E:/Project Twiiter Analysis/sentiment analysis Negative word list.csv",header=FALSE,stringsAsFactors=FALSE)
> str(negativeText)

```
> str(negativeText)
'data.frame':   4783 obs. of  1 variable:
 $ V1: chr  "2-faced" "2-faces" "abnormal" "abolish" ...
> |
```

> negativeText=negativeText$V1
> negativeText<-unlist(lapply(negativeText, function(x){ str_split(x,"\n")}))

Now, add some more words into positiveText and negativeText :
> pos.words=c(positiveText,"upgrade")
> neg.words=c(negativeText,"wtf","wait","waiting","epicfail","mechanical")

iv. Extracting Tweets with hashtag:

To demonstrate sentiment analysis, we analyzed tweets relating to Amazon, Flipkart and Myntra.

> Amazon_tweets=searchTwitter('@Amazon',n=1000)
> Amazon_tweets

```
Console  Terminal ×
~/
[[998]]
[1] "MartinhaPina: Christmas Meditations: Making Christmas a Christian Holiday Again Independent..
. https://t.co/kRhBONiTgC via @amazon"

[[999]]
[1] "kranti_keshav: @amazonIN @JeffBezos @amazon I order a show and what I received is totally dif
ferent in color and design I don't wa… https://t.co/dROBAbToLJ"

[[1000]]
[1] "omlopes: @OnePerfectShot @amazon \"Where's... Spielberg?\""
```

> flipkart_tweets=searchTwitteR('@flipkart',n=1000)
> flipkart_tweets

```
Console  Terminal ×
~/
[1] "BommalingM: RT @ARYA98241902: @Flipkart से #SurfExcel\nका सूपड़ा साफ \nआज तक #SurfExcel ने दाग साफ
किये है पर आज पहली बार हिंदुओ ने सर्फ एक्सेल को ही साफ…"

[[998]]
[1] "vishalfrmbnaras: ..@Flipkart ये  सर्फ़ एक्सेल को कहाँ छुपा दिया? \nमित्रों मुहिम रंग ला रही है... \U0001f604\U000
1f604\U0001f604\U0001f44c\U0001f44d\U0001f44d\n#BoycottSurfExcel https://t.co/USNG6V2qtu"

[[999]]
[1] "AliHaid82410539: @flipkartsupport @Flipkart \nI had orderd for a bedsheet, return was done bu
t no refund has been professed yet. This… https://t.co/VLi1E8axQN"

[[1000]]
[1] "iamsatish955: @ASUSIndia @Flipkart Any exchange plans  with bank cashbacks?"

> |
```

> Myntra_tweets=searchTwitter('@Myntra',n=1000)
> Myntra_tweets

```
 @becurefit, @hotstartweets and… https://t.co/i6RLkyvfBl"

[[998]]
[1] "FinMay05: RT @reebokindia: .@myntra turns 12! #HappyBirthdayMyntra.\n5 lucky winners stand a
chance to win Reebok goodies. #ContestAlert\n\n- Visit the…"

[[999]]
[1] "AlmairahMaruhom: @NushBrand @AnushkaSharma @shoppersstop @myntra @jabong @AmazonFashionIn I l
ove you Anushka\U0001f618\U0001f618❤"

[[1000]]
[1] "bliss_berri: RT @myntra: #MyntraInsider, participate in our exclusive Fan Council contest tha
t celebrates street style, using your social media swag!\nCl…"

> |
```

v. Processing Tweets:

a. Convert the tweets into text format:

```
> Amazon_text=sapply(Amazon_tweets,function(t) t$getText())
> flipkart_text=sapply(flipkart_tweets,function(t) t$getText())
> Myntra_text=sapply(Myntra_tweets,function(t) t$getText())
```

b. Calculate the number of tweets for each e-commerce company:

```
>
no_of_tweets=c(length(Amazon_text),length(flipkart_text),length(Myntra_te
xt))
```

c. Combining the text of all these e-commerce companies:

```
> Shopping_Site<-c(Amazon_text,flipkart_text,Myntra_text)
```

vi. Sentiment Analysis application code:

The code below will show how sentiment analysis is written and executed. Before we proceed with sentiment analysis, a function needs to be defined, which will calculate the sentiment score.

parameters of function:
sentences -- vector of text to score
pos.words -- vector of words of positive sentiment
neg.words -- vector of words of negative sentiment
sent.score -- is the simple array with sapply()
-- acts as comments which is not processed by R.

```
> score.sentiment=function(sentences, pos.words,neg.words){
+
+    sent.score=sapply(sentences,
function(sentence,pos.words,neg.words){
+       # removing punctuations
+       sentences=gsub("[[:punct:]]","",sentence)
+       # removing control charaters
+       sentences=gsub("[[:cntrl:]]","",sentence)
+       # removing digits
+       sentences=gsub("\\d+","",sentence)
+       # error handling function when trying to convert lower case
+       tryTolower=function(x){
+          y=NA
+          try_error=tryCatch(tolower(x),error=function(e) e)
+          if(!inherits(try_error,"error")){
+             y=tolower(x)
+          }
+          return(y)
+       }
+       sentence=sapply(sentence,tryTolower)
+       # split sentence into words with str_split (stringr package)
+       word.list = str_split(sentences, "\\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 )
+
+    sent.scores.datafrm = data.frame(text=sentences, score=sent.score)
+    return(sent.scores.datafrm)
+ }
```

v. Start processing the tweets to calculate the sentiment score

```
> sent.scores = score.sentiment(Shopping_Site, pos.words,neg.words)
```

a. Step 1) Create a variable in the data frame.

```
> sent.scores$Shopping_Site = factor(rep(c("Amazon","flipkart","Myntra"),
no_of_tweets))
```

b. Step 2) calculate positive, negative and neutral sentiments.

```
> sent.scores$positive <- as.numeric(sent.scores$score >0)
> sent.scores$negative <- as.numeric(sent.scores$score <0)
> sent.scores$neutral <- as.numeric(sent.scores$score==0)
```

c. Step 3) Split the data frame into individual datasets for each Shopping Site.

```
> Amazon_Shopping_Site <- subset(sent.scores,
sent.scores$Shopping_Site=="Amazon")
> Flipkart_Shopping_Site <-
subset(sent.scores,sent.scores$Shopping_Site=="flipkart")
```

```
> Myntra_Shopping_Site <-
subset(sent.scores,sent.scores$Shopping_Site=="Myntra")
```

d. Step 4 - Create polarity variable for each data frame.

```
> Amazon_Shopping_Site$polarity <- ifelse(Amazon_Shopping_Site$score
>0,"positive",ifelse(Amazon_Shopping_Site$score <
0,"negative",ifelse(Amazon_Shopping_Site$score==0,"Neutral",0)))
> Flipkart_Shopping_Site$polarity <- ifelse(Flipkart_Shopping_Site$score
>0,"positive",ifelse(Flipkart_Shopping_Site$score <
0,"negative",ifelse(Flipkart_Shopping_Site$score==0,"Neutral",0)))
> Myntra_Shopping_Site$polarity <- ifelse(Myntra_Shopping_Site$score
>0,"positive",ifelse(Myntra_Shopping_Site$score <
0,"negative",ifelse(Myntra_Shopping_Site$score==0,"Neutral",0)))
```

<u>vi. Generating Graphs :</u>

After the above steps have been executed, we will go ahead and create insightful
graphs. The steps below outline the process to create graphs.

Plot 1- Polarity Plot – Customer Sentiments (Amazon)

```
> qplot(factor(polarity), data=Amazon_Shopping_Site, geom="bar",
fill=factor(polarity))+xlab("Polarity Categories") +
ylab("Frequency") + ggtitle("Customer Sentiments - Amazon Shopping
Site")
```

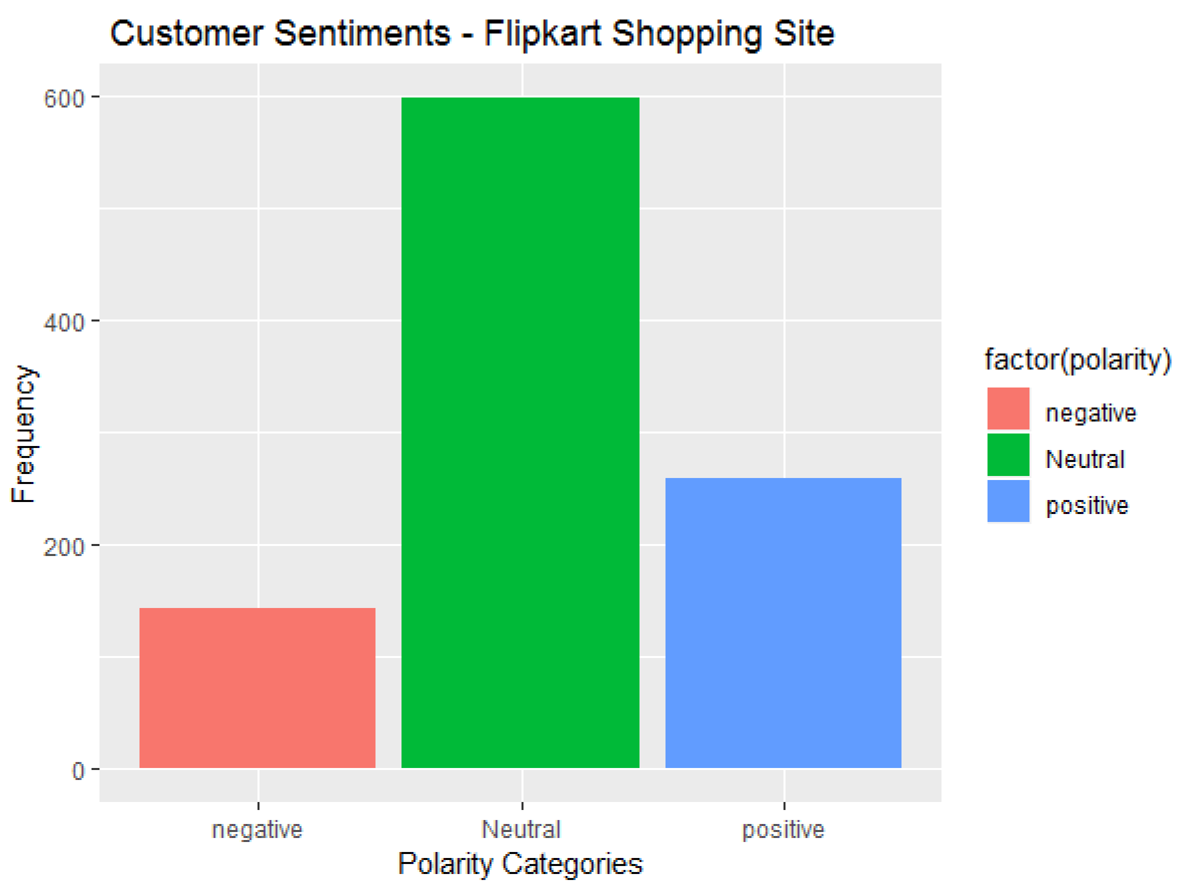
The bar graph above depicts polarity if we closely analyze the graph. It reveals that out
of 1,000 Twitter users, 50 users have commented in a negative way while 800 something users are neutral.
However, 140 users are pretty positive about Amazon.

Plot 2- Customer Sentiment Scores (Amazon Shopping Site)>
>qplot(factor(score), data=Amazon_Shopping_Site, geom="bar",
fill=factor(score))+xlab("Sentiment Score") + ylab("Frequency") +
ggtitle("Customer Sentiment Scores - Amazon Shopping Site")

The bar graph above depicts a Twitter user's sentiment score. The negative score
denoted by the (-) symbol, indicates the unhappiness of users with Amazon, and the
positive score denotes that users are happy with Amazon. Zero represents that Twitter
users are neutral.

Plot 3 - Polarity Plot – Customer Sentiments (Flipkart)

```
> qplot(factor(polarity), data=Flipkart_Shopping_Site, geom="bar",
fill=factor(polarity))+xlab("Polarity Categories") +
ylab("Frequency") + ggtitle(" Customer Sentiments - Flipkart Shopping Site
")
```



The bar graph above represents polarity. In this case, out of the 1,000 Twitter users,
140 users have commented negatively, 600 users remain neutral, and 270 users are
positive about Flipkart.

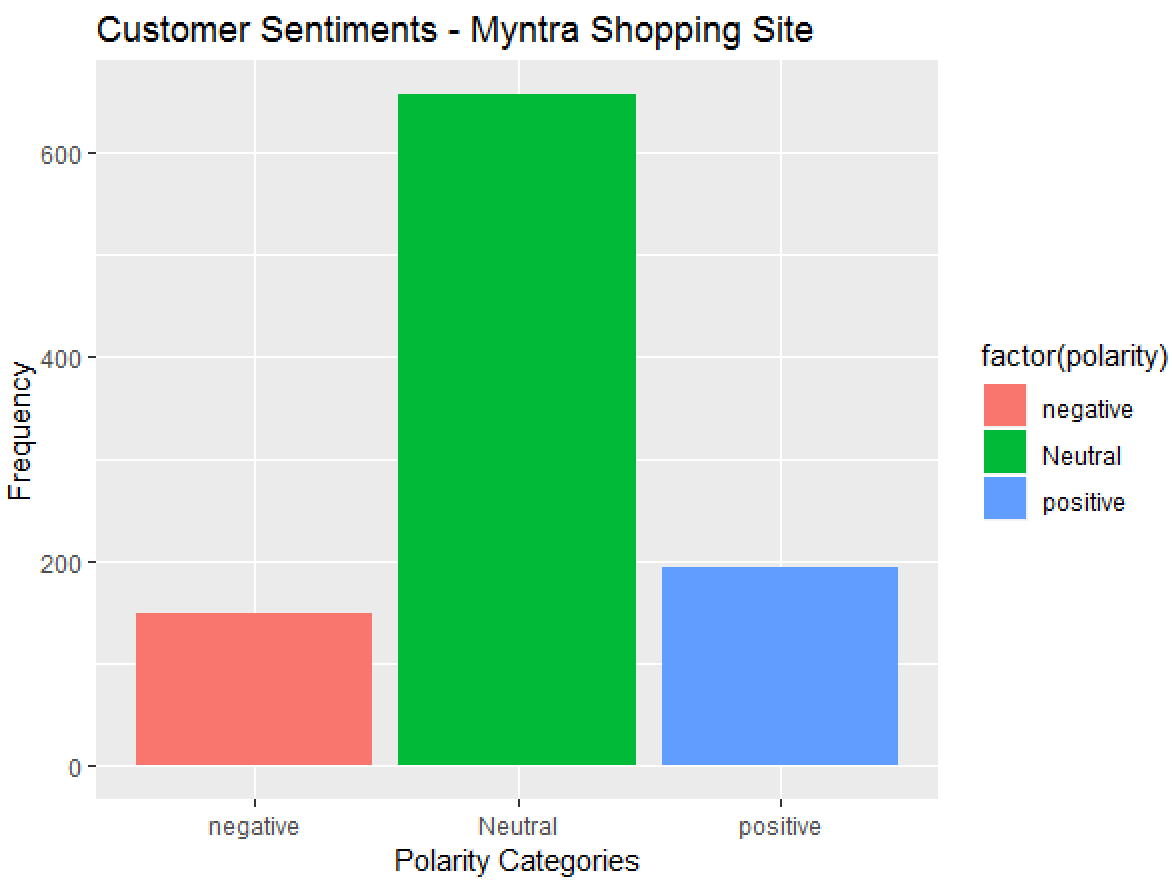
Plot 4 - Customer Sentiment Scores (Flipkart)

```
> qplot(factor(score), data=Flipkart_Shopping_Site,
geom="bar",fill=factor(score))+xlab("Sentiment Score") + ylab("Frequency")
+ ggtitle("Customer Sentiment Scores - Flipkart Shopping Site")
```

The bar graph above depicts a Twitter user's sentiment score. The negative score,
denoted by the (-) symbol, indicates unhappiness with the Flipkart Shopping Site and
the positive score denotes that users are quite happy. The zero here represents that
users are neutral.

Plot 5 - Polarity Plot – Customer Sentiments (Myntra)

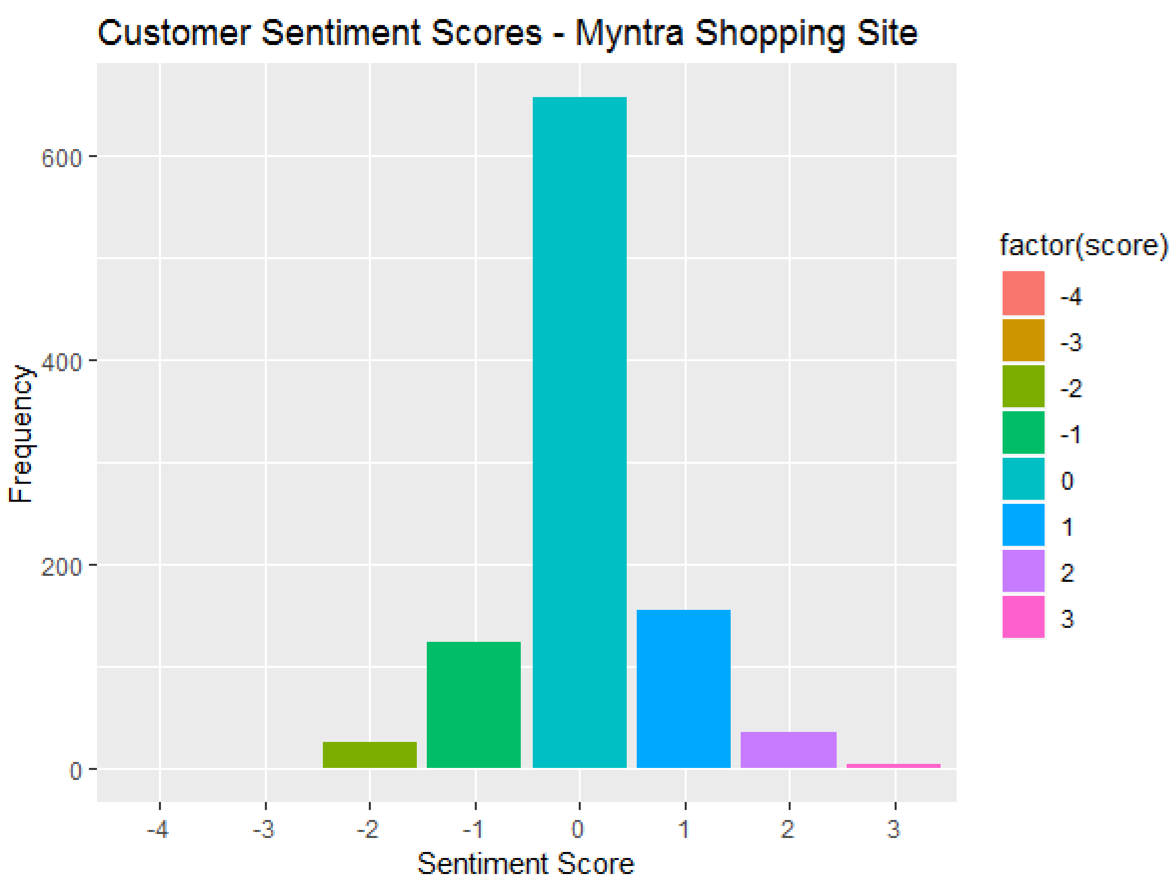
> qplot(factor(polarity), data=Myntra_Shopping_Site, geom="bar", fill=factor(polarity))+xlab("Polarity Categories") + ylab("Frequency") + ggtitle("Customer Sentiments - Myntra Shopping Site")



The bar graph above represents polarity. In this case, out of the 1,000 Twitter users,150
users have commented negatively, 670 users are neutral, and the remaining 190 users
remains positive about the e-commerce company.

Plot 6 - Customer Sentiment Scores (Myntra)

```
> qplot(factor(score), data=Myntra_Shopping_Site, geom="bar",
fill=factor(score))+xlab("Sentiment Score") + ylab("Frequency") +
ggtitle("Customer Sentiment Scores - Myntra Shopping Site ")
```



The bar graph above depicts the Twitter user's sentiment score. The negative score
denoted by the (-) symbol indicates the unhappiness of users with the e-commerce
company while the positive score denotes that users are quite happy. Zero represents

that users are neutral about their opinion.

vii. Summarizing Scores:

The code below will help us to summarize the overall positive, negative, and
neutral scores:

> datafrm = ddply(sent.scores, c("Shopping_Site"), summarise,pos_count=sum(positive), neg_count=sum(negative),neu_count=sum(neutral))
> datafrm

```
> datafrm
  Shopping_Site pos_count neg_count neu_count
1        Amazon       133        52       815
2      flipkart       259       143       598
3        Myntra       194       150       656
>
```

To put it in another way, we will create the total count by adding the positive,
negative, and neutral sums.

> datafrm$total_count = datafrm$pos_count +datafrm$neg_count + datafrm$neu_count
> datafrm$total_count

```
> datafrm$total_count
[1] 1000 1000 1000
>
```

Additionally, we will calculate the positive, negative, and neutral percentages
using the code below:

> datafrm$pos_percent_score = round(
100*datafrm$pos_count/datafrm$total_count)
> datafrm$pos_percent_score

```
> datafrm$pos_percent_score
[1] 13 26 19
>
```

> datafrm$neg_percent_score = round(
100*datafrm$neg_count/datafrm$total_count)
> datafrm$neg_percent_score

```
> datafrm$neg_percent_score
[1]  5 14 15
>
```

> datafrm$neu_percent_score = round(
100*datafrm$neu_count/datafrm$total_count)
> datafrm$neu_percent_score

```
> datafrm$neu_percent_score
[1] 82 60 66
>
```

viii. Comparison Charts :

> attach(datafrm)

```
> attach(datafrm)
The following object is masked _by_ .GlobalEnv:

    Shopping_Site
```

Comparison 1 - Positive Comparative Analysis

Here is the code to create a positive comparison pie chart for these three ecommerce companies:

> pie.chart.abc
<-paste(datafrm$Shopping_Site,datafrm$pos_percent_score)
> pie.chart.abc

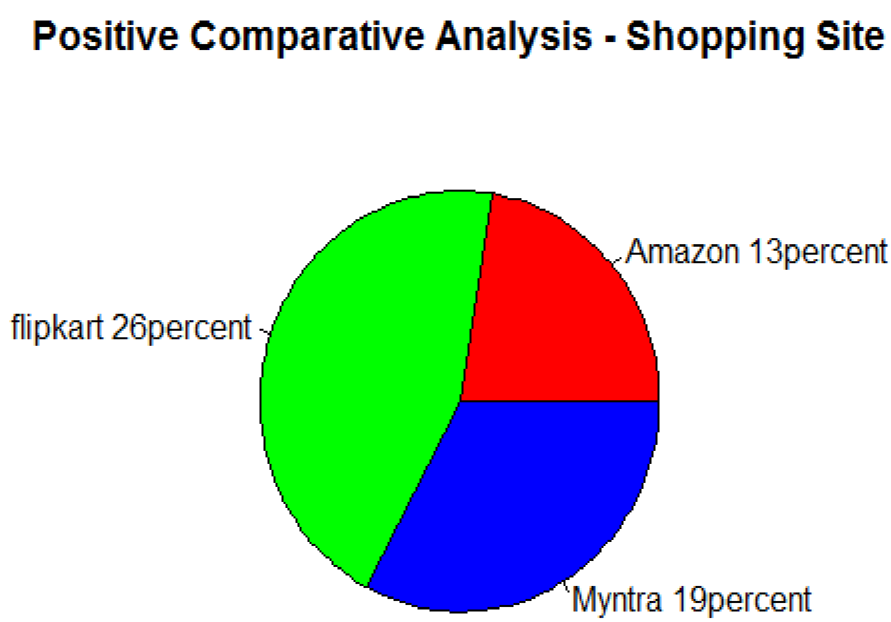
```
> pie.chart.abc
[1] "Amazon 13"   "flipkart 26" "Myntra 19"
> 
```

> pie.chart.abc <- paste(pie.chart.abc,"percent",sep="")
> pie.chart.abc

```
> pie.chart.abc
[1] "Amazon 13percent"   "flipkart 26percent" "Myntra 19percent"
> 
```

> pie(pos_percent_score, labels = pie.chart.abc, col = rainbow(length(pie.chart.abc)), main = "Positive Comparative Analysis - Shopping Site")

The pie chart below represents the positive percentage score of these companies:



Comparison 2 - Negative Comparative Analysis

Here is the code to create a negative comparison pie chart for these three ecommerce companies:

```
>
pie.chart.abc<-paste(datafrm$Shopping_Site,datafrm$neg_percent_score)
> pie.chart.abc
```

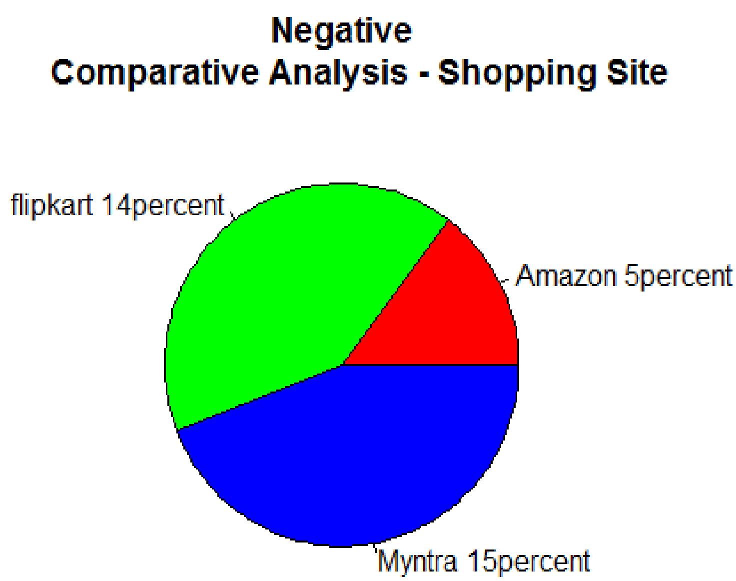
```
> pie.chart.abc
[1] "Amazon 5"    "flipkart 14" "Myntra 15"
> |
```

```
> pie.chart.abc <- paste(pie.chart.abc,"percent",sep="")
> pie.chart.abc
```

```
> pie.chart.abc
[1] "Amazon 5percent"    "flipkart 14percent" "Myntra 15percent"
> |
```

```
> pie(neg_percent_score, labels = pie.chart.abc, col =
  rainbow(length(pie.chart.abc)), main = " Negative
  Comparative Analysis - Shopping Site")
```

The pie chart below represents the negative percentage score of these three
companies:



Comparison 3 - Neutral Comparative Analysis

Here is the code to create a neutral comparison pie chart:

> pie.chart.abc
<-paste(datafrm$Shopping_Site,datafrm$neu_percent_score)
> pie.chart.abc

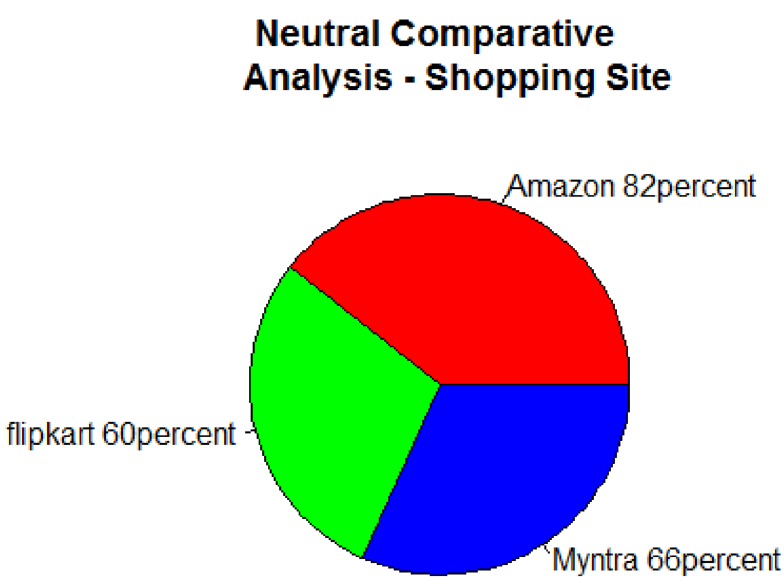
```
> pie.chart.abc
[1] "Amazon 82"   "flipkart 60" "Myntra 66"
>
```

> pie.chart.abc <- paste(pie.chart.abc,"percent",sep="")
> pie.chart.abc

```
> pie.chart.abc
[1] "Amazon 82percent"   "flipkart 60percent" "Myntra 66percent"
>
```

> pie(neu_percent_score, labels = pie.chart.abc, col =
rainbow(length(pie.chart.abc)), main = "Neutral Comparative
Analysis - Shopping Site")

The pie chart below represents the neutral percentage score of these three companies:



Part II:

i. Data Preprocessing:

a. Writing and Reading the Data as ‘Amazon_Shopping_Site’

> write.table(Amazon_Shopping_Site,"E:/Project Twiiter Analysis/Amazon_Shopping_Site.csv", append=T, row.names=F, col.names=T,sep=",")

```
> write.table(Amazon_Shopping_Site,"E:/Project Twiiter Analysis/Amazon_Shopping_Site.csv", append=T
, row.names=F, col.names=T,sep=",")
Warning message:
In write.table(Amazon_Shopping_Site, "E:/Project Twiiter Analysis/Amazon_Shopping_Site.csv",  :
  appending column names to file
```

> Amazon_Shopping_Site_csv <-read.csv("E:/Project Twiiter Analysis/Amazon_Shopping_Site.csv", header = TRUE, encoding = "UCS-2LE")
> View(Amazon_Shopping_Site_csv)

	text	score	Shopping_Site	positive	negative
1	RT @OnePerfectShot: MAD MAX: FURY ROAD (2015) Cine...	0	Amazon	0	0
2	RT @DebsNovels: She finds out that first loves are not al...	1	Amazon	1	0
3	RT @OnePerfectShot: SPIDER-MAN 2 (2004) Cinematogra...	0	Amazon	0	0
4	RT @Angel_City_Buzz: Live Art Decor - #LosAngeles Skyli...	0	Amazon	0	0
5	RT @GUcomicshop: We could not be happier for all the ...	2	Amazon	1	0
6	RT @gracieversusdad: Century Star Women's Vintage Wi...	0	Amazon	0	0
7	RT @AuthorPearlTate: #NewRelease CALLIM'S CHALLENG...	0	Amazon	0	0
8	RT @darkskyfilms: \Crackling with tension	@level16film is a chilling and empowering film.\ - @Cine...	0	Amazon	0
9	Neutral				
10	RT @VoteSawant: "Comcast has pumped money into ma...	0	Amazon	0	0
11	Why is @amazon's #customercentric growth sustainable...	0	Amazon	0	0
12	I suspect it's a pricing error but we'll see: Rick and Morty...	-2	Amazon	0	1
13	@FLAutismMom @RepAdamSchiff @amazon https://t.co...	0	Amazon	0	0
14	My local grocery store doesn't carry toilet paper without...	0	Amazon	0	0
15	Book recommendation: Dynamic Lecturing: Research-Ba...	0	Amazon	0	0
16	@amazon your customer is appalling atrocious and quit...	-3	Amazon	0	1
17	RT @Jason: Don't worry America, @ewarren will make s...	-1	Amazon	0	1
18	RT @DreadfulPaige: Well, this is me! Please be kind. The ...	0	Amazon	0	0

Showing 1 to 19 of 2,882 entries

> dataf$class <- as.factor(dataf$class)

Now, read the new file, "Amazon_Shopping_Site_classif2.csv".

> dataf<- read.csv("E:/Project Twiiter Analysis/Amazon_Shopping_Site_classif2.csv", stringsAsFactors = FALSE)

> head(dataf)

```
> head(dataf)

                                                                    text
1 RT @OnePerfectShot: MAD MAX: FURY ROAD (2015) \n\nCinematography by John Seale\nDirected
by George Miller\nBuy it via @amazon: https://t.co/U8f…
2     RT @DebsNovels: She finds out that first loves are not always the answer to a woman's
 prayer. #romance #newLove #drama  https://t.co/UFaSyQ…
3 RT @OnePerfectShot: SPIDER-MAN 2 (2004)\n\nCinematography by Bill Pope\nDirected by Sam R
aimi\nBuy or rent via @amazon: https://t.co/4zD2Z8qZ2R…
4     RT @Angel_City_Buzz: Live Art Decor - #LosAngeles Skyline Picture Canvas Prints Moder
n California Summer Moon Night City Scene Wall Art for…
5  RT @GUcomicshop: We could not be happier for all the success that @zackkaps @andreamutti
9 &amp; @VPopov_Artworks are having with the continued…
6     RT @gracieversusdad: Century Star Women's Vintage Winter Soft Wool Warm Comfort Cozy
Crew Socks 5 Pack  https://t.co/Kxoo8NLh3a via @amazon…
     class
1  Neutral
2 positive
3  Neutral
4  Neutral
5 positive
6  Neutral
> |
```

b. Randomize the dataset and convert the 'class' variable from character to factor.

```
> set.seed(1)
> dataf <- dataf[sample(nrow(dataf)), ]
> head(dataf)
```

```
> head(dataf)

                                                                          text
529                       A dona @amazon começou a semana do consumidor com umas promoções
pica. look this #senmanadoconsumidor https://t.co/CHQcfXxjyi
741                  Knock knock! Proactive Recovery is here <U+0001F44F>\n\nFind RESQWATER
 on @amazon: https://t.co/uxrbobe93r https://t.co/bm3CwmwcuW
1140         The Cursed Jackson Green (e-book) is now available exclusively on #amazonkindle
 for #FREE through #KindleUnlimited… https://t.co/exh4BGVNy3
1807                                                                                    @A
aaamberr2255 @AmazonKDP @amazon @JeffBezos blocked by who???
401   RT @DrPChouinard: I didn’t forget how @amazon was hosting &amp; seemingly endorsing Mi
racle Mineral Solution sellers as recently as 2018.\n\nThes…
1786                Take a new #novel on #springbreak2019, Reflections in the Mist https://t
.co/cyUsx5c2qK via @amazon #sailing… https://t.co/v3CYcopn9I
        class
529   Neutral
741   Neutral
1140 positive
1807  Neutral
401  positive
1786  Neutral
> |
```

> str(dataf)

```
> str(dataf)
'data.frame':   1992 obs. of  2 variables:
 $ text : chr  "A dona @amazon começou a semana do consumidor com umas promoções pica. look
 this #senmanadoconsumidor https://t.co/CHQcfXxjyi" "Knock knock! Proactive Recovery is her
e <U+0001F44F>\n\nFind RESQWATER on @amazon: https://t.co/uxrbobe93r http"| __truncated__ "
The Cursed Jackson Green (e-book) is now available exclusively on #amazonkindle for #FREE t
hrough #KindleUnlimi"| __truncated__ "@Aaaamberr2255 @AmazonKDP @amazon @JeffBezos blocked
by who???" ...
 $ class: chr  "Neutral" "Neutral" "positive" "Neutral" ...
> |
```

> dataf$class=as.factor(dataf$class)
> str(dataf)

```
> str(dataf)
'data.frame':   1992 obs. of  2 variables:
 $ text : chr  "A dona @amazon começou a semana do consumidor com umas promoções pica. look
 this #senmanadoconsumidor https://t.co/CHQcfXxjyi" "Knock knock! Proactive Recovery is her
e <U+0001F44F>\n\nFind RESQWATER on @amazon: https://t.co/uxrbobe93r http"| __truncated__ "
The Cursed Jackson Green (e-book) is now available exclusively on #amazonkindle for #FREE t
hrough #KindleUnlimi"| __truncated__ "@Aaaamberr2255 @AmazonKDP @amazon @JeffBezos blocked
by who???" ...
 $ class: Factor w/ 3 levels "negative","Neutral",..: 2 2 3 2 3 2 3 2 2 2 ...
> |
```

c. Bag of Words Tokenization

In this approach, we represent each word in a document as a token (or feature) and each document as a vector of features. In addition, for simplicity, we disregard word order and focus only on the number of occurrences of each word, which means that we represent each document as a multi-set ‘bag’ of words.

We first prepare a corpus of all the documents in the dataframe.

> corpuss <- VCorpus(VectorSource(dataf$text))
> inspect(corpuss[1:3])

```
> inspect(corpuss[1:3])
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 3

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 125

[[2]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 128

[[3]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 139

> |
```

c. Data Cleanup

We clean up the corpus by eliminating numbers, punctuation, and white space and by converting to lowercase. In addition, we discard common stop words, such as “his”, “our”, “hadn’t”, couldn’t“, etc.

```
> corpus.clean <- corpuss %>% tm_map(content_transformer(tolower))
%>% tm_map(removePunctuation) %>% tm_map(removeNumbers) %>%
tm_map(removeWords, stopwords(kind="en")) %>%
tm_map(stripWhitespace)      #dplyr package
```

d. Matrix representation of Bag of Words: The Document Term Matrix (DTM)

We represent the bag of words tokens with a document term matrix (DTM). The rows of the DTM correspond to the documents in the collection, the columns correspond to the terms, and its elements are the term frequencies.

> dtm <- DocumentTermMatrix(corpus.clean)
> dtm

```
> dtm
<<DocumentTermMatrix (documents: 1992, terms: 3800)>>
Non-/sparse entries: 23074/7546526
Sparsity           : 100%
Maximal term length: 62
Weighting          : term frequency (tf)
> |
```

> inspect(dtm[40:50, 10:15])

```
> inspect(dtm[40:50, 10:15])
<<DocumentTermMatrix (documents: 11, terms: 6)>>
Non-/sparse entries: 0/66
Sparsity           : 100%
Maximal term length: 11
Weighting          : term frequency (tf)
Sample             :
    Terms
Docs “defiant “democracy” “everyone “grimmest “huge “jumping
  40        0           0         0         0     0        0
  41        0           0         0         0     0        0
  42        0           0         0         0     0        0
  43        0           0         0         0     0        0
  44        0           0         0         0     0        0
  45        0           0         0         0     0        0
  46        0           0         0         0     0        0
  47        0           0         0         0     0        0
  48        0           0         0         0     0        0
  49        0           0         0         0     0        0
  50        0           0         0         0     0        0
> |
```

ii. Partitioning the Data

We create 70:30 partitions of the dataframe, corpus, and DTM for training and
testing purposes.

```
> dataf.train <- dataf[1:1600,]
> dataf.test <- dataf[1601:1992,]
> dtm.train<-dtm[1:1600,]
> dtm.test<-dtm[1601:1992,]
> corpus.clean.train <- corpus.clean[1:1600]
> corpus.clean.test <- corpus.clean[1601:1992]
> dim(dtm.train)
```

```
> dim(dtm.train)
[1] 1600 3800
>
```

The DTM contains many features, but not all of them are useful for classification. We reduce the number of features by ignoring the words that appear in less than five reviews. To do this, we use the 'findFreqTerms' function to indentify frequent words, and then we restrict the DTM to use only the frequent words using the 'dictionary' option.

```
> fivefreq <- findFreqTerms(dtm.train, 5)
> fivefreq
```

```
> fivefreq
  [1] "-amp"              "'re"                "'ve"
  [4] "aaaamberr"         "absolutely"         "add"
  [7] "advertising"       "agnes"              "airguns"
 [10] "albertobagnai"     "alcott"             "alert"
 [13] "allen"             "already"            "also"
 [16] "always"            "amazon"             "amazon…"
 [19] "amazonhelp"        "amazonin"           "amazonkdp"
 [22] "amazons"           "amerheroesradio"    "america"
 [25] "american"          "amir"               "amor"
 [28] "amp"               "amp…"               "andrew"
 [31] "angelaleechizum"   "angelcitybuzz"      "answer"
 [34] "anything"          "aoc"                "apocalypse"
 [37] "apple"             "art"                "asking"
 [40] "assumono"          "augustine"          "aunt"
 [43] "authentics"        "author"             "authorpearltate"
 [46] "available"         "away"               "awesome"
 [49] "back"              "backyard"           "bandersdavidson"
 [52] "bands"             "banks"              "basics"
```

```
[757] "world"             "worry"              "wrangler"
[760] "wright"            "written"            "yafex"
[763] "yankeenets"        "yankees"            "yardsales"
[766] "year"              "yearman"            "years"
[769] "yesnetwork"        "yield"              "zacksnyder"
[772] "zakkwyldebls"
```

> length(fivefreq)

```
> length(fivefreq)
[1] 772
>
```

> dtm.train.nb <- DocumentTermMatrix(corpus.clean.train, control=list(dictionary = fivefreq))
> dim(dtm.train.nb)

```
> dim(dtm.train.nb)
[1] 1600  772
>
```

> dtm.test.nb <- DocumentTermMatrix(corpus.clean.test, control=list(dictionary = fivefreq))
> dim(dtm.test.nb)

```
> dim(dtm.test.nb)
[1] 392 772
>
```

The Naive Bayes text classification algorithm is essentially an application of Bayes theorem (with a strong independence assumption) to documents and classes. In this method, the term frequencies are replaced by Boolean presence/absence features. The logic behind this is that for sentiment classification, word occurrence matters more than word frequency.

a. Function to convert the word frequencies to yes (presence) and no (absence)labels:

```
> convert_count <- function(x) {
+     y <- ifelse(x > 0, 1,0)
+     y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
+     y
+ }
```

b. Applying the convert_count function to get the final training and testing DTMs:

```
> train_NB <- apply(dtm.train.nb, 2, convert_count)
> test_NB <- apply(dtm.test.nb, 2, convert_count)
```

iii. Training the Naive Bayes Model

To train the model, we use the Naive Bayes function from the ‘e1071’ package. Since
Naive Bayes evaluates the products of probabilities, we need some way of assigning nonzero probabilities to words that do not appear in the sample.

Train the classifier.
> system.time(classifier <- naiveBayes(train_NB, dataf.train$class,laplace = 1)

```
> system.time( classifier <- naiveBayes(train_NB, dataf.train$class,laplace = 1))
   user  system elapsed 
   0.29    0.00    0.29 
> 
```

iv. Testing the Predictions

Use the Naïve Bayes classifier we built to make predictions on the test set:

> system.time(pred <- predict(classifier, newdata=test_NB))

```
> system.time( pred <- predict(classifier, newdata=test_NB))
   user  system elapsed 
   6.94    0.08    7.13 
> 
```

Create a truth table by tabulating the predicted class labels with the actual class labels:

> table("Predictions"= pred, "Actual" = dataf.test$class)

```
> table("Predictions"= pred, "Actual" = dataf.test$class )
           Actual
Predictions negative Neutral positive
   negative        4       7        0
   Neutral         4     302       10
   positive        3      19       43
> 
```

Create a truth table by tabulating the predicted class labels with the actual class labels:

> conf_matrix <- confusionMatrix(pred, dataf.test$class)

#this function is in “caret” package

> conf_matrix

```
> conf_matrix
Confusion Matrix and Statistics

          Reference
Prediction negative Neutral positive
  negative        4       7        0
  Neutral         4     302       10
  positive        3      19       43

Overall Statistics

               Accuracy : 0.8903
                 95% CI : (0.8551, 0.9195)
    No Information Rate : 0.8367
    P-Value [Acc > NIR] : 0.001714

                  Kappa : 0.6371
 Mcnemar's Test P-Value : 0.085376

Statistics by Class:

                     Class: negative Class: Neutral Class: positive
Sensitivity                  0.36364         0.9207          0.8113
Specificity                  0.98163         0.7812          0.9351
Pos Pred Value               0.36364         0.9557          0.6615
Neg Pred Value               0.98163         0.6579          0.9694
Prevalence                   0.02806         0.8367          0.1352
Detection Rate               0.01020         0.7704          0.1097
Detection Prevalence         0.02806         0.8061          0.1658
Balanced Accuracy            0.67263         0.8510          0.8732
> 
```

<u>CONCLUSION:</u>

- In the first part, we analyzed tweets for competing e-commerce brands and characterized the sentiment score for each tweet as positive,
  negative, and neutral. With this polarity data, we have created a variety of charts to enable a comparative study of brand value, in terms of the customer's response on Twitter.
  Our analysis shows that **flipkart is the most-liked brand out of the Three Brands** (Amazon, Myntra, and flipkart) we analyzed for this project. Customers tweets for flipkart were mostly of positive sentiment, then Myntra and the comes Amazon.
- In the <u>second part</u>, we trained the Naïve Bayes algorithm, using the tweet and polarity data from part one of the sentiment analysis for the prediction of new tweets.

Our results show an accuracy of 89.30%; higher accuracy can be achieved with more training on a larger dataset.
We also calculated sensitivity, specificity, and the P-Value of test data through Confusion matrix for better insights.