SENTIMENT ANALYSIS USING TWITTER DATA ON

SHOPPING SITES

DEVELOPED BY:

**RAMYA .M**

CENTRE NAME:

**CHENNAI T. NAGAR**

REGISTRATION ID:

**R191176900064**

STUDENT ID:

**S191176900061**

BATCH CODE:

**S190392**

CERTIFICATE OF ORIGINALITY

This is to certify that the project work, ”**Sentiment Analysis using Twitter data on Shopping sites**” submitted by me is an outcome of my independent and original work. I have duly acknowledged all the sources from which the ideas and extracts have been taken. The project is free from any plagiarism and has not been submitted elsewhere for publication.

Name: Ramya.M

Institution: NIIT

Date: 13/03/2019

ACKNOWLEDGEMENT

I would like to express my gratitude to my faculty, **Ms Lopamudra Bera** for her guidance and support throughout the project, my tech mentor, **Ms Preethi** for her assistance throughout the term at my centre.

I would like to thank my family for their moral support.

Lastly but mostly I would like to show my sincere gratitude to my centre, **NIIT T.NAGAR CHENNAI,** for letting me have this opportunity.

TABLE OF CONTENTS

|  |  |
| --- | --- |
| TOPIC | PAGE |
| Backgroung of project | 5 |
| Problem definition | 7 |
| Implementation prequsits | 8 |
| Solution Statement | 11 |
| Methodology – Part 1 | 12 |
| Methodology – Part 2 | 30 |
| Findings and Conclusions | 40 |

BACKGROUND OF PROJECT

Social media, as a platform for socializing and communicating, has greatly evolved over the past decade. It now serves as a medium for people to express their views and displeasures and appreciation to people and companies about their services and their products. Because of this openness and ease of sharing feedback, companies target social media to understand their customers better. This project will help students understand how consumer sentiment extracted from tweets through machine learning can be used to generate insights regarding product accessibility and performance in the market.

Twitter is a microblogging platform where users can post their messages (140 UTF – 8 characters) called Twitter and read other’s messages or tweets on a single page on reverse chronology or timeline. By default, all tweets are publicly visible unless users tweak their privacy settings. Because of constraints on the length of tweets, twitter follows a few conventions for users to add a structure to their tweets. For example, the @ symbol is used to refer to users in tweets and # symbol is used to categorize tweets.

* **What is sentiment analysis?**

Sentiment analysis is contextual mining of text to systematically identifying and categorizing opinions expressed towards big companies through social media in a piece of text in order to determine whether the writer’s attitude towards a particular topic, product, etc. is positive, negative or neutral and to what degree.

* **Why do we need sentiment analysis?**

Companies take feedback from their customers to understand how they feel about their products and (or) services. To get a better understanding of how the customers feel, sentiment analysis helps to uncover those feelings. This will help with the company’s growth, product improvements, train sales, customer care agents and building new marketing campaigns.

* **Twitter analysis using R**

Twitter has not only become a social media platform where people talk to each other, but it has become very vast and serves many more purposes like where people Express their interests, share their views and displeasures and comment on the companies services. To do that, using R we extract tweets from twitter application. Cleaning the tweets and segregating them according to positive and negative tweets.

PROBLEM DEFINITION

The aim of this project is to create insightful graphs that indicate consumer sentiment towards e-commerce websites, such as Amazon and Myntra. In addition, training will be undertaken for the Naïve Bayes classifiers to classify tweets according to their overall sentiment and check the accuracy of results.

Part 1:

Around 1,000 tweets will be extracted of competing brands and data will be prepared using various cleaning steps. The polarity and sentiment of the score for each tweet are calculated and used for creating individual and comparative plots. Packages requied for data cleaning methods are installed for data preparation.

Part 2:

In this part, Naïve Bayes (Supervised Algorithm) will be used for prediction of tweet sentiment as positive, negative, or neutral (in terms of emotion such as happy, sad, excited). Confusion matrix is also created to calculate the accuracy of their algorithm.

Implementation prerequisites

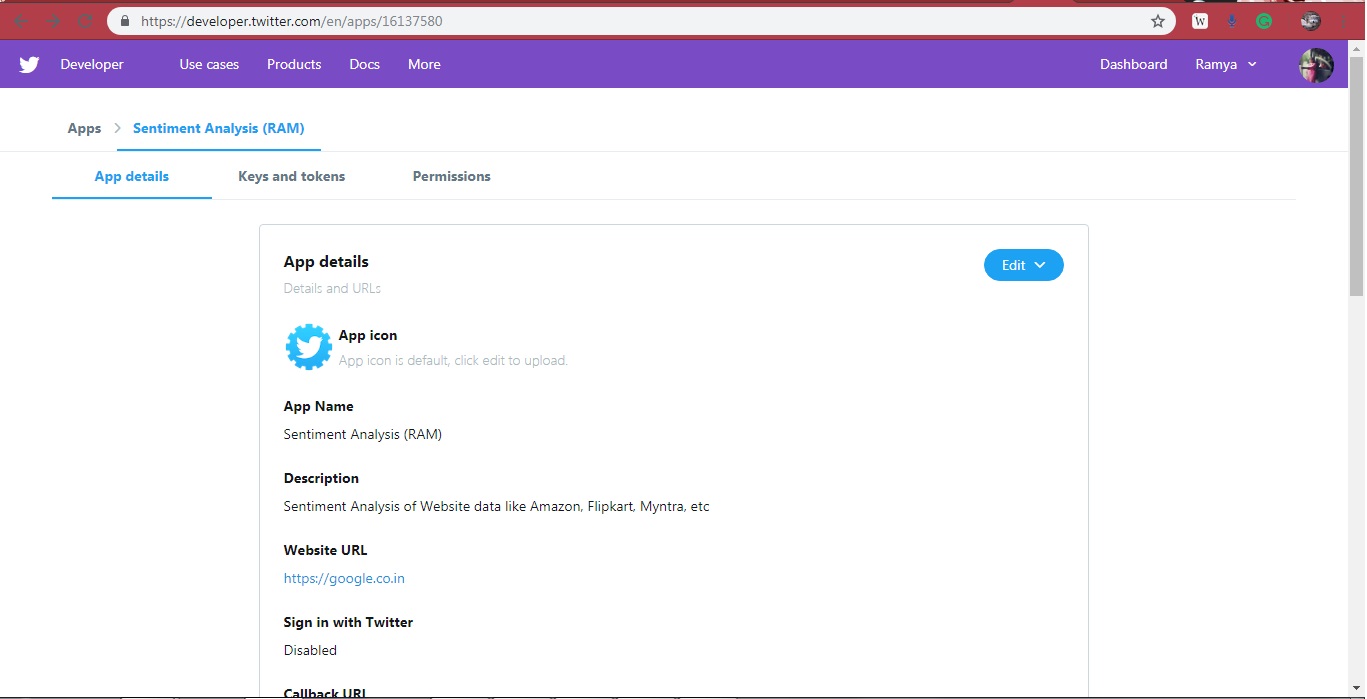
1. Accessing API in Twitter

Steps to create an app in twitter and generate consumer key(API), consumer secret(API secret key), Access token Key, Access secret token key. We need these keys to get access to API in twitter through R.

**Step1**: Visit <https://apps.twitter.com>

**Step2**: Log on to a twitter account (if not present create an account using e-mail id)

**Step3**: Click Create New App.

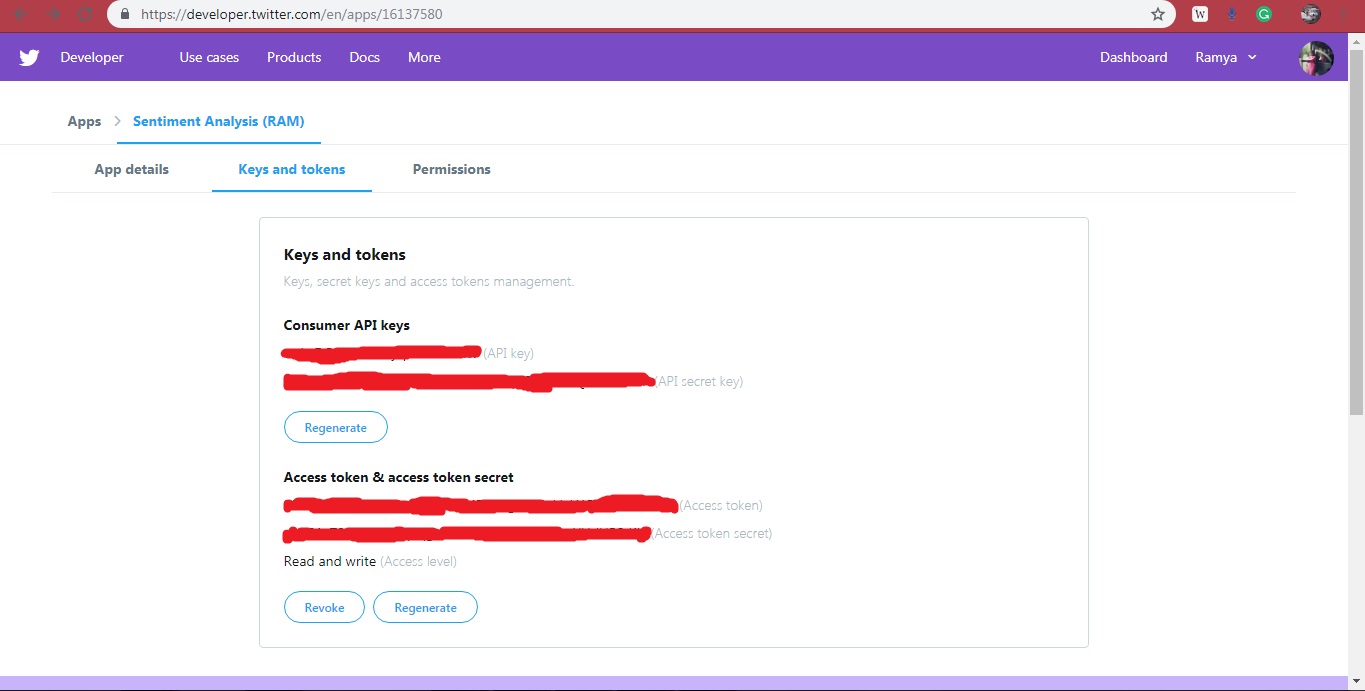


**Step4**: Provide a name for the application and add a description to it.

**Step5**: Give a valid URL for the application, generic URL like <https://www.google.co.in/>

**Step6**: Click the create button

**Step7**: API key, API secret key, Access token key, Access token secret key will be generated. Copy the Keys and use them for extracting tweets.



1. Packages required for Twitter analysis

The following packages will be used in R for sentiment analysis

**twitteR**: Provides access to Twitter API

**plyr**: used in data manipulation. Makes it easy to split and combine data

**ROAuth**: Provides an interface to OAuth 1.0 specification allowing users to authenticate via OAuth to the server of their choice.

**stringr**: A consistent, simple and easy to use set of wrappers around the fantastic 'stringi' package. All function and argument names (and positions) are consistent, all functions deal with "NA"'s and zero length vectors in the same way, and the output from one function is easy to feed into the input of another.

**ggplot2**: Creating complex graphs

**RTextTools**: Automatic text classification that makes it simple for novice users to get started with ML.

**e1071**: Naïve Bayes classifier

**tm**: A framework for text mining applications within R

**dplyr**: Can manupilate datasets efficiently and faster in R and has a more consistent API.

**caret** : Misc functions for training and plotting classification and regression models.

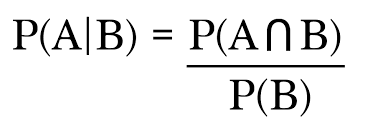
SOLUTION STATEMENT:

SENTIMENT ANALYSIS:

The sentiment score for the extracted tweets are calculated using sent.score and categorized into positive, negative and neutral measures. This polarized data is used to create charts and used to compare shopping sites.

NAÏVE BAYES CLASSIFICATION:

A Naïve Bayes classifier is a probabilistic machine learning model that’s used for classification task. The classifier is based on Bayes theorem:



Where **B=** event occurred

**A=** hypothesis

we train Naïve Bayes algorithm using the tweets and polarity data to predict new tweets.

METHODOLOGY

PART I:

* 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.

**==** Setting the Authorization for Extracting Tweets:

**>api\_key<-"wvkvDQMu1rIMSyqJ6kdocTxRV"**

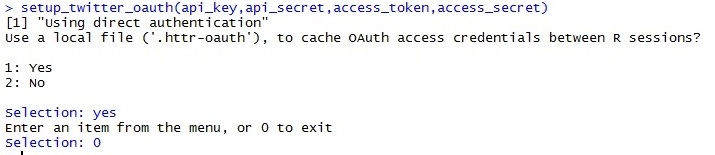
**>api\_secret<-"3jx8ChXcc4WVJjspxI72LJWDRfJcqC0qU3ihaaRXQr5bWtILVh"**

**>access\_token<-"2912970973-u7bvgqsp1rMt9MFU1ug7Cs5qVaLXCKKr5EGNTML"**

**>access\_token\_secret<-"z1N01zZ3q7yxKckvq1qgnMsLkWwDxCkG8WvyYHdU5StKH"**

**==** Setting connection between Twitter and R

**>setup\_twitter\_oauth(api\_key,api\_secret,access\_token,access\_token\_secret)**



* Required libraries

**==**Installing and loading packages

**>install.packages("twitteR")**

**>install.packages("ROAuth")**

**>install.packages("plyr")**

**>install.packages("tm")**

**>install.packages("caret")**

**>install.packages("e1071")**

**>install.packages("ggplot2")**

**>install.packages("stringr")**

**>install.packages("dplyr")**

**>install.packages("RTextTools")**

**==**Attaching Packages

**>library(twitteR)**

**>library(ROAuth)**

**>library(plyr)**

**>library(tm)**

**>library(caret)**

**>library(e1071)**

**>library(ggplot2)**

**>library(stringr)**

**>library(dplyr)**

**>library(RTextTools)**

**==**Importing files

We have to now import files containing the dictionary of positive and negetive words. we have already two files, onefor positive and another for negetive sentiments can be imported using the below code

**>posText<-read.csv("C:/Users/FAMILY/Desktop/term 2 project/Sentiment Analysis/positive.csv.csv",header = FALSE,stringsAsFactors = FALSE)**

**>str(posText)**

**>negText<-read.csv("C:/Users/FAMILY/Desktop/term 2 project/Sentiment Analysis/negative.csv.csv",header = FALSE,stringsAsFactors = FALSE)**

**>str(negText)**

Search for V1 feature:

**>posText<-posText$V1**

**>negText<-negText$V1**

Split the word and unlist:

**>posText<-unlist(lapply(posText, function(x){str\_split(x,"\n")}))**

**>negText<-unlist(lapply(negText, function(x){str\_split(x,"\n")}))**

Adding words to posText and negText

**>pos.words=c(posText,"upgrade")**

**>neg.words=c(negText,"wtf","wait","waiting","epicfail","mechanical")**

* Extracting Tweets with hasgtag:

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

**>Amazon\_tweets=searchTwitter('@Amazon',n=1000)**

**>Flipkart\_tweets=searchTwitter('@Flipkart',n=1000)**

**>Myntra\_tweets=searchTwitter('@Myntra',n=1000)**

* Processing tweets:

**==**Convert the tweets into text format:

**>Amazon\_txt<-sapply(Amazon\_tweets, function(t) t$getText())**

**>Flipkart\_txt<-sapply(Flipkart\_tweets, function(t) t$getText())**

**>Myntra\_txt<-sapply(Myntra\_tweets, function(t) t$getText())**

**==**Calculate the number of tweets for each e-commerce company:

**>noof\_tweets=c(length(Amazon\_txt),length(Flipkart\_txt),length(Myntra\_txt))**

**==**Combining the text of all these e-commerce companies:

**>Shopping\_Site<-c(Amazon\_txt,Flipkart\_txt,Myntra\_txt)**

* 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.scores=sapply(sentences, function(sentence,pos.words,neg.words){**

**#removing punctuations**

**sentences=gsub("[[:punct:]]","",sentence)**

**#removing control characters**

**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(sentence,"\\s+")**

**words=unlist(word.list)**

**#compare words to the dictionaries of positive and 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 ant 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 )**

**#dataframe with sent.scores fo each sentence**

**sent.scores.datafrm= data.frame(text=sentences,score=sent.scores)**

**return(sent.scores.datafrm)**

**}**

**>** **sent.scores = score.sentiment(Shopping\_Site, pos.words,neg.words)**

**Step 1 - Create a variable in the data frame**

**>sent.scores$Shopping\_Site= factor(rep(c("Amazon","Flipkart","Myntra"),noof\_tweets))**

**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)**

**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")**

**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)))**

* Generating Graphs

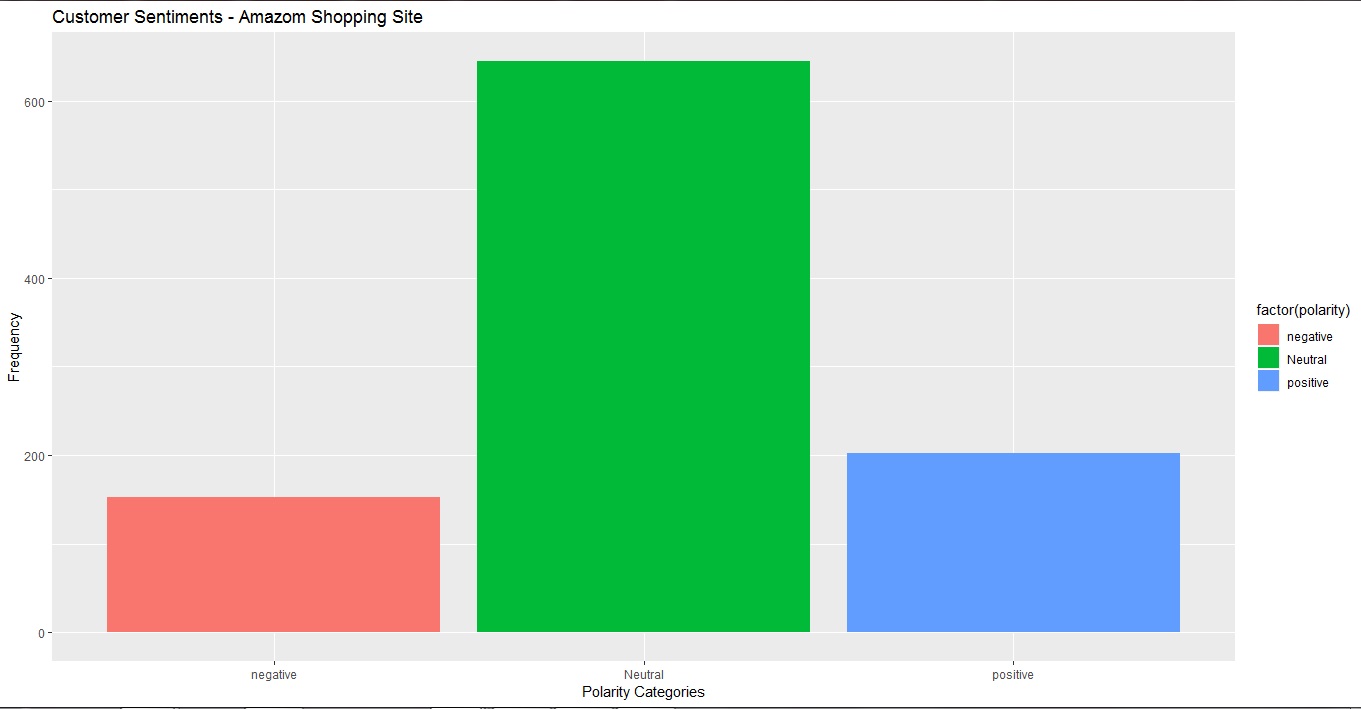
Plot 1- Polarity Plot – Customer Sentiments (Amazon)

**>qplot(factor(polarity), data=Amazon\_Shopping\_Site, geom="bar",**

**fill=factor(polarity))+xlab("Polarity Categories") +**

**ylab("Frequency") + ggtitle("Customer Sentiments - Amazon**

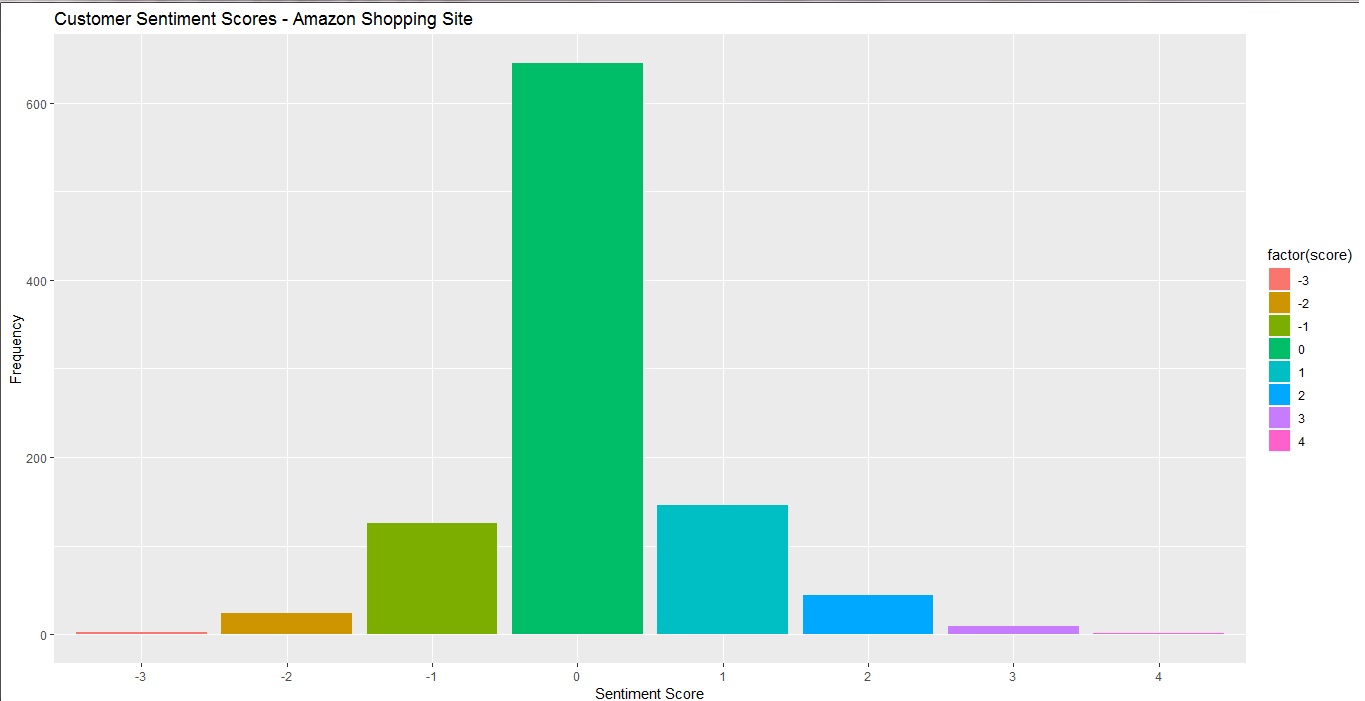
**Shopping Site")**

****

The bar graph above depicts polarity if we closely analyze the graph. It reveals that out of 1,000 Twitter users, 100 users have commented in a negative way while 660 users are neutral. However, 240 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.

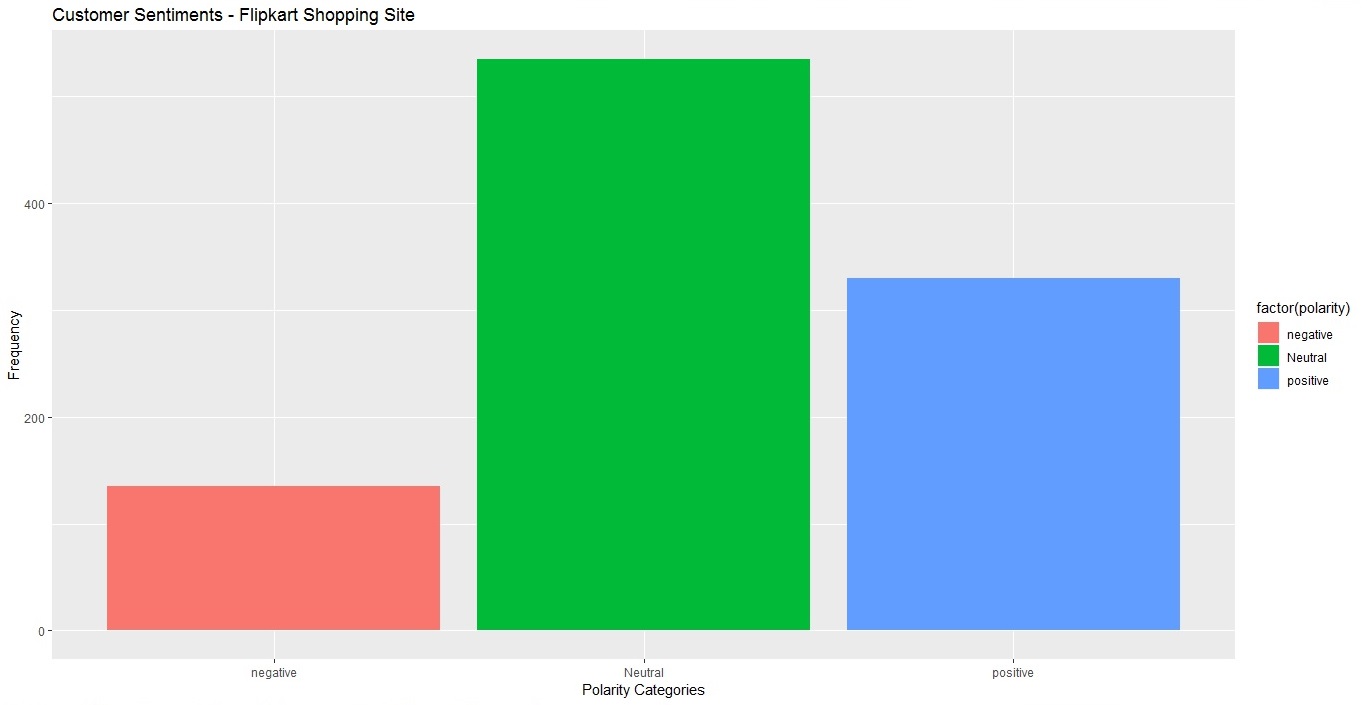
Plot 3 - Polarity Plot – Customer Sentiments (Flipkart)

**>qplot(factor(polarity), data=Flipkart\_Shopping\_Site, geom="bar",**

**fill=factor(polarity))+xlab("Polarity Categories") +**

**ylab("Frequency") + ggtitle(" Customer Sentiments - Flipkart**

**Shopping Site ")**

****

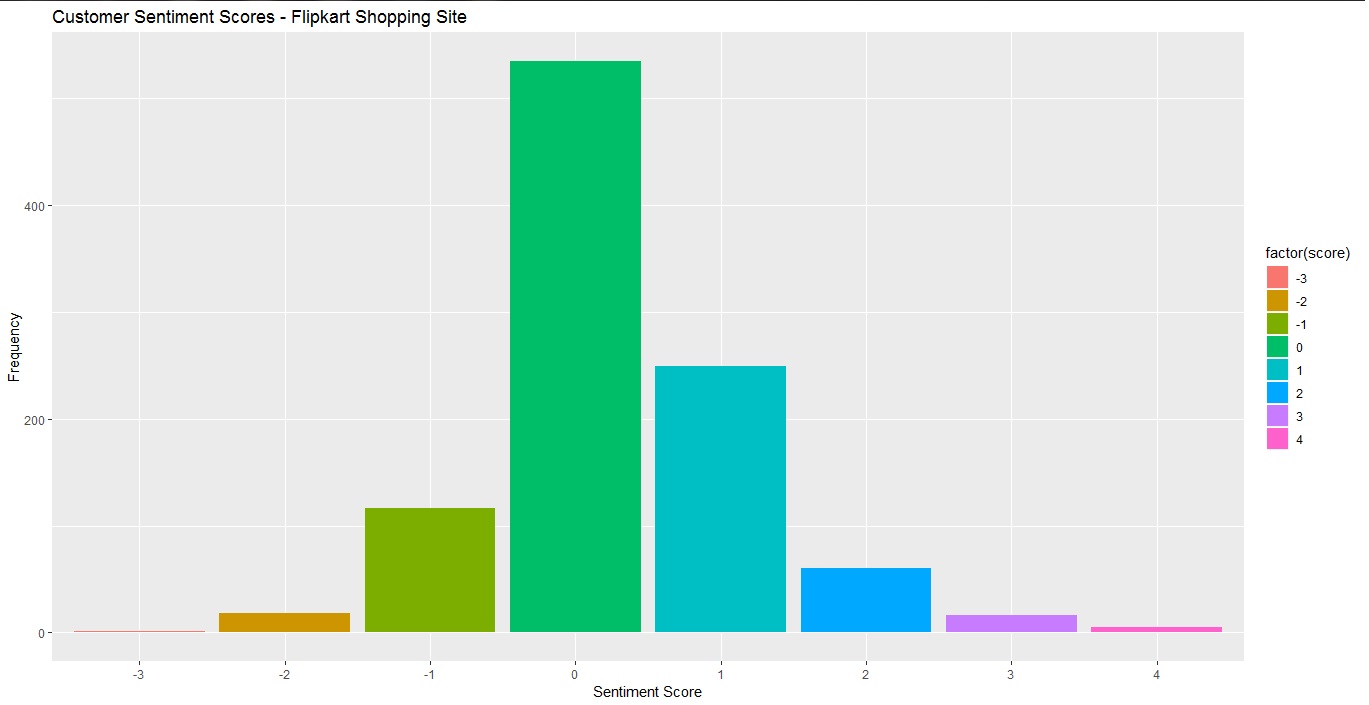
The bar graph above represents polarity. In this case, out of the 1,000 Twitter users, 240 users have commented negatively, 460 users remain neutral, and 300 users are positive about Flipkart.

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.

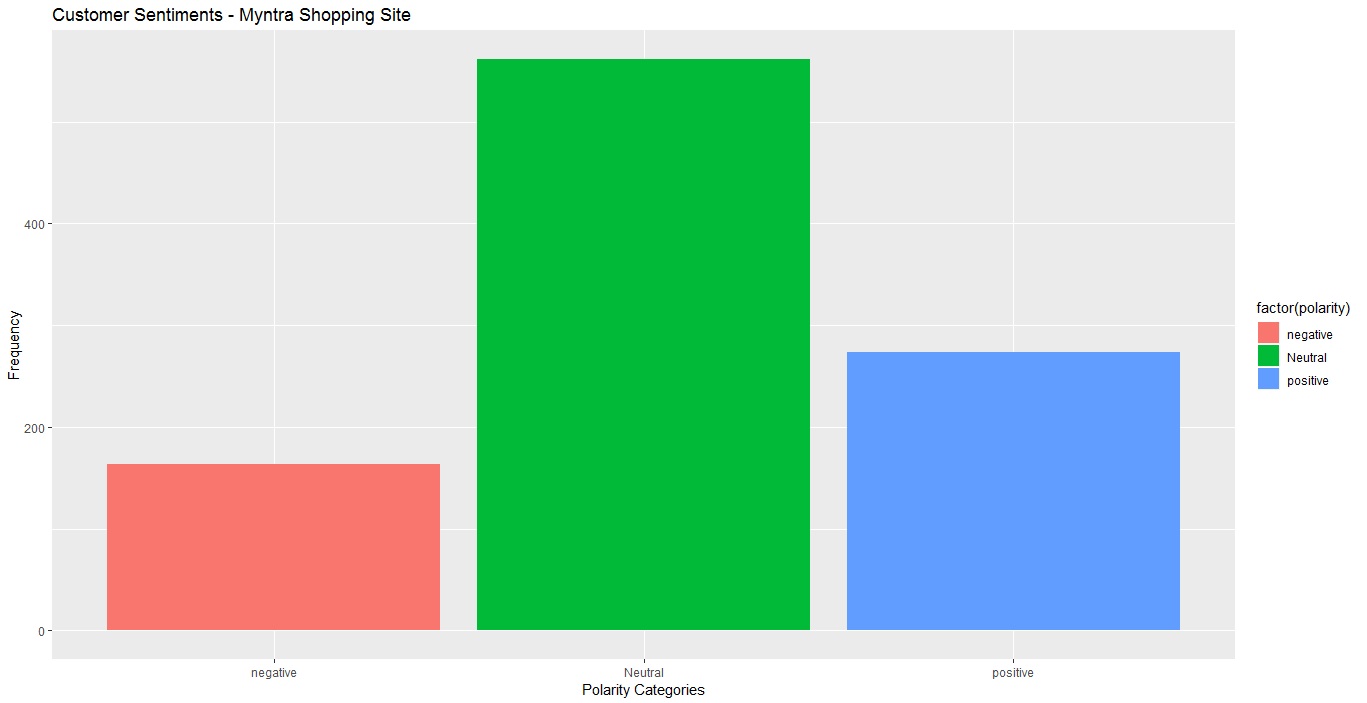
Plot 5 - Polarity Plot – Customer Sentiments (Myntra)

**>qplot(factor(polarity), data=Myntra\_Shopping\_Site, geom="bar",**

**fill=factor(polarity))+xlab("Polarity Categories") +**

**ylab("Frequency") + ggtitle("Customer Sentiments - Myntra**

**Shopping Site")**

****

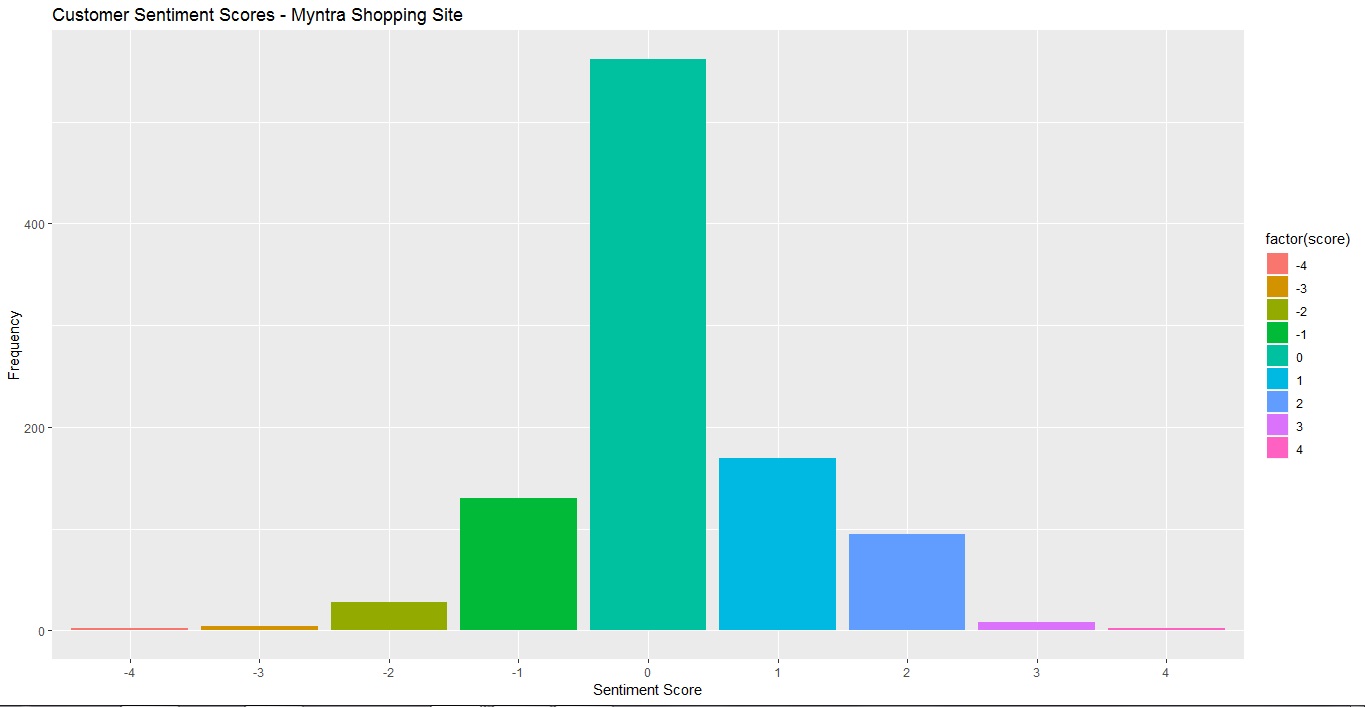
The bar graph above represents polarity. In this case, out of the 1,000 Twitter users, 80 users have commented negatively, 380 users are neutral, and the remaining 540 users remain positive about the e-commerce company**.**

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.

* 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))**

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**

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

**> datafrm$pos\_percent\_score = round(100\*datafrm$pos\_count/datafrm$total\_count)datafrm$neg\_percent\_score = round(100\*datafrm$neg\_count/datafrm$total\_count)datafrm$neu\_percent\_score = round(100\*datafrm$neu\_count/datafrm$total\_count)**

* Comparison Charts

Comparison 1 - Positive Comparative Analysis

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

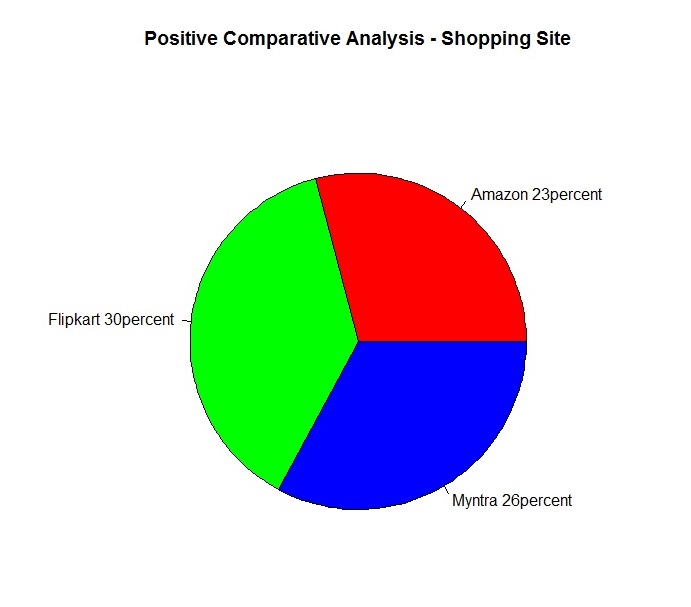
**>attach(datafrm)**

**>pie.chart.lbls<-paste(datafrm$Shopping\_Site,datafrm$pos\_percent\_score)**

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

**>pie(pos\_percent\_score, labels = pie.chart.lbls, col=rainbow(length(pie.chart.lbls)), main="Positive Comparative Analysis - Shopping Site")**

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



Comparison 2 - Negative Comparative Analysis

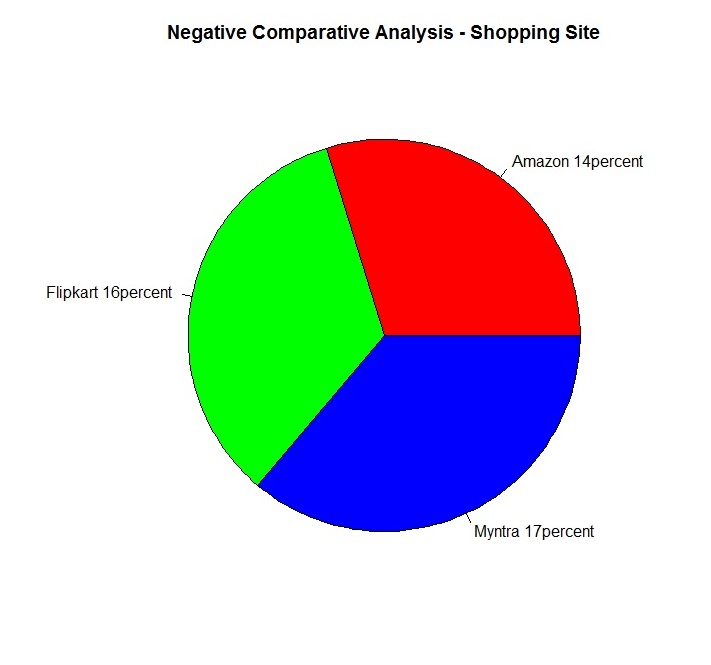
Here is the code to create a negative comparison pie chart for these three ecommerce companies:

**>pie.chart.lbls<-paste(datafrm$Shopping\_Site,datafrm$neg\_percent\_score)**

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

**>pie(neg\_percent\_score, labels = pie.chart.lbls, col=rainbow(length(pie.chart.lbls)), main="Negative Comparative Analysis - Shopping Site")**

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



Comparison 3 - Neutral Comparative Analysis

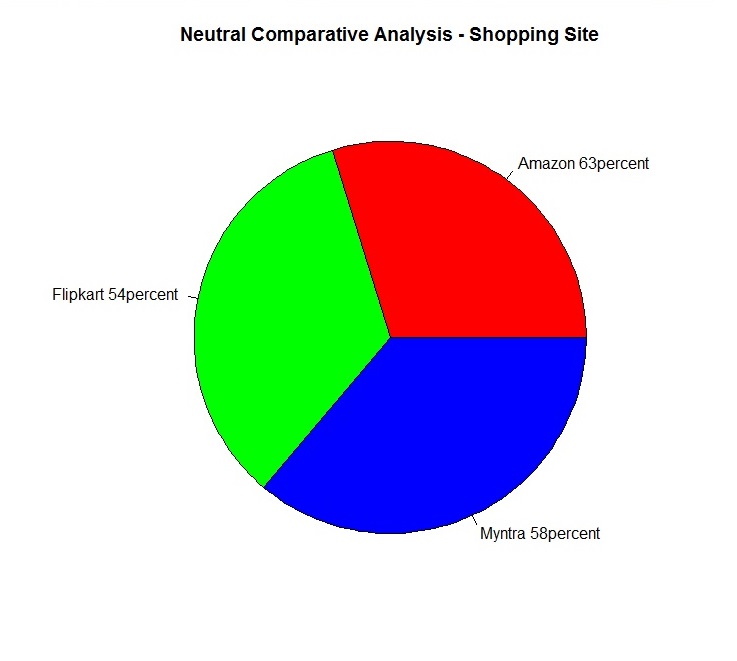
Here is the code to create a neutral comparison pie chart:

**>pie.chart.lbls<-paste(datafrm$Shopping\_Site,datafrm$neu\_percent\_score)**

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

**>pie(neu\_percent\_score, labels = pie.chart.lbls, col=rainbow(length(pie.chart.lbls)), main="Neutral Comparative Analysis - Shopping Site")**

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

****

PART II:

NAÏVE BAYES

* Data Preprocessing

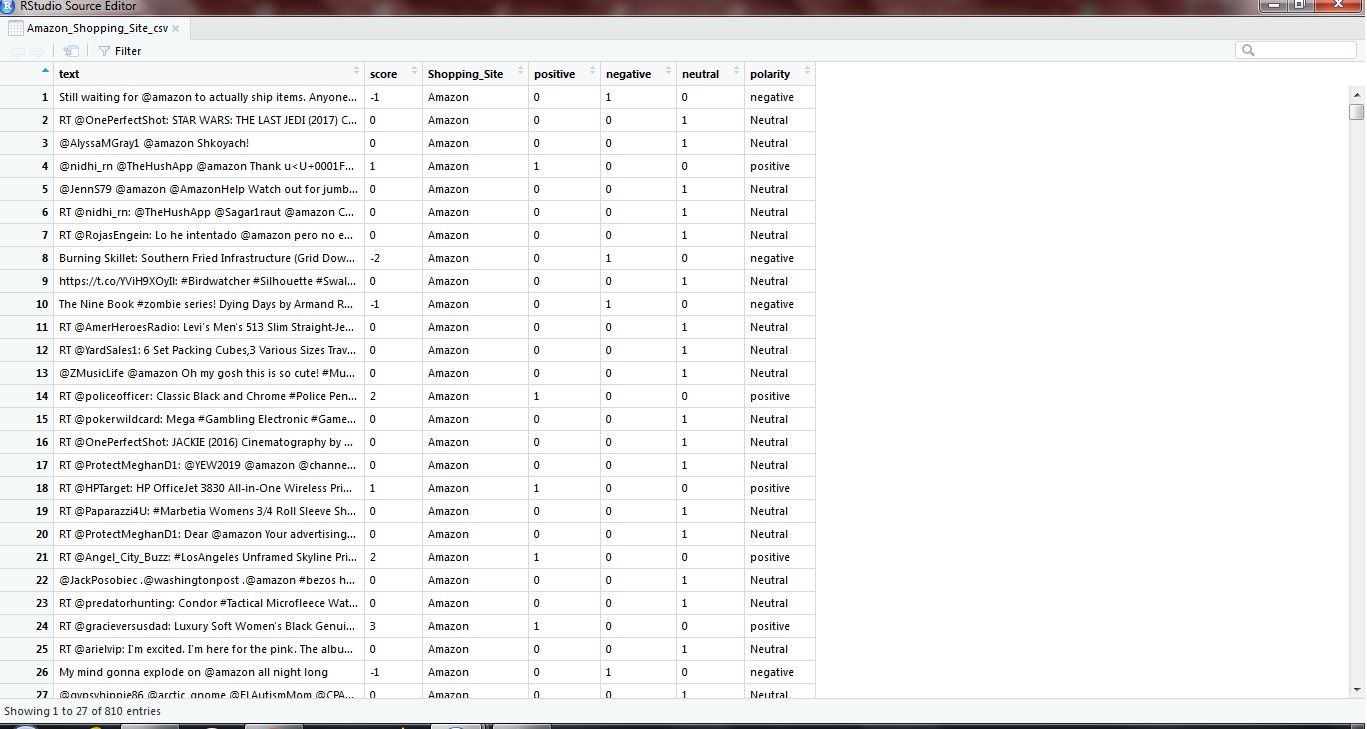
We will first load all the required libraries (packages).

**==** Writing and Reading the Data a‘Amazon\_Shopping\_Site’

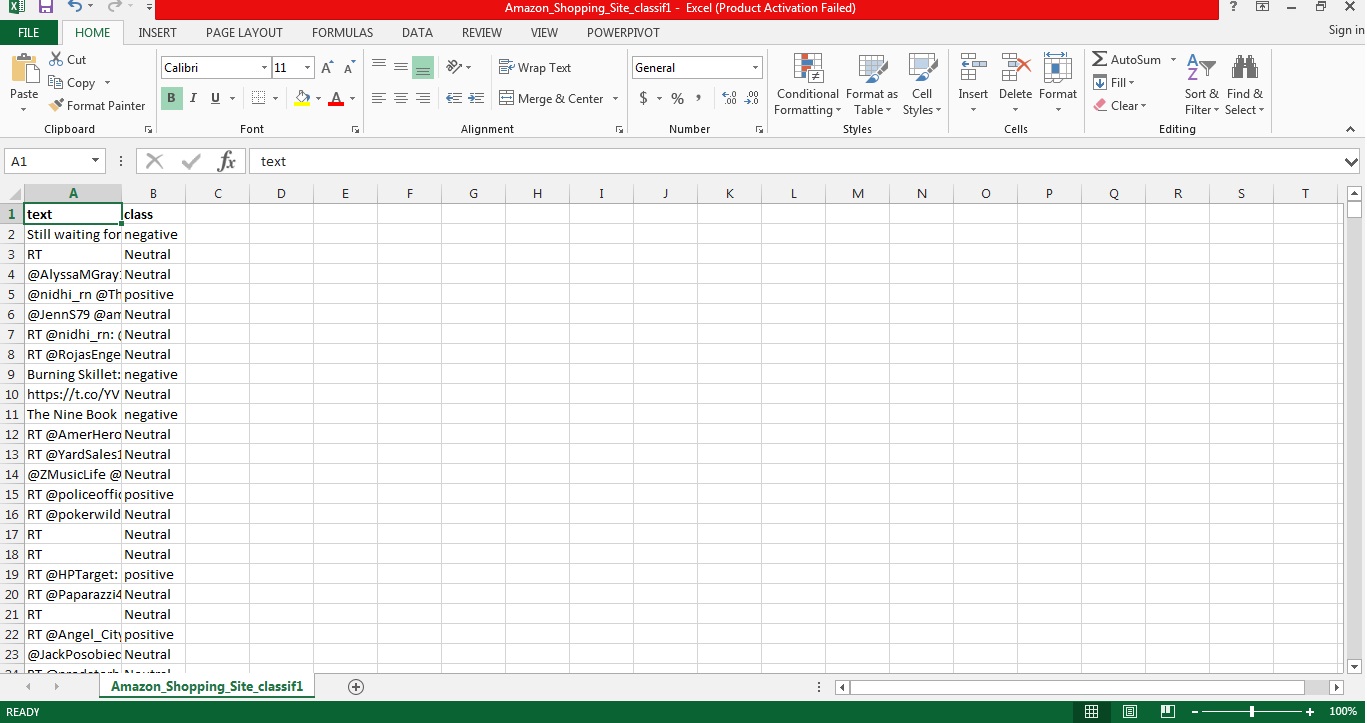
**>write.table(Amazon\_Shopping\_Site,"C:/Users/FAMILY/Desktop/term 2 project/Sentiment Analysis/Amazon\_Shopping\_Site.csv", append = T, row.names = F,col.names = T,sep = ",",)**

**>Amazon\_Shopping\_Site\_csv <-read.csv("C:/Users/FAMILY/Desktop/term 2 project/Sentiment Analysis/Amazon\_Shopping\_Site.csv",header = TRUE, encoding = "UCS-2LE")**

**>View(Amazon\_Shopping\_Site\_csv)**



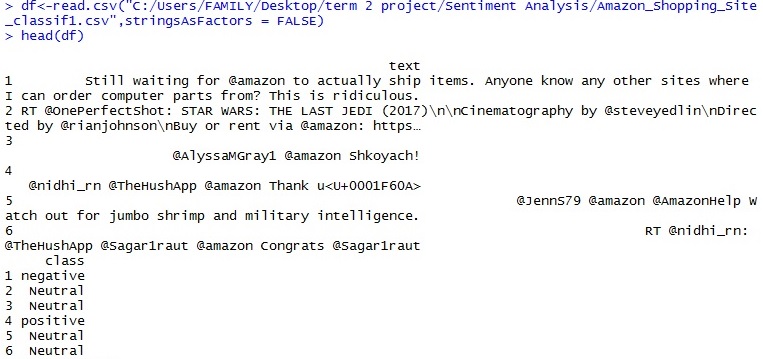
Amazon\_Shopping\_Sites\_classif1.csv



Read the Amazon\_Shopping\_Sites\_classif1.csv

**>df<-read.csv("C:/Users/FAMILY/Desktop/term 2 project/Sentiment Analysis/Amazon\_Shopping\_Site\_classif1.csv",stringsAsFactors = FALSE)**

**>head(df)**

****

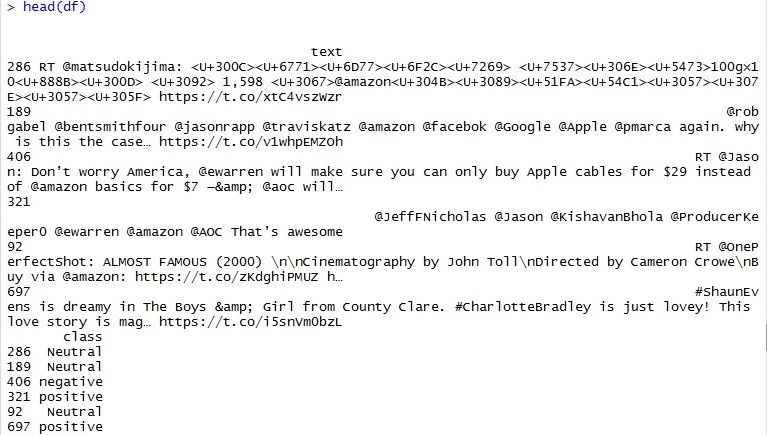
Randomize the dataset and convert the 'class' variable from character to factor.

**>set.seed(1)**

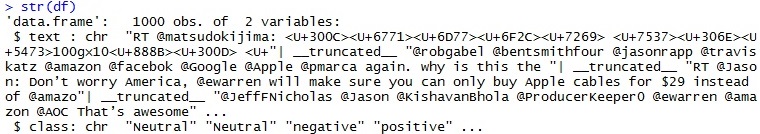
**>df<- df[sample(nrow(df)), ]**

**>df<- df[sample(nrow(df)), ]**

**>head(df)**

****

**>str(df)**

****

**>df$class <- as.factor(df$class)**

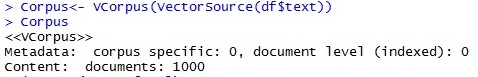
* Bag of Words Tokenization

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.

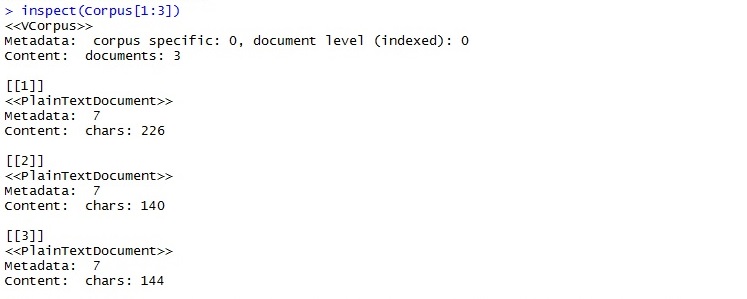
We first prepare a corpus of all the documents in the dataframe.

>**Corpus<- VCorpus(VectorSource(df$text))**

**>Corpus**

****

**>inspect(Corpus[1:3])**

****

* 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.

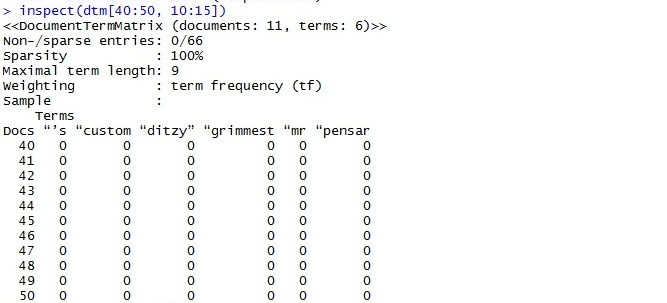
**>corpus.clean<-Corpus %>% tm\_map(content\_transformer(tolower)) %>% tm\_map(removePunctuation) %>% tm\_map(removeNumbers) %>% tm\_map(removeWords, stopwords(kind = "en")) %>% tm\_map(stripWhitespace)**

* 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)**

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

****

* Partitioning the Data

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

**>df.train<- df[1:697,]**

**>df.test<- df[698:1000,]**

**>dtm.train<- dtm[1:697,]**

**>dtm.test<- dtm[698:1000,]**

**>corpus.clean.train <- corpus.clean[1:697]**

**>corpus.clean.test <- corpus.clean[698:1000]**

Feature Selection

**>dim(dtm.train)**

****

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)**

**>length((fivefreq))**

****

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

**>dim(dtm.train.nb)**

****

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

**>dim(dtm.test.nb)**

****

* The Naive Bayes algorithm

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.

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**

**}**

Applying the convert\_count function to get the final training and testing DTMs**:**

**>trainNB <- apply(dtm.train.nb, 2, convert\_count)**

**>testNB <- apply(dtm.test.nb, 2, convert\_count)**

* 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(trainNB, df.train$class,laplace = 1))**

****

**==**Testing the Predictions

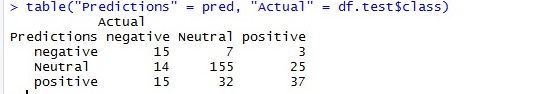
Use the NB classifier we built to make predictions on the test set:

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

****

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

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

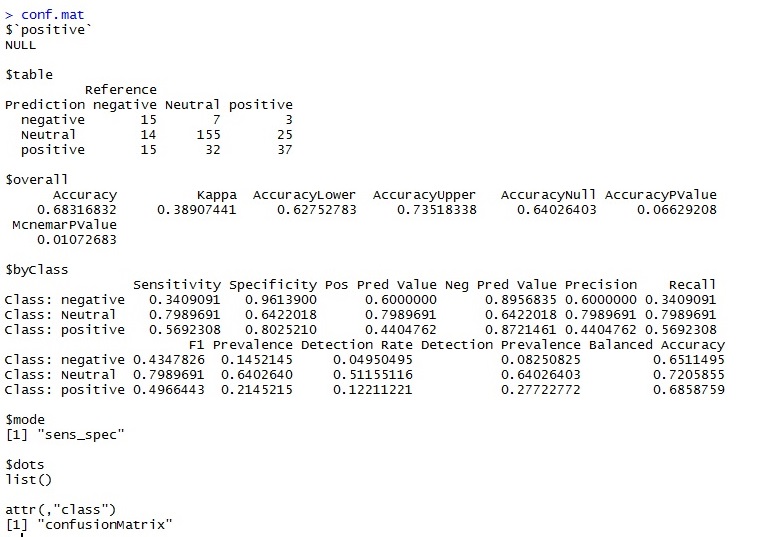
****

**==**Confusion Matrix

Prepare the confusion matrix:

**>conf.mat<- confusionMatrix(pred, df.test$class)**

**>conf.mat**

****

FINDINGS AND CONCLUSIONS

In the first part, we analysed tweets for competing e-commerce brands and characterised 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 Myntra is the most-liked brand out of the three brands (Amazon, Myntra, and Flipkart) we analyzed for this project . Customer tweets for Myntra were mostly of positive sentiment as opposed to Flipkart, which had tweets mostly of negative sentiment and Amazon, which had tweets mostly of neutral sentiment.

In the second part, 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 73.73%; 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.

===========================================