Sentiment Analysis on Shopping Sites Using R

Developed by

C Aishwarya

S190020100271

NIIT South Extension

New Delhi

CONTENTS:

* Abstract
* Introduction
* What is Sentiment Analysis
* Why Sentiment Analysis
* Sentiment Analysis in Twitter
* Challenges performing SA on twitter tweets
* Overview
* Implementation
* Finding and Conclusion
* Bibliography and Reference

Abstract

Our day-to-day life has always been influenced by what people think. Ideas and opinions of others have always affected our own opinions. An important part of our information-gathering behavior has always been to find out what other people think. With the growing availability and popularity of opinion-rich resources such as online review sites and personal blogs, new opportunities and challenges arise as people now can, and do, actively use information technologies to seek out and understand the opinions of others. The sudden eruption of activity in the area of opinion mining and sentiment analysis, which deals with the computational treatment of opinion, sentiment, and subjectivity in text, has thus occurred at least in part as a direct response to the surge of interest in new systems that deal directly with opinions as a first-class object.

And now Internet has made it possible to find out the opinions of millions of people on everything from latest gadgets to political philosophies. Internet is increasingly both the forum for discussion and source of information for a growing number of people

Ready availability of opinionated text has created a new area in text analysis, expanding the subject of study from traditionally fact- and information-centric view of text to enable sentiment-aware applications. In the past decade, extraction of sentiment from text has been getting a lot of attention in both industry and academia. Increasingly businesses realize the importance of Internet users’ opinions about their product and services.

Aim of this project is to create insightful graphs that indicate consumer sentiment towards e-commerce websites, such as Amazon, Myntra and flipkart.

entiment polarity is a particular feature of text. It is usually dichotomised

into two – positive and negative – but polarity can also be thought of as a range. A

document containing several opinionated statements would have a mixed polarity

overall, which is different from not having a polarity at all (being objective). Fur-

thermore, a distinction must be made between the polarity of sentiment and of its

strength. One may feel strongly about a product being OK, not particularly good

or bad; or weakly about a product being very good (because perhaps one owned it

for too short time to form a strong opinion).

Sentiment polarity is a particular feature of text. It is usually dichotomised

into two – positive and negative – but polarity can also be thought of as a range. A

document containing several opinionated statements would have a mixed polarity

overall, which is different from not having a polarity at all (being objective). Fur-

thermore, a distinction must be made between the polarity of sentiment and of its

strength. One may feel strongly about a product being OK, not particularly good

or bad; or weakly about a product being very good (because perhaps one owned it

for too short time to form a strong opinion).

**INTORDUCTIOn**

Opinions are central to almost all human activities and are key influencers of our behaviors. Our beliefs and perceptions of reality, and the choices we make, are, to a considerable degree, conditioned upon how others see and evaluate the world. For this reason, when we need to make a decision we often seek out the opinions of others. This is not only true for individuals but also true for organizations. Opinions and its related concepts such as sentiments, evaluations, attitudes, and emotions are the subjects of study of sentiment analysis and opinion mining. The inception and rapid growth of the field coincide with those of the social media on the Web, e.g., reviews, forum discussions, blogs, microblogs, Twitter, and social networks, because for the first time in human history, we have a huge volume of opinionated data recorded in digital forms. Since early 2000, sentiment analysis has grown to be one of the most active research areas in natural language processing. It is also widely studied in data mining, Web mining, and text mining. In fact, it has spread from computer science to management sciences and social sciences due to its importance to business and society as a whole. In recent years, industrial activities surrounding sentiment analysis have also thrived. Numerous startups have emerged. Many large corporations have built their own inhouse capabilities. Sentiment analysis systems have found their applications in almost every business and social domain.

What is Sentiment Analysis?

Sentiment Analysis is a Natural Language Processing and Information Extraction task that aims to obtain writer’s feelings expressed in positive or negative comments, questions and requests, by analyzing a large numbers of documents. Generally speaking, sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall tonality of a document. In recent years, the exponential increase in the Internet usage and exchange of public opinion is the driving force behind Sentiment Analysis today. The Web is a huge repository of structured and unstructured data. The analysis of this data to extract latent public opinion and sentiment is a challenging task.

Sentiment Analysis is a branch of computer science, and overlaps heavily with Machine Learning, and Computational Linguistics . Why? One seeks to understand the general opinion across many documents within a corpus (e.g., all tweets relating to a given brand). This is labor intensive, so we use ML to automatically label documents via classifier through a labeled dataset (supervised learning).

### Why sentiment analysis?

Let’s look from a company’s perspective and understand why would a company want to invest time and effort in analyzing sentiments of the posts. Analyzing each post and understanding the sentiment associated with that post helps us find out which are the key topics or themes which resonate well with the audience.

If the sentiment around the post is very positive, then people want to talk about the topic in that post. The topic could be a product or a service or a social message or any other thing. Understanding this can help us decide the kind of posts the company needs to put on social media platforms to increase the user engagement.

Also, analyzing the sentiment of a company over a period could help us relate its sales data with the overall sentiment. Was there a negative campaign at some time which resulted in the negative sentiment of the company.

Twitter Sentiment Analysis

Twitter is an amazing micro blogging tool and an extraordinary communication medium. In addition, twitter can also be an amazing open mine for text and social web analyses. Among the different softwares that can be used to analyze twitter, R offers a wide variety of options to do lots of interesting and fun things.Sentiment Analysis is a technique used in text mining. Twitter Sentiment Analysis may, therefore, be described as a text mining technique for analyzing the underlying sentiment of a text message, i.e., a tweet. Twitter sentiment or opinion expressed through it may be positive, negative or neutral. However, no algorithm can give you 100% accuracy or prediction on sentiment analysis.

As a part of Natural Language Processing, algorithms like SVM, Naive Bayes is used in predicting the polarity of the sentence. sentiment analysis of Twitter data may also depend upon sentence level and document level.

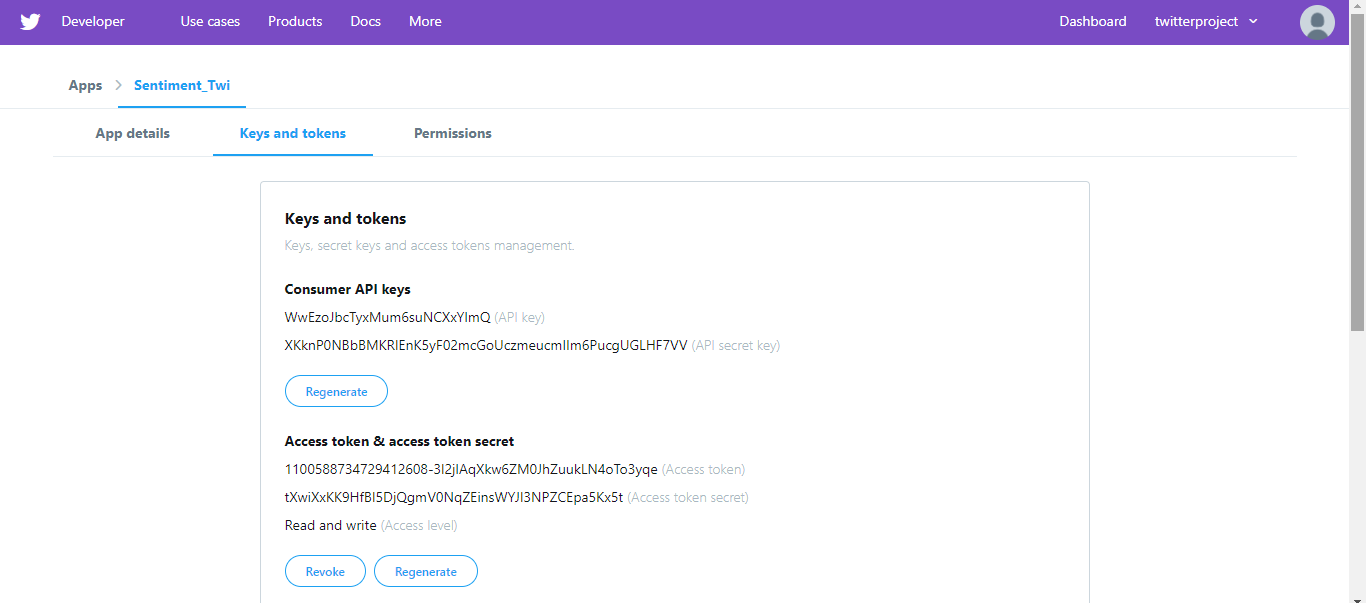
Methods like, positive and negative words to find on the sentence is however inappropriate, because the flavor of the text block depends a lot on the context.

Twitter Sentiment Analysis in R

R, a programming language intended for deep statistical analysis, is open source and available across different platforms, e.g., Windows, Mac, Linux.  You can use R to extract and visualize Twitter data.

**Implementation Prerequisites:**

1. Accessing all API Key, API Secret, Access Token Key, Access Token Secret



2. Packages for twitter analysis::

• **twitteR** :It is an interface to access Twitter API. Most functionality of the API is supported, with a bias towards API calls that are more useful in data analysis as opposed to daily interaction.

**• plyr**: This package is a collection of tools which can solve general set of problems like when we need to break down a large problem to various pieces and each piece is operated separately and then all the pieces put back together.

• **Stringr:** It is a collection of simple wrappers which help us in doing basic operations on strings like removing special characteristics , converting uppercase alphabets to lower case alphabet and by not considering spaces.

• **ggplot2** : Used to plot graphics using grammar in R. Each plot can be build up step by step from various different sources. A system for 'declaratively' creating graphics, based on "The Grammar of Graphics". You provide the data, tell 'ggplot2' how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

**ROAuth:** Provides an interface to the OAuth 1.0 specification allowing users to authenticate via OAuth to the server of their choice. Class OAuth wraps and handles OAuth handshakes and signatures for the user within R

* **e1071:** Functions for latent class analysis, short time Fourier transform, fuzzy clustering, support vector machines, shortest path computation, bagged clustering, naive Bayes classifier, ...
* **tm:** A framework for text mining applications within R. The **tm package** offers functionality for managing text documents, abstracts the process of document manipulation and eases the usage of heterogeneous text formats in **R**.
* **dplyr:** The R package***dplyr*** is an extremely useful resource for data cleaning, manipulation, visualisation and analysis. It provides simple “verbs”, functions that correspond to the most common data manipulation tasks, to help you translate your thoughts into code.
* **Caret:** The [**caret**](http://cran.r-project.org/web/packages/caret/index.html) package (short for *C*lassification *A*nd *RE*gression *T*raining) is a set of functions that attempt to streamline the process for creating predictive models. . The package contains tools for:
* data splitting
* pre-processing
* feature selection
* model tuning using resampling
* variable importance estimation

as well as other functionality.

* 1. 2 .csv files of positive and negative words, download and save in your project folder. you have to create one more for neutral words.

**Naive Bayes Classification:**

Many language processing tasks are tasks of classiﬁcation, although luckily our classes are much easier to deﬁne than those of Borges. In this classification we present the naive Bayes algorithms classiﬁcation, demonstrated on an important classiﬁcation problem: text categorization, the task of classifying an entire text by assigning it a text categorization label drawn from some set of labels.

We focus on one common text categorization task, sentiment analysis, the ex-sentiment analysis traction of sentiment, the positive or negative orientation that a writer expresses toward some object. Are view of a movie, book, or product on the web expresses the author’s sentiment toward the product, while an editorial or political text expresses sentiment toward a candidate or political action. Automatically extracting consumer sentiment is important for marketing of any sort of product, while measuring public sentiment is important for politics and also for market prediction. The simplest version of sentiment analysis is a binary classiﬁcation task ,and the words of there

view provide excellent cues. Consider, for example, the following phrases extracted from positive and negative reviews of movies and restaurants,. Words like great, richly, awesome, and pathetic, and awful and ridiculously are very informative cues:

+ ...zany characters and richly applied satire, and some great plot twists

− It was pathetic. The worst part about it was the boxing scenes...

+ ...awesome caramel sauce and sweet toasty almonds. I love this place!

− ...awful pizza and ridiculously overpriced...

Naive Bayes is a probabilistic classiﬁer, meaning that for a document d, out of all classes c∈C the classiﬁer returns the class ˆ c which has the maximum posterior probability given the document. In Eq. 1 we use the hat notation to mean “our estimate of the correct class”.

c = argmax P(c|d) where c∈C

This idea of Bayesian inference has been known since the work of Bayes (1763),Bayesian inference and was ﬁrst applied to text classiﬁcation by Mosteller and Wallace (1964). It gives us a way to break down any conditional probability P(x|y) into three other probabilities:

P(x|y) =P(y|x)P(x) /P(y)

implementation

1.Extracting and Analysing 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:

1. Setting the Authorization for Extracting Tweets:

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

api\_key<-"WwEzoJbcTyxMum6suNCXxYlmQ"

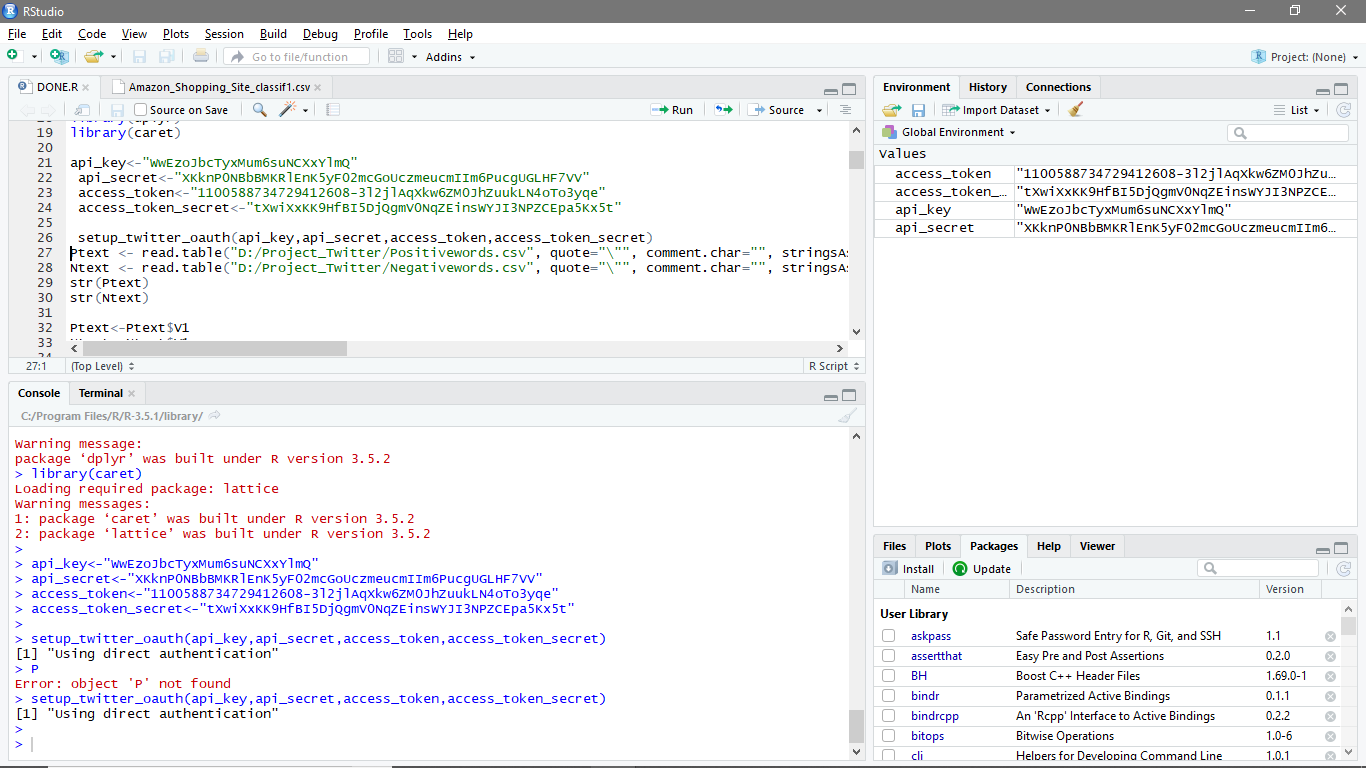
api\_secret<-"XKknP0NBbBMKRlEnK5yF02mcGoUczmeucmIIm6PucgUGLHF7VV"

access\_token<-"1100588734729412608-3l2jlAqXkw6ZM0JhZuukLN4oTo3yqe"

access\_token\_secret<-"tXwiXxKK9HfBI5DjQgmV0NqZEinsWYJI3NPZCEpa5Kx5t"

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

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

Output:

1. Required Libraries

install.packages("twitteR", lib="C:/Program Files/R/R-3.5.1/library")

install.packages("plyr", lib="C:/Program Files/R/R-3.5.1/library")

install.packages("ROAuth", lib="C:/Program Files/R/R-3.5.1/library")

install.packages("stringr", lib="C:/Program Files/R/R-3.5.1/library")

install.packages("ggplot2", lib="C:/Program Files/R/R-3.5.1/library")

install.packages("e1071", lib="C:/Program Files/R/R-3.5.1/library")

install.packages("tm", lib="C:/Program Files/R/R-3.5.1/library")

install.packages("dplyr", lib="C:/Program Files/R/R-3.5.1/library")

install.packages("caret", lib="C:/Program Files/R/R-3.5.1/library")

library(twitteR)

library(plyr)

library(ROAuth)

library(stringr)

library(ggplot2)

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 sentiments and another for negative sentiments, which can be imported using the below code:

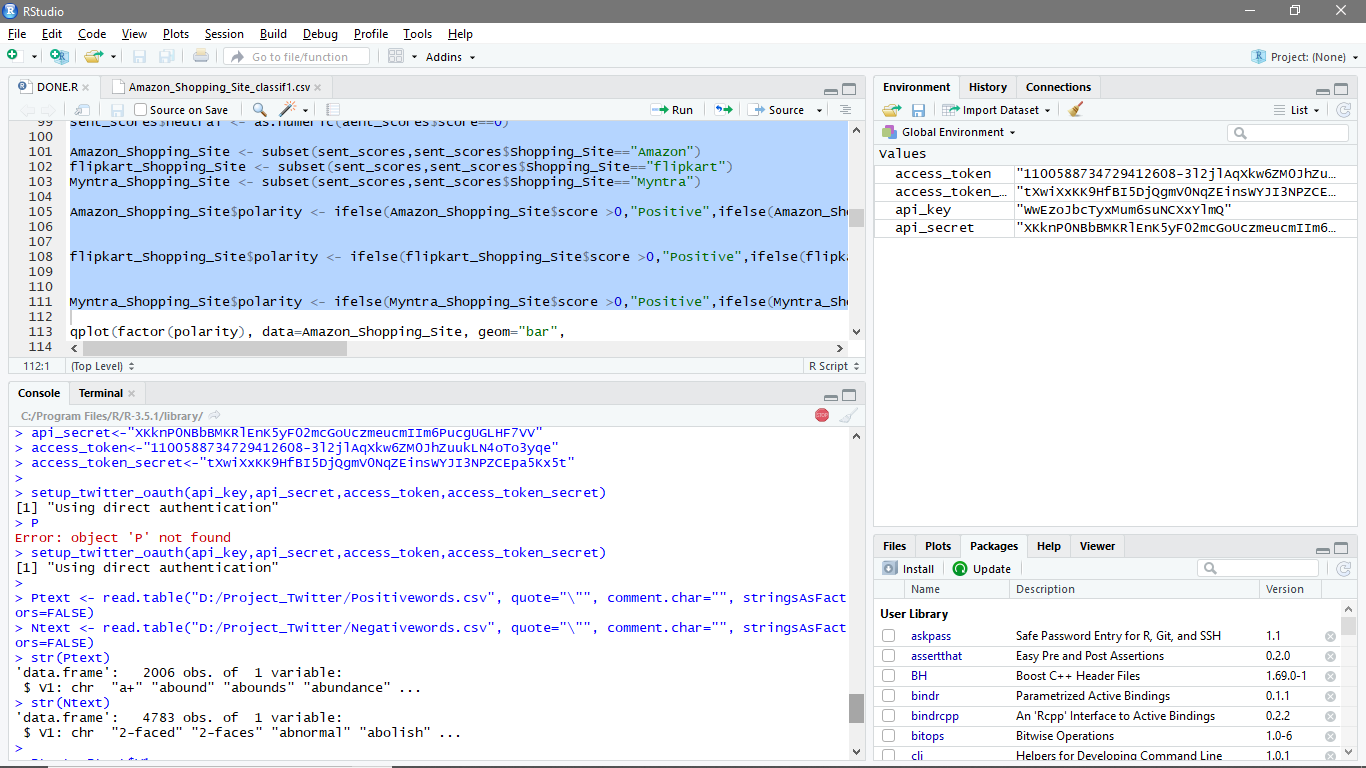
Ptexts <- read.csv("D:/Project\_Twitter/Positivewords.csv", stringsAsFactors=FALSE)

Ntext <- read.csv("D:/Project\_Twitter/Negativewords.csv",stringsAsFactors=FALSE)

str(Ptext)

str(Ntext)

Output:



Search for V1 feature, if there just refer this column/feature to the posText object.

Ptext<-Ptext$V1

Ntext<-Ntext$V1

\*\* split the words and store the unlisted data into Ntext object.

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

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

\*\* add some more words into Ptext and Ntext

pos.words = c(Ptext, 'upgrade')

neg.words=c(Ntext,"wtf","wait","waiting","epicfail","mechanical")

iv. Extracting Tweets with hashtag:

To demonstrate sentiment analysis, we analysed 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)

v. Processing Tweets

a. 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())

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

no.of.tweets=c(length(Amazon\_txt),length(flipkart\_txt),length(Myntra\_txt))

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

Shopping\_Site<-c(Amazon\_txt,flipkart\_txt,Myntra\_txt)

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 whose parameters are:-

sentences -- vector of text to score

pos.words -- vector of words of positive sentiment

neg.words -- vector of words of negative sentiment

.progress -- passed to lapply() to control the progress bar

sent.score -- is the simple array with lapply()

# -- acts as comments which is not processed by R.

score.sentiment=function(sentences, pos.words,neg.words){

require(plyr)

require(stringr)

sent\_scores=sapply(sentences, function(sentence,pos.words,neg.words){

# removing punctuations

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

# removing control characters

sentences=gsub("[[:cntrl:]]","",sentences)

# removing digits

sentences=gsub("\\d+","",sentences)

# split sentence into words with str\_split (stringr package)

word.list=str\_split(sentence, "\\s+")

words=unlist(word.list)

# 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)

# 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)

# data frame with sent\_scores for each sentence

sent\_scores.df<-data.frame(sent\_score=scores, text=sentences)

return(sent\_scores.df)

}

sent\_scores= score.sentiment(Shopping\_Site,pos.words,neg.words)

vii.. Start processing the tweets to calculate the sentiment score.

Step 1 - Create a variable in the data frame.

sent\_scores$Shopping\_Site = factor(rep(c("Amazon", "flipkart","Myntra"), no.of.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)))

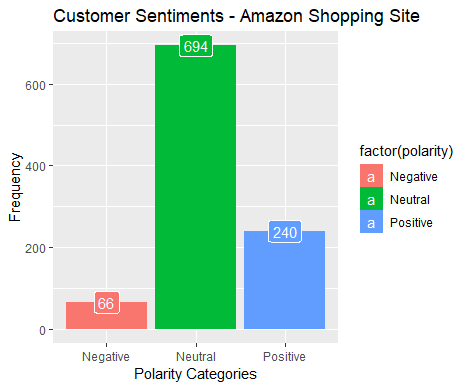
viii.. Generating Graphs: After the above steps have been executed, we will go ahead and create insightful graphs. The steps below outline the process to create graphs.

Plot 1- Polarity Plot – Customer Sentiments (Amazon)

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

fill=factor(polarity))+xlab("Polarity Categories") + geom\_label(aes(label = ..count..), stat "count", color = "white") + ylab("Frequency") + ggtitle("Customer Sentiments - Amazon

Shopping Site")



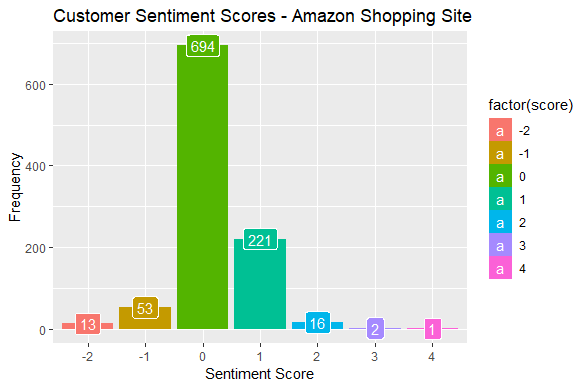
The bar graph above depicts polarity if we closely analyze the graph. It reveals that out

of 1,000 Twitter users, 66 users have commented in a negative way while 694 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") +geom\_label(aes(label = ..count..), stat = "count", color = "white")+ 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.

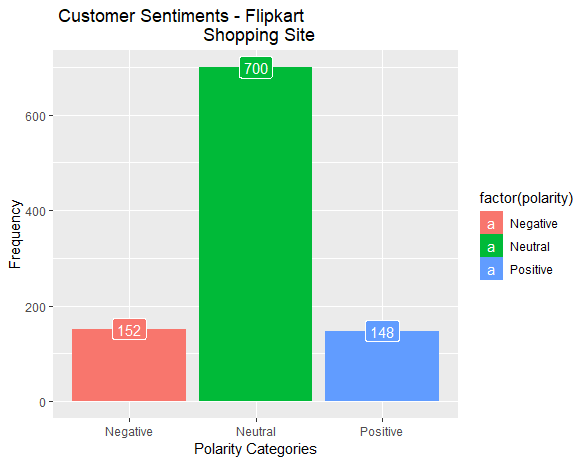
Plot 3 - Polarity Plot – Customer Sentiments (Flipkart)

qplot(factor(polarity), data=flipkart\_Shopping\_Site, geom="bar",

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

ylab("Frequency") +geom\_label(aes(label = ..count..), stat = "count", color = "white") +ggtitle(" Customer Sentiments - Flipkart

Shopping Site ")



The bar graph above represents polarity. In this case, out of the 1,000 Twitter users,

152 users have commented negatively, 700 users remain neutral, and 148 users are

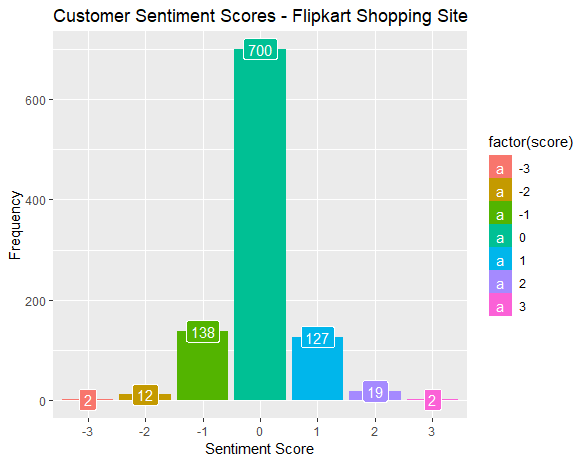
positive about Flipkart.

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

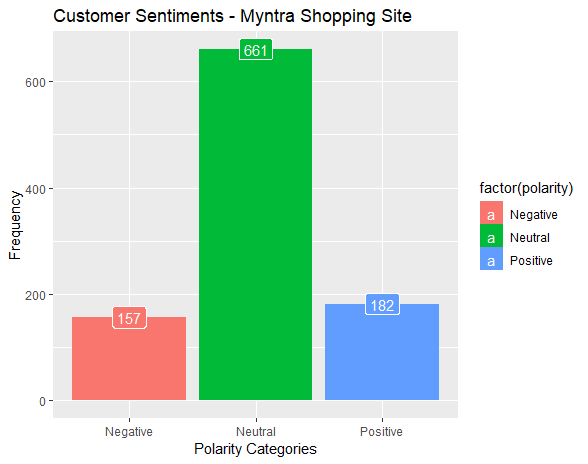
the positive score denotes that users are quite happy. The zero here represents that

users are neutral.

Plot 5 - Polarity Plot – Customer Sentiments (Myntra)

qplot(factor(score), data=flipkart\_Shopping\_Site, geom="bar", fill=factor(score))+

xlab("Sentiment Score") + ylab("Frequency")+geom\_label(aes(label = ..count..), stat = "count", color = "white")+ ggtitle("Customer Sentiment Scores - Flipkart Shopping Site")



The bar graph above represents polarity. In this case, out of the 1,000 Twitter users, 157

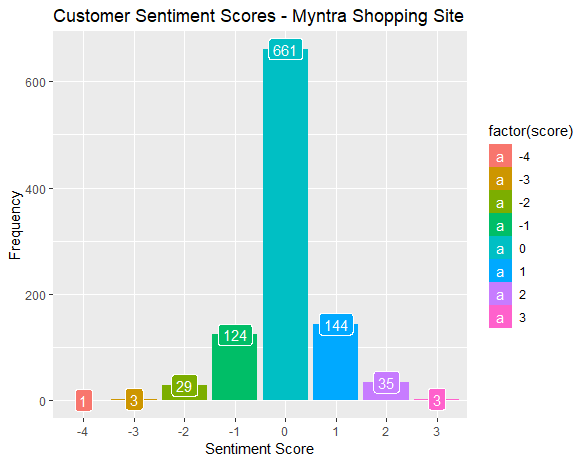
users have commented negatively, 681 users are neutral, and the remaining 182 users

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") +geom\_label(aes(label = ..count..), stat = "count",color="white")+ 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.

9.. 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 )

10.. Comparison Charts

Comparison 1 - Positive Comparative Analysis

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

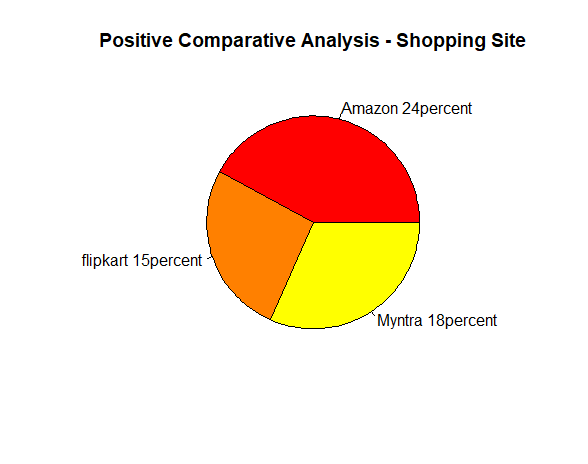
> attach(datafrm)

> 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 = heat.colors(length(pie.chart.lbls)),

main = "Positive Comparative Analysis - Shopping Site")



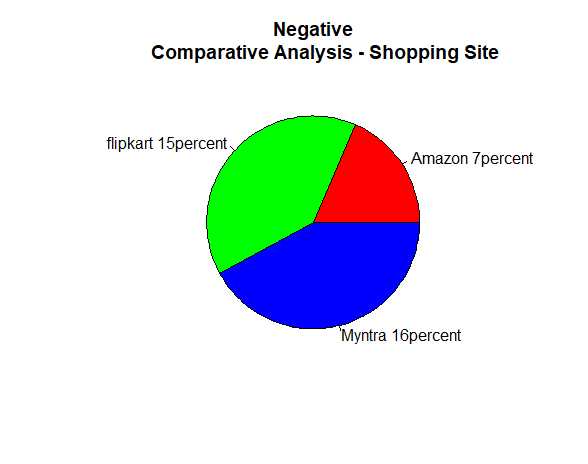
The pie chart below represents the positive percentage score of these companies:

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

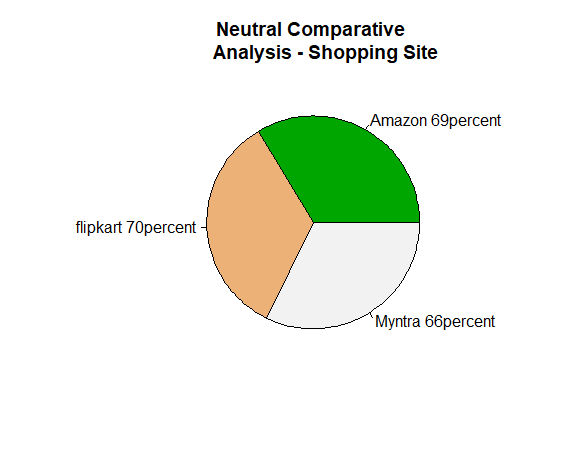
Here is the code to create a neutral comparison pie chart:

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 = terrain.colors(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

i. Data Preprocessing

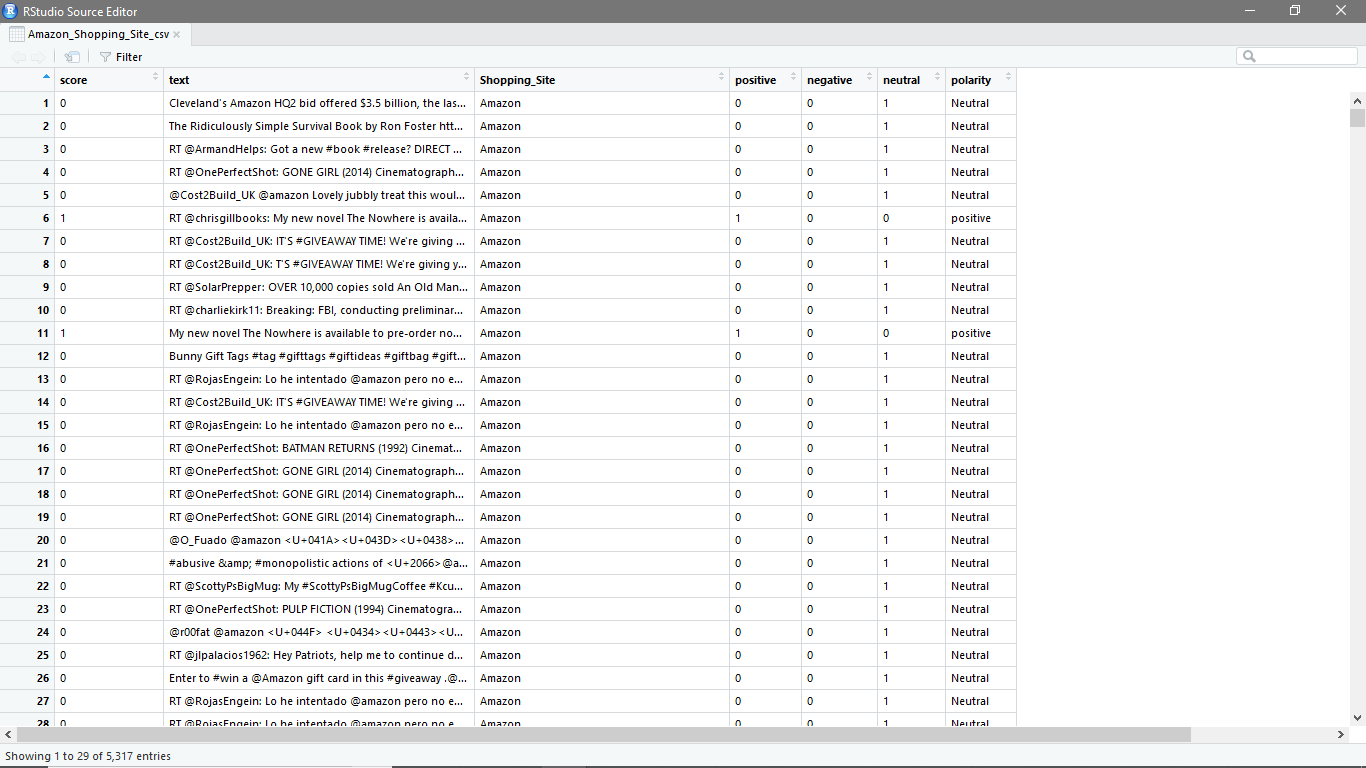
We will first load all the required libraries (packages).

a. Writing and Reading the Data as ‘Amazon\_Shopping\_Site’

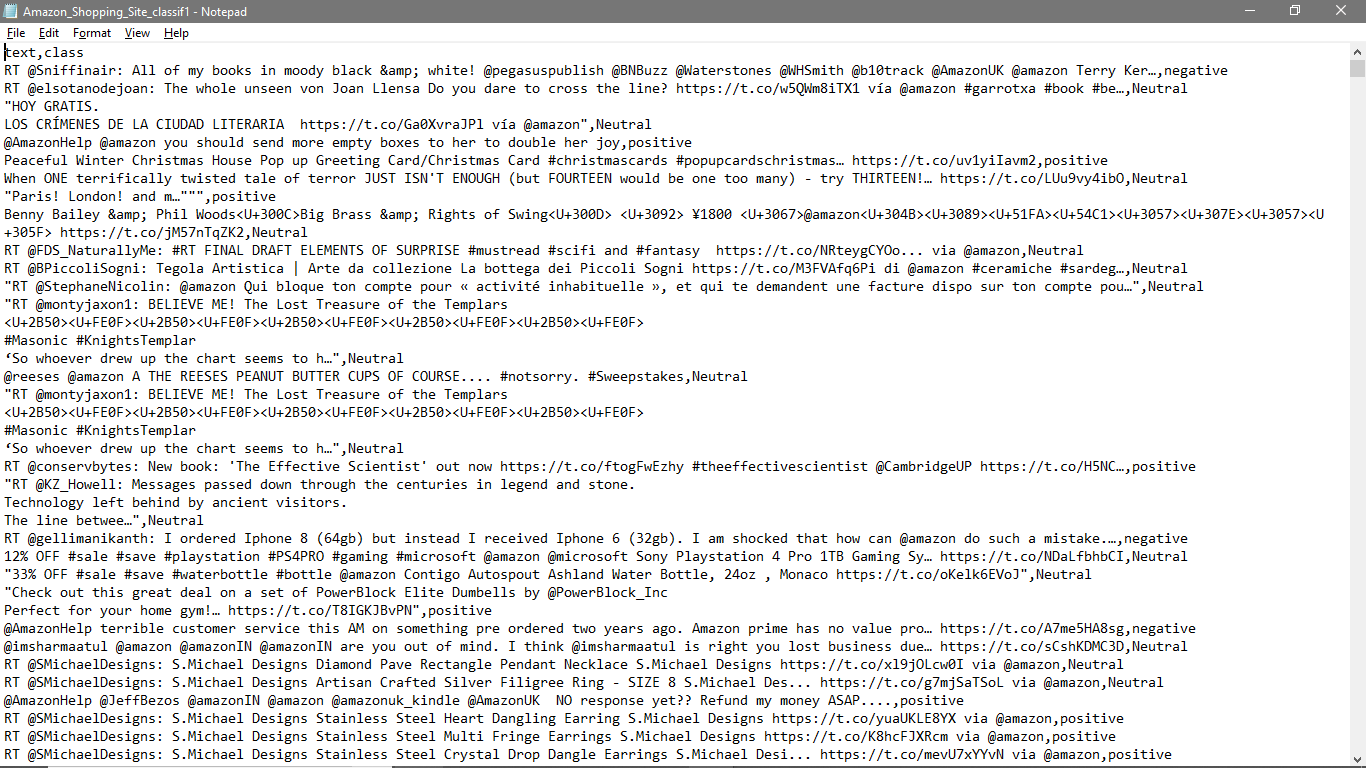
write.table(Amazon\_Shopping\_Site,"D:/Project\_Twitter/Amazon\_Shopping\_Site.csv", append=T, row.names =F, col.names=T,sep=",",)

Amazon\_Shopping\_Site\_csv <-read.csv("D:/Project\_Twitter/Amazon\_Shopping\_Site.csv", header = TRUE, encoding = "UCS-2LE")

View(Amazon\_Shopping\_Site\_csv)



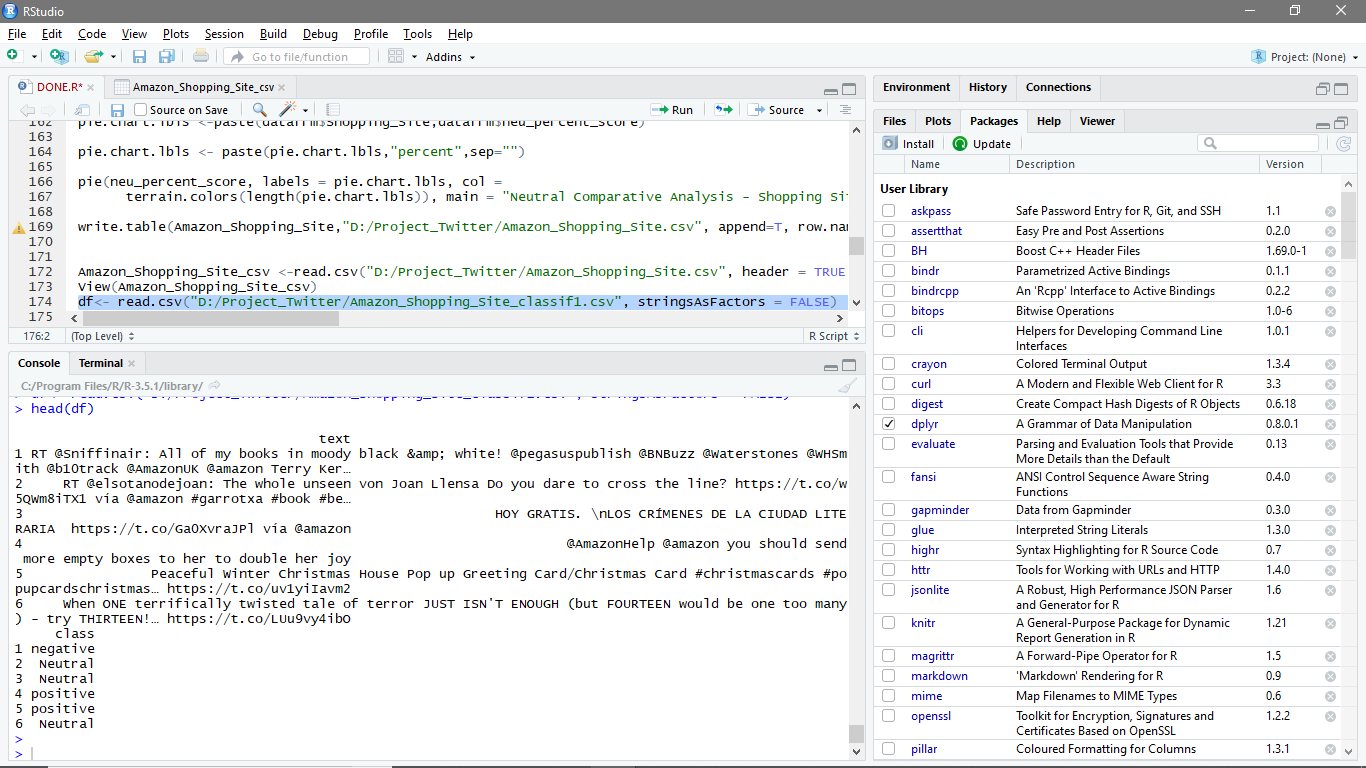
Note: Create a .csv file with only one column, class (Open Amazon\_Shopping\_Site.csv and create a new .csv file and save it as “Amazon\_Shopping\_Site\_classif1.csv” file; open this file in excel and delete all the columns except polarity; now, change the column name polarity to class and select the filter to delete all the rows other than positive, negative, and neutral tweets and save it.)



Now, read the new file, "Amazon\_Shopping\_Site\_classif1.csv".

df<- read.csv("D:/Project\_Twitter/Amazon\_Shopping\_Site\_classif1.csv", stringsAsFactors = FALSE)

head(df)



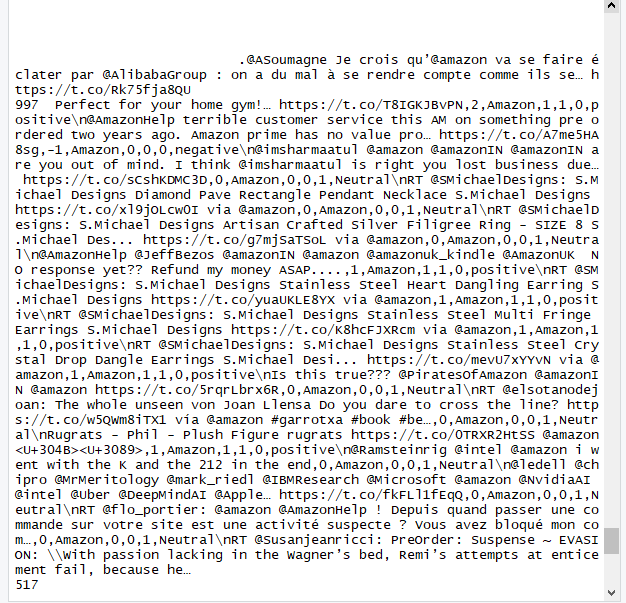
b. Randomize the dataset and convert the 'class' variable from character to factor.

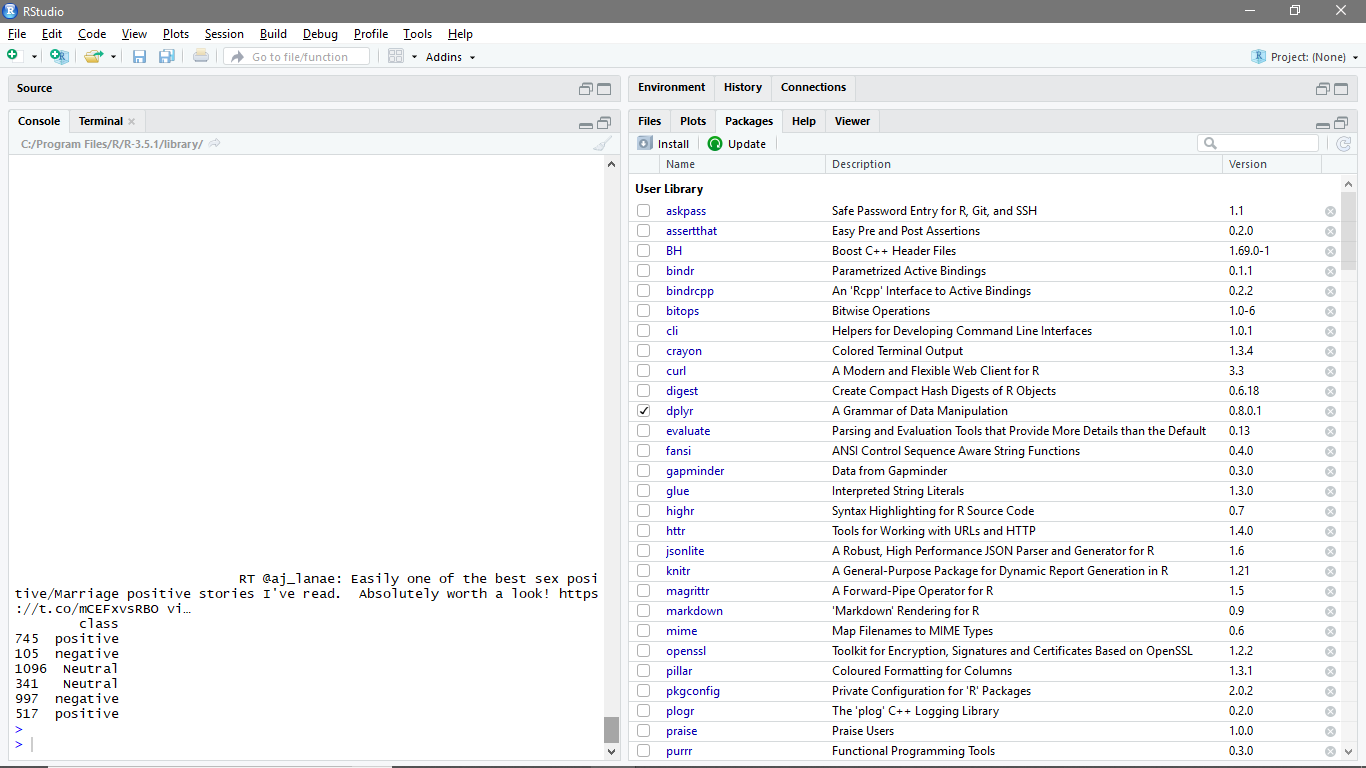
set.seed(1)

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

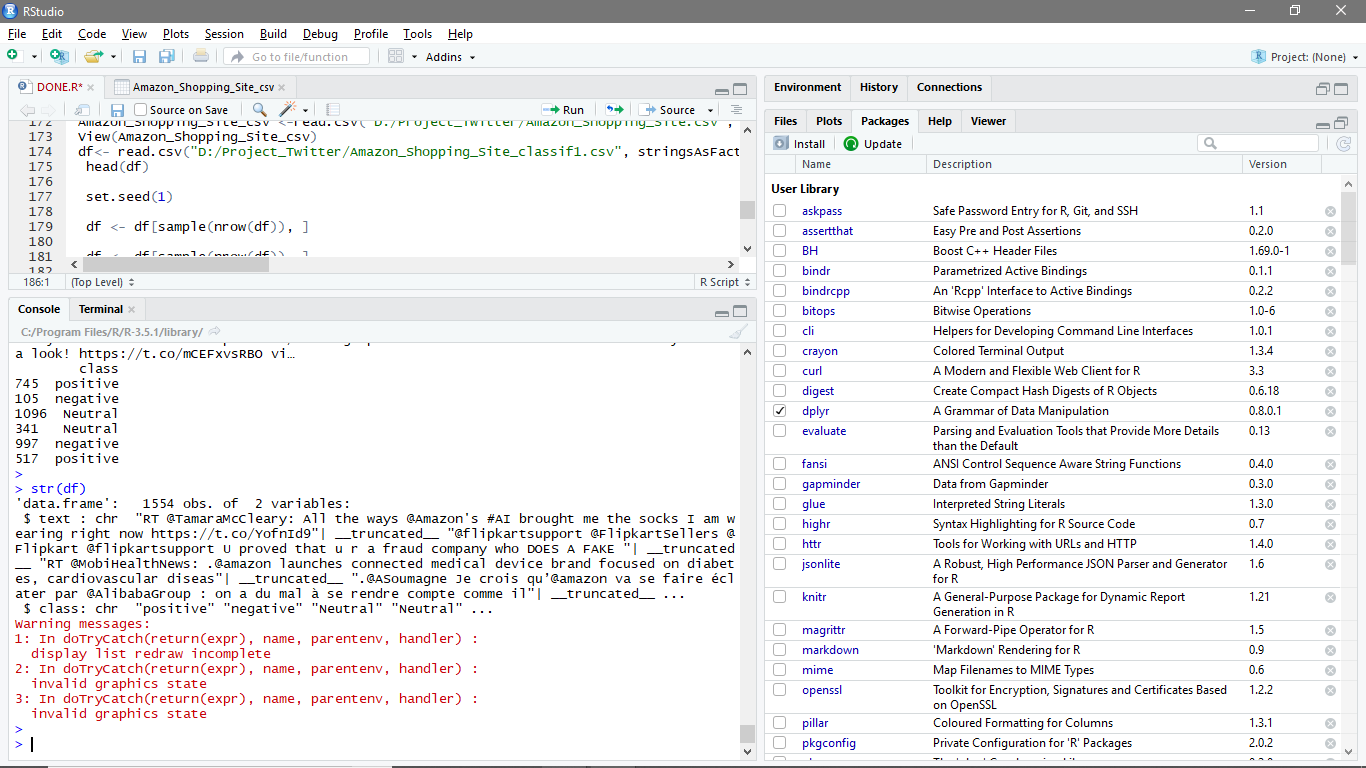
df <- df[sample(nrow(df)), ]

head(df)





str(df)



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

c. Bag of Words Tokenization

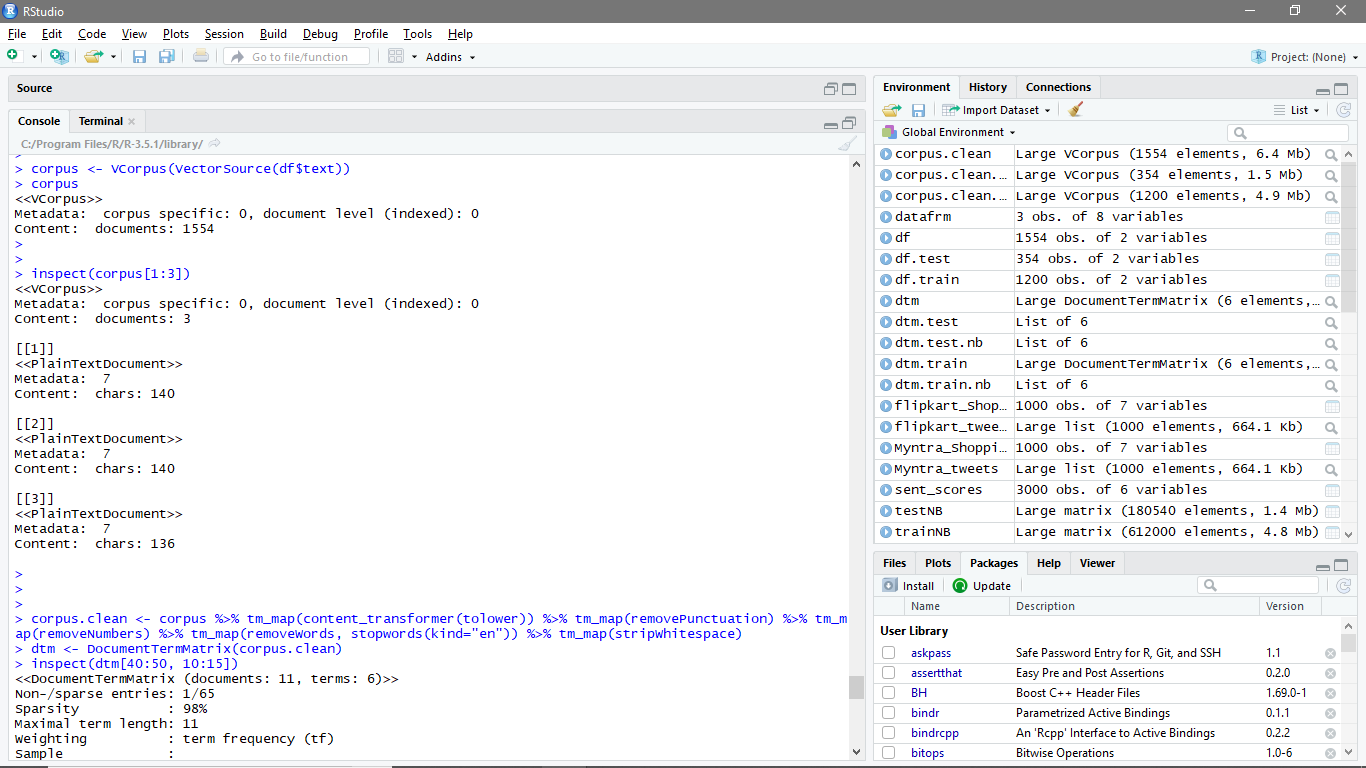
In this approach, we represent each word in a document as a token (or feature) and each document as a vector of features. In addition, for simplicity, we disregard word order and focus only on the number of occurrences of each word, which means that we represent each document as a multi-set ‘bag’ of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

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

corpus <- VCorpus(VectorSource(df$text))

corpus

inspect(corpus[1:3])



c. Data Cleanup

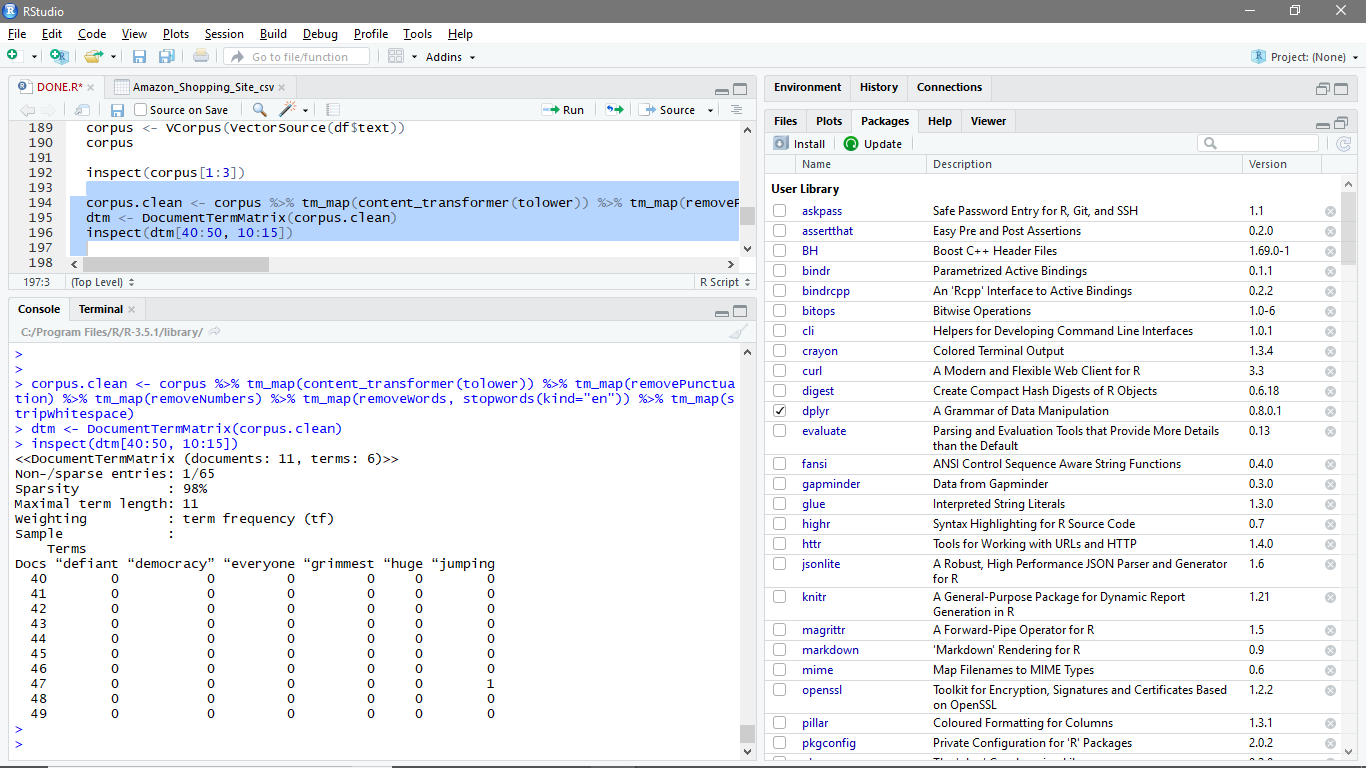
We clean up the corpus by eliminating numbers, punctuation, and white space and by converting to lowercase. In addition, we discard common stop words, such as “his”, “our”, “hadn’t”, couldn’t“, etc.

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

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)

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

ii. Partitioning the Data

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

df.train <- df[1:1200,]

df.test <- df[1201:1554,]

dtm.train <- dtm[1:1200,]

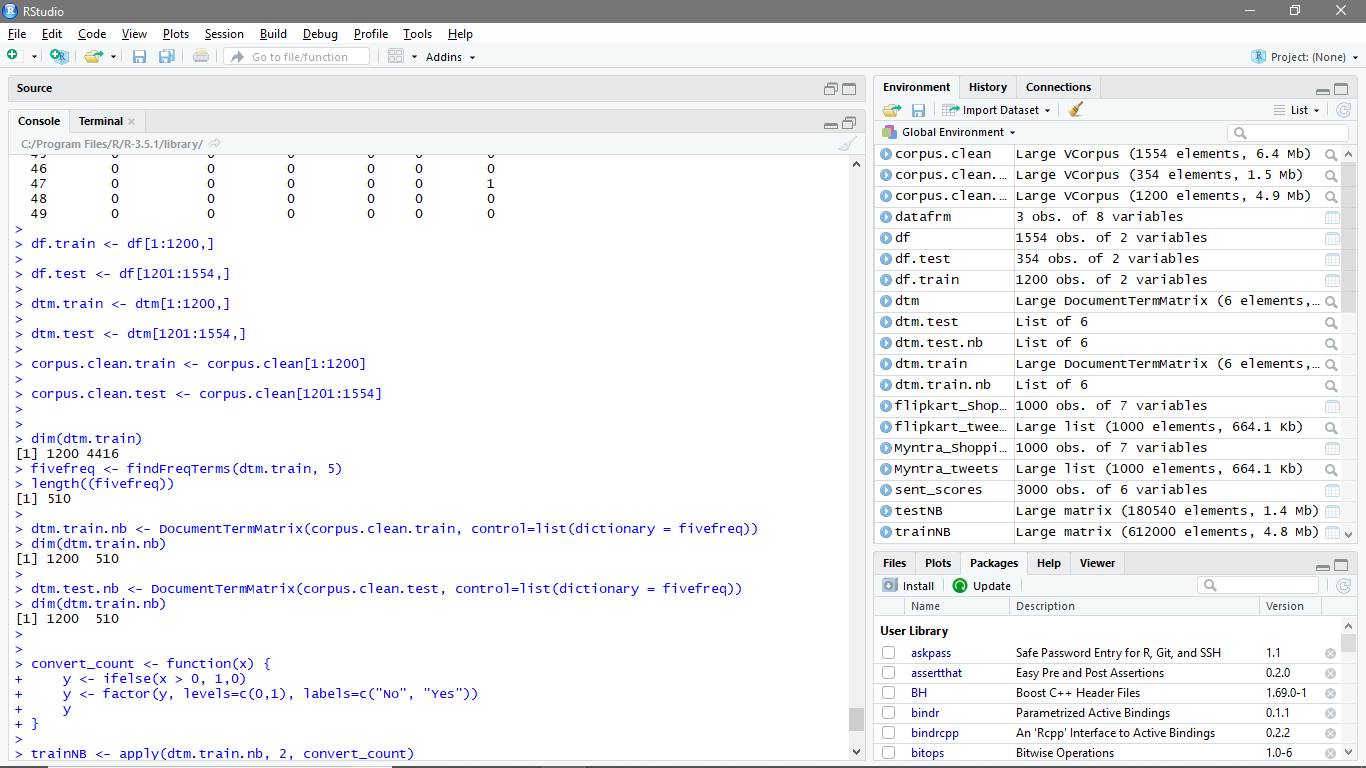
dtm.test <- dtm[1201:1554,]

corpus.clean.train <- corpus.clean[1:1200]

corpus.clean.test <- corpus.clean[1201:1554]

Feature Selection:

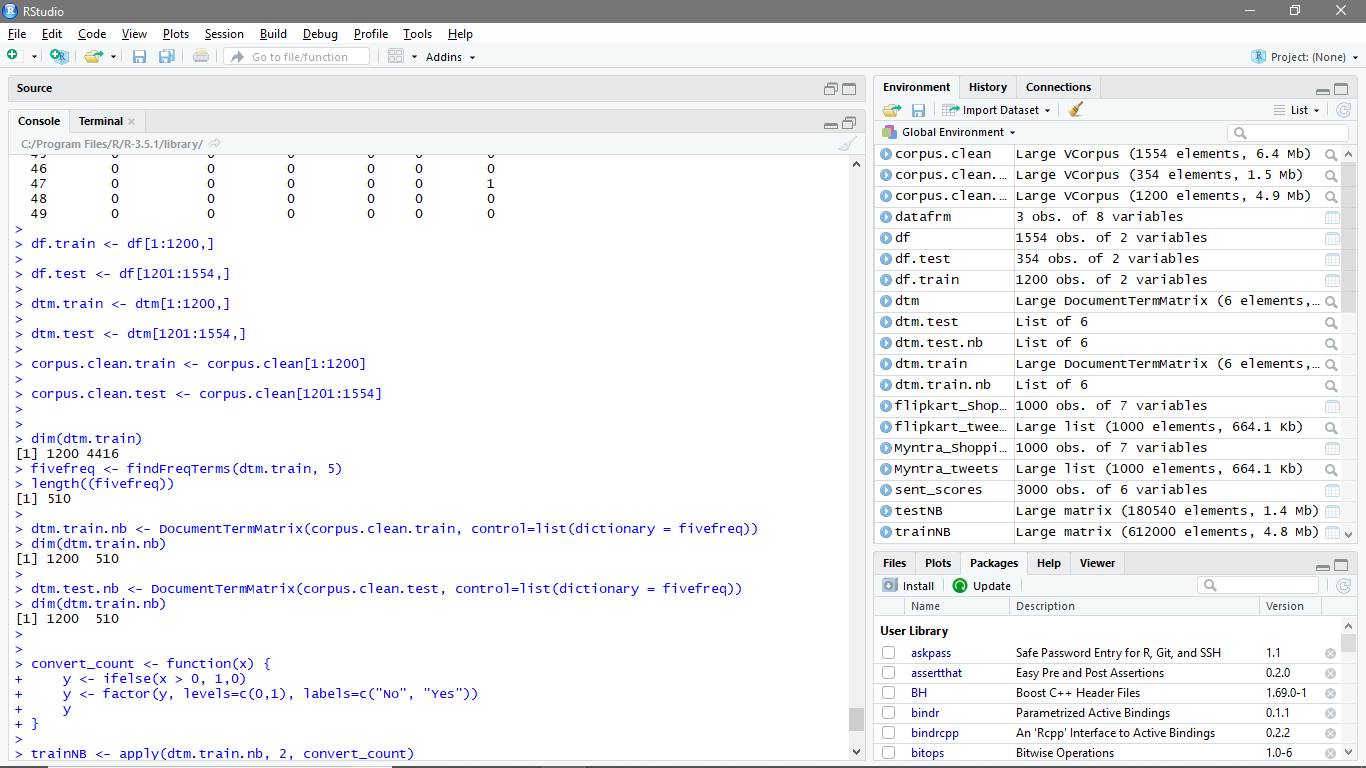
dim(dtm.train)



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.

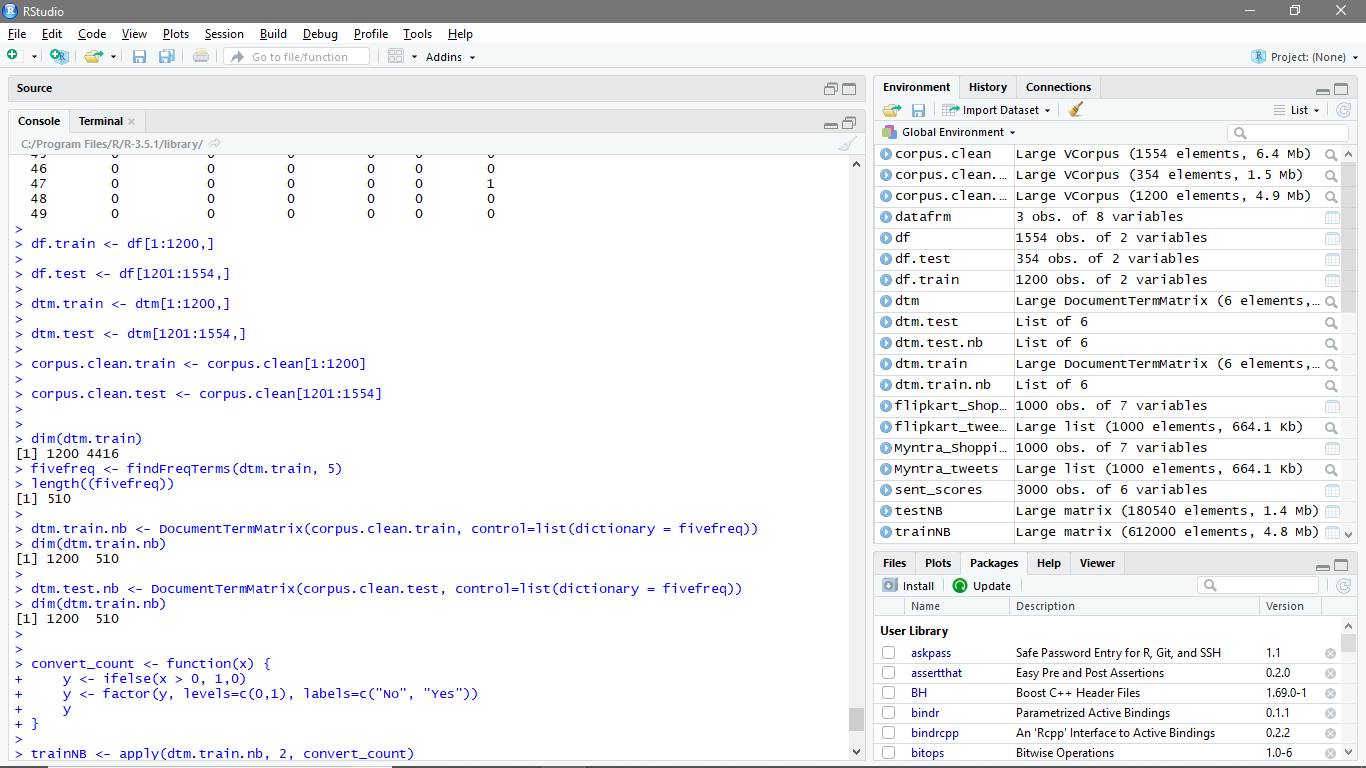
fivefreq <- findFreqTerms(dtm.train, 5)

length((fivefreq))



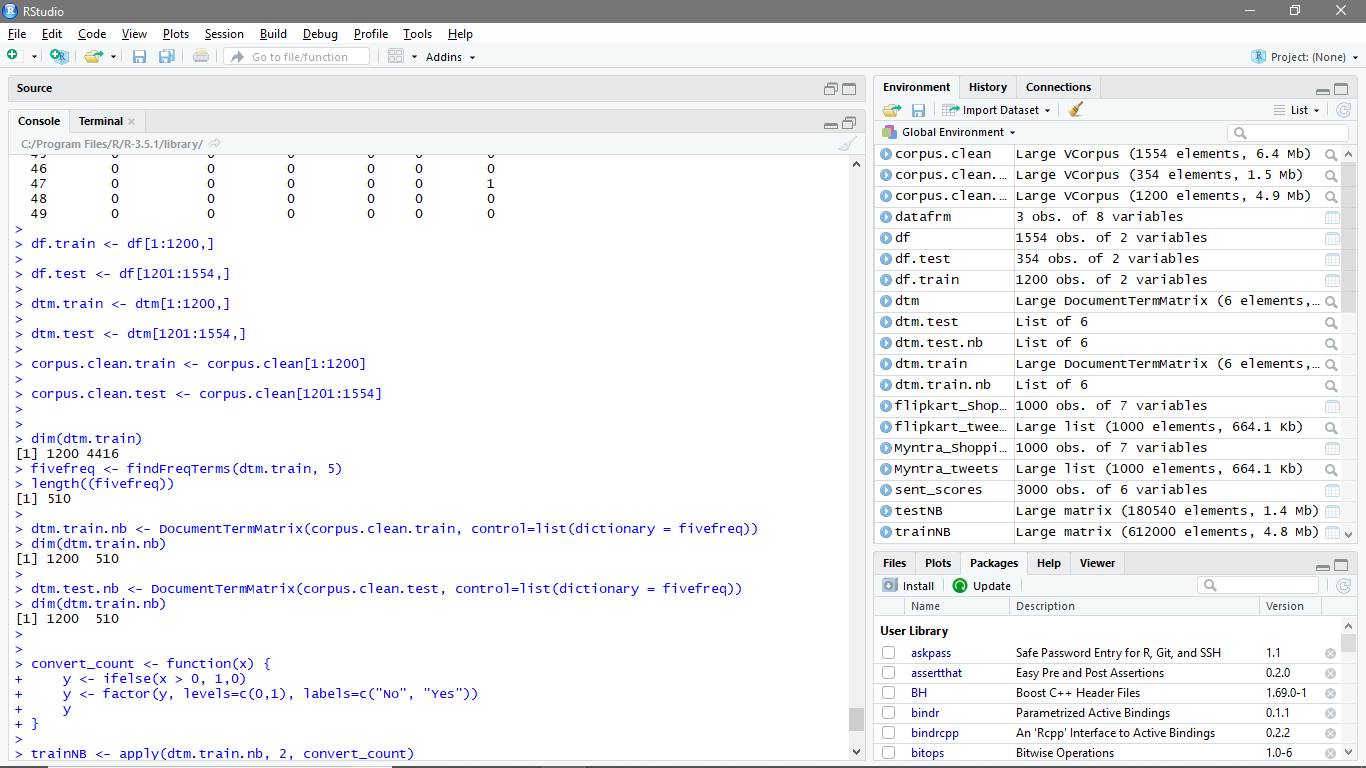
dtm.train.nb <- DocumentTermMatrix(corpus.clean.train, control=list(dictionary = fivefreq))

dim(dtm.train.nb)



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

dim(dtm.train.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.

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:

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

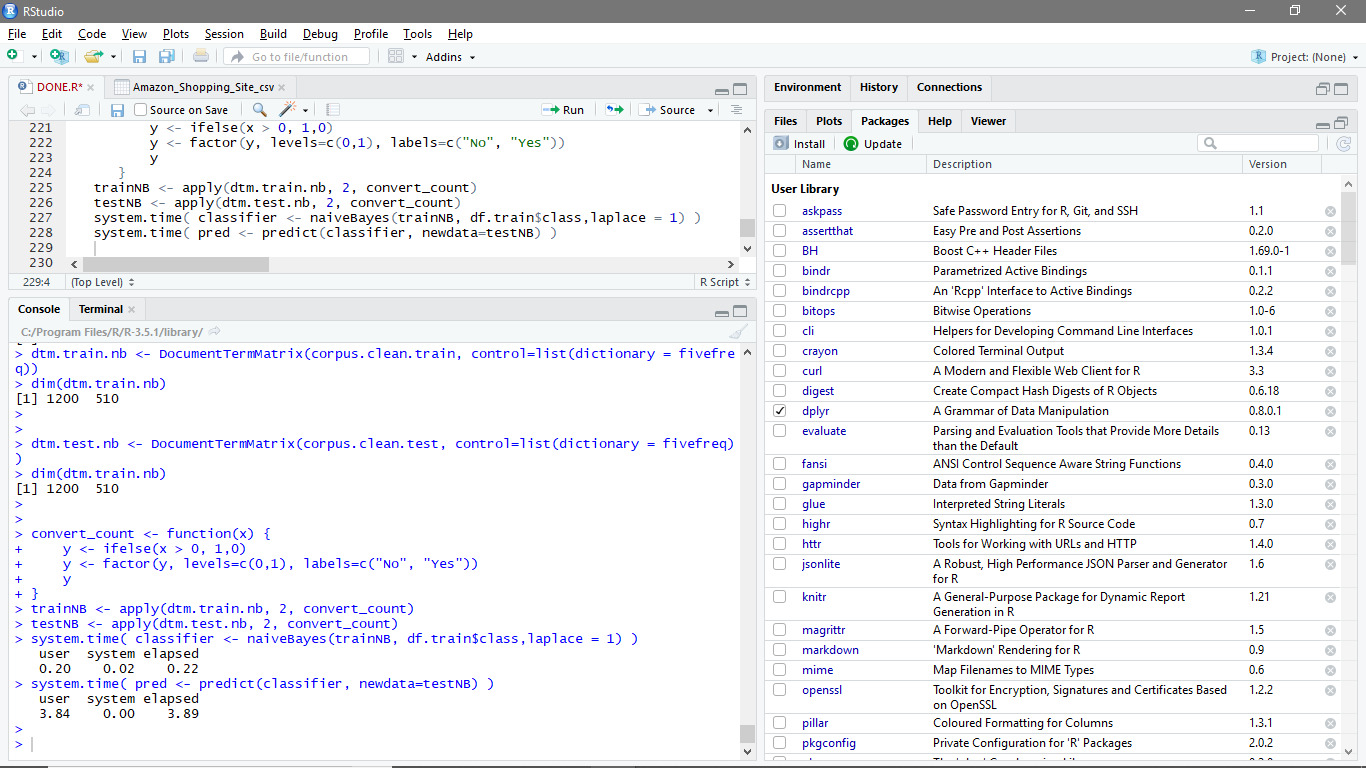
testNB <- apply(dtm.test.nb, 2, convert\_count)

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.

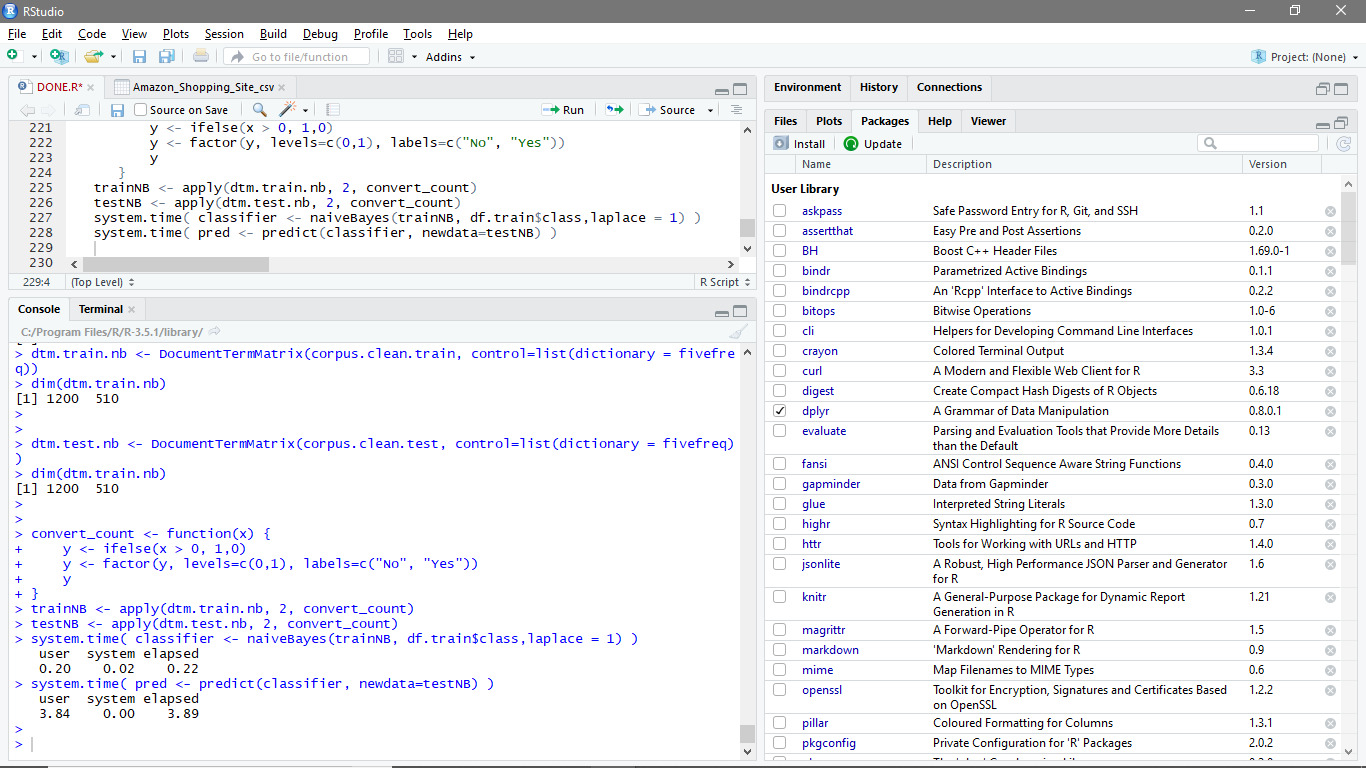
system.time( classifier <- naiveBayes(trainNB, df.train$class,laplace = 1) )



iv. Testing the Predictions

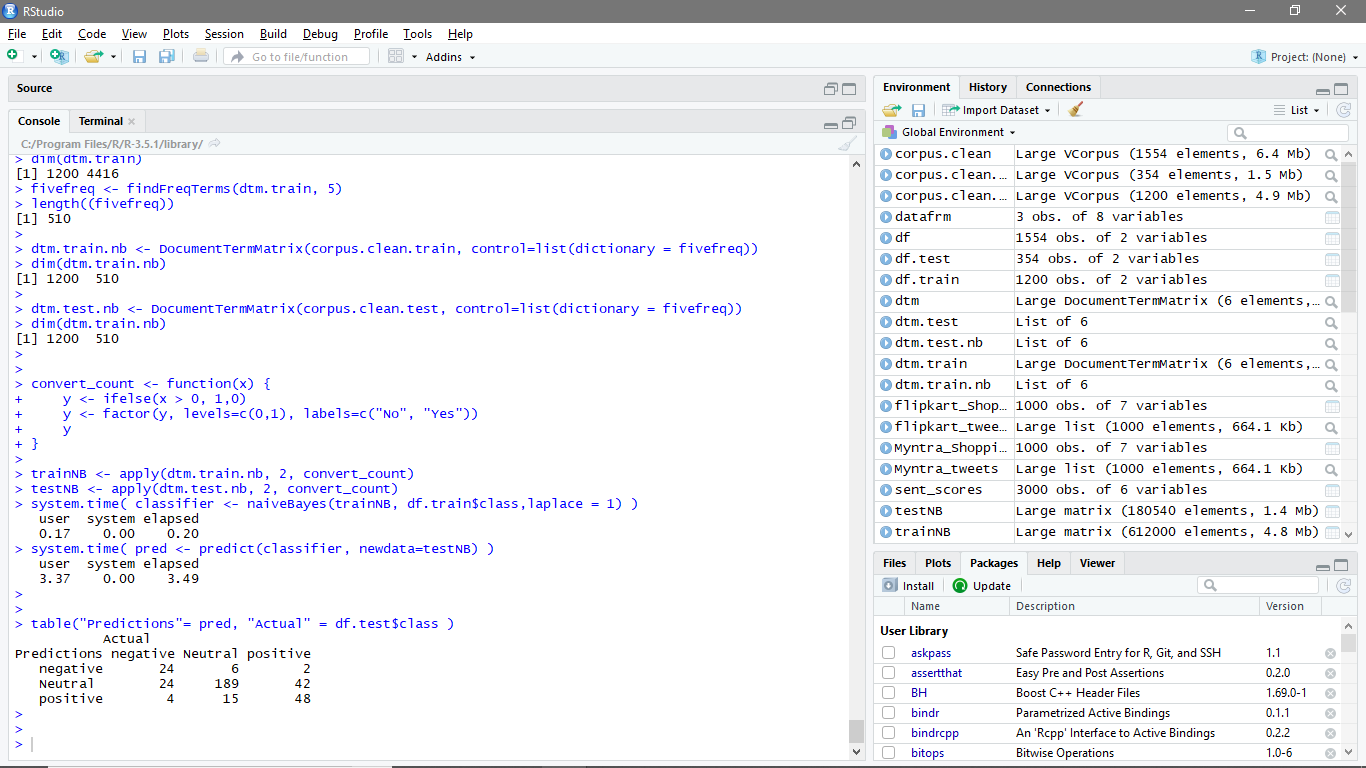
Use the NB classifier we built to make predictions on the test set:

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 )

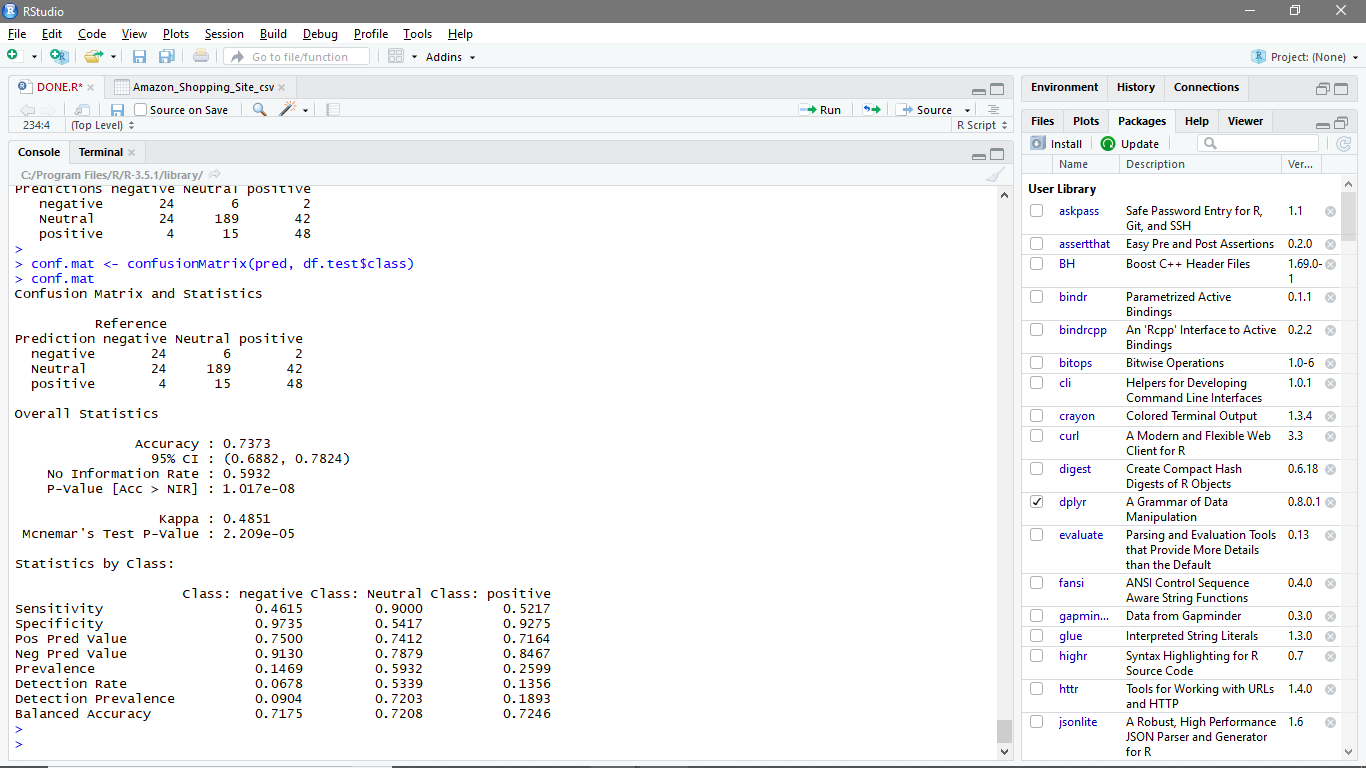


v. Confusion Matrix

Prepare the confusion matrix:

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

conf.mat



Finding and Conclusion

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.

Bibliography and

Reference:

<http://www.cfilt.iitb.ac.in/resources/surveys/SA-Literature%20Survey-2012-Subhabratam.pdf>

<https://www.ethz.ch/content/dam/ethz/special-interest/gess/computational-social-science-dam/documents/education/Spring2017/ML/Sentiment.pdf>

<https://www.cs.uic.edu/~liub/FBS/SentimentAnalysis-and-OpinionMining.pdf>

<https://www.researchgate.net/profile/Yelena_Mejova/publication/264840229_Sentiment_Analysis_An_Overview/links/590ad68e0f7e9b1d0823eff2/Sentiment-Analysis-An-Overview.pdf?origin=publication_detail>

<http://rcciit.org/students_projects/projects/it/2018/GR33.pdf>

<https://www.digitalvidya.com/blog/twitter-sentiment-analysis-introduction-and-techniques/>