Customer Engagement and Brand Perception of Indian Ecommerce- A Social Media Approach

PGPBA-BI Capstone Project Report

Submitted by:	**Anshuman Biswal , Akash Alex, Mithun Babu, Mohamed Habibulla, Srinivas K**
Group No:	**10**
Batch:	**PGPBA-BI 2016**
Project Mentor:	**Rajesh Jakhotia**
Program Director:	**Dr. P K Vishwanathan**

Acknowledgements

We wish to place on record our deep appreciation for the guidance and help provided to us by our Mentor Mr. Rajesh Jakhotia from Great Lakes Institute of Management for guiding us in completing this project on time.

We would also like to place on record our appreciation for the guidance provided by Dr. P K Vishwanathan for giving us valuable feedback and being a source of inspiration in helping us to work on this project.

We certify that the work done by us for conceptualizing and completing this project is original and authentic.

Date: 19 November 2016

Place: Bangalore

Anshuman Biswal
Akash Alex
Mithun Babu
Mohamed Habibulla
Srinivas K

GREAT LAKES INSTITUTE OF MANAGEMENT

Postgraduate Program Business Analytics And Business Intelligence

Certificate

This is to certify that the PGPBA Capstone Project titled "Customer Engagement and Brand Perception of Indian Ecommerce- A Social Media Approach" is a bonafide record of the Project work carried out by Anshuman Biswal, Akash Alex, Mithun Babu, Mohamed Habibulla and Srinivas K in fulfilment of requirements for the award of PGPBA-BI in Great Lakes Institute Of Management .

January 2016.

Rajesh Jakhotia	*Dr. P K Vishwanathan*
Project Mentor	*Program Director*

Date: December 12, 2016

Place – Bangalore

Declaration

Customer Engagement and Brand Perception of Indian Ecommerce -

A Social Media Approach

The Project report is submitted in partial fulfilment of academic requirements for PGPBA-BI in Great Lakes Institute Of Management. This report is a result of our analysis of twitter tweets of various Indian ecommerce brands. All sections of the text and results, which has been obtained from other sources, are fully referenced. We understand that cheating and plagiarism constitute a breach of institute regulations and will be dealt with accordingly.

Signature:

Name of the Student:	**Anshuman Biswal, Akash Alex, Mithun Babu, Mohamed Habibulla and Srinivas K**
Group No:	**10**
Batch:	**January 2016**
Date:	**December 12, 2016**

Abstract

Customer engagement is a connection between a business communication and an external stakeholders.The term can also be used to define customer to customer correspondence regarding a communication, product, service, or brand. (Ryan & Jones, 2011).In the past customer engagement has been generated irresolutely through television, radio, media, outdoor advertising, and various other touchpoints ideally during peak and/or high trafficked allocations. In more recent times the internet has significantly enhanced the processes of customer engagement, in particular, the way in which it can now be measured in different ways on different levels of engagement. Social media channels have played a major role in this.

As per figure 1 there are 1.59 billion facebook users and 320 million tweeter active users. 432 out of top 500 retailers have facebook page and 350 out of top 500 retailers use twitter and youtube. We can say that the online retailers are tapping into this goldmine.

This report documents the effort to analyse the customer engagement and the brand perception of indian ecommerce mainly for Amazon india, Flipkart, Snapdeal, Ebay India and Shop Clues. R was used for programming and churning out the data. Analysis is based on the tweets from the twitter for these brands. Data mining was done using R for the twitter analytics. Tableau was used to draw some graphs. Among analytical approach & tools used, Term frequency chart,wordcloud to find the most frequent words used. Word correlation using heat map is done and also word association of highly correlated word is done . Topic modelling is done using Gibbs sampling and K-Means & Hierarchical Clustering is employed on the themes of tweets to have a cluster dendogram. It also has sentiment analysis of tweets based on polarity and also based on emotions.Tableau is used to create word clouds. Microsoft Excel and Word is used to perform spellchecks on extracted wordsTo draw the inferences from the tweets data analytics life cycle was followed starting from data collection,data cleaning,data exploration and data analysis and finally drawing conclusion. Finally this report also recommends the future works and use cases that can be done on predictive analytics using this as the base.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

“53% of people on twitter recommends companies and/or product in their tweets,with 48% of them delivering on their intention to buy the product.” (ROI Research for performance, June 2010)

Customer value management focusses on optimising the value of the customer. It is also known as Customer Value Maximization. It can be described as the upgraded version of loyalty. Instead of simply rewarding its loyal customers, it focuses on how to maximise the value of every customer, at their every interaction with the company. It is the way of evaluating individual customers in term of their overall value and profitability and increasing it. It is a process which refines and leverages the benefits of Customer Relationship Management. It includes customer identification, advanced data modelling and customer scoring, contact management, campaign management. CVM relies on customer Pay Back period, budget rebalancing, tailored customer rewards, and cross and up-selling campaigns. ("ICME - Publications Telecom & Media" 2016)

Only understanding the CVM is not sufficient for business success. With the introduction of new technologies and changes in customer expectations, experiences and behaviour, firms need to look to the future. The three emerging perspectives on CVM: (1) managing customer engagement, (2) managing customer networks, and (3) managing the customer experience.

In an increasingly networked society in which customers can interact easily with other customers and firms through social networks and other new media, no transactional customer behaviour is likely to become more important (Hennig-Thurau et al. 2010). According to a recent study almost one in every 5 minutes spent their time online and reaches 82% of the entire world’s online population, which represents an audience of nearly 1.2 billion internet users. The industry has 94% of all businesses continuously pursue engaging their customer base on at least one of the “Big Three” social media platforms: Facebook, Twitter and Google+.

With the introduction of social media, online stores etc. the ecommerce experience has changed dramatically. As a result, businesses need to ensure they are ready for those changes and are prepared to focus on customer engagement to drive long-term success ("5 Ways to Increase Mobile Marketing Conversion Rates" 2014).

Online consumers interacts across different channels - retail stores, online stores, mobile stores and social media and uses his perception and experience to choose the right product from the right online shopping site. Why they use the online site more than the brick and mortar? Its because of the convenience. They get the right product from a variety of choice at a single place which are rightly and competiatively priced.

Ecommerce industries bridge this gap by providing right products which are rightly priced and also provide the recommendations based on their buying behaviour or past history. They study the purchase sentiments from clicks, analyse the positive or negative sentiments from the comments made by their customers in various social media channels.

As now a days the customers are more moving towards the ecommerce sites it became necessary for the businesses to study the customer engagement rate and the brand perception of their brands. Customer engagement is the depth of the relationship a customer has with a brand. Every interaction should affirm customer's decision to make a brand part of their life. Brand perception is the extent to which customers are able to recall or recognise a brand.

This report is segregated as follows. Chapter two has the data mining or collection technique and data cleaning in R. Chapter 3 has the data exploration. Chapter 4 has the analysis and inferences drawn using the cleaned data for the project objectives. This report ends with the conclusion and future recommendations.

Chapter 2

Background Theory

There is no single definition for the concept of customer engagement. Though social media platforms usually present engagement in terms of level of interaction between the brand and the fan base, it is in fact a more complicated concept comprising three distinct dimensions: behavioral engagement(actions), emotional engagement (feelings) and cognitive engagement (thoughts). These three dimensions are also expected to play different roles in the overall process of engaging a customer with a brand on social media. It is believed that a number of different factors may be driving the customers to engage with a brand on social media and a number of different outcomes can be expected dependent on the type of engagement., the concept of customer engagement has become commonplace among marketers. More and more resources are being invested in social media and brands from all over the world continue to grow their online fan base. However, if we take a better look at the engagement metrics, such as “Talking about it” on Facebook, we will see that even the most popular brands struggle to engage with more than 1% of their entire fan base on Facebook. ("Do Brands Engage Their Customers Through Social Media? - Market Community" 2016).

Social media not only makes it easier to create a memorable brand, which is key to success for customer engagement, but also makes it cost-effective ("Social Media and Mobile Marketing: A Marriage Made in Heaven - Launch & Hustle" 2014). The number of social media sites offering the ability to make purchases directly from their platform is increasing. Internet Retailer recently reported that Tumblr has added a number of action buttons to posts, including a Buy button. (Stambor 2016).

How important is social media when it comes to the consumer purchasing process?

Business2Community reports that 63 percent of millennials state they actually stay updated on brands via social networks. Another 46 percent stated that they rely on social media when making online purchases. ("104 Fascinating Social Media and Marketing Statistics for 2014 (And 2015)" 2016). Clearly, it has now become vital for brands to ensure they not only have a presence on the major social media networks but that they also work consistently toward engaging their customer base on those networks. The rationale behind this assertion is the prevailing conception of customer engagement as a way to create deeper and more lasting customer brand relationships (Kumar et al., 2010). Almost every brand is in the mainstream social media platforms such as Facebook, Twitter or Google+ and some others are in Instagram, Pinterest or Foursquare. There are multiple different ways and tools that businesses can use in order to engage their customers. One recent practitioner study of the most popular brands on Facebook has discovered that less than 5% of brands were able to attract repeated fan visits to their page within a 30-day period, meaning that under one in 20 fans in a month chose to return to the brand page more than once (WARC, 2012b). Brands are willing to take a leap of faith building on the core premise of social media paradigm, which suggests that brands need to engage their customers in order to sustain growth.

Problem statement and objective

Brodie et al. (2011a) suggest that under different circumstances the importance of the cognitive, emotional, and behavioural customer engagement dimensions may vary. Therefore, it is likely that customer engagement in different contexts, such as online versus offline environments, would manifest in different expressions. The behavioural measures of engagement currently available on online social media platforms such as number of fans, repeated visits or interactions with the brand page provide little information about the returns to be expected (Nelson-Field & Taylor, 2012)

The main Objective of this project is to bridge this gap by conceptualizing customer brand engagement on online social media platforms and answering the following important questions:

- Understand the level of customer engagement of a business or a brand from the data of twitter tweets.
- Measuring the organization's share of voice and brand perception compared with the competition.
- Measuring metrics like reach of a brand, responsiveness to customers, customers interest in the content/campaigns etc. to be mined and compare these metrics against the competitions (companies/brand in the same business)
- Quantify the brand's reach, influence and sentiment; and
- Understanding sentiment drivers.
- Finally give insights and recommendations about things like:

a> influences (social media audience who have a great network or reach)

b>in which metrics and areas a brand is better or worse than the competitor

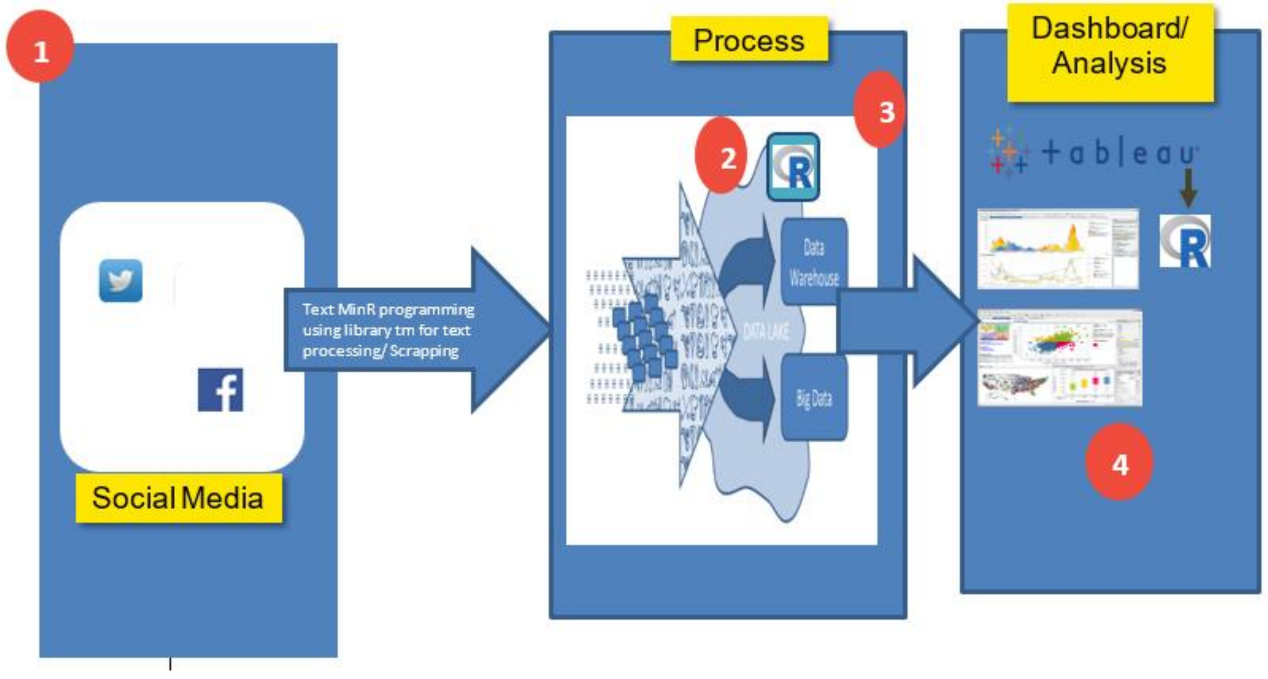
d> sentiment analysis of content as how the audience respond to different types of contents.

Scope And Data Sources

69 percent of online shoppers use social media sites. Of that group, 56 percent choose to proactively interact with companies on social sites by "liking" or "following" at least one retailer, according to Foresee Results. One-fifth of social media users interact with 11 or more retailers on social sites. Social networks have become an integral part of retailers' marketing strategies because retailers can use them to engage with customers, and learn how to serve their customers better. Social media analytics is helping retailers (Improving Revenue and Customer Engagement with Social Media Analytics 2016)

This project will focus on online ecommerce like Amazon India, Flipkart, Snapdeal, Ebay India,Shop Clues. Text mining will be done from twitter to collect data and use appropriate statistical tools and technique to meet the project objectives.

Analytical Approach



1
- Social media data extracted using R programming using library tm for text processing

2
- Structure and unstructured data in data lake (db/files etc)

3
- R programming using library tm for text processing

4
- R/SAS/Tableau dashboard for exploratory analysis.
- Tableau integrated with R for clustering and predictive modelling

Figure 2. Analytic process for twitter analytics

Process

Obtaining twitter data

To begin ingesting social media data from Twitter, you will need a developer account on Twitter. Once you have a Twitter account, return to that page and enter your username and password. Now, simply click on the Create New Application button and enter the requested information. Note that these inputs are neither important nor binding. You simply need to provide a name, description, and website (even just a personal blog) in the required fields. Once finished, you should see a page with a lot of information about your application. Included here is a section called OAuth settings. These are crucial in helping you authenticate your application with Twitter, thus allowing you to mine tweets. More specifically, these bits of information will authenticate you with the Twitter application programming interface (API). You'll want to copy the consumer key, consumer secret, access token and access token secret.

You can download the twitteR package and load it into your R session as follows

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(data.table)
```

Setting up the handshake between twitter app and R

The handshake will be done using OAUTH mechanism. For this the consumer secret key, consumer secret, auth token and auth token secret was created using the twitter app management portal and then using setup_twitter_oauth () the connection was established to pull the data.

```
consumer_key = 'vitqhIlWGWfRUOUP8JIixb7OB'
consumer_secret = 'gJrNUmTl99eU3bTpUQTnBhO1zs3TST1qluwSjyozfkTm4FOGjW'
access_token = '3306555647-NpzZt0Af4T06E5ea9EWlVQeGRfE4wF4fvQBOBMg'
access_token_secret = '5Y4ojfc0LO8UtHmjrGVdhMlsF9AjkjWIxluBmofH45v2y'
setup_twitter_oauth(consumer_key,consumer_secret,access_token, access_token_secret)
```

Data collection

Data collection was done using the twitter api , searchTwitteR and the tweets were collected for the time frame between 2016-01-01 to 2016-10-28 .

```
amazon_handle<-searchTwitteR("@amazonIN",n = 3000,lang = "en",since = '2016-01-01',until = '2016-10-28')
```

It was then converted to data frame using do.call and lapply and then the tweets were saved as csv file for future use.

```
amazon_handle.df <- do.call(rbind, lapply(amazon_handle, as.data.frame))
save(amazon_handle, file = "amazon_handle.RData")
write.csv(amazon_handle.df,"amazon_handle.csv")
```

The data was collected for the search terms: @amazonIN , #amazonIN, amazonIN for amazon tweets, @Flipkart, @flipkartsupport, #flipkart for flipkart tweets, @snapdeal, #snapdeal and snapdeal for snapdeal tweets and @ebayindia, for ebay tweets and @shopclues for shop clues tweets

Data cleaning

All the data extracted from the tweeter may not be needed by us and so cleaning is needed.

Remove character string between < >

```
amazon_combined.df$text=genX(amazon_combined.df$text, " <", ">")
```

Text data, such as tweets, comes with little structure compared to spreadsheets and other typical types of data. One very useful way to impose some structure on text data is to turn it into a document-term matrix. This is a matrix where each row represents a document and each term is represented as a column. Each element in the matrix represents the number of times a particular term (column) appears in a particular document (row). Put differently, the i, jth element counts the number of times the term j appears in the document i. Document-term matrices get their length from the number of input documents and their width from the number of unique words used in the collection of documents, which is often called a corpus.

So the document corpus was created and was converted to lower case.

# Create document corpus with tweet text

```
amazon_combined.corpus<- Corpus(VectorSource(amazon_combined.df$text))
#####convert to Lowercase
amazon_combined.corpus <- tm_map(amazon_combined.corpus, content_transformer(stri_trans_tolower))
```

The URL links were removed.

```
#####Remove the links (URLs)
removeURL <- function(x) gsub("http[^[:space:]]*", "", x)
amazon_combined.corpus <- tm_map(amazon_combined.corpus, content_transformer(removeURL))
```

And then except English language and space rest all were removed.

```
removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)
amazon_combined.corpus <- tm_map(amazon_combined.corpus, content_transformer(removeNumPunct))
```

Stop words, single letter words and extra white spaces were removed. This makes the basic cleaning. Similar cleaning was done for tweets of other brands too.

```
#####Remove anything except the english language and space
removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)
amazon_combined.corpus <- tm_map(amazon_combined.corpus, content_transformer(removeNumPunct))
#####Remove Stopwords
myStopWords<- c((stopwords('english')),c("rt", "use", "used", "via", "amp","should","go","then","what","when","where","than"))
```

Further cleaning was done by stemming the words. Documents are going to use different forms of a word, such as organize, organizes, and organizing. Additionally, there are families of derivationally related words with similar meanings, such as democracy, democratic, and democratization. In many situations, it seems as if it would be useful for a search for one of these words to return documents that contain another word in the set. The goal of stemming is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For instance: car, cars, car's, cars' = car.

```
#####Stem words in the corpus
amazon_combined.corpus=tm_map(amazon_combined.corpus, stemDocument)
writeLines(strwrap(amazon_combined.corpus[[250]]$content,60))
```

Then inspect the words after stemming

```
for (i in c(1:100, 6109)) {
  cat(paste0("[", i, "] "))
  writeLines(strwrap(as.character(amazon_combined.corpus[[i]]), 60) )
}
```

Then after stemming the text was completed and was converted to corpus again

```
#####Function to correct/complete the text after stemming
stemCompletion2 <- function(x, dictionary) {
  x <- unlist(strsplit(as.character(x), " "))
  # Unexpectedly, stemCompletion completes an empty string to
  # a word in dictionary. Remove empty string to avoid above issue.
  x <- x[x != ""]
  x <- stemCompletion(x, dictionary=dictionary)
  x <- paste(x, sep="", collapse=" ")
  PlainTextDocument(stripWhitespace(x))
}
```

This process was done for all the tweets.

Chapter 3

Data Exploration

Most frequent terms – Frequency plot and Word cloud

Finding the most frequent words (for example, n, where n is the minimum frequency) will help us build out our most frequent library of words in order to refine our analyses and learn more about our corpus. This is the lexicon approach. A word such as strong would appear to be a good predictor of positive sentiment. Exploring the corpus itself can allow our lexicon to grow inductively, allowing us to augment our domain-specific dictionary or build one from a general purpose dictionary. The any function returns logical vectors if at least one of the values is true. Using the results from the frequent words, we can begin to test for the presence of words from within our corpus and our lexicons as follows: findFreqTerms(tdm, lowfreq = 200) . Word cloud is used to get to know the most frequent words more easily using wordcloud function.

Frequency plot

Amazon

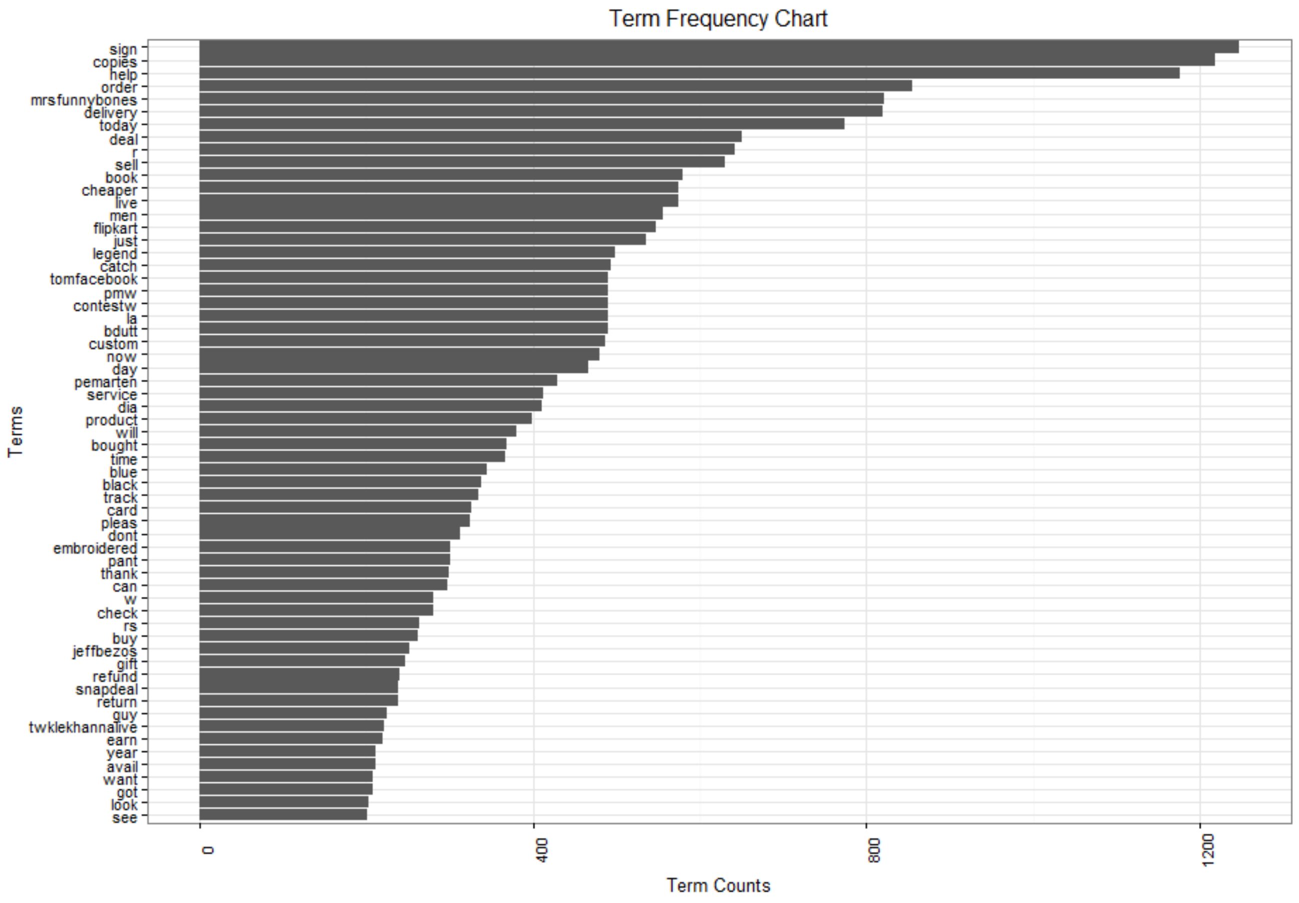


Figure 3.1 Term frequencr chart of terms used in the tweets for the brand Amazon

Inference: Sign, Copies, Help and Order are the most frequent terms from Amazon tweets

Flipkart

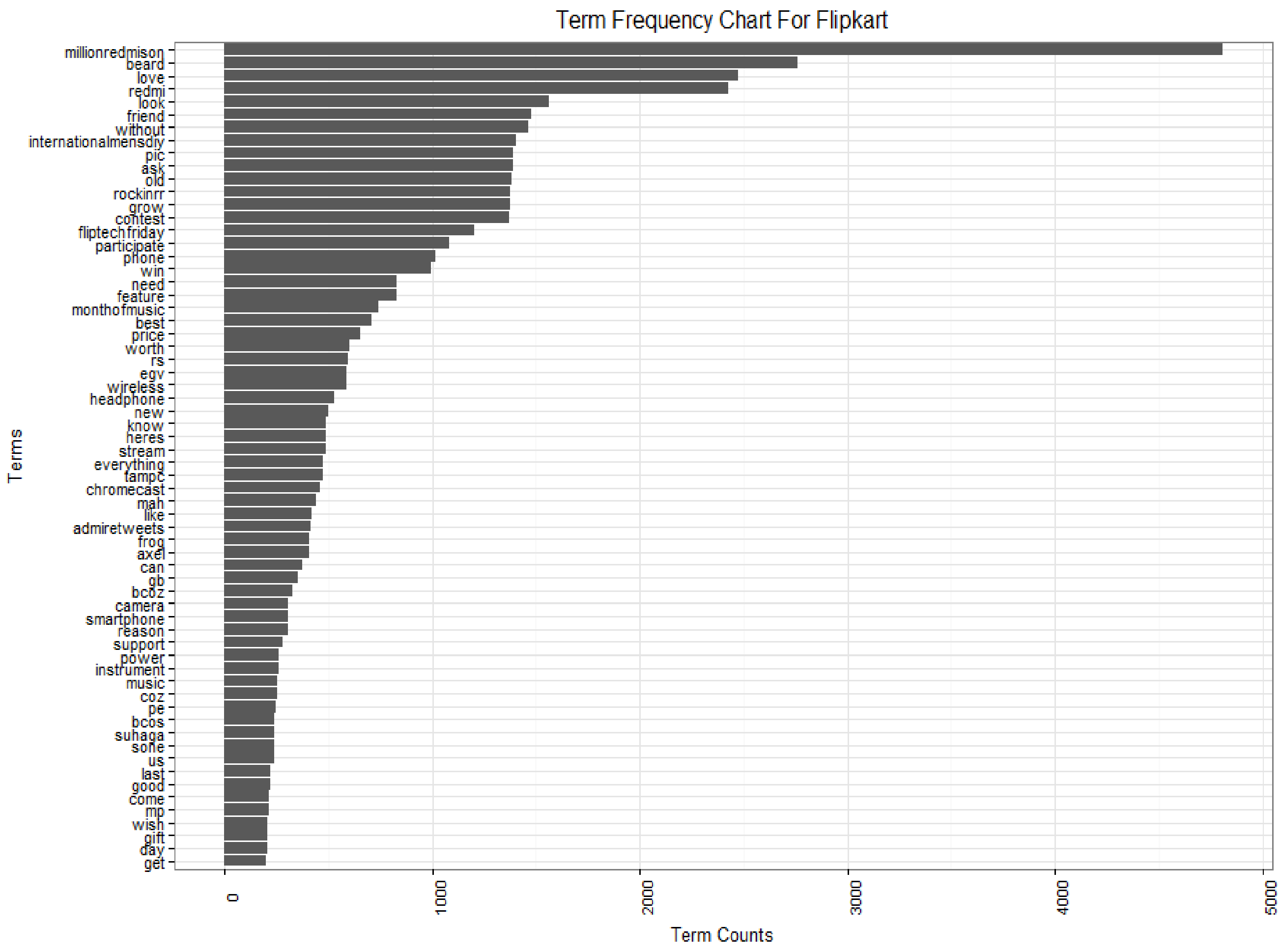


Figure 3.2 Term frequencr chart of terms used in the tweets for the brand Flipkart

Inference: millionredmison, beard, love and redmi are the most frequent terms from flipkart tweets

Snapdeal

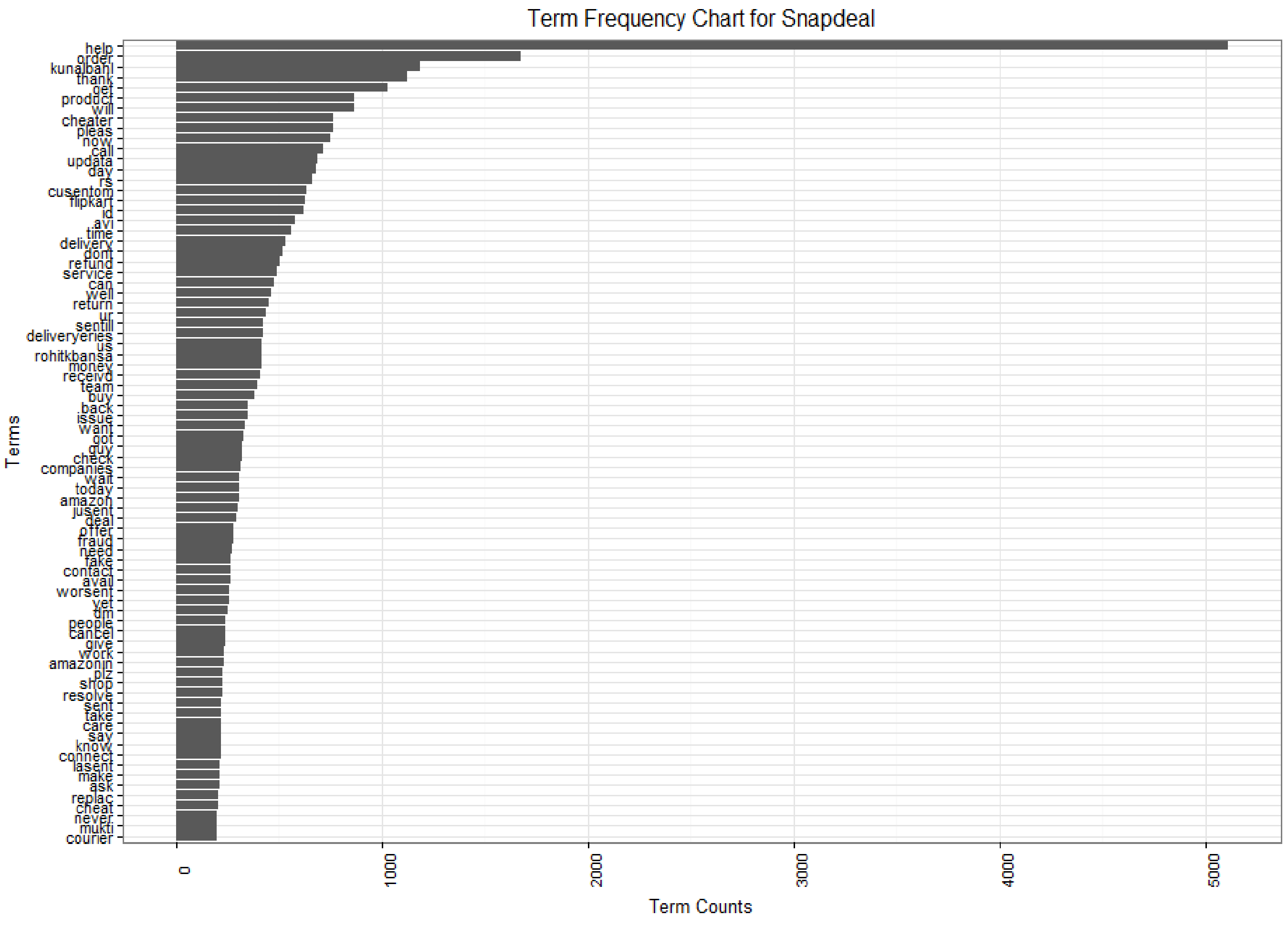


Figure 3.3 Term frequencr chart of terms used in the tweets for the brand Snapdeal

Inference: help, order, thank and get are the most frequent terms from Snapdeal tweets

Ebay

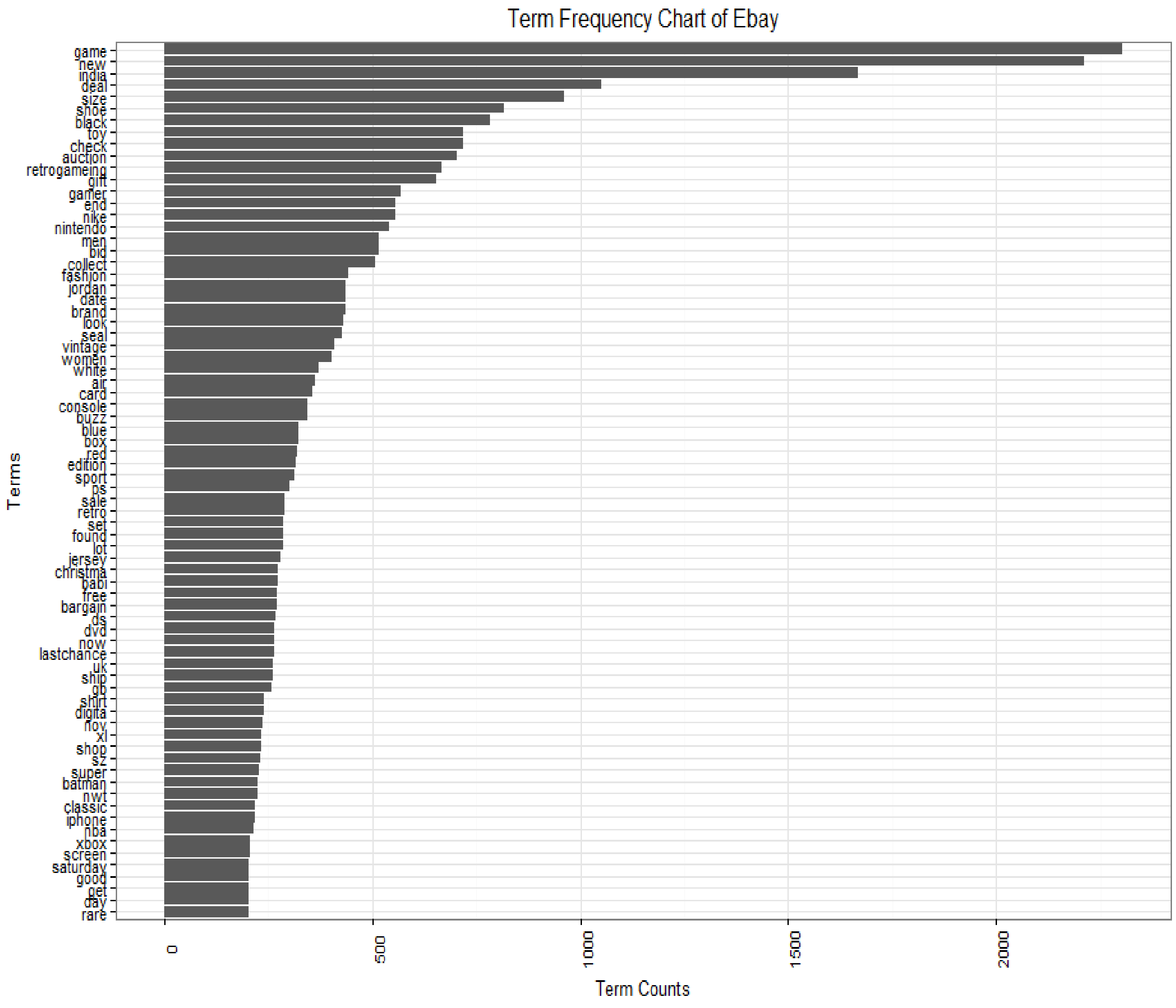


Figure 3.4 Term frequencr chart of terms used in the tweets for the brand Ebay

Inference: game, new, india and deal are the most frequent terms from ebay tweets

Shop Clues

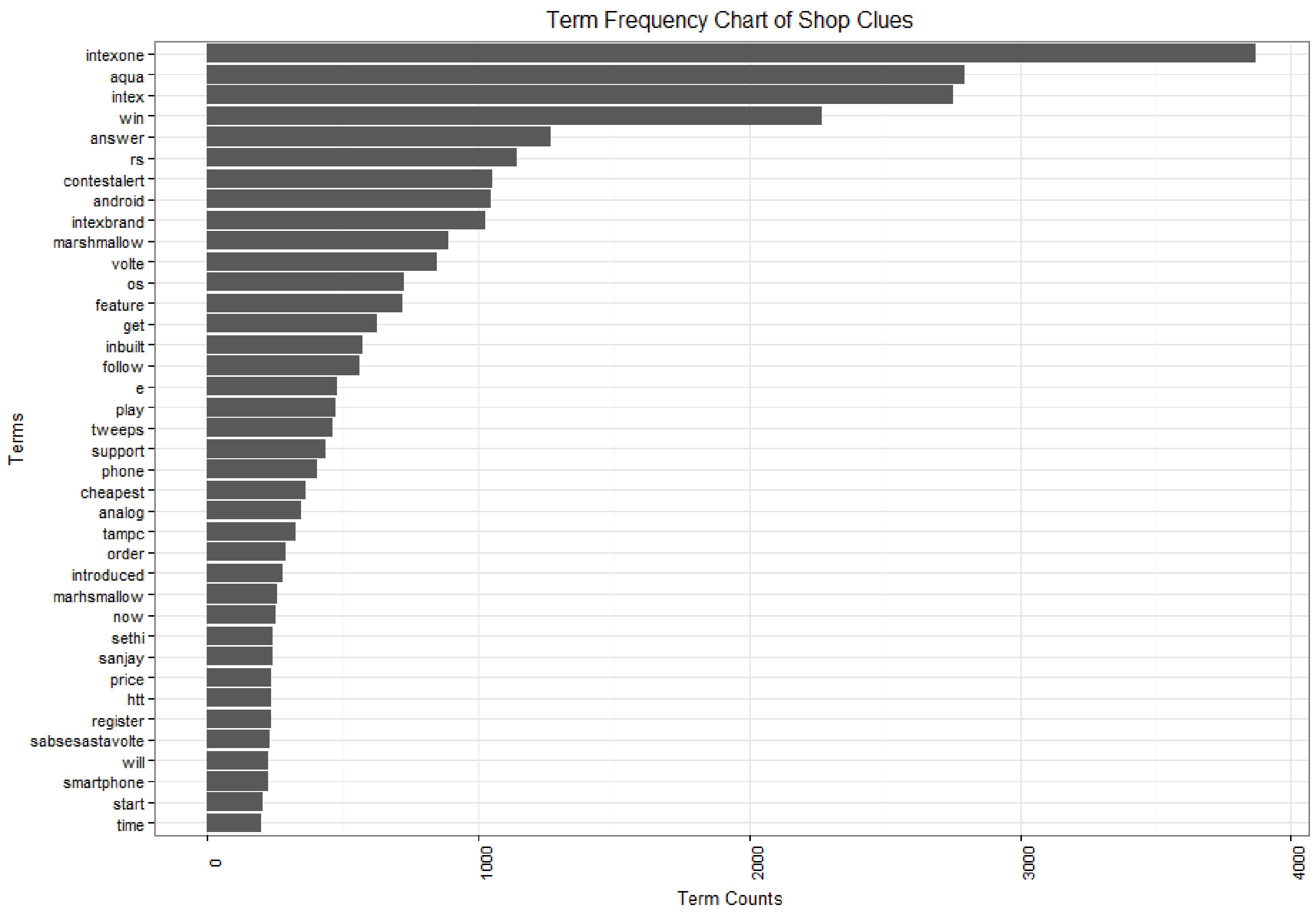


Figure 3.5 Term frequencr chart of terms used in the tweets for the brand Shop Clues

Inference: intexone, aqua, intex and win are the most frequent terms from shopclue tweets

Word Cloud

Amazon

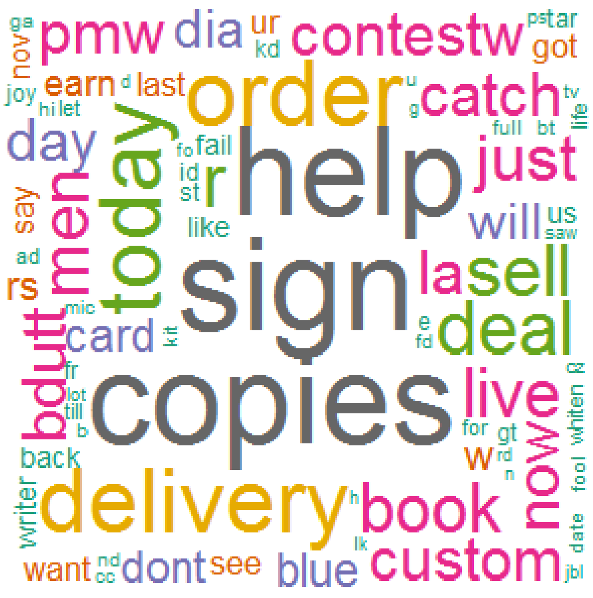


Figure 3.6 Word cloud of terms used in the tweets for the brand Amazon

Flipkart

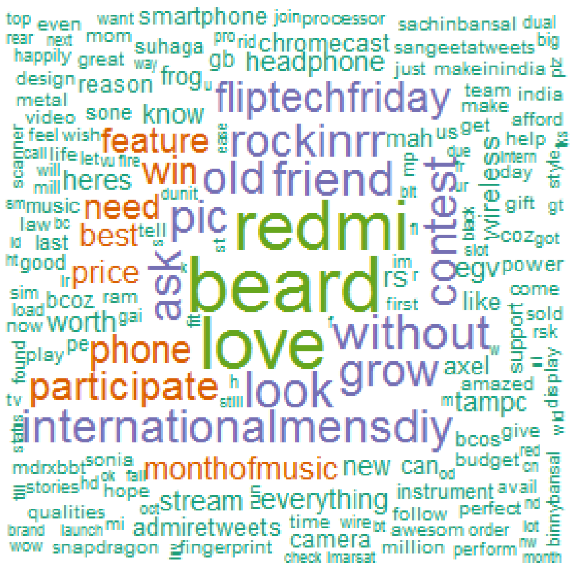


Figure 3.7 Word cloud of terms used in the tweets for the brand Flipkart

Snapdeal

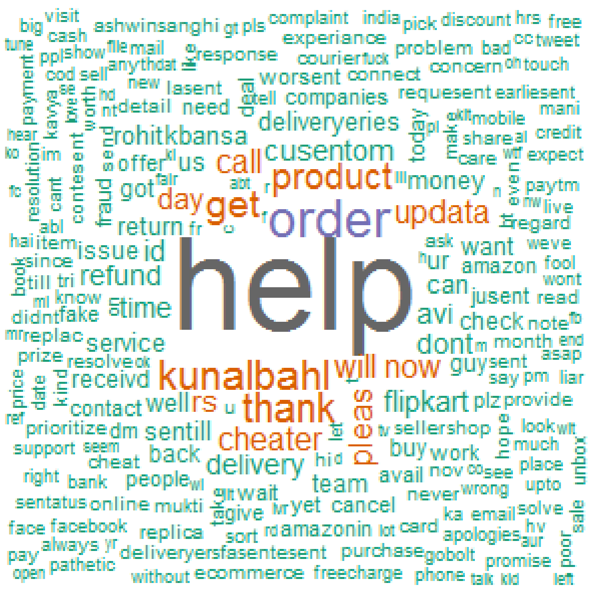


Figure 3.8 Word cloud of terms used in the tweets for the brand Snapdeal

Ebay

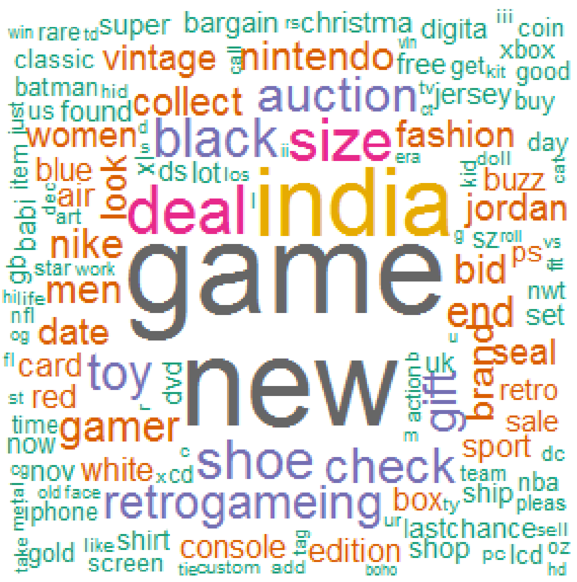


Figure 3.9 Word cloud of terms used in the tweets for the brand Ebay

Shop Clues

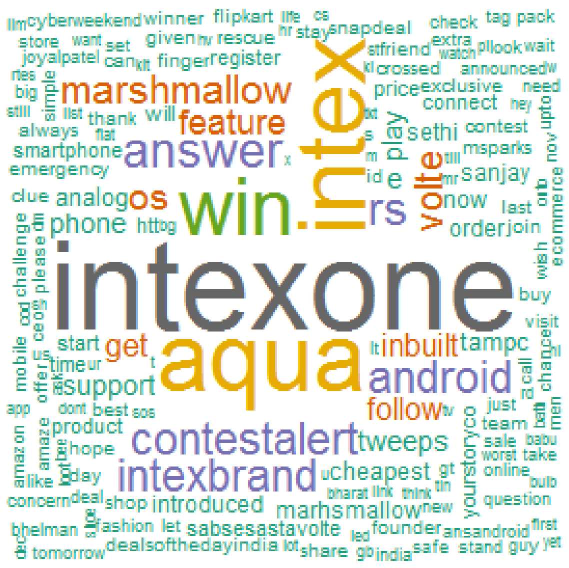


Figure 3.10 Word cloud of terms used in the tweets for the brand Shop clues

Inference

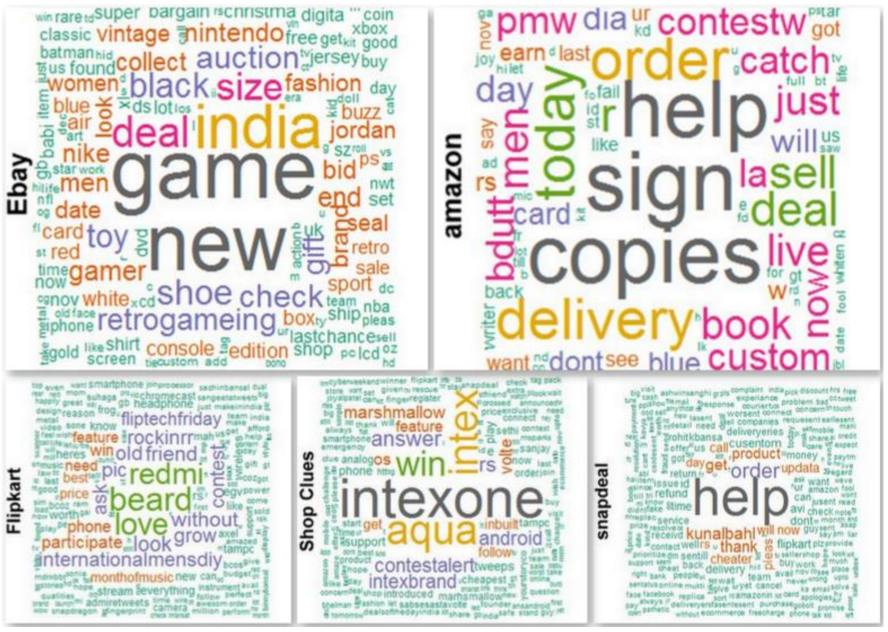


Figure 3.11 Word cloud comapriosn of terms used in the tweets for various brands

For amazon the word help,sign,copies are frequent. For Ebay the word game,new are frequent. For Flipkart the word redmi,beard and love are frequent but the colour is green and size is small which says that there frequency is not that high.For shop clues intex one is frequent and for snapdeal we see the word help as frequent.The word order is seen as frequent terms for some brands collection.

Word Corelation

Certain words are correlated with other. So it would be good to study what all words are correlated with the frequent occurring terms to get a clue on the situation of the business or what customers are speaking on. For this we will use the heat map to visualize the result. To generate the heat map the higly correlated words are found out for a particular word usingly apply_as_df function of package qdap with word_cor as argument. Here the words which are having a correlation factor r of value 0.25 and above is found out and is fed to the qheat function of the qdap package.

Amazon

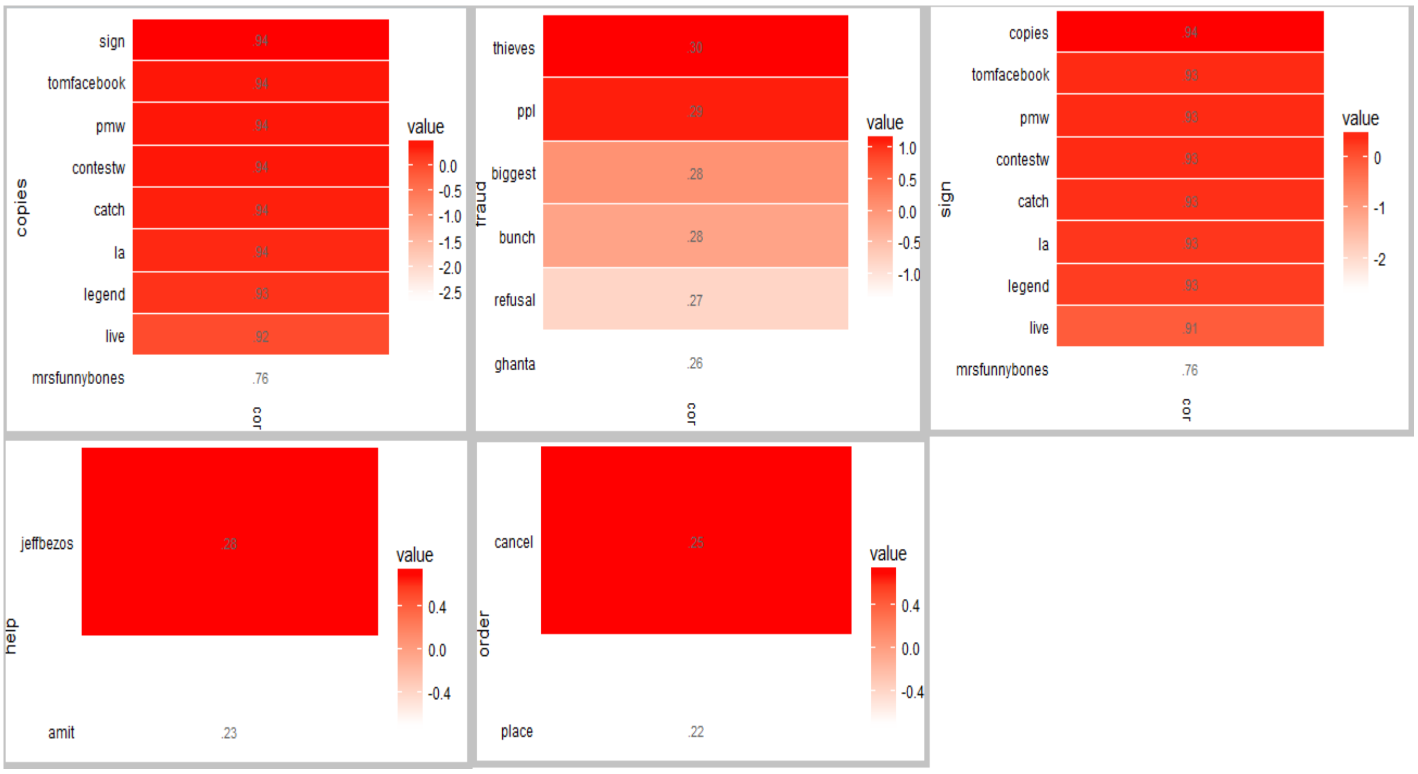


Figure 3.12 Heat map of highly correlated words of Amazon brand

Lets find the tweets of these correlated words

Find messages with high corelated word for Amazon tweets

Help

[1] "help feedback make sense loser money refusal owe yr yr partner fraud jeffbezos"

[2] "milanbarai help valuable custom ask whatever plan ternal men day mensdaynov"

[3] "milanbarai help will celebrate mensdaynov prostate cancer nd killer men fundi"

[4] "help jeffbezos even me lenovo delivery yet won spot fact contest relax yet now worried"

[5] "help romensmith crazyamita jeffbezos plz look bcoz last time play leeco hunt saw mandar tweet"

order

[1] " order mouse work proper return replace product kd help regrd"

[2] "help jeffbezos share issue order nothg reach"

[3] " paid fr piece refurbished shiti custom suck last orderwant"

[4] "help retrnd order almost month back havent recvd refund yetno suprt frm custom srvord id"

[5] "jeffbezos help confirm order cancel day false reason givenprice crease shall pay"

sign

[1] "mrsfunnybones sign copies catch tomfacebook live pmw contestw sign copies legend la"

[2] "jo earn click sign refer and earn "

[3] " dont for twklekhannalive entries w sign copies latest book cont"

[4] "srishtipub grab authorsigned copies latest book last wish ajaypandey exclusive "

[5] "help height fraud guy forged signature hope know illegal "

copies

[1] "mrsfunnybones sign copies catch tomfacebook live pmw contestw sign copies legend la"

[2] "grab discount copies day week halfbakedbeans pustakmandi sale"

[3] " dont for twklekhannalive entries w sign copies latest book cont"

[4] "srishtipub grab authorsigned copies latest book last wish ajaypandey exclusive "

[5] " mrsfunnybones will goa live fb page pm want w sign copies stay tuned twklekhannalive"

fraud

[1] "help feedback make sense loser money refusal owe yr yr partner fraud jeffbezos"

[2] "help havent cancel ityr courier didnt show mark fraud remark refusal custom jeffbezos"

[3] " prime fraud order delivery day refund money now"

[4] "vipgupta flipkartsupport flipkart snapdeal flipkart fraud ecommerce dont buy flipkart harassi"

[5] "flipkartsupport flipkart snapdeal flipkart fraud ecommerce dont buy flipkart harass replace"

Flipkart

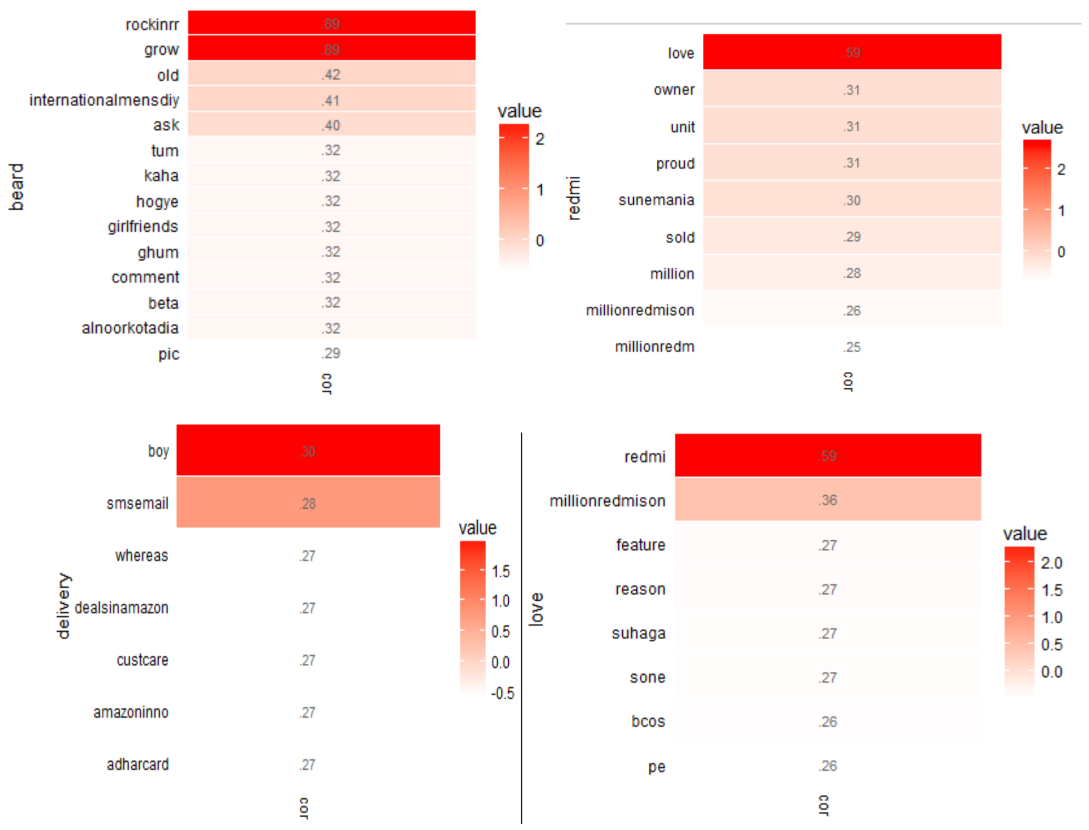


Figure 3.13 Heat map of highly correlated words of Flipkart brand

Find messages with high corelated word for Flipkart tweets

Redmi

[1] "best feature big still light weight sleekmillionredmison"
[2] "heres everything need know participate millionredmison contest"
[3] "anitakhushlani congratulate millionredmison"
[4] "winner millionredmison contest congratulate"
[5] "participate millionredmison contest win egv worth rs tampc"

Beard

[1] "alnoorkotadia dollar time cringed look beard id millionaire internationalmensdiy"
[2] "rockinrr internationalmensdiy friend look old pic without beard ask grow beard"
[3] "beta tum kaha ghum hogye beard comment girlfriends mom internationalmensdiy"
[4] "internationalmensdiy keep lincolnneckbeard mountain man sirlet trim"

[5] "never like beard internationalmensdiy"

Love

[1] "fliptechfriday wireless like love can hear without strings attached like"

[2] "fliptechfriday monthofmusic love listen music time id need wirelesslooooffice"

[3] "fliptechfriday love mobile dont like tethered physically"

[4] "mess god anywhere wire headset love first wireless headphone fliptechfriday"

[5] "gurvinderm mess god anywhere wire headset love first wireless headphone"

Delivery

[1] "chanchal snapdealhelp pathetic customer service wrt amazonin face colossal delay snapdeal delivery team"

[2] "delivery boy cancel order without informing send sms ur order cancel"

[3] "complaint irresponsible delivery service order product nov cancel"

[4] "today got delivery delivery boy request adharcard just want know u require adharcard"

[5] "worst online experience ever delivery delay month now support say cant even cancel order disgusted"

Snapdeal

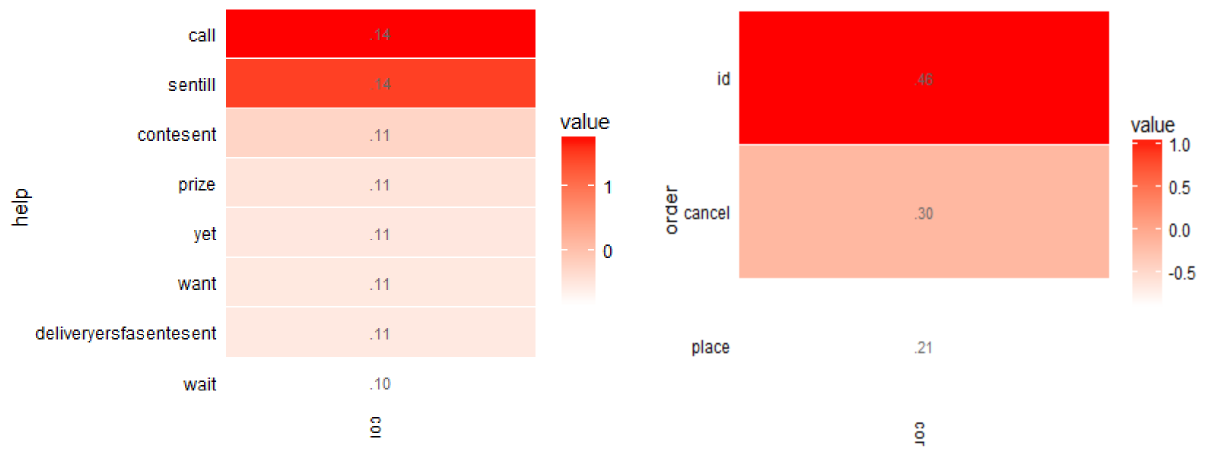


Figure 3.14 Heat map of highly correlated words of Snapdeal brand

Find messages with high corelated word for Snapdeal tweets

For Word Help

[1] "help guy never chk send order"
[2] "help order got spandeal awaited"
[3] "activisentonjob fraudgivn mble balvindersinghs nt rturng wthn dys ufteryou help cus entom"
[4] "help kunalbahl rohitkbansa sdgold founder focus core business operate media presen ce"
[5] "help kunalbahl rohitkbansa sdgold caller say matter close twitter dm say work mislea ding"

For Word Order

[1] "help guy never chk send order"
[2] "help order got spandeal awaited"
[3] "place redmi note order id im face issue ph nt replac ph"
[4] "hv place prepaid order rd novsentill prodct nt dispatch email replyno way contact"
[5] "pawanmca ufteryou take order day refusal give losent mani diwali offer sitenot r"

Ebay

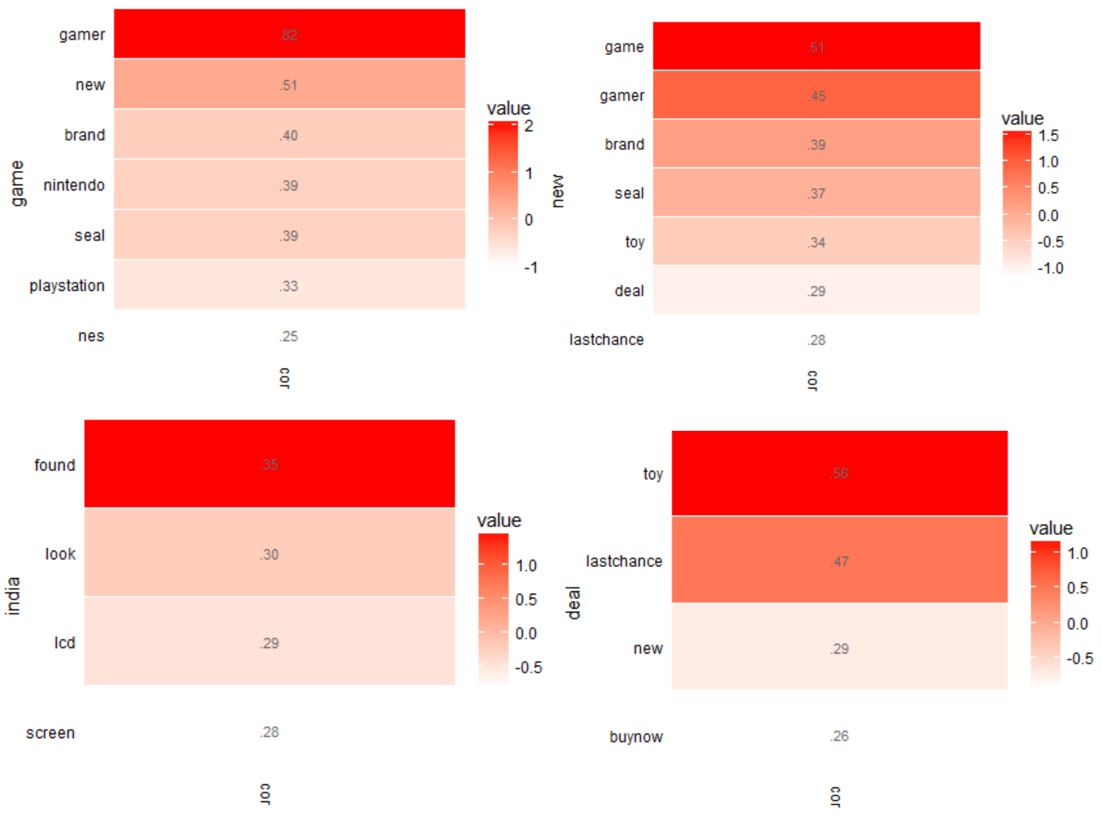


Figure 3.15 Heat map of highly correlated words of Ebay brand

Find messages with high corelated word for Ebay tweets

For word game

[1] "super turrican snes nintendo ocean test rare uk esp au india indiedev gamedev"
[2] "india game special dad gift birthday thingsdontjugde"
[3] "india india game special dad gift birthday"
[4] "india devinwenig uk ask den d call given widout resolution u play game wid"
[5] "run segasaturn jp game retrogameing"

For word new

[1] "india grievance redressing system worst gov cheat wait refund day claim id toiindian ews"
[2] "india newsroom uk look past mnthsi wait solution frm sideits month"
[3] "get small hidden camera spice camera lowest cost shopclues flipkart india abpnewstv paytm"
[4] "india newsroom see standard ur service"
[5] "ask india newsroom book prodct od oct still didnt receive payment given

For word india

[1] "dear india guy take note suspicious seller"
[2] "india power bank usa just rs now thats darwazatod deal grab"
[3] "india fraud fake companies"
[4] "look found india"
[5] "india serious wud now pick good work just cosmetic lip service"

For word deal

[1] "india power bank usa just rs now thats darwazatod deal grab"
[2] "india dont need coffee refresh deal like stop think get click"
[3] "promo sale dailydeals shoplive retweetsshoppershourtweetmaster sariukhashtags indi a"
[4] "india wont beg plead crib get darwazatoddeal iphone youve always want click heregt "
[5] "india gotta refresh second get darwazatod deal dont worried patience will rewarded"

Shop Clues

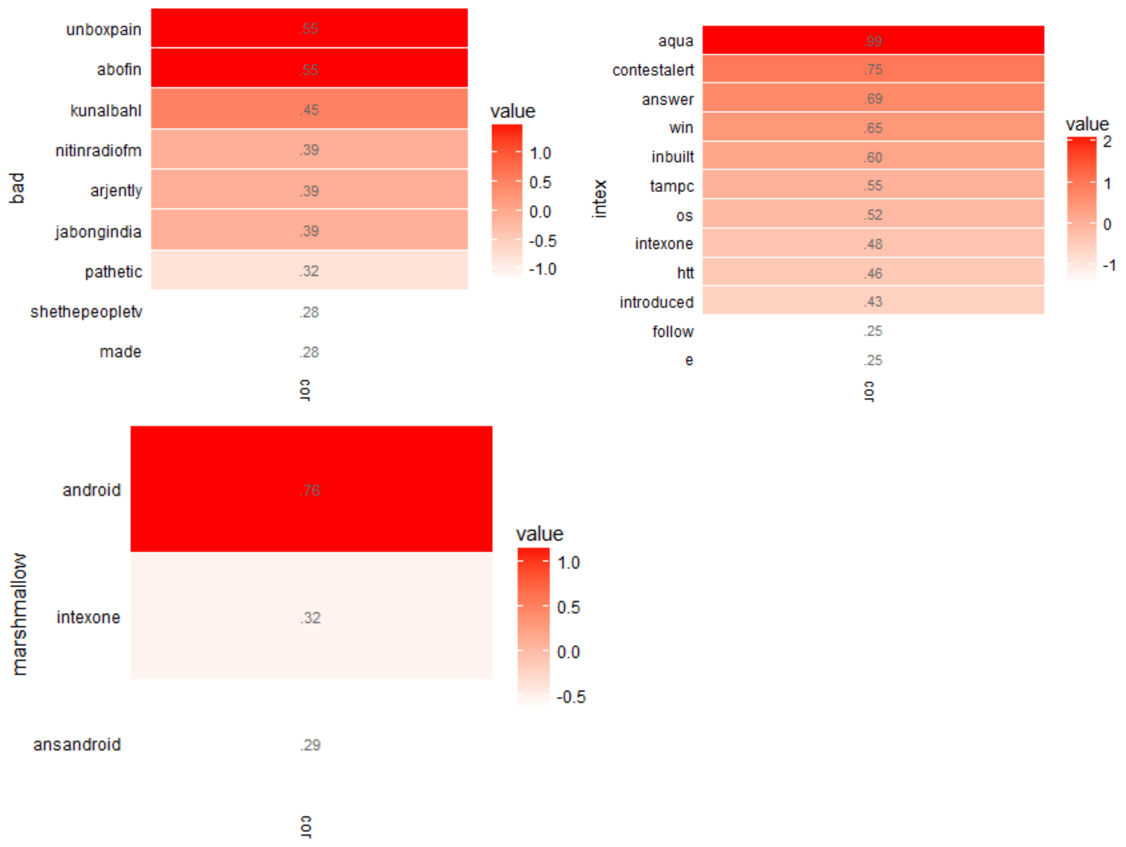


Figure 3.16 Heat map of highly correlated words of Shop Clues brand

Find messages with high corelated word for ShopClues tweets

For word intex

[1] "intexbrand new aqua smartphone smart nation buy now intexone s"
[2] " intexone contestalert answer win intex aqua cheapest volte phone rs"
[3] "intexbrand last chance get rs intex aqua smartphone register intexone"
[4] " inbuilt os intex aqua sailfish"
[5] " congratulation winner intexone contest havent register intex aqua will"

For word bad

[1] "nitinradiofm bad unboxpain pathetic snapdeal kunalbahl amazon flipkart paytm abof in jabongindia"
[2] "possibleindia tweetbadshah tweeps follow play intexone win win intex aqua e"
[3] " android marshmallow intexone badgujarparth iamakashrajput"
[4] "badmashdip kiranideal tweeps follow play intexone win win intex aqua e"
[5] " extremely bad product deliveryer"

For word marshmallow

```
[1] " come android marshmallow intexone"
[2] "joyalpatel  ansandroid os marshmallow intexone"
[3] " intexbrand android marshmallow intexone sabsesastavolte"
[4] " intexbrand android marshmallow intexone"
[5] " android marshmallow intexone intexbrand"
```

Word Association

We can also explore the data in an associational sense by looking at collocation, or those terms that frequently co-occur. Using word associations, we can find the correlation between certain terms and how they appear in tweets (documents). We can perform word associations with the findAssocs() function. Let us find the word associations for most frequent words that we found for all brands in above steps and return the word terms with correlations higher than 0.2.

Find word association for high corelated words from Amazon tweets

For word help

```
> findAssocs(tdm, "help", 0.2)
$help
jeffbezos      amit
     0.28      0.23
```

For word order

```
> findAssocs(tdm, "order", 0.2)
$order
cancel  place
  0.25   0.22
```

For word sign

```
> findAssocs(tdm, "sign", 0.2)
$sign
       copies          catch       contestw             la         legend            pmw
         0.94           0.93           0.93           0.93           0.93           0.93
  tomfacebook           live mrsfunnybones
         0.93           0.91           0.76
```

For word copies

```
> findAssocs(tdm, "copies", 0.2)
$copies
        catch      contestw            la           pmw          sign   tomfacebook
         0.94          0.94          0.94          0.94          0.94          0.94
       legend          live mrsfunnybones
         0.93          0.92          0.76
```

For word fraud

```
> findAssocs(tdm, "fraud", 0.2)
$fraud
  thieves        ppl    biggest      bunch    refusal     ghanta  fraudyou amitgupta      dukan
     0.30       0.29       0.28       0.28       0.27       0.26      0.24      0.23       0.23
    loser       apni   vipgupta
     0.23       0.22       0.22
```

Find word association for high corelated words from Flipkart tweets

For Word redmi

```
> findAssocs(tdm, "redmi", 0.2)
$redmi
             love             owner             proud              unit         sunemania
             0.59              0.31              0.31              0.31              0.30
             sold           million millionredmison        millionredm           express
             0.29              0.28              0.26              0.25              0.23
          explain               law              mill    sangeetatweets
             0.22              0.21              0.21              0.21
```

For Word beard

```
> findAssocs(tdm, "beard", 0.2)
$beard
                  ask                  grow                   old              rockinrr
                 1.00                  1.00                  1.00                  1.00
internationalmensdiy                   pic                friend               without
                 0.99                  0.99                  0.96                  0.96
                 look
                 0.93
```

For word Love

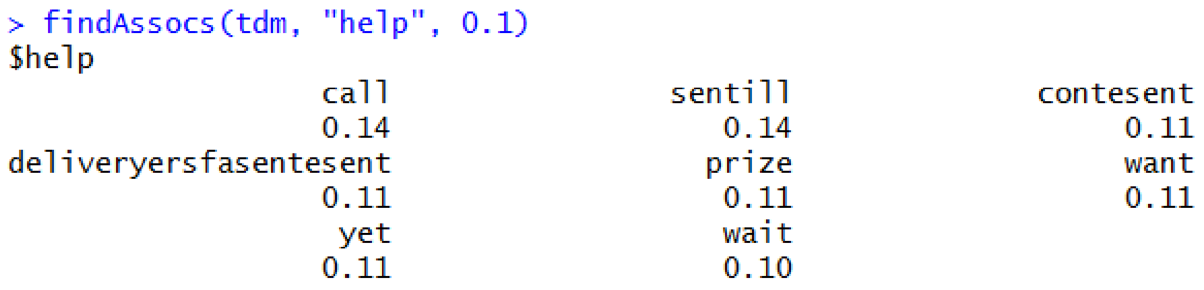
```
> findAssocs(tdm, "love", 0.2)
$love
          redmi millionredmison         feature          reason            sone
           0.59            0.36            0.27            0.27            0.27
         suhaga            bcos              pe             law  sangeetatweets
           0.27            0.26            0.26            0.23            0.22
         follow           price
           0.21            0.21
```

For word delivery

```
> findAssocs(tdm, "delivery", 0.2)
$delivery
          boy      smsemail      adharcard    amazoninno       custcare dealsinamazon
         0.30          0.28           0.27          0.27           0.27          0.27
      whereas        sumonc          delay
         0.27          0.24           0.21
```

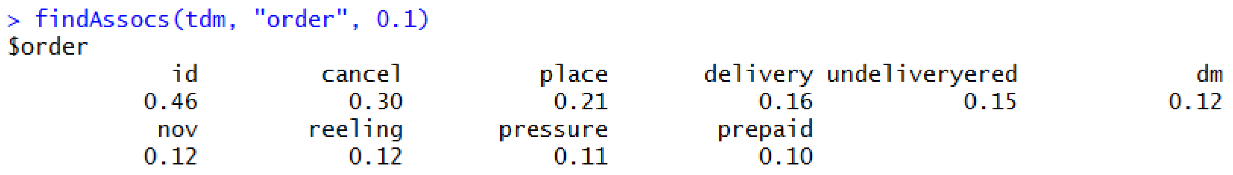
Find word association for high corelated words from Snapdeal tweets

For Word help



```
> findAssocs(tdm, "help", 0.1)
$help
                   call                sentill               contesent
                   0.14                   0.14                    0.11
deliveryersfasentesent                   prize                    want
                   0.11                   0.11                    0.11
                    yet                   wait
                   0.11                   0.10
```

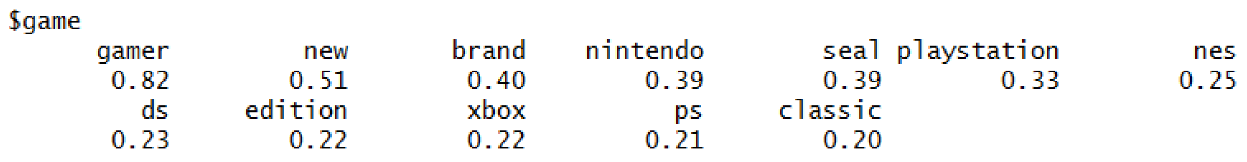
For Word order



```
> findAssocs(tdm, "order", 0.1)
$order
           id          cancel           place        delivery undeliveryered              dm
         0.46            0.30            0.21            0.16           0.15            0.12
          nov         reeling        pressure         prepaid
         0.12            0.12            0.11            0.10
```

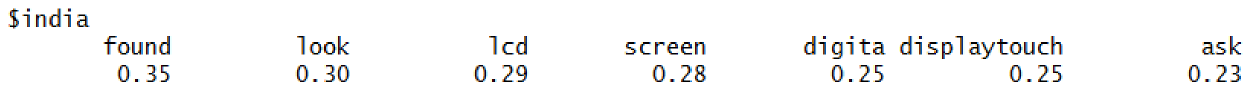
Find word association for high corelated words from Ebay tweets

For word game



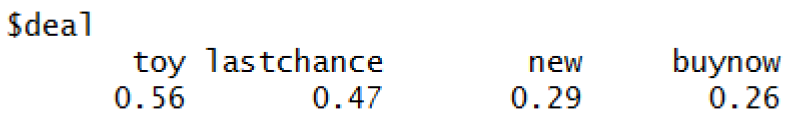
```
$game
      gamer         new       brand    nintendo        seal playstation         nes
       0.82        0.51        0.40        0.39        0.39        0.33        0.25
         ds     edition        xbox          ps     classic
       0.23        0.22        0.22        0.21        0.20
```

For word India



```
$india
      found        look         lcd      screen      digita displaytouch        ask
       0.35        0.30        0.29        0.28        0.25        0.25        0.23
```

For word deal



```
$deal
        toy lastchance        new     buynow
       0.56       0.47       0.29       0.26
```

Find word association for high corelated words from Shop Clues tweets

For word intex

```
$intex
        aqua contestalert       answer          win      inbuilt        tampc           os
        0.99         0.75         0.69         0.65         0.60         0.55         0.52
    intexone          htt   introduced            e       etampc       follow       tweeps
        0.48         0.46         0.43         0.25         0.25         0.25         0.25
        play     cheapest
        0.24         0.21
```

For word bad

```
$bad
        abofin      unboxpain       kunalbahl        arjently     jabongindia    nitinradiofm
          0.55           0.55            0.45            0.39            0.39            0.39
      pathetic           made shethepeopletv       extremely
          0.32           0.28            0.28            0.23
```

For Word marshmallow

```
$marshmallow
   android   intexone ansandroid
      0.76       0.32       0.29
```

Topic modelling

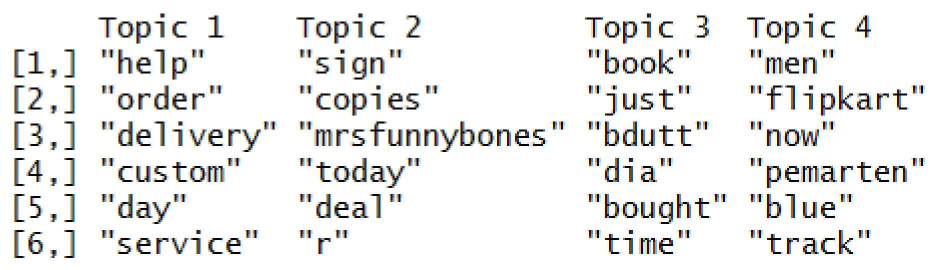
When there is a large collection of documents but have no idea what they are about. One of the first things you might want to do is to classify these documents into topics or themes or clusters or bags. This helps to figure out if there is anything interest while also directing you to the relevant subset(s) of the corpus. For small collections, one could do this by simply going through each document but this is clearly infeasible for corpuses containing thousands of documents. Topic models allow the probabilistic modeling of term frequency occurrences in documents. The fitted model can be used to estimate the similarity between documents as well as between a set of specified keywords using an additional layer of latent variables which are referred to as topics.

Topic Modelling using LDA with Gibbs sampling

Here we are using LDA function of topicmodels package with Gibbs sampling for the topic modelling. LDA is a technique that helps in automatic discovery of themes from collection of documents. The basic assumption behind LDA is that each of the documents in a collection consist of a mixture of collection-wide topics. LDA does the segregation of terms in various topics by recreating the documents in the corpus by adjusting the relative importance of topics in documents and words in topics iteratively. The iterative process uses a technique called Gibbs sampling. How LDA and Gibbs sampling works in details is beyong the scope of this report.

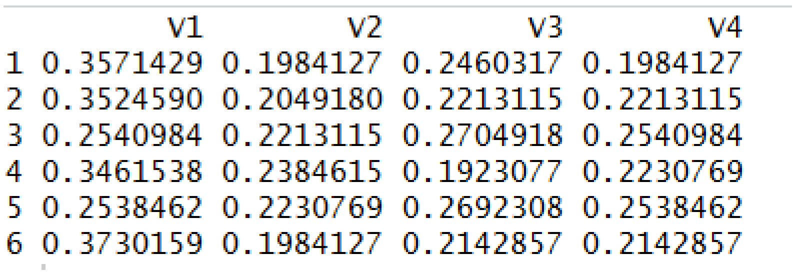
Gibbs sampling works by performing a random walk in such a way that reflects the characteristics of a desired distribution. Because the starting point of the walk is chosen at random, it is necessary to discard the first few steps of the walk (as these do not correctly reflect the properties of distribution). This is referred to as the burn-in period. We set the burn-in parameter to 4000. Following the burn-in period, we perform 2000 iterations, taking every 500th iteration for further use. The reason we do this is to avoid correlations between samples. We use 5 different starting points (nstart=5) – that is, five independent runs. Each starting point requires a seed integer (this also ensures reproducibility), so I have provided 5 random integers in my seed list. Finally I've set best to TRUE (actually a default setting), which instructs the algorithm to return results of the run with the highest posterior probability. The settings above do not guarantee the convergence of the algorithm. Using this approach we found the top 6 terms in each topic and then the probabilities associated with each topic and the terms.

Top 6 terms in each topic for Amazon



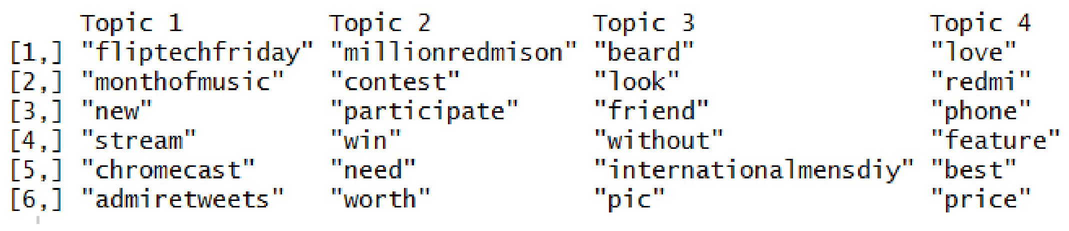
```
     Topic 1    Topic 2         Topic 3  Topic 4
[1,] "help"     "sign"          "book"   "men"
[2,] "order"    "copies"        "just"   "flipkart"
[3,] "delivery" "mrsfunnybones" "bdutt"  "now"
[4,] "custom"   "today"         "dia"    "pemarten"
[5,] "day"      "deal"          "bought" "blue"
[6,] "service"  "r"             "time"   "track"
```

Probabilities associated with each topic for Amazon tweets



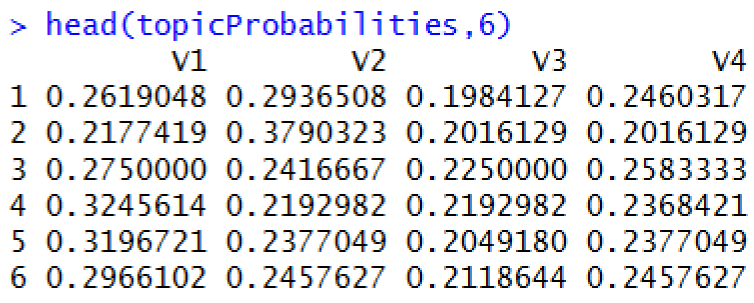
```
         V1        V2        V3        V4
1 0.3571429 0.1984127 0.2460317 0.1984127
2 0.3524590 0.2049180 0.2213115 0.2213115
3 0.2540984 0.2213115 0.2704918 0.2540984
4 0.3461538 0.2384615 0.1923077 0.2230769
5 0.2538462 0.2230769 0.2692308 0.2538462
6 0.3730159 0.1984127 0.2142857 0.2142857
```

Top 6 terms in each topic for Flipkart



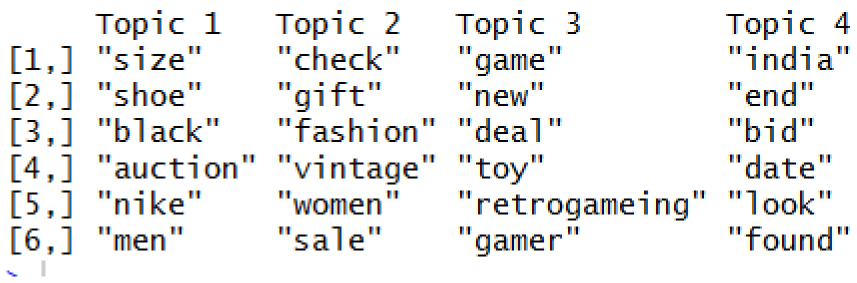
```
     Topic 1          Topic 2           Topic 3                Topic 4
[1,] "fliptechfriday" "millionredmison" "beard"                "love"
[2,] "monthofmusic"   "contest"         "look"                 "redmi"
[3,] "new"            "participate"     "friend"               "phone"
[4,] "stream"         "win"             "without"              "feature"
[5,] "chromecast"     "need"            "internationalmensdiy" "best"
[6,] "admiretweets"   "worth"           "pic"                  "price"
```

Probabilities associated with each topic for Flipkart tweets



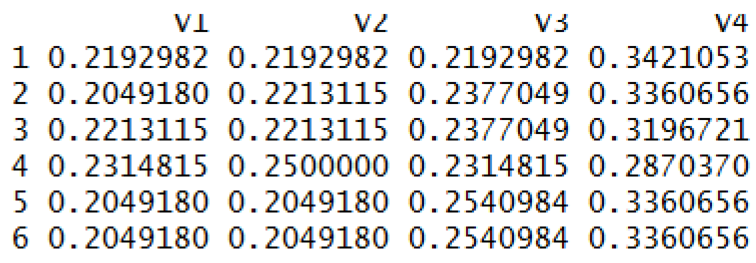
```
> head(topicProbabilities,6)
         V1        V2        V3        V4
1 0.2619048 0.2936508 0.1984127 0.2460317
2 0.2177419 0.3790323 0.2016129 0.2016129
3 0.2750000 0.2416667 0.2250000 0.2583333
4 0.3245614 0.2192982 0.2192982 0.2368421
5 0.3196721 0.2377049 0.2049180 0.2377049
6 0.2966102 0.2457627 0.2118644 0.2457627
```

Top 6 terms in each topic for Ebay



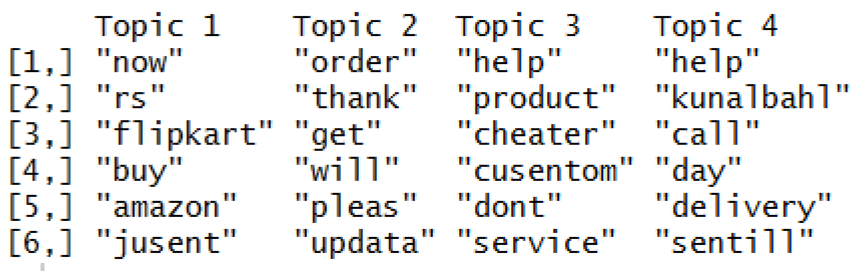
```
     Topic 1   Topic 2   Topic 3        Topic 4
[1,] "size"    "check"   "game"         "india"
[2,] "shoe"    "gift"    "new"          "end"
[3,] "black"   "fashion" "deal"         "bid"
[4,] "auction" "vintage" "toy"          "date"
[5,] "nike"    "women"   "retrogameing" "look"
[6,] "men"     "sale"    "gamer"        "found"
```

Probabilities associated with each topic for Ebay tweets



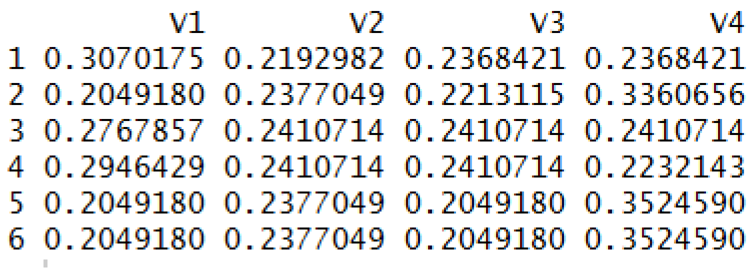
```
         V1        V2        V3        V4
1 0.2192982 0.2192982 0.2192982 0.3421053
2 0.2049180 0.2213115 0.2377049 0.3360656
3 0.2213115 0.2213115 0.2377049 0.3196721
4 0.2314815 0.2500000 0.2314815 0.2870370
5 0.2049180 0.2049180 0.2540984 0.3360656
6 0.2049180 0.2049180 0.2540984 0.3360656
```

Top 6 terms in each topic for Snapdeal



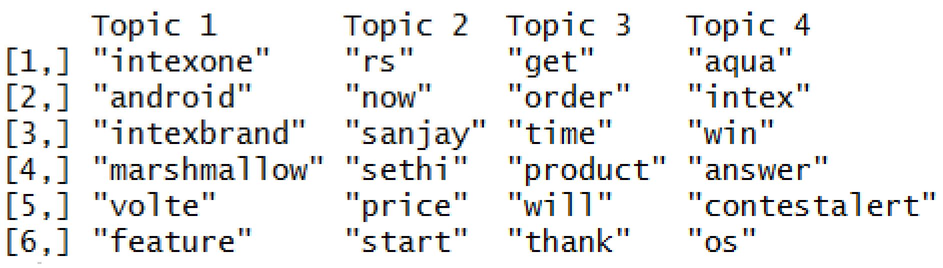
```
     Topic 1   Topic 2  Topic 3    Topic 4
[1,] "now"     "order"  "help"     "help"
[2,] "rs"      "thank"  "product"  "kunalbahl"
[3,] "flipkart" "get"   "cheater"  "call"
[4,] "buy"     "will"   "cusentom" "day"
[5,] "amazon"  "pleas"  "dont"     "delivery"
[6,] "jusent"  "updata" "service"  "sentill"
```

Probabilities associated with each topic for Snapdeal tweets



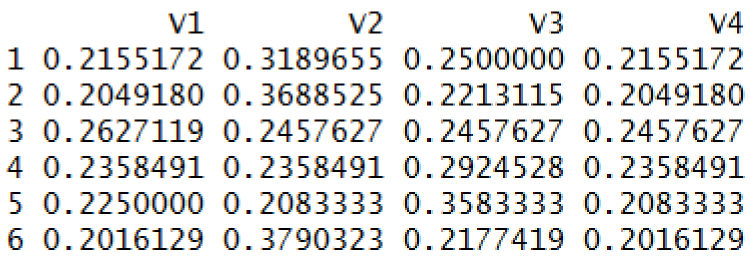
```
         V1        V2        V3        V4
1 0.3070175 0.2192982 0.2368421 0.2368421
2 0.2049180 0.2377049 0.2213115 0.3360656
3 0.2767857 0.2410714 0.2410714 0.2410714
4 0.2946429 0.2410714 0.2410714 0.2232143
5 0.2049180 0.2377049 0.2049180 0.3524590
6 0.2049180 0.2377049 0.2049180 0.3524590
```

Top 6 terms in each topic for ShopClues



```
     Topic 1        Topic 2  Topic 3   Topic 4
[1,] "intexone"     "rs"     "get"     "aqua"
[2,] "android"      "now"    "order"   "intex"
[3,] "intexbrand"   "sanjay" "time"    "win"
[4,] "marshmallow"  "sethi"  "product" "answer"
[5,] "volte"        "price"  "will"    "contestalert"
[6,] "feature"      "start"  "thank"   "os"
```

Probabilities associated with each topic for Shop Clues tweets



```
         V1        V2        V3        V4
1 0.2155172 0.3189655 0.2500000 0.2155172
2 0.2049180 0.3688525 0.2213115 0.2049180
3 0.2627119 0.2457627 0.2457627 0.2457627
4 0.2358491 0.2358491 0.2924528 0.2358491
5 0.2250000 0.2083333 0.3583333 0.2083333
6 0.2016129 0.3790323 0.2177419 0.2016129
```

Topic modelling using Hierarchical Agglomerative Clustering dendogram

Dendogram is a visual representation of clusters. It is a tree diagram frequently used to illustrate the arrangement of the clusters produced by hierarchical clustering. Dendrograms are often used in computational biology to illustrate the clustering of genes or samples, sometimes on top of heat maps.

Clusters are formed according to the Euclidean distance between the points. The height of the curve determines the distance. Lowest the distance, more is the association between the points.

The top row of nodes represents data (individual observations), and the remaining nodes represent the clusters to which the data belong, with the arrows representing the distance (dissimilarity).

The distance between merged clusters is monotone increasing with the level of the merger: the height of each node in the plot is proportional to the value of the intergroup dissimilarity between its two daughters (the top nodes representing individual observations are all plotted at zero height).

To make a Hierarchical Agglomerative cluster plot, we need to reduce the number of terms (which otherwise wouldn't fit on a page or the screen) and build a data frame. Use removeSparseTerms(tdm,sparse=0.98) and pass the term document matrix. This function returns a term-document matrix where those terms from tdm are removed which have at least a sparse percentage of empty (i.e., terms occurring 0 times in a document) elements. I.e., the resulting matrix contains only terms with a sparse factor of less than sparse.Now convert the term document matrix to a matrix using as.matrix().There are a wide range of hierarchical clustering approaches. We have used Ward's method.First create a distance matrix using dist() of stats package after scaling the matrix obtained from the as.matrix() in above step. Then use the hclust() of stats package to get the cluster dendogram and to see the graph use the plot().

```
distmatrix=dist(scale(m2))
fit=hclust(distmatrix,method="ward.D2")
plot(fit,cex=0.9)
```

And then the tree is cut for number of groups using rect.hclust() of stats package.

Hierarchical Agglomerative Cluster Dendogram for Amazon

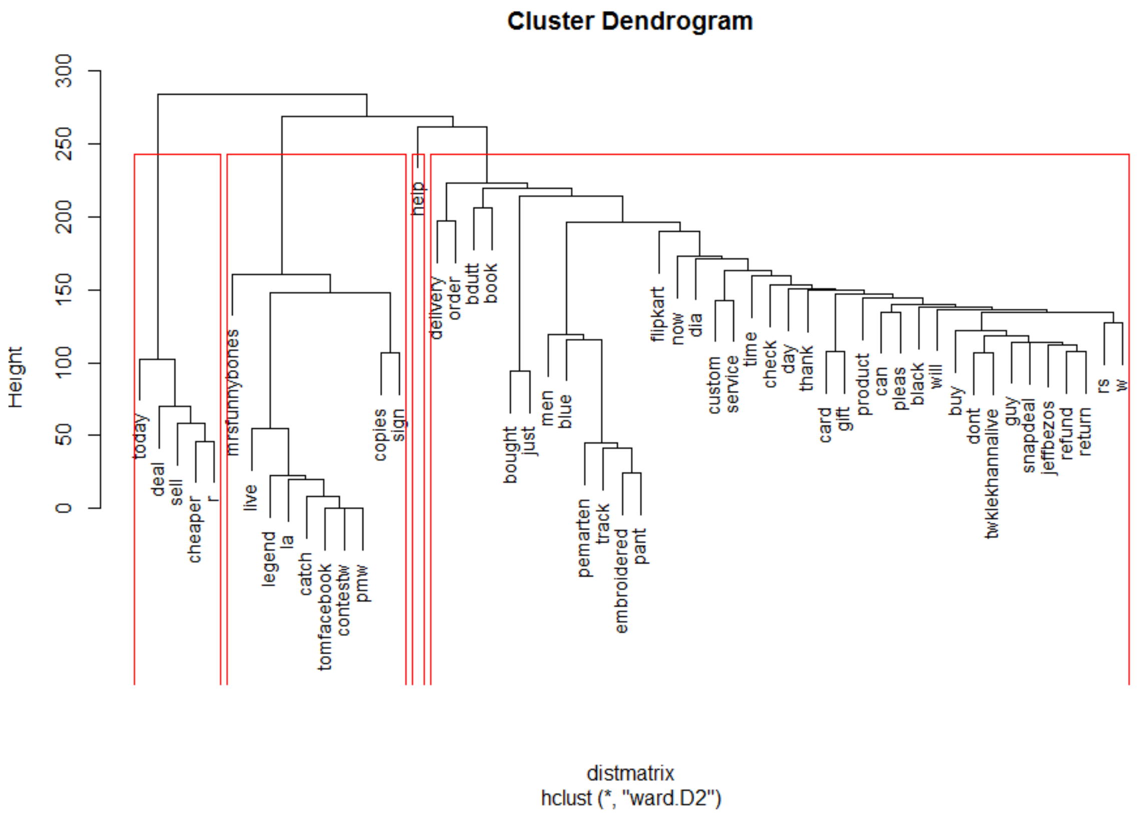


Figure 3.17 Hierarchical Agglomerative Cluster Dendogram for Amazon

The following distinct clusters of tweets are observable from Figure 3.17:

This represents the hierarchical structure of data. 1st cluster are the tweets that talk about deal and price . Here words cheaper & r are closest. Followed by this pair & sell. This cluster of 3 words are then closest to deal & today.

In this cluster today & r are the farthest in the hierarchy.

Cluster 2 are the tweets that talk about some books. In this cluster copies & signs closest. Mrstunnybones & pmw are the farthest in the hierarchy.

Cluster 3 is tweet asking help from the customer care. It is a singleton node.

Cluster 4 is segment regarding tweets for delivery related queries and complaints.

Hierarchical Agglomerative Cluster Dendogram for Flipkart

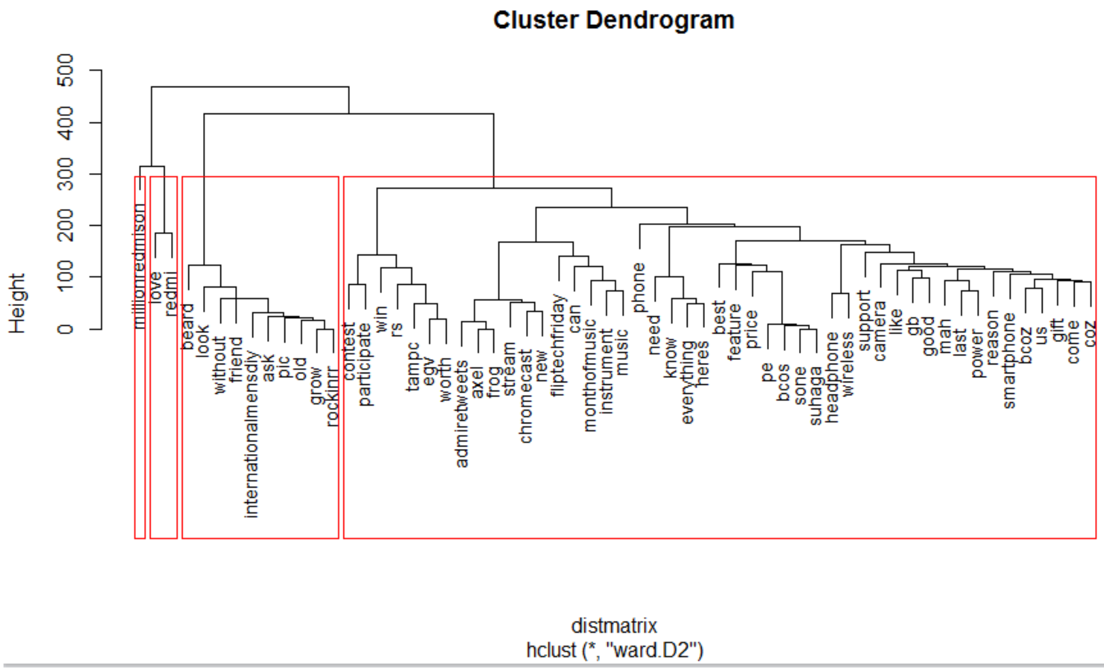


Figure 3.18 Hierarchical Agglomerative Cluster Dendogram for Flipakrt

The following distinct clusters of tweets are observable from Figure 3.18:

1st cluster is tweet that talk about redmi flash sale.It has a singleton node.

2nd cluster is tweets that talk about redmi phone.

3rd cluster is tweets that talk about some beard grooming product. Words grow & rockinrr are the closest in this cluster. Beard & rocknrr are the farthest.

4th cluster are tweets asking flipkart support from the customer care regarding some camera,headphones, cd's ,wireless etc. i.e regarding Electronic category.

Hierarchical Agglomerative Cluster Dendogram for Snapdeal

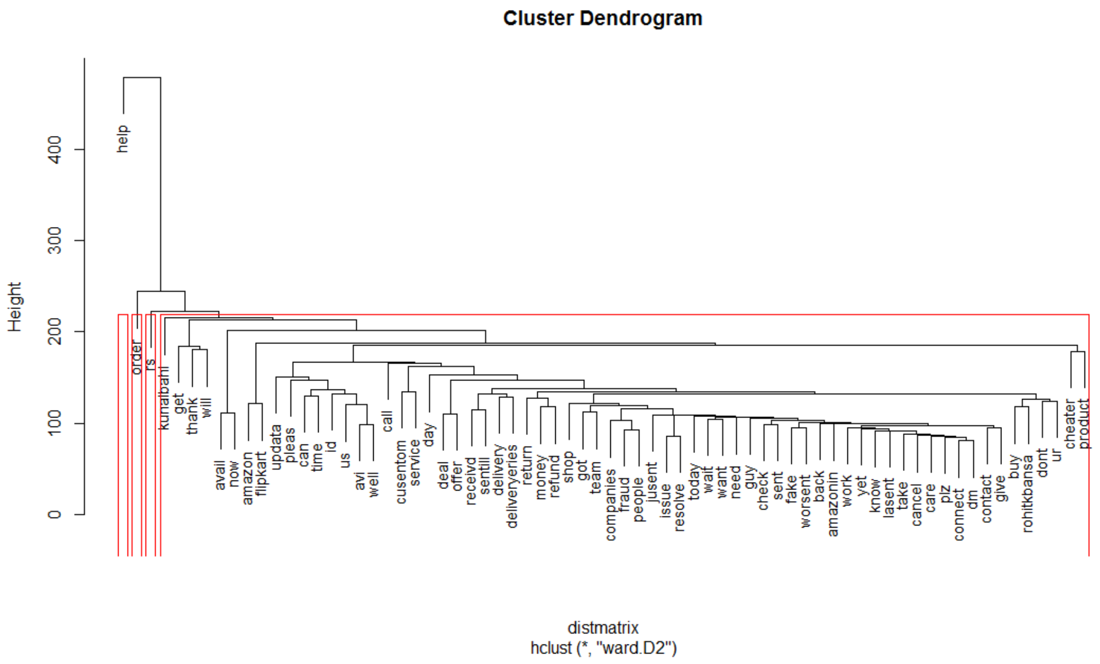


Figure 3.19 Hierarchical Agglomerative Cluster Dendogram for Snapdeal

The following distinct clusters of tweets are observable from Figure 3.19:

1st, 2nd & 3rd clusters are singleton node with help, order & rs respectively.

Tweets that talk about snapdeal help. Tweets that talk about orders and some tweets about the price of the products. Tweets that has some complaints for a deal and delivery and also some tweets from irritated customers regarding the snap deal deal and offer.

Hierarchical Agglomerative Cluster Dendogram for Ebay

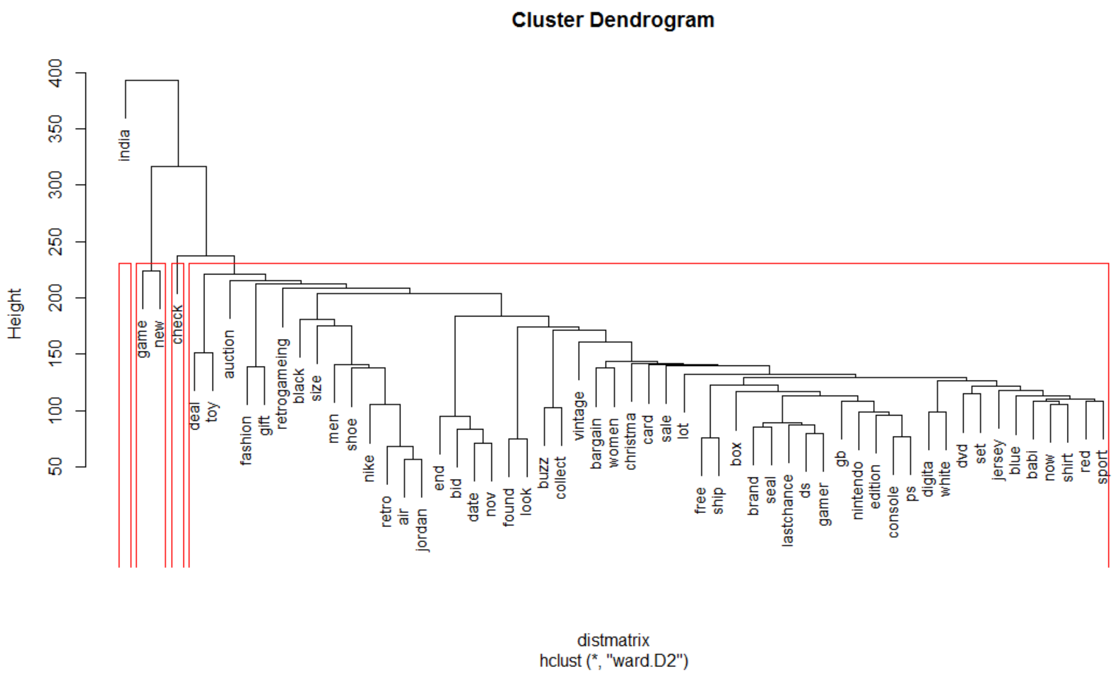


Figure 3.20 Hierarchical Agglomerative Cluster Dendogram for Ebay

The following distinct clusters of tweets are observable from Figure 3.20:

1st cluster is the tweet that talk about India. It is a singleton node.

2nd cluster are the tweets that talk about some new game. Game and new are the closest words in the hierarchy.

3rd cluster is a singleton node having word check.

4th cluster are tweets about the price of the products. Tweets regarding some deal,auction of fashion and sport products

Hierarchical Agglomerative Cluster Dendogram for Shopclues

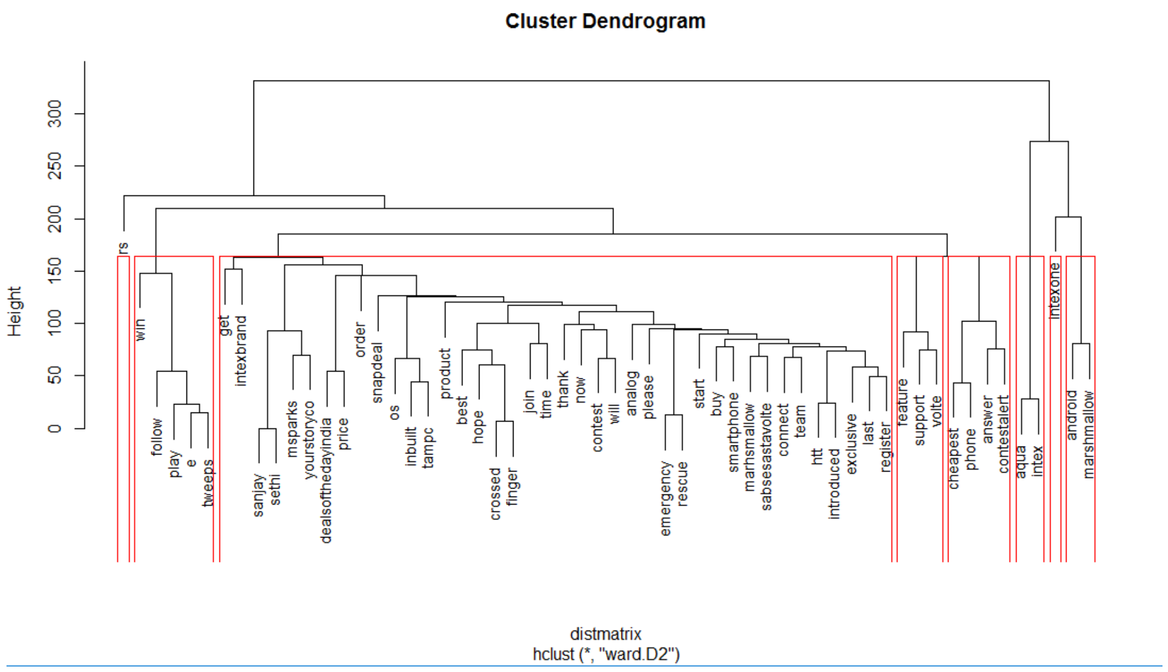


Figure 3.21 Hierarchical Agglomerative Cluster Dendogram for Shop clues

The following distinct clusters of tweets are observable from Figure 3.21:

Tweets that talk about price. Tweets that talk about some deal and offer. and some tweets about customer support. Tweets regarding some comparison with Snapdeal and order . Tweets regarding some android phones.

Sentiment Analysis

Sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document.

A basic task in sentiment analysis is classifying the polarity of a given text at the document, sentence, or feature/aspect level — whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral. Advanced, "beyond polarity" sentiment classification looks, for instance, at emotional states such as "angry," "sad," and "happy."

We use the polarity() of the qdap package. Then we convert it to a dataframe. We segregate the texts into two polarity levels i.e positive or negative.

Sentiment analysis by polarity for tweets for brand Amazon

```
negative positive
    7097     3298
```

Sentiment analysis by date for tweets for brand Amazon

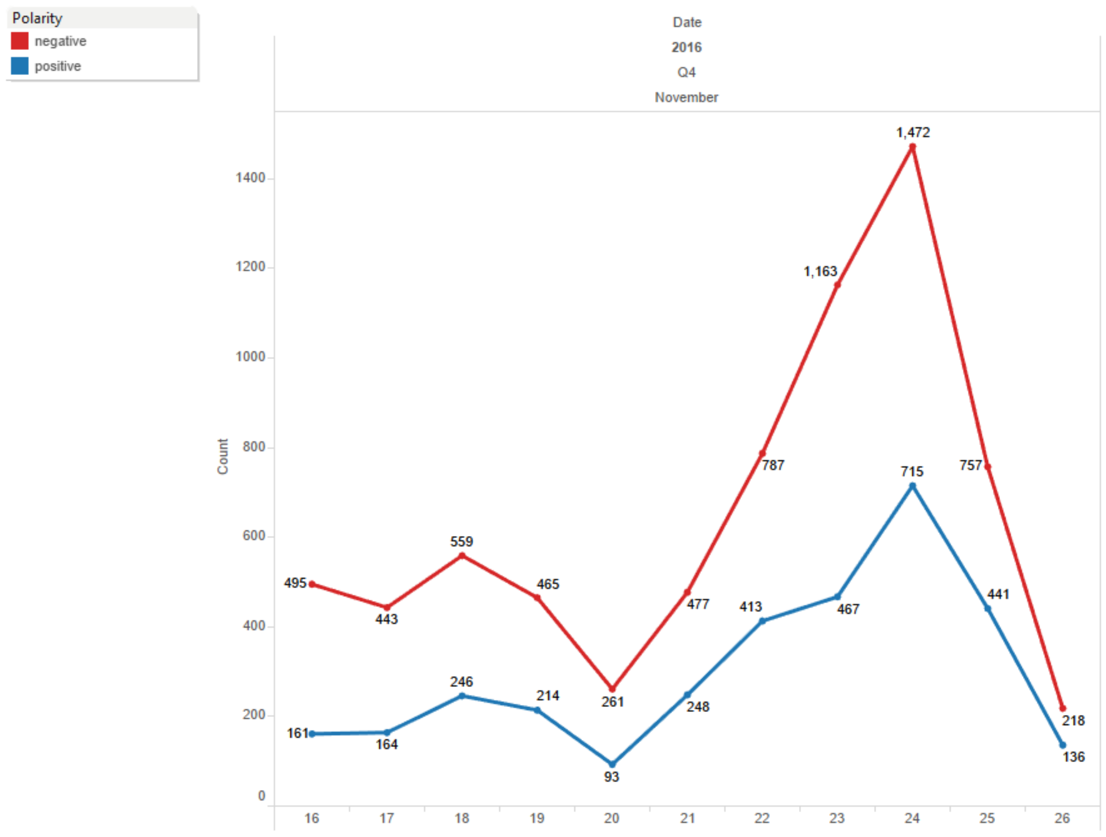


Figure 3.22 Sentment analysis for Amazon tweets by date

The polarity rate is on the negative side and for 24 november we see the negative polarity of tweets has increased twice that of the positive polarity.

Sentiment Analysis of tweets of Amazon by emotions

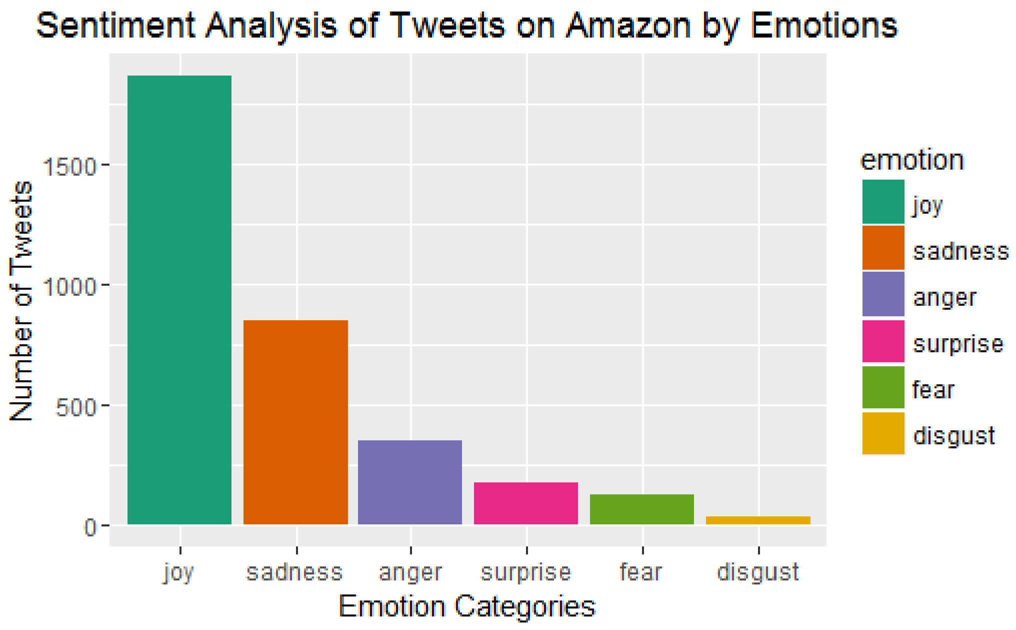


Figure 3.23 Sentment analysis by emotions for Amazon tweets

The number of tweets with **joy** constitute the largest part of tweets, indicating that Amazon is trying their best to provide good business in the country. The **sadness** tweets are less numerous than the **joy** tweets. However, if compared with each other, they indicate the overall market share versus the level of customer satisfaction.

Word cloud of Sentiment Analysis of tweets of Amazon by emotions

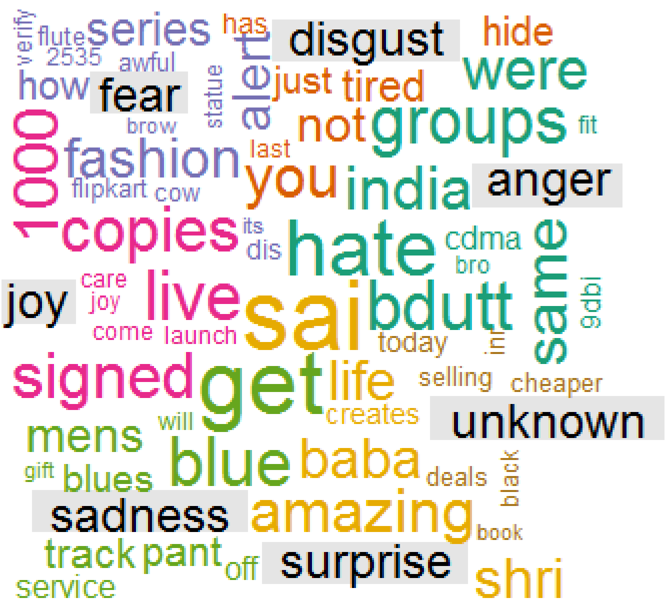


Figure 3.24 Word cloud of sentment analysis for Amazon tweets

The most frequent terms for joy are 1000, copies, live etc. for fear are awful, flipkart etc

Sentiment analysis by polarity for tweets for brand Flipkart

```
negative positive 
    4804     5196 
```

Sentiment analysis by date for tweets for brand Flipkart

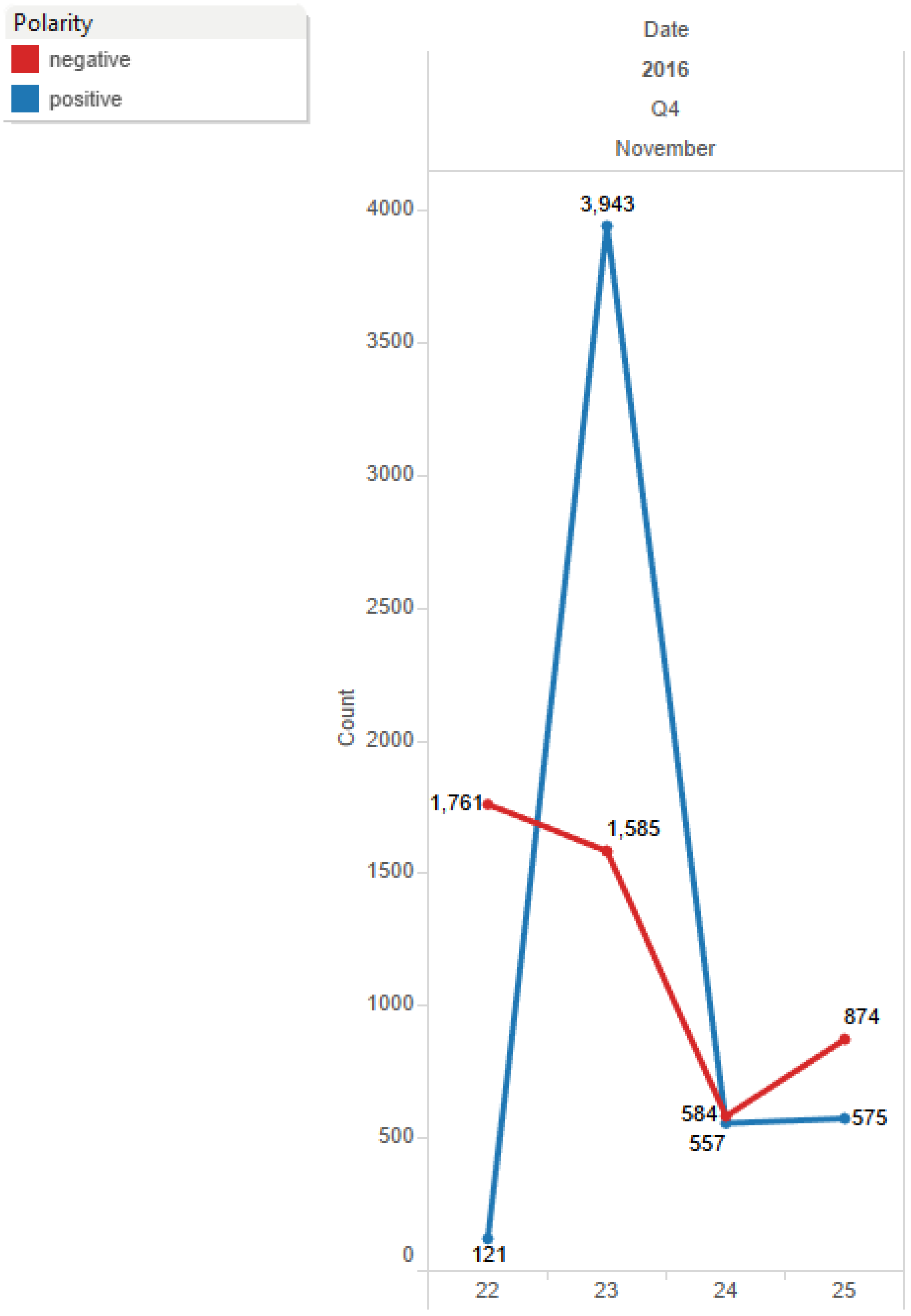


Figure 3.25 Sentment analysis for Flipkart tweets by date

The polarity rate is on the positive side, which shows high customer satisfaction index.

Sentiment Analysis of tweets of Flipkart by emotions

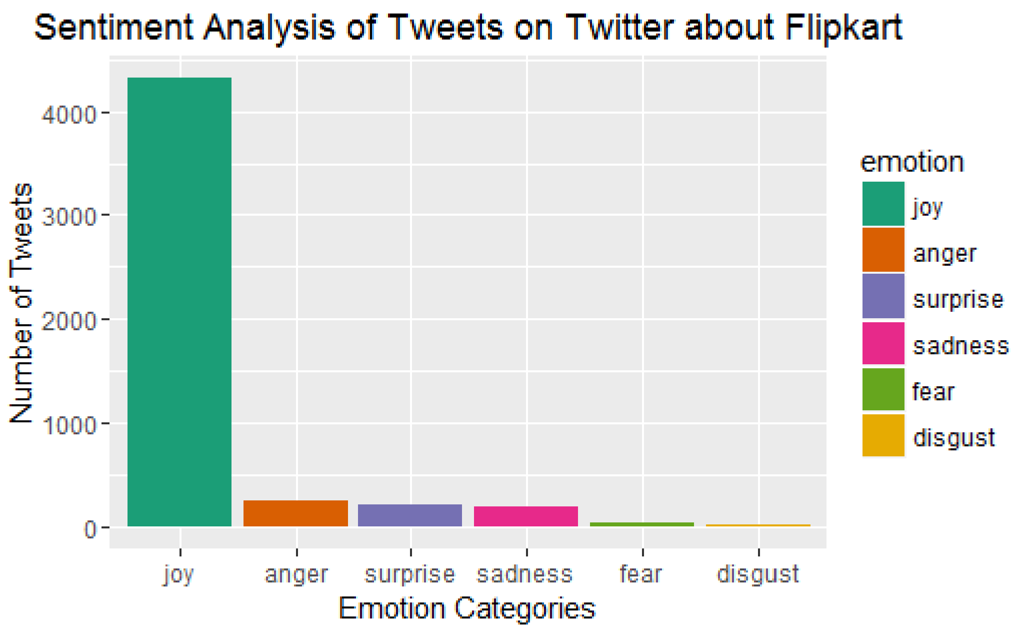


Figure 3.26 Sentment analysis by emotions for flipkart tweets

The number of tweets with **joy** constitute the largest part of tweets compared to all other emotions, showing the strong emotional connect customers are having with Flipkart

Word cloud of Sentiment Analysis of tweets of flipkart by emotions

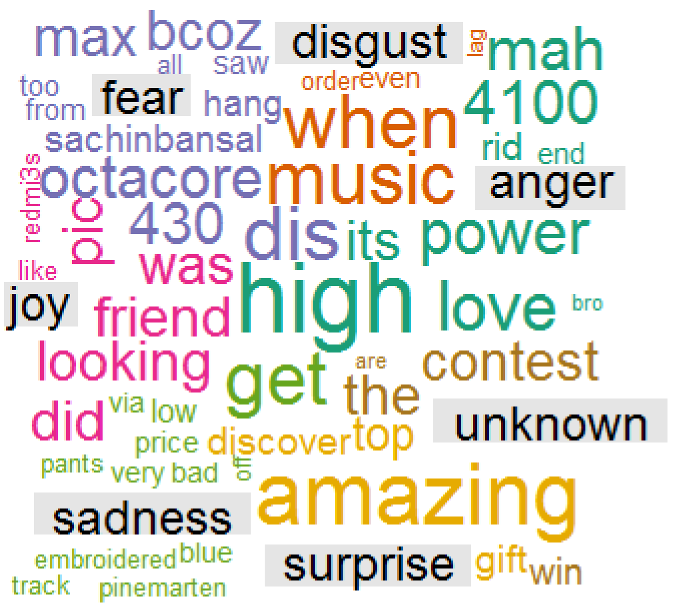


Figure 3.27 Word cloud of sentment analysis by emotions for flipkart tweets

The most frequent terms for joy emotions are: friend,looking ect. For surprise are amazing, discover etc.

Sentiment analysis by polarity for tweets for brand Snapdeal

```
negative positive
    7883     2509
```

Sentiment analysis by date for tweets for brand Snapdeal

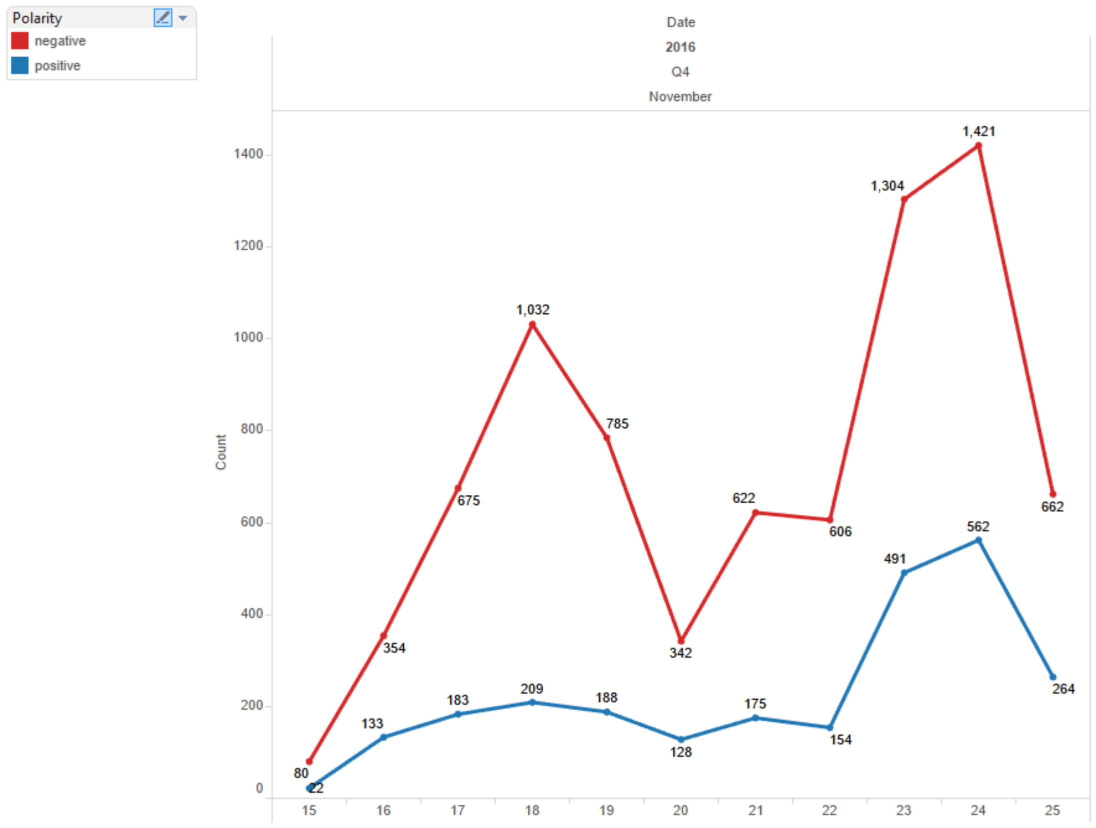


Figure 3.28 Sentment analysis for Snapdeal tweets by date

There is high negative polarity and snapdeal needs to look into what customers are telling to understand the reasons for negative tweets

Sentiment Analysis of tweets of Snapdeal by emotions

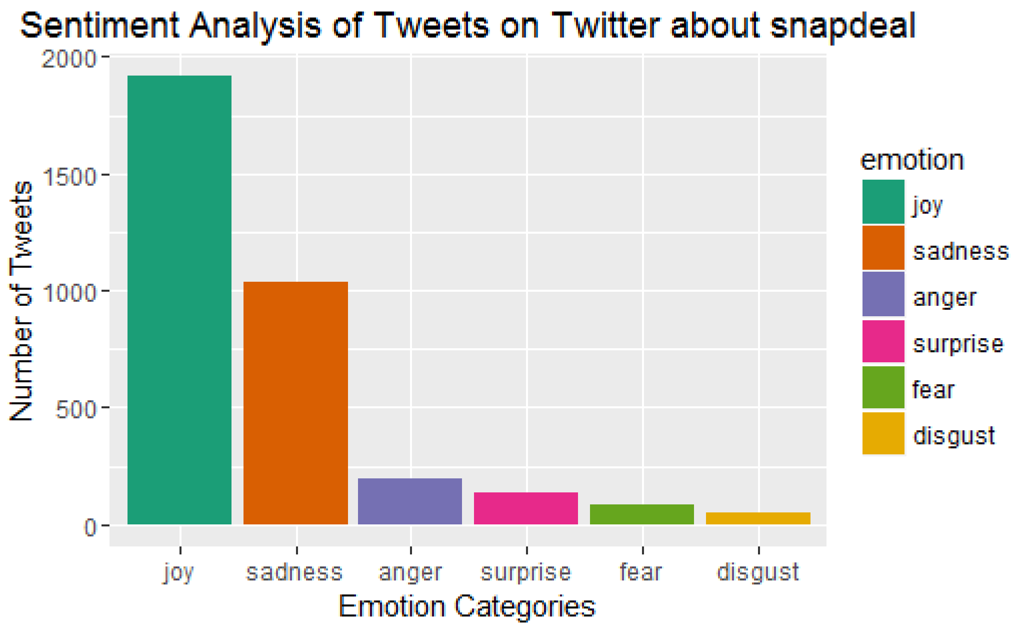


Figure 3.29 Sentment analysis by emotions for snapdeal tweets

Sadness tweets are just less than half the number of joy tweets

Word cloud of Sentiment Analysis of tweets of Snapdeal by emotions

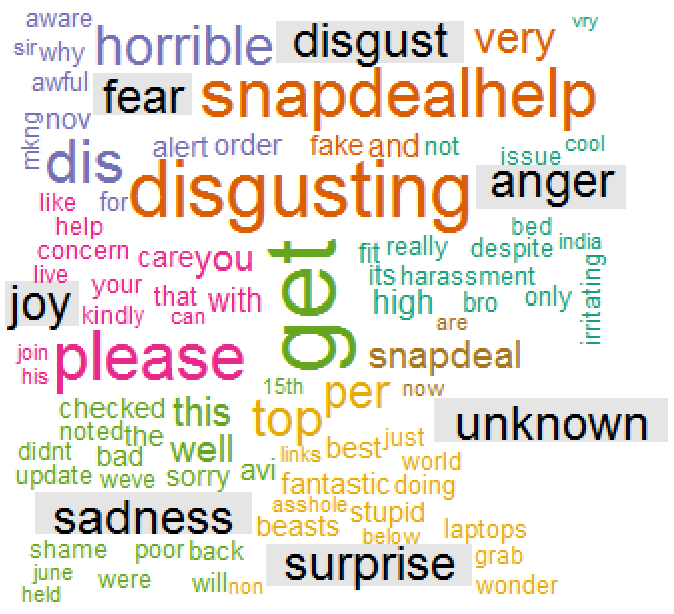


Figure 3.30 Word cloud of Sentment analysis by emotions for snapdeal tweets

For sadness, the common words are: bad, shame, poor etc.

Sentiment Analysis by polarity (positive and negative) for brand Ebay

```
negative positive
    7731     3675
```

Sentiment analysis by date for tweets for brand Ebay

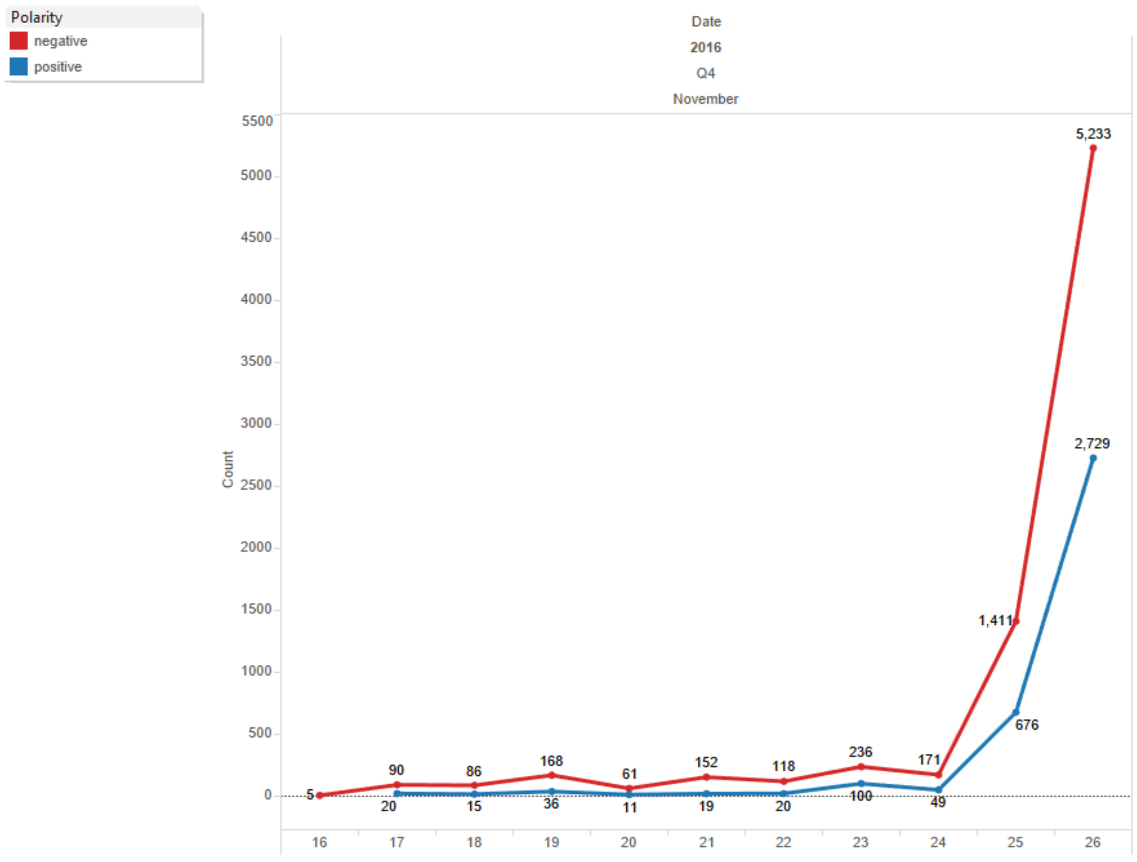


Figure 3.31 Sentment analysis for Ebay tweets by date

The polarity rate had a sudden jump with nearly 5000 negative tweets against 2800 positive tweets on 26th Nov 2016

Sentiment Analysis of tweets of Ebay by emotions

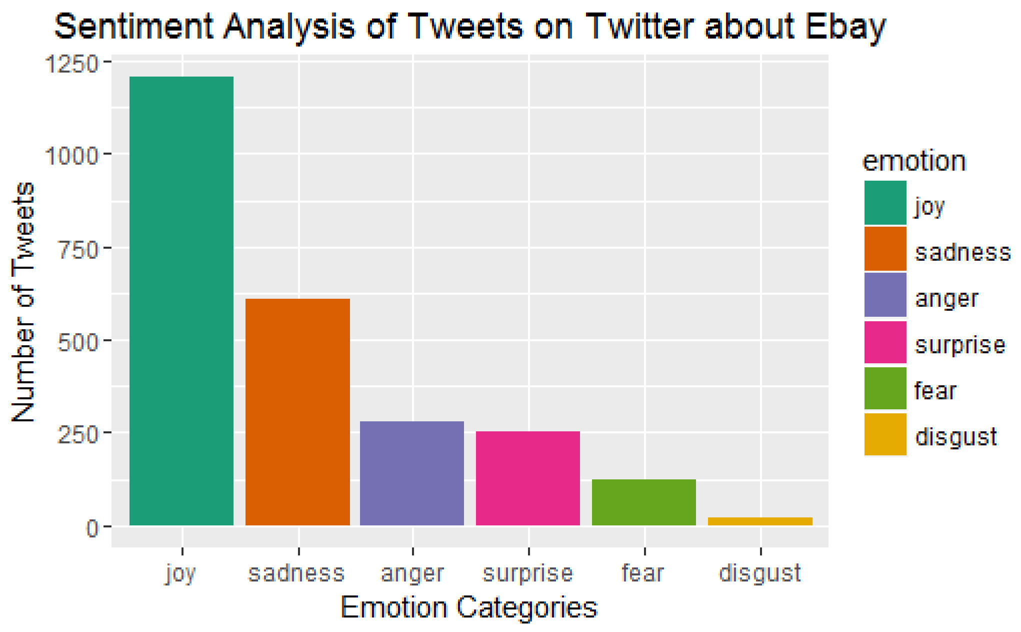


Figure 3.32 Sentment analysis by emotions for Ebay tweets

There are numerous sadness, anger, surprise tweets for ebay.

Word cloud of Sentiment Analysis of tweets of Ebay by emotions

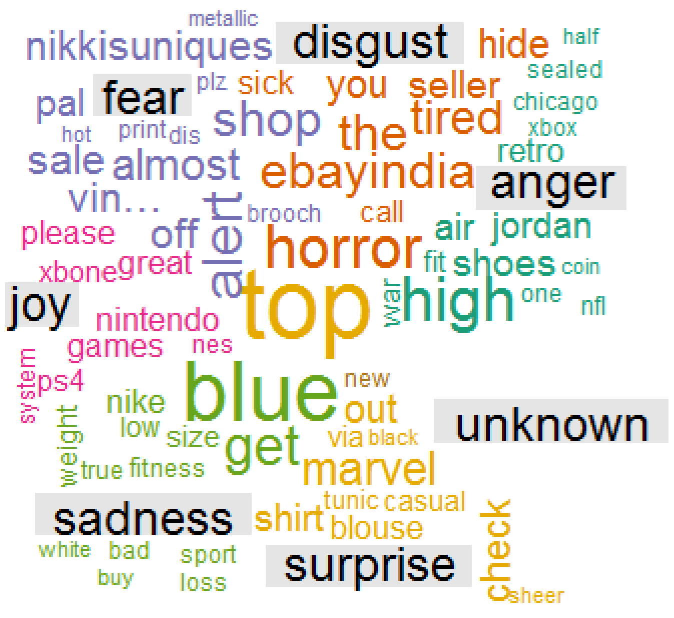


Figure 3.33 Word cloud of Sentment analysis by emotions for Ebay tweets

Blue, get are common terms in sadness tweets

Sentiment Analysis by polarity (positive and negative) for brand shop clues

```
negative positive
    3599     3325
```

Sentiment analysis by date for tweets for brand Shop clues

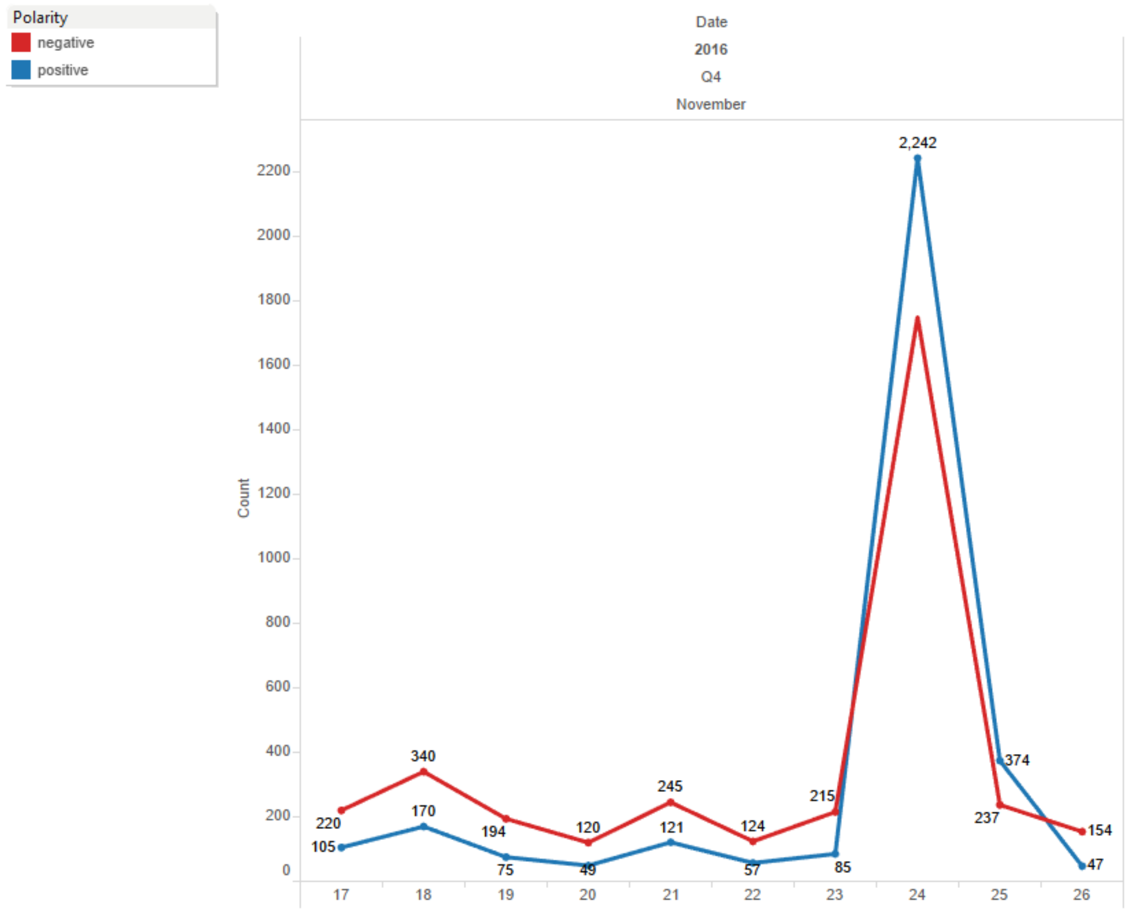


Figure 3.34 Sentment analysis for Shop Clues tweets by date

Positive tweets improved suddenly on 24th November compared to other days

Sentiment Analysis of tweets of Shop Clues by emotions

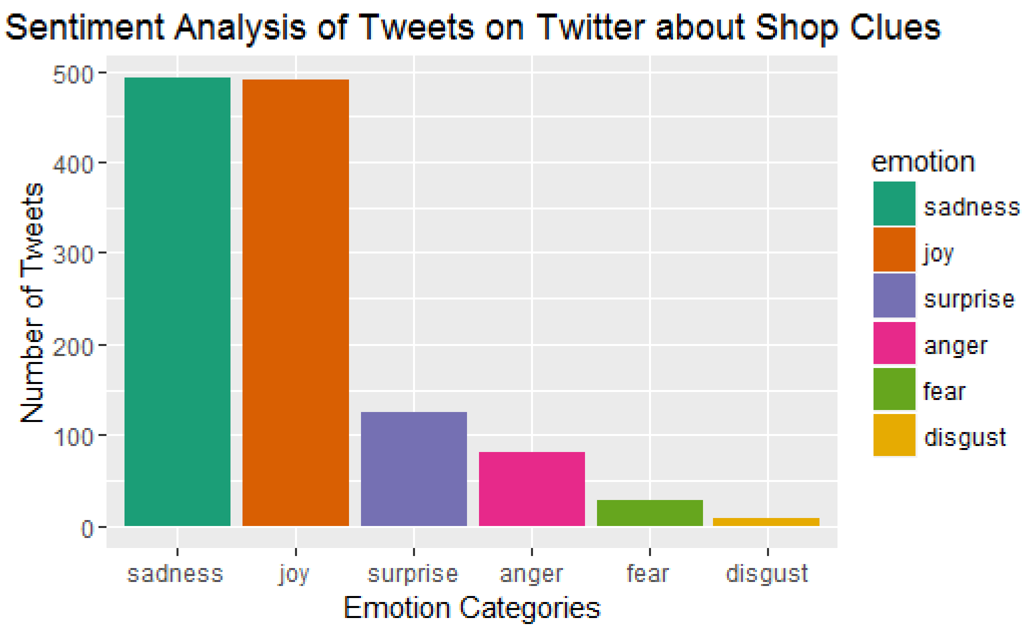


Figure 3.35 Sentiment Analysis of tweets of Shop Clues by emotions

Sadness tweets are slightly more than joy tweets

Word cloud of Sentiment Analysis of tweets of Shop Clues by emotions

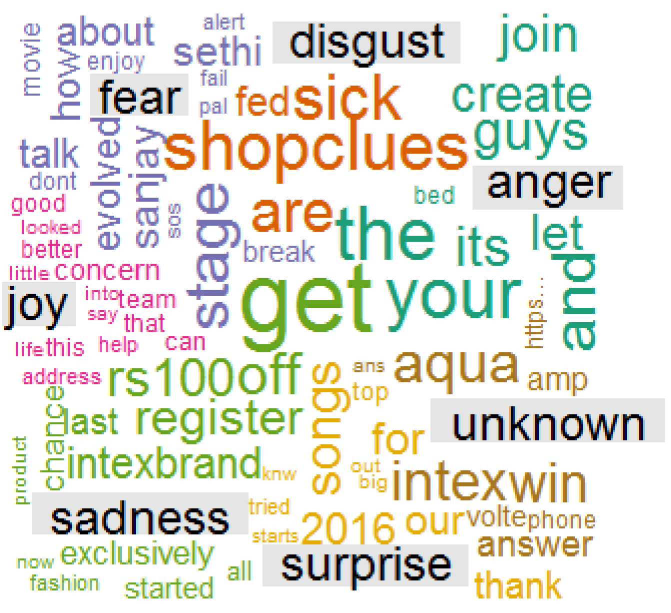


Figure 3.36 Word cloud of Sentiment Analysis of tweets of Shop Clues by emotions

From figure 3.36 Intexbrand, rs100off are common terms in sadness tweets

Sentiment Analysis comparison

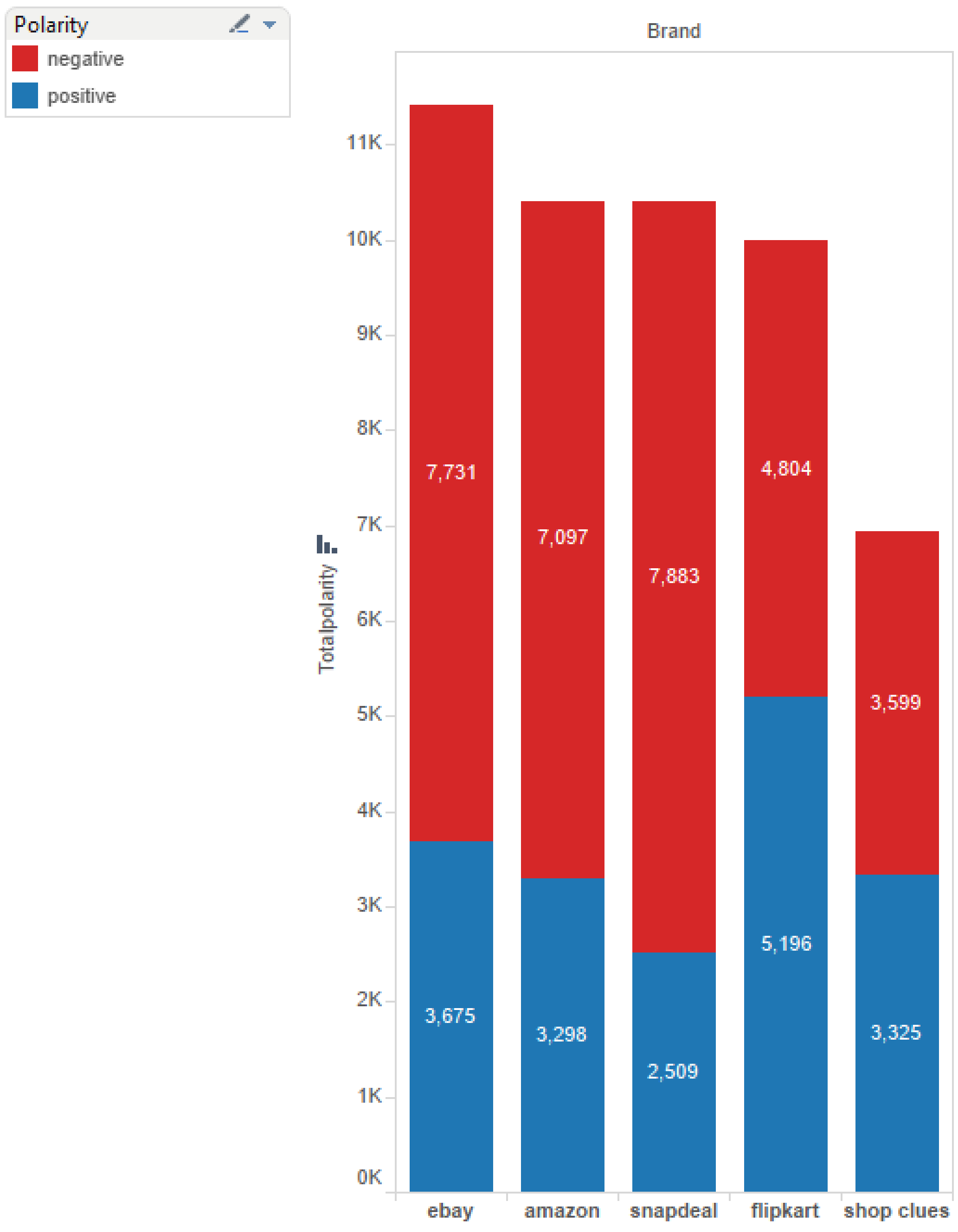
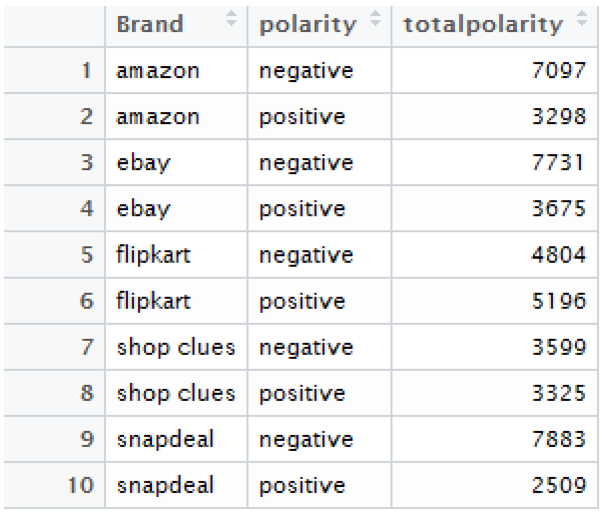


Figure 3.37 Comparison of all brands by sentimens polarity

Ebay needs to focus as there are more negative tweets when compared to other brands



	Brand	polarity	totalpolarity
1	amazon	negative	7097
2	amazon	positive	3298
3	ebay	negative	7731
4	ebay	positive	3675
5	flipkart	negative	4804
6	flipkart	positive	5196
7	shop clues	negative	3599
8	shop clues	positive	3325
9	snapdeal	negative	7883
10	snapdeal	positive	2509

Table 3.1 Sentiment polarity of all brands

Sentiment Analysis polarity trend by date of various brands

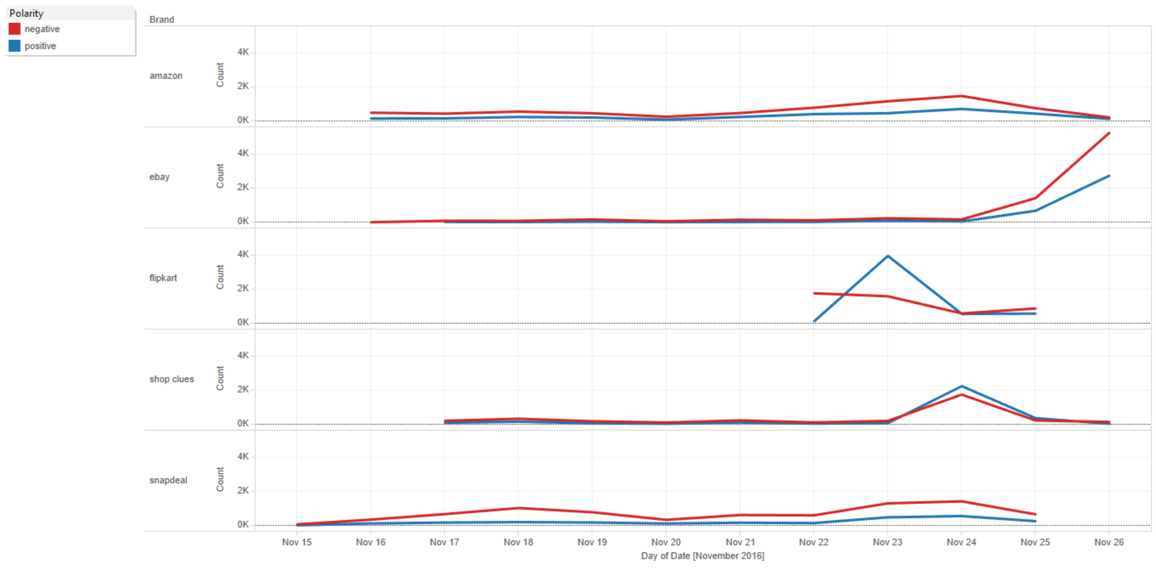


Figure 3.38 Sentiment polarity trend by date of all brands

Based on trend Snapdeal should work on improving the polarity rate.

Sentiment Analysis comparison by polarity rate (positive /negative)

Brand	positive	negative	p/n	p/n(%)
amazon	3298	7097	0.464703	46
flipkart	5196	4804	1.081599	108
snapdeal	2509	7883	0.31828	32
ebay	3675	7731	0.475359	48
shop clues	3325	3599	0.923868	92

Table 3.2 Sentiment Analysis comparison by polarity rate

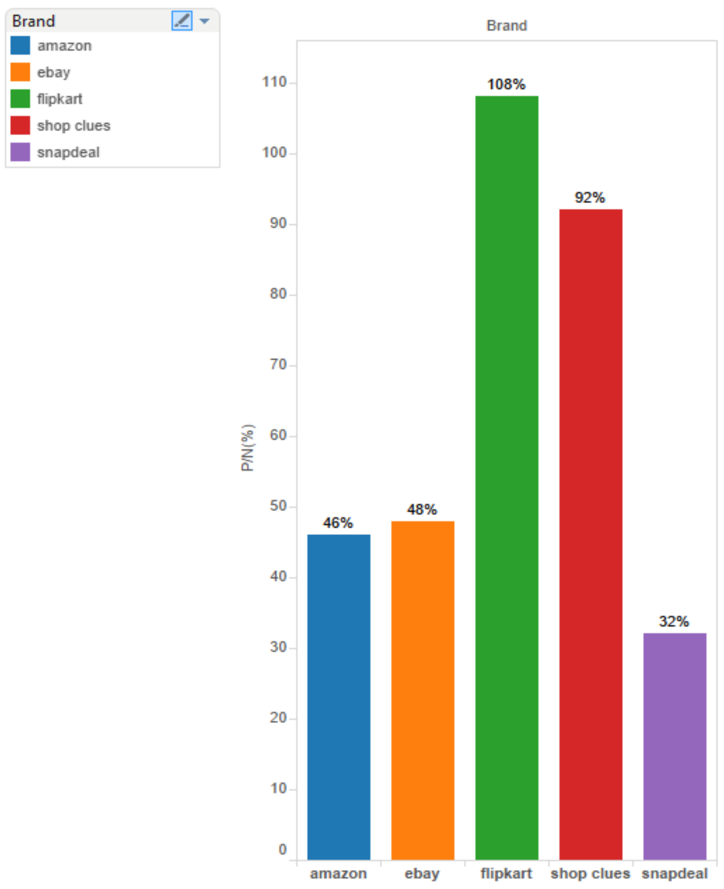


Figure 3.39 Sentiment Analysis comparison by polarity rate

Flipkart tweets could be collected only for few days – as a result flipkart is showing highest polarity rate here.

Sentiment Analysis comparison by emotions

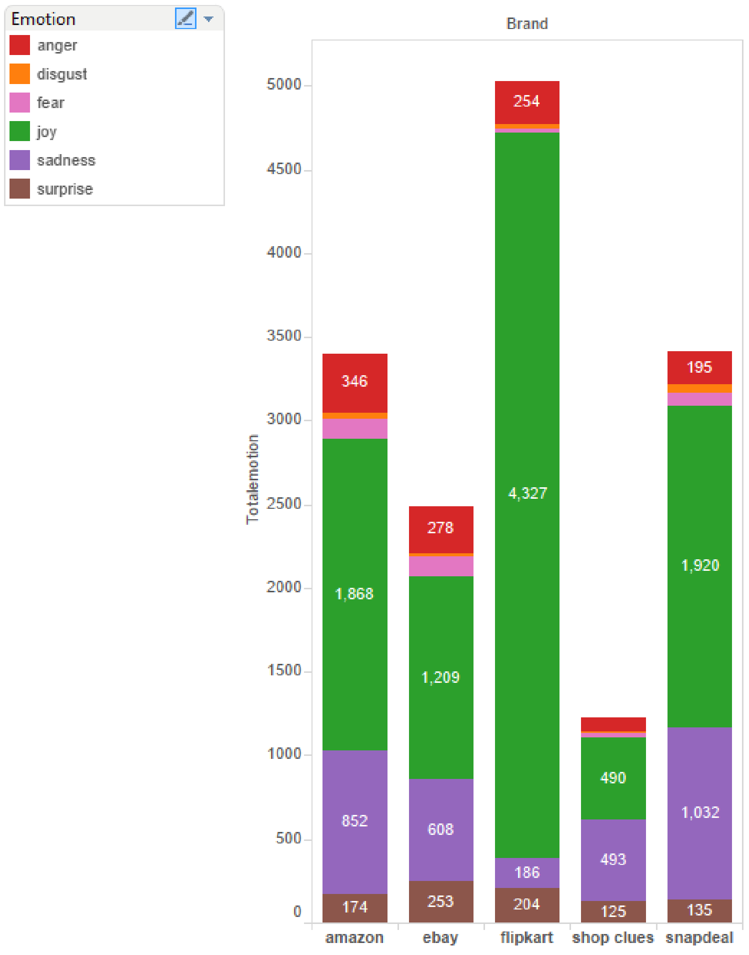


Figure 3.40 Sentiment Analysis comparison by emotions

Shop clues needs to work on better customer satisfaction – as number of joyness tweets are very less compared to other brands.

Sentiment Analysis comparison by emotion percentage

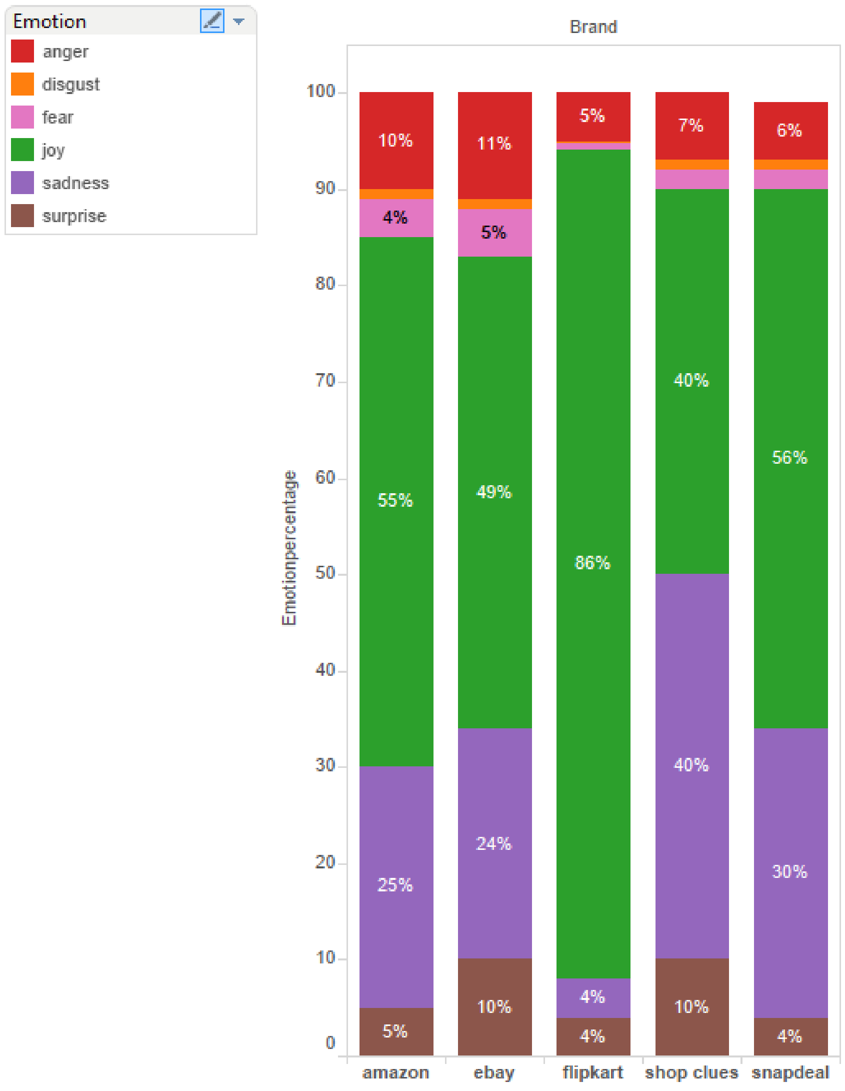


Figure 3.41 Sentiment Analysis comparison by emotions percentage

Shopclues has the highest fear percentage and flipkart has the lowest

Chapter 4

Analysis of the data for customer engagement and brand perception

Brand perception analysis

To analyse the tweets for brand perception the following KPI's or metric were calculated. And each metric were given a ranking or priority number after discussion and from various blogs based on the imporatance of the metric for brand perception.

KPIs	Definition	Priority Factor
Followers count	Followers of the brand	9
Favorite count	Number of Favorites	3
Friends count	Number of friends	5
Listed count	number of public lists where the brand is a member of	7
Polarity count	ratio of positive sentiments/negative sentiments	10

Table 4.1 Brand perception analysis metric details

Favorite count comparison of various brands

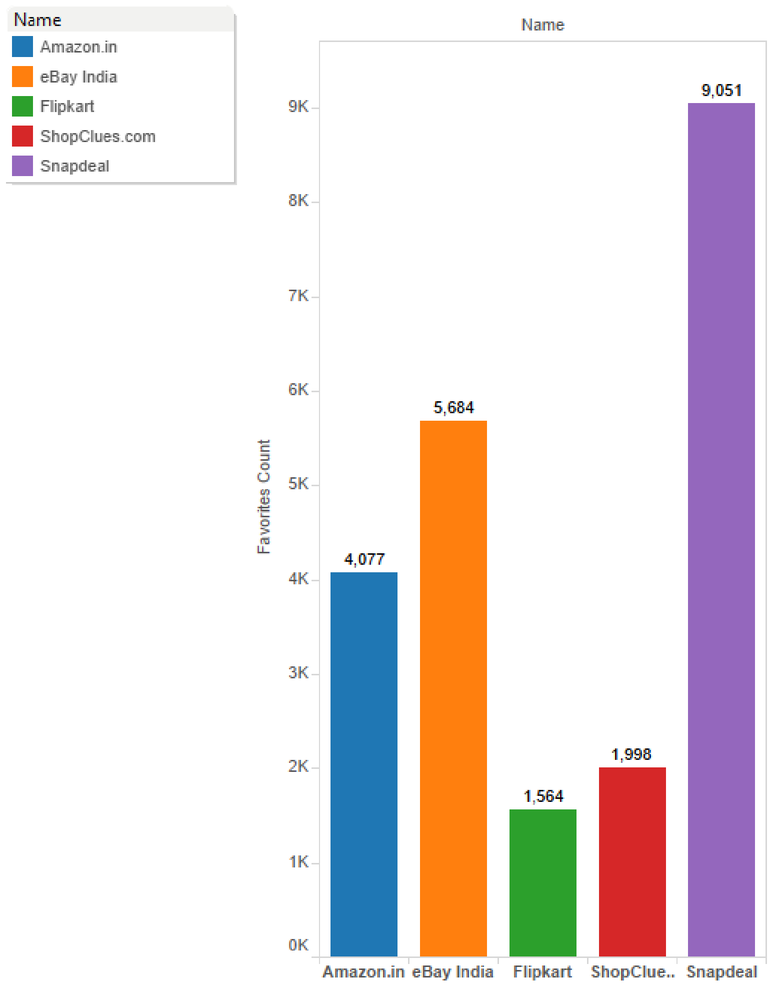


Figure 4.1 Favorite count comparison indian ecommerce

Snapdeal leads in this perspective followed by ebay india

Followers count comparison of various brands

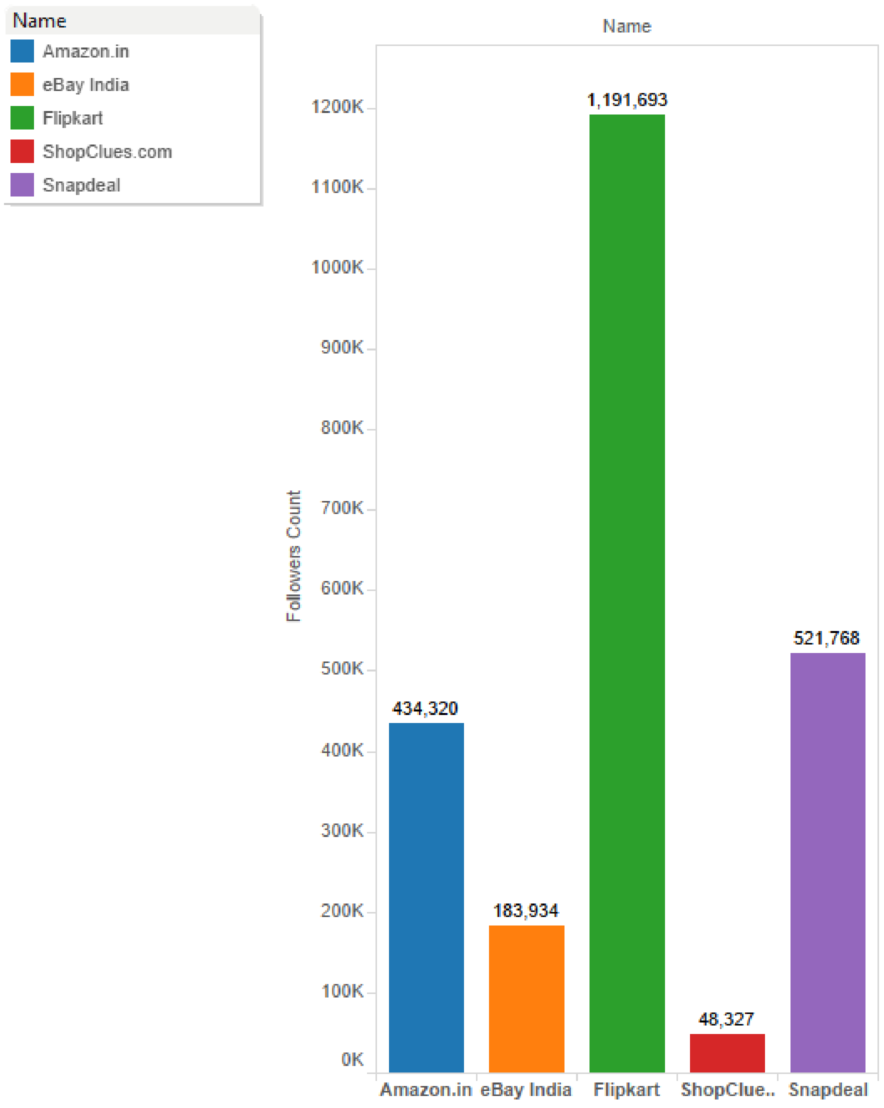


Figure 4.2 Followers count comparison indian ecommerce

Flipkart has the highest follower count followed by snapdeal and Amazon

Friends count comparison of various brands

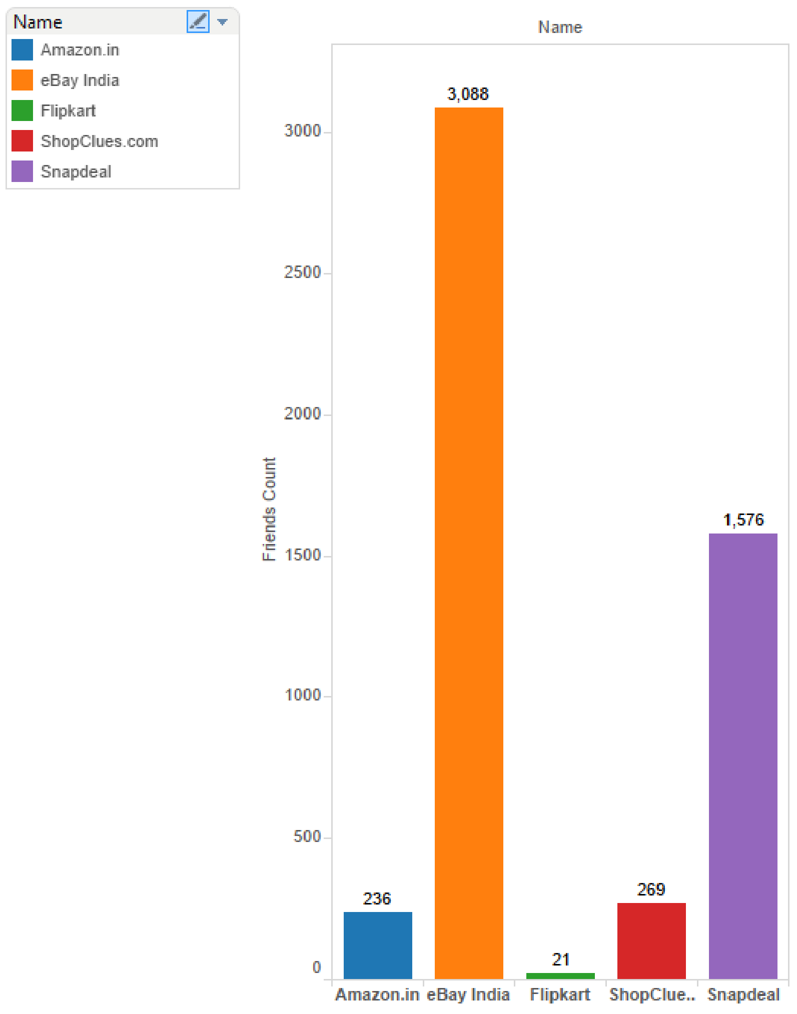


Figure 4.3 Friends count comparison indian ecommerce

Flipkart needs to improve on this aspect, followed by amazon and shopclues as well.

Listed count comparison of various brands

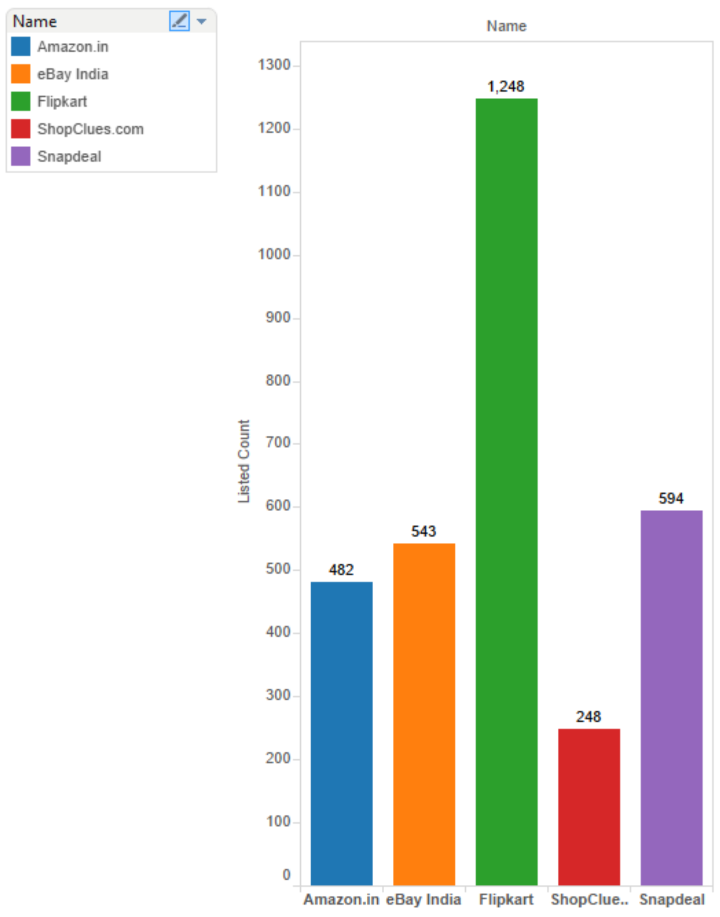


Figure 4.4 Listed count comparison indian ecommerce

Flipkart has the highest followed by snapdeal and amazon which means more customers are looking for flipkart tweets than others and so they have listed it.

Brand perception by all metrics

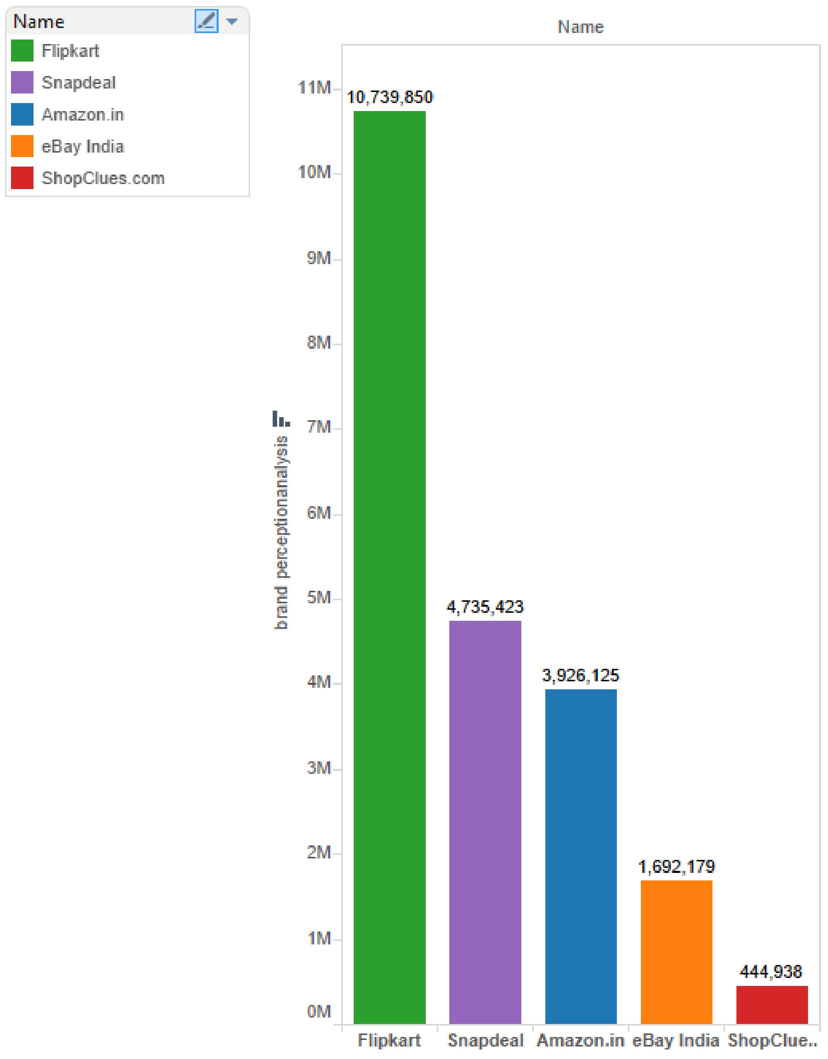


Figure 4.5 Brand perception by all metrics

As per brand perception analysis is considered Flipkart is doing good in maintaining its brand reputation and perception in social media followed by Snapdeal and AmazonIndia.

Customer engagement analysis

To analyse the customer engagement of various brands from twitter analysis the following KPI's or metrics were used. And each metric were given a ranking or priority number after discussion and from various blogs based on the imporatance of the metric for customer engagement.

KPIs	Definition	Formulas	Priority Factor
Response Rate	conversation rate/ number of replies per tweet	Total number of replies to the customer from the brand/Total number of queries from the customer	10
Amplification Rate	How frequently are you tapping into second level n/w i.e. reaching followers of followers	Total number of retweets/Total number of tweets	5
Applause Rate	helps you understand what the audience likes	Total number of favorites/Total number of tweets	9
Status count	number of posts being posted by brand		7

Table 4.2 Customer engagement analysis metric details

Retweet counts of various brands

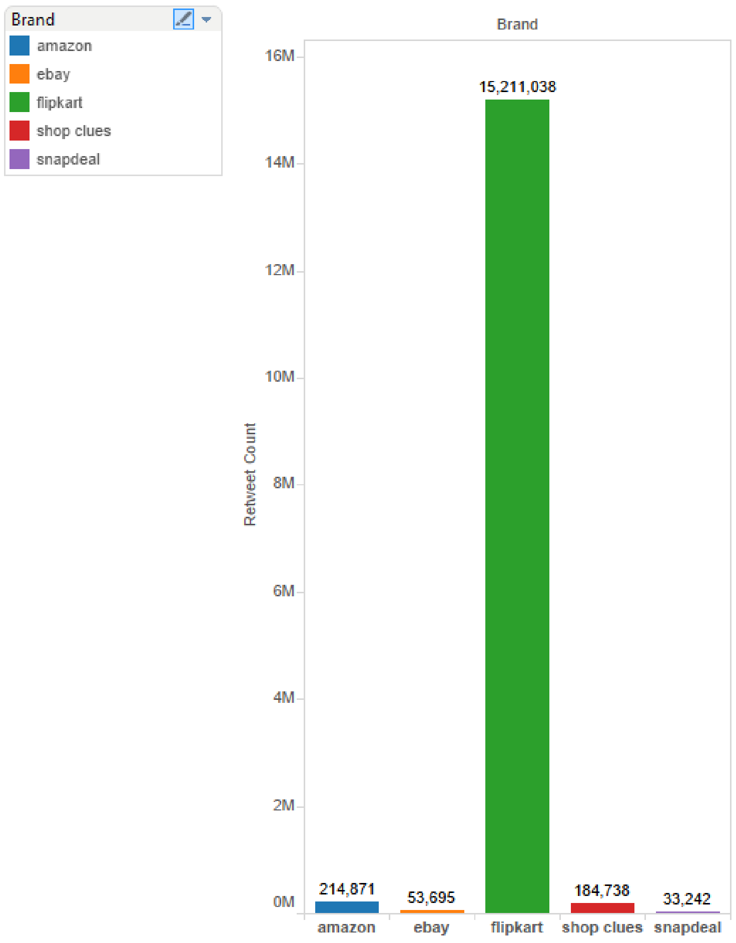


Figure 4.6 retweet count comparison of indian ecommerce

There is a high retweet of flipkart as compared to others.

Favourite count by brand

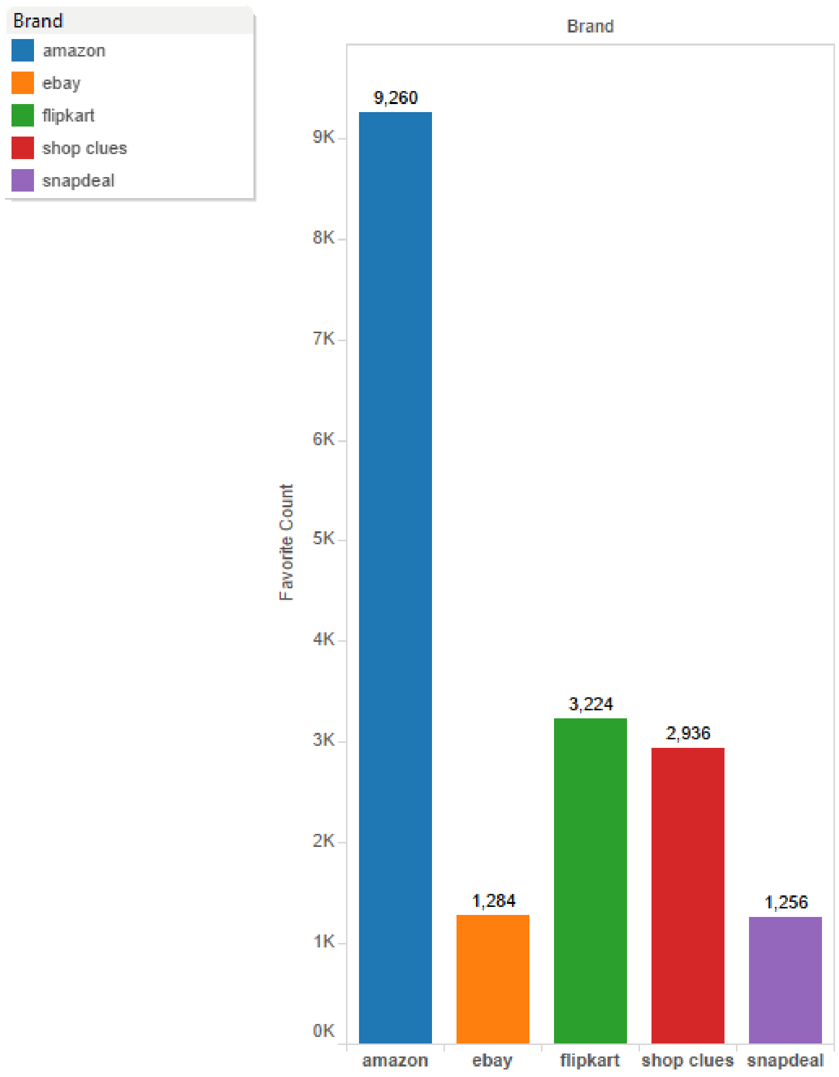


Figure 4.7 Favorite count comparison of indian ecommerce

Ebay has the least and Amazon has the highest favorite count

Query count from customer to particular brand

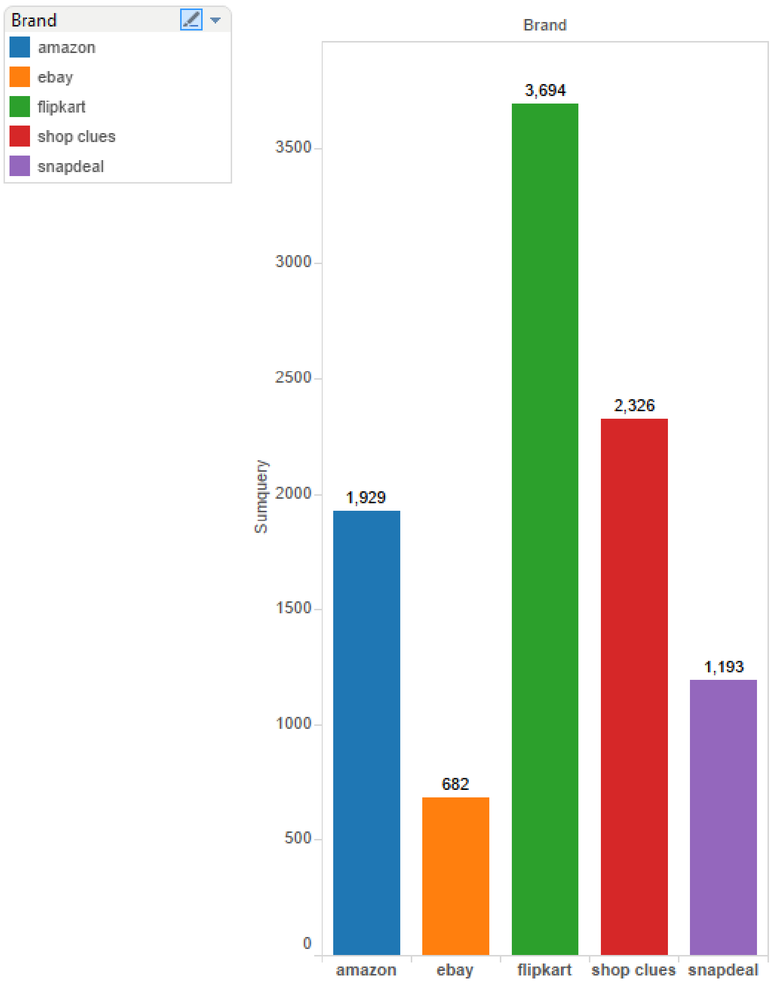


Figure 4.8 Query count comparison of indian ecommerce

Flipkart has more customer queries over twitter than rest all brand followed by Amazon and Shop Clues.

Reply count by brand given to customer

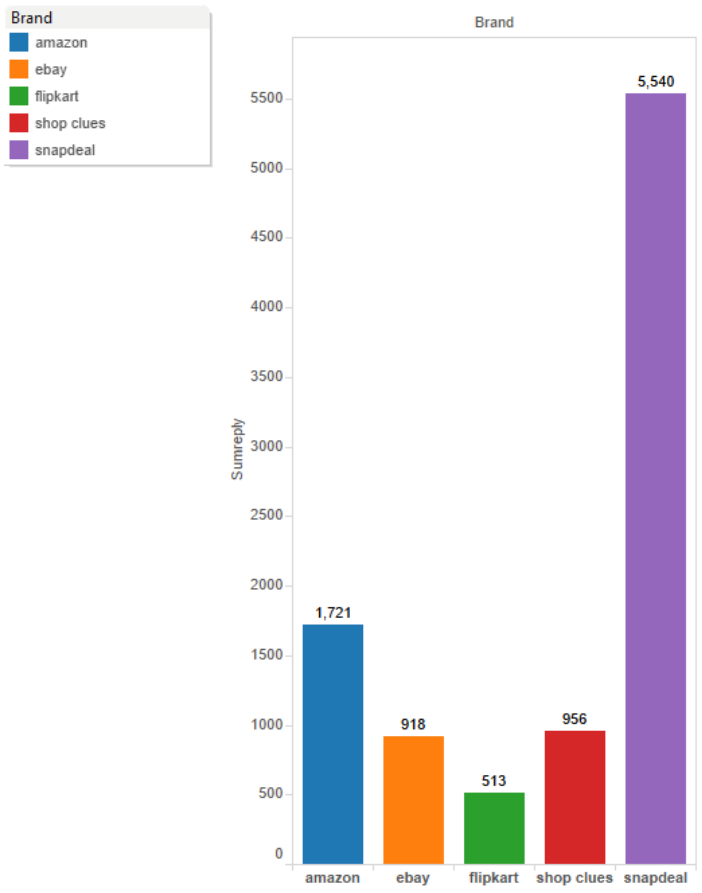


Figure 4.9 Reply count comparison of indian ecommerce

Snapdeal shows great responsiveness to their customers followed by amazon by responding to the queries of the customer where as flipkart replies 8 times less than the queries they received from the customer.

Response rate by brand

	brand	sumreply	sumquery	responserate
1	amazon	1721	1929	89.21721
2	ebay	918	682	134.6041
3	flipkart	513	3694	13.88738
4	shop clues	956	2326	41.1006
5	snapdeal	5540	1193	464.3755

Table 4.3 Response rate comparison table of indian ecommerce

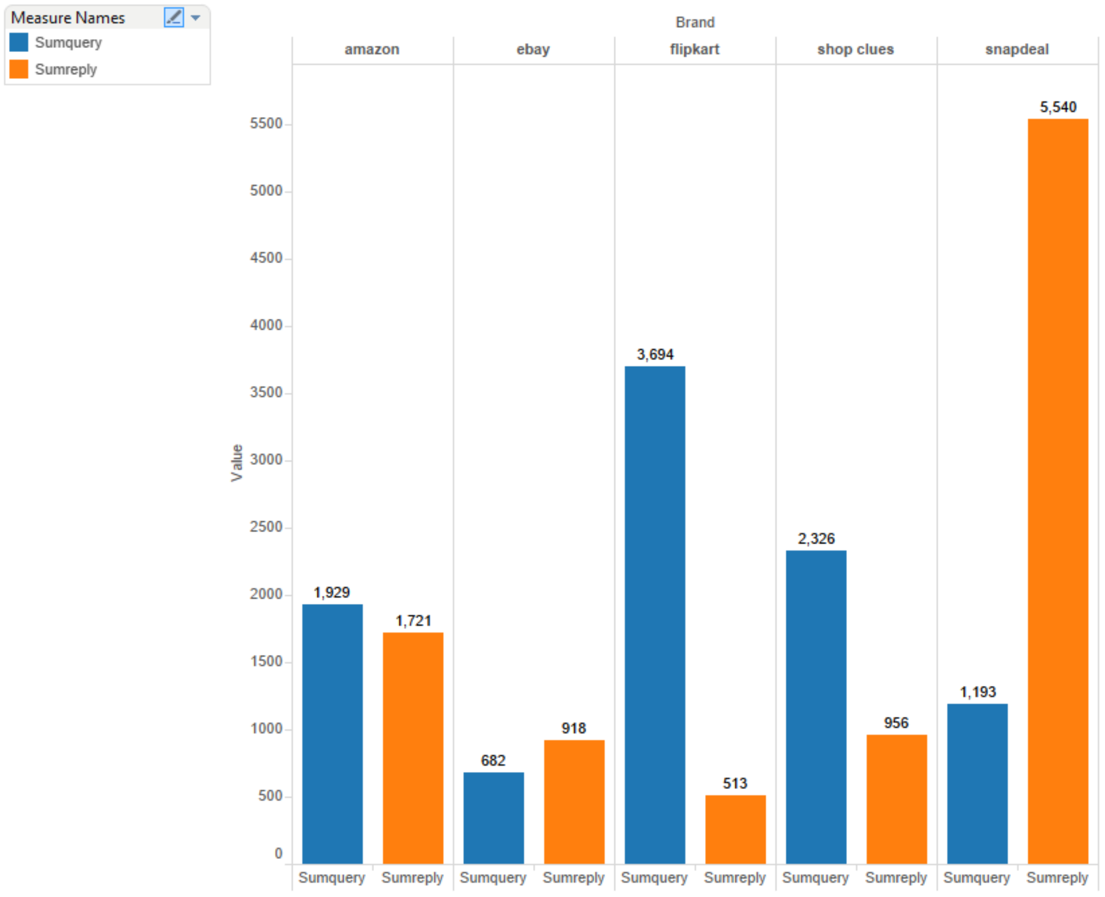


Figure 4.10 Response rate comparison of indian ecommerce

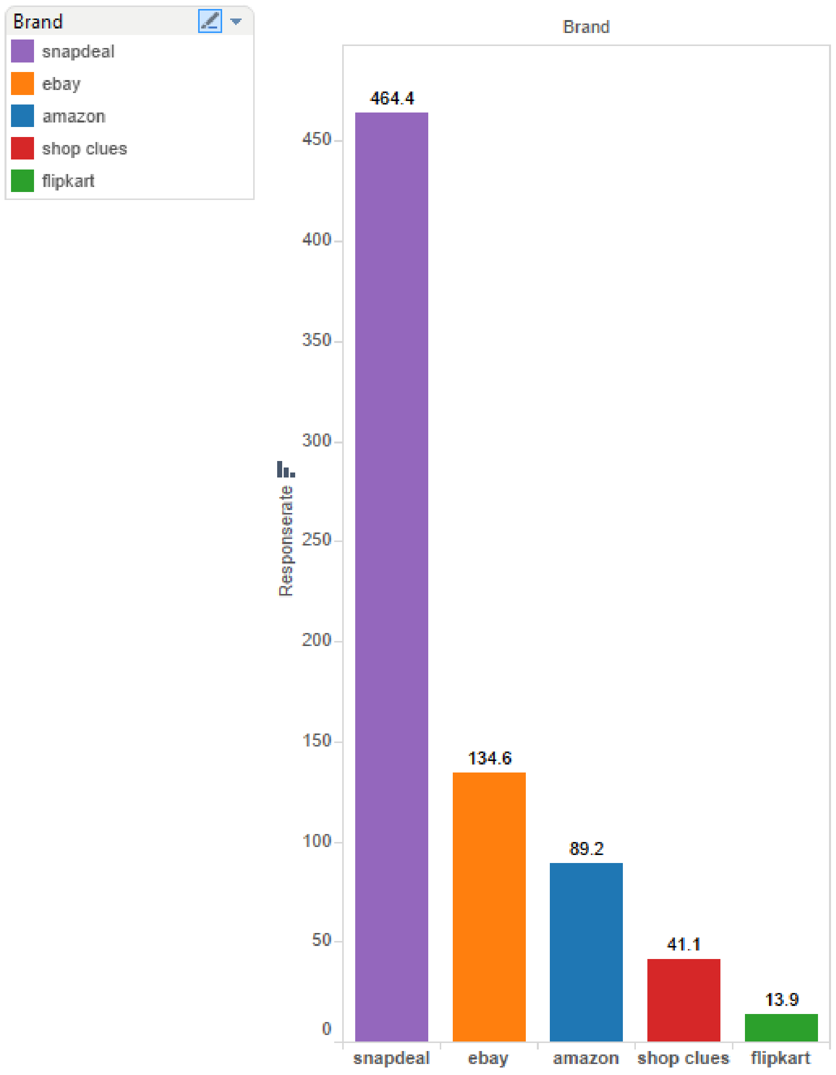


Figure 4.11 Response rate comparison of indian ecommerce (sorted)

Flipkart and shopclues needs to work hard on the reply aspect when compared to the queries they receive.Snapdeal,Ebay and Amazon perform better in responding to customer queries over twitter.

Amplification rate of various brands

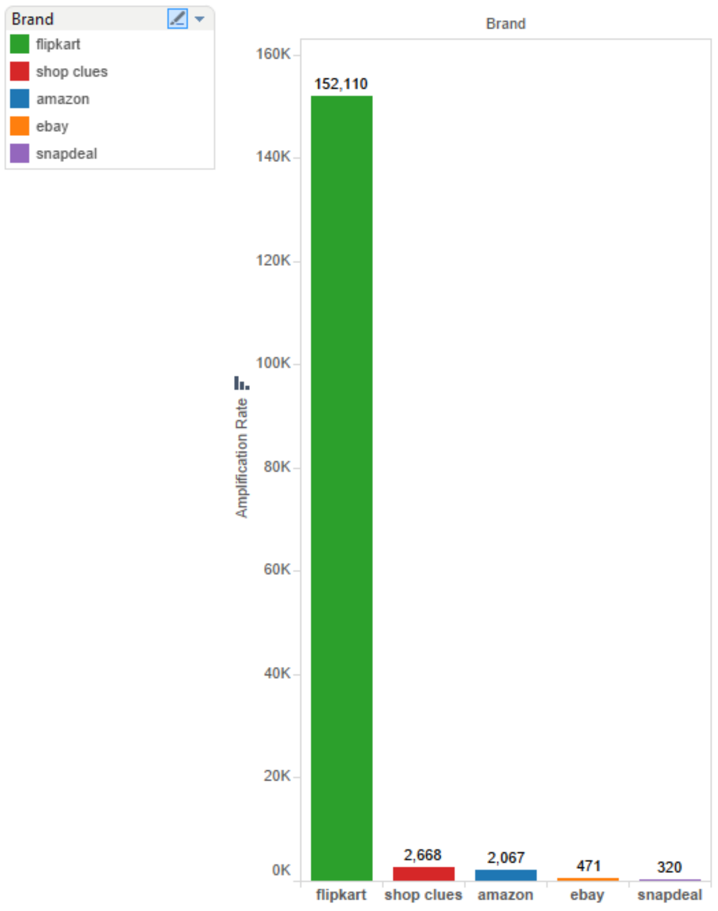


Figure 4.12 Amplification rate comparison of indian ecommerce (sorted)

Flipkart has the best amplification rate and snapdeal the least.

Applause rate of various brands

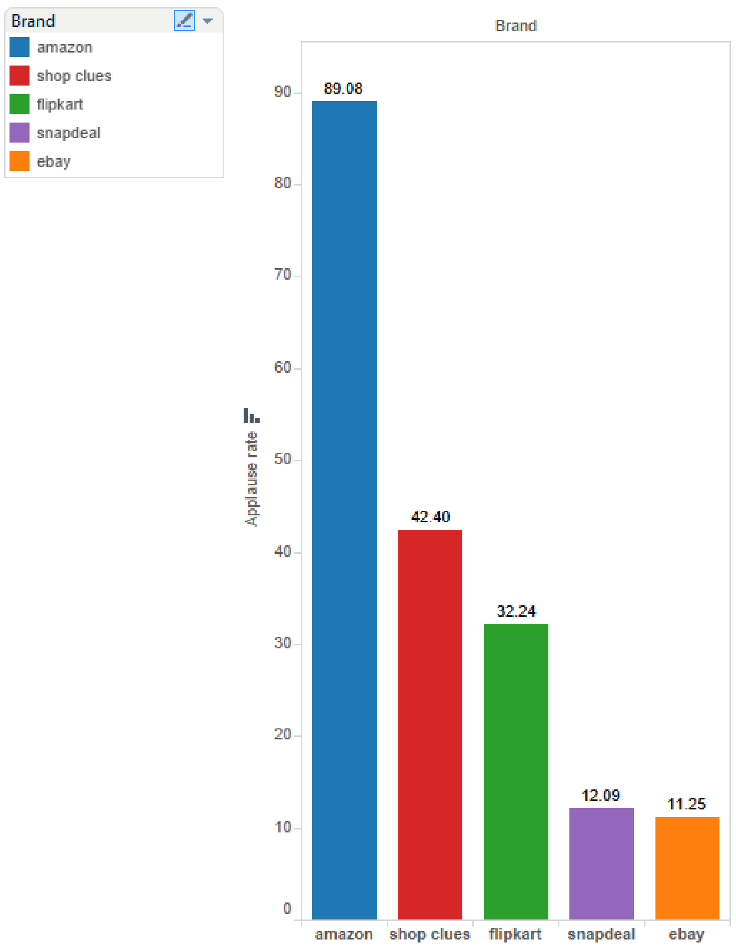


Figure 4.13 Applause rate comparison of indian ecommerce (sorted)

Applause rate is least for ebay and snapdeal and amazon does good.

Status Count of various brands

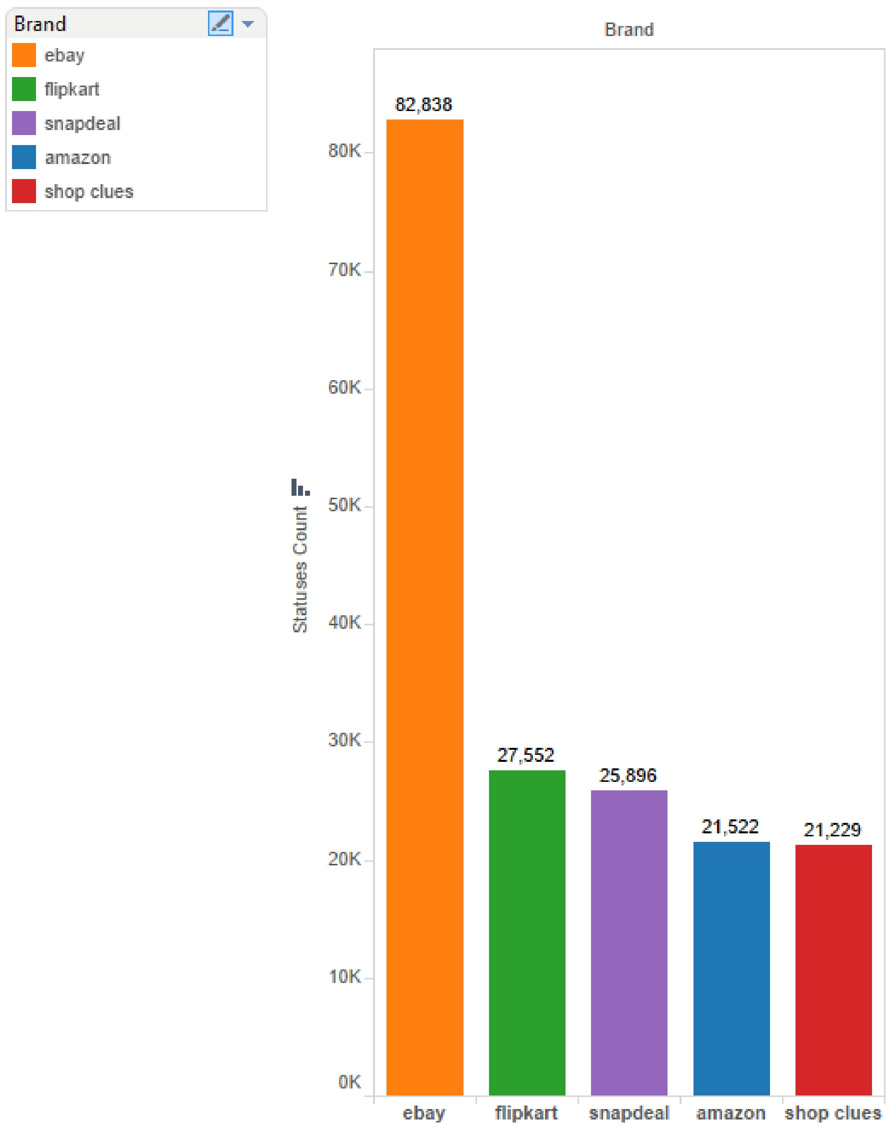


Figure 4.14 Status count comparison of indian ecommerce (sorted)

Ebay leads in this aspect followed by flipkart and snapdeal.

Customer engagement by overall metrics

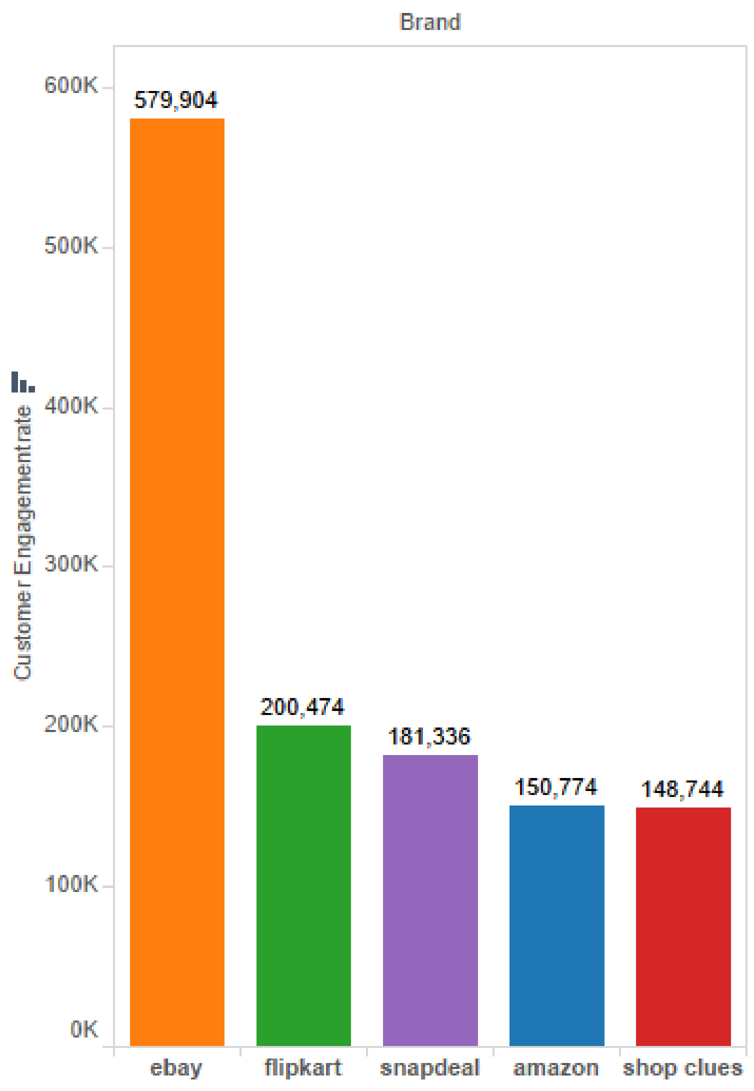


Figure 4.15 Customer engagement analysis of indian ecommerce (sorted)

The overall metrics shows amazon and shop clues need to work on this aspect.

Chapter 5

Conclusions

1. As per brand perception analysis is considered Flipkart is doing good in maintaining its brand reputation and perception in social media followed by Snapdeal and AmazonIndia. Being a big ecommerce player ebay needs to improve a lot in this aspect by continuously and actively engaging its business in social media channel like twitter. Shop Clues being a small fish should agressively build up its brand awareness and perception in social media channels like twitter.

2. As per customer engagement analysis Ebay stands at the top and its steals the show in a long mile .Amazon needs to work hard in this aspect to come at the level of Ebay. Flipkart performs beter than Amazon in Customer engagement analysis. So when done a root cause analysis it was found that the time frame in which the tweets were extracted were during Diwali season where the Amazon had seen a huge rush in customer buying behaviour and they were unable to handle so many requests. Because of this the orders of the customers during that period started piling up and late delivery started and so there was bit negative emotions among amazon users during that period of time.

Recommendations and applications

The above analysis can be used for

- Competition Analysis - Dimension analysis based on social media and news articles which will help in identifying the strengths, weaknesses, and opportunities in the market and compare the performance with competitors.
- Campaign analysis - Measure the impact of campaigns on conversations volume, brand perception & sales.
- Content Analysis - What are the trending topics on social platforms which can engage the customers more on owned social media and what contents are working or not working

- Persona Analysis: Develop personas based on demographic, psychographic sentiments
- Brand perception analysis: Calculate the brand perception index to see how the brand value is performing and changing over time.
- Product dimension analysis – What are the main attributes correlated with sentiment of the personal sets
- Market completion analysis - in which metrics and areas a brand is better or worse than the competitor
- Sentiment Analysis - sentiment analysis of content as how the audience respond to different types of contents and provide the reviews.
- Campaign analysis – Measure the performance of the campaign from the conversation rate, brand perceptions and sale.

References

- "104 Fascinating Social Media And Marketing Statistics For 2014 (And 2015)". 2016. Business 2 Community. http://www.business2community.com/social-media/104-fascinating-social-media-marketing-statistics-2014-2015-01084935#QjqLc44BxyrAMpS8.97.
- "5 Ways To Increase Mobile Marketing Conversion Rates". 2014. Launch & Hustle. http://www.launchandhustle.com/7-mobile-strategies-to-boost-conversion-rates.
- "Articles In Marketing Technology Category | Emarketer". 2016. Emarketer.Com. http://www.emarketer.com/topics/Marketing-Technology/1335.
- Brodie, R. J., Hollebeek, L. D., Jurić, B., & Ilić, A. (2011a). Customer Engagement. Journal of Service Research, 14(3), 252-271. doi: 10.1177/1094670511411703
- "Do Brands Engage Their Customers Through Social Media? - Market Community". 2016. Marketcommunity.Com. http://www.marketcommunity.com/Content/articles/MARKET-68---Brand-Engagement/Do-brands-engage-their-customers-through-social-media.
- Hennig-Thurau, T., E. C. Malthouse, C. Friege, S. Gensler, L. Lobschat, A. Rangaswamy, and B. Skiera. 2010. "The Impact Of New Media On Customer Relationships". Journal Of Service Research 13 (3): 311-330. doi:10.1177/1094670510375460.
- "ICME - Publications Telcom & Media". 2016. Icme.Com. http://www.icme.com/publications-telecom-and-media.htm.
- Improving Revenue And Customer Engagement With Social Media Analytics. 2016. Ebook. 1st ed. saas. http://www.sas.com/content/dam/SAS/en_us/doc/whitepaper2/improving-revenue-customer-engagement-social-media-105004.pdf.
- Kumar, V., Aksoy, L., Donkers, B., Venkatesan, R., Wiesel, T., & Tillmanns, S. (2010). Undervalued or Overvalued Customers: Capturing Total Customer

Engagement Value. Journal of Service Research, 13(3), 297-310. doi: 10.1177/1094670510375602

- Nelson-Field, K., & Taylor, J. (2012). Facebook fans: A fan for life? Admap, 25-27
- Ryan, D., & Jones, C. (2011). Best Digital Marketing Campaigns in the World: Mastering The Art of Customer Engagement. Retrieved from http://aut.eblib.com.au/patron/FullRecord.aspx?p=730278
- "Social Media And Mobile Marketing: A Marriage Made In Heaven - Launch & Hustle". 2014. Launch & Hustle. http://www.launchandhustle.com/social-media-mobile-marketing-marriage-made-heaven.
- Stambor, Zak. 2016. "Tumblr Plunges Into E-Commerce". Internetretailer.Com. https://www.internetretailer.com/2014/12/04/tumblr-plunges-e-commerce.
- WARC. (2012b). Brands need Facebook "stickiness". Retrieved August 13, 2012, from http://www.warc.com/LatestNews/News/EmailNews.news?ID=29986&Origin=WA RCNewsEmail

Appendix A

step1_capstone_collection_twitter.r

```
		###import required packages
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(data.table)

setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")


consumer_key = 'vitqhIlWGWfRUOUP8JIixb7OB'
consumer_secret = 'gJrNUmTl99eU3bTpUQTnBhO1zs3TST1qluwSjyozfkTm4FOGjW'
access_token = '3306555647-NpzZt0Af4T06E5ea9EWlVQeGRfE4wF4fvQBOBMg'
access_token_secret = '5Y4ojfc0LO8UtHmjrGVdhMlsF9AjkjWIxluBmofH45v2y'


setup_twitter_oauth(consumer_key,consumer_secret,access_token, access_token_secret)



###Amazon data collection
Sys.sleep(5000)
amazon_handle<-searchTwitteR("@amazonIN",n = 10000,lang = "en",since='2016-11-
01')
amazon_handle.df <- do.call(rbind, lapply(amazon_handle, as.data.frame))
save(amazon_handle, file = "amazon_handle.RData")
write.csv(amazon_handle.df,"amazon_handle.csv")

amazon_word<-searchTwitteR("amazonIN",n = 4000,lang = "en",since='2016-11-01')
amazon_word.df <- do.call(rbind, lapply(amazon_word, as.data.frame))
save(amazon_word, file = "amazon_word.RData")
write.csv(amazon_word.df,"amazon_word.csv")

amazon_combined.df= rbind(amazon_handle.df,amazon_word.df)
save(amazon_combined.df, file = "amazon_combined.RData")
write.csv(amazon_combined.df,"amazon_combined.csv")

####Flipkart data collection
flipkart_handle<-searchTwitteR("@Flipkart",n = 10000,lang = "en",since='2016-11-01')
save(flipkart_handle, file = "flipkart_handle.RData")
```

```
flipkart_handle.df <- do.call(rbind, lapply(flipkart_handle, as.data.frame))
write.csv(flipkart_handle.df,"flipkart_handle.csv")



####Snapdeal data collection
snapdeal_handle<-searchTwitteR("@snapdeal",n = 10000,lang = "en",since='2016-11-
01')
save(snapdeal_handle, file = "snapdeal_handle.RData")
snapdeal_handle.df <- do.call(rbind, lapply(snapdeal_handle, as.data.frame))
write.csv(snapdeal_handle.df,"snapdeal_handle.csv")

snapdeal_hashtag<-searchTwitteR("#snapdeal",n = 3000,lang = "en",since='2016-11-01')
save(snapdeal_hashtag, file = "snapdeal_hashtag.RData")
snapdeal_hashtag.df <- do.call(rbind, lapply(snapdeal_hashtag, as.data.frame))
write.csv(snapdeal_hashtag.df,"snapdeal_hash.csv")

snapdeal_word<-searchTwitteR("snapdeal",n = 3000,lang = "en",since='2016-11-01')
save(snapdeal_word, file = "snapdeal_word.RData")
snapdeal_word.df <- do.call(rbind, lapply(snapdeal_word, as.data.frame))
write.csv(snapdeal_word.df,"snapdeal_word.csv")


snapdeal_help<-searchTwitteR("@Snapdeal_Help",n = 3000,lang = "en",since='2016-11-
01')
save(snapdeal_help, file = "snapdeal_help.RData")
snapdeal_help.df <- do.call(rbind, lapply(snapdeal_help, as.data.frame))
write.csv(snapdeal_help.df,"snapdeal_help.csv")

snapdeal_combined.df                                                          =
rbind(snapdeal_handle.df,snapdeal_hashtag.df,snapdeal_word.df,snapdeal_help.df)
write.csv(snapdeal_combined.df,"snapdeal_combined.csv")
save(snapdeal_combined, file = "snapdeal_combined.RData")


####Ebay data collection
Sys.sleep(5000)
ebay_handle<-searchTwitteR("@ebayindia",n = 10000,lang = "en",since='2016-11-01')
save(ebay_handle, file = "ebay_handle.RData")
ebay_handle.df <- do.call(rbind, lapply(ebay_handle, as.data.frame))
write.csv(ebay_handle.df,"ebay_handle.csv")

ebay_hash<-searchTwitteR("#ebay",n = 10000,lang = "en",since='2016-11-01')
save(ebay_hash, file = "ebay_hash.RData")
ebay_hash.df <- do.call(rbind, lapply(ebay_hash, as.data.frame))
```

```
write.csv(ebay_hash.df,"ebay_hash.csv")


ebay_word<-searchTwitteR("ebay",n = 3000,lang = "en",since='2016-11-01')
save(ebay_word, file = "ebay_word.RData")
ebay_word.df <- do.call(rbind, lapply(ebay_word, as.data.frame))
write.csv(ebay_word.df,"ebay_word.csv")

ebay_combined.df = rbind(ebay_handle.df,ebay_hash.df,ebay_word.df)
write.csv(ebay_combined.df,"ebay_combined.csv")
save(ebay_combined.df, file = "ebay_combined.RData")


####shopclues data collection
sc_handle<-searchTwitteR("@ShopClues  ",n = 10000,lang = "en",since='2016-11-01')
save(sc_handle, file = "sc_handle.RData")
sc_handle.df <- do.call(rbind, lapply(sc_handle, as.data.frame))
write.csv(ebay_handle.df,"sc_handle.csv")

sc_word<-searchTwitteR("ShopClues  ",n = 10000,lang = "en",since='2016-11-01')
save(sc_word, file = "sc_word.RData")
sc_word.df <- do.call(rbind, lapply(sc_word, as.data.frame))
write.csv(sc_word.df,"sc_word.csv")

sc_hash<-searchTwitteR("#ShopClues  ",n = 10000,lang = "en",since='2016-11-01')
save(sc_hash, file = "sc_hash.RData")
sc_hash.df <- do.call(rbind, lapply(sc_hash, as.data.frame))
write.csv(sc_hash.df,"sc_hash.csv")

sc_hash2<-searchTwitteR("#ShopClues!  ",n = 10000,lang = "en",since='2016-11-01')
save(sc_hash2, file = "sc_hash2.RData")
sc_hash2.df <- do.call(rbind, lapply(sc_hash2, as.data.frame))
write.csv(sc_hash2.df,"sc_hash2.csv")


sc_combined.df<-rbind(sc_handle.df,sc_word.df,sc_hash.df,sc_hash2.df)
write.csv(sc_combined.df,"sc_combined.csv")
save(sc_combined.df, file = "sc_combined.RData")
```

Appendix B

step2_capstone_data_cleaning.r

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")

#Clean amazon data
amazon_combined.df=read.csv("amazon_combined.csv",header =
TRUE,stringsAsFactors = FALSE,sep = ",")
amazon_combined.df=amazon_combined.df[,-c(1)]
# Remove character string between < >
amazon_combined.df$text=genX(amazon_combined.df$text, " <", ">")

# Create document corpus with tweet text
amazon_combined.corpus<- Corpus(VectorSource(amazon_combined.df$text))

#####convert to Lowercase
amazon_combined.corpus <- tm_map(amazon_combined.corpus,
content_transformer(stri_trans_tolower))
```

```
#####Remove the links (URLs)
removeURL <- function(x) gsub("http[^[:space:]]*", "", x)
amazon_combined.corpus <- tm_map(amazon_combined.corpus,
content_transformer(removeURL))

#####Remove anything except the english language and space
removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)
amazon_combined.corpus <- tm_map(amazon_combined.corpus,
content_transformer(removeNumPunct))

#####Remove Stopwords
myStopWords<- c((stopwords('english')),c("rt", "use", "used", "via",
"amp","should","go","then","what","when","where","than","amazon","amazonIn"))
amazon_combined.corpus <- tm_map(amazon_combined.corpus,
removeWords,myStopWords)

#####Remove Single letter words
removeSingle <- function(x) gsub(" . ", " ", x)
amazon_combined.corpus <- tm_map(amazon_combined.corpus,
content_transformer(removeSingle))

#####Remove Extra Whitespaces
amazon_combined.corpus<- tm_map(amazon_combined.corpus, stripWhitespace)
#####keep a copy of "amazon.corpus" for stem completion later
amazon_combined.corpus.copy<- amazon_combined.corpus
save(amazon_combined.corpus.copy, file = "amazon_combined.corpus.copy.RData")


#Clean flipkart data
flipkart_handle.df=read.csv("flipkart_handle.csv",header = TRUE,stringsAsFactors =
FALSE,sep = ",")
flipkart_handle.df=flipkart_handle.df[,-c(1)]
# Remove character string between < >
flipkart_handle.df$text=genX(flipkart_handle.df$text, " <", ">")

# Create document corpus with tweet text
flipkart_handle.df.corpus<- Corpus(VectorSource(flipkart_handle.df$text))

#####convert to Lowercase
flipkart_handle.df.corpus <- tm_map(flipkart_handle.df.corpus,
content_transformer(stri_trans_tolower))

#####Remove the links (URLs)
removeURL <- function(x) gsub("http[^[:space:]]*", "", x)
```

```
flipkart_handle.df.corpus <- tm_map(flipkart_handle.df.corpus,
content_transformer(removeURL))

#####Remove anything except the english language and space
removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)
flipkart_handle.df.corpus <- tm_map(flipkart_handle.df.corpus,
content_transformer(removeNumPunct))

#####Remove Stopwords
myStopWords<- c((stopwords('english')),c("rt", "use", "used", "via",
"amp","should","go","then","what","when","where","than","flipkart"))
flipkart_handle.df.corpus <- tm_map(flipkart_handle.df.corpus,
removeWords,myStopWords)

#####Remove Single letter words
removeSingle <- function(x) gsub(" . ", " ", x)
flipkart_handle.df.corpus <- tm_map(flipkart_handle.df.corpus,
content_transformer(removeSingle))

#####Remove Extra Whitespaces
flipkart_handle.df.corpus<- tm_map(flipkart_handle.df.corpus, stripWhitespace)
#####keep a copy of "amazon.corpus" for stem completion later
flipkart_handle.df.corpus.copy<- flipkart_handle.df.corpus
save(flipkart_handle.df.corpus.copy, file = "flipkart_handle.df.corpus.copy.RData")


# Clean snapdeal
snapdeal_combined.df=read.csv("snapdeal_combined.csv",header =
TRUE,stringsAsFactors = FALSE,sep = ",")
snapdeal_combined.df=snapdeal_combined.df[,-c(1)]
# Remove character string between < >
snapdeal_combined.df$text=genX(snapdeal_combined.df$text, " <", ">")

# Create document corpus with tweet text
snapdeal_combined.df.corpus<- Corpus(VectorSource(snapdeal_combined.df$text))

#####convert to Lowercase
snapdeal_combined.df.corpus <- tm_map(snapdeal_combined.df.corpus,
content_transformer(stri_trans_tolower))

#####Remove the links (URLs)
removeURL <- function(x) gsub("http[^[:space:]]*", "", x)
snapdeal_combined.df.corpus <- tm_map(snapdeal_combined.df.corpus,
content_transformer(removeURL))
```

```
#####Remove anything except the english language and space
removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)
snapdeal_combined.df.corpus <- tm_map(snapdeal_combined.df.corpus,
content_transformer(removeNumPunct))

#####Remove Stopwords
myStopWords<- c((stopwords('english')),c("rt", "use", "used", "via",
"amp","should","go","then","what","when","where","than","snapdeal"))
snapdeal_combined.df.corpus <- tm_map(snapdeal_combined.df.corpus,
removeWords,myStopWords)

#####Remove Single letter words
removeSingle <- function(x) gsub(" . ", " ", x)
snapdeal_combined.df.corpus <- tm_map(snapdeal_combined.df.corpus,
content_transformer(removeSingle))

#####Remove Extra Whitespaces
snapdeal_combined.df.corpus<- tm_map(snapdeal_combined.df.corpus, stripWhitespace)
#####keep a copy of "amazon.corpus" for stem completion later
snapdeal_combined.df.corpus.copy<- snapdeal_combined.df.corpus
save(snapdeal_combined.df.corpus.copy, file =
"snapdeal_combined.df.corpus.copy.RData")



#Cleaning ebay data

ebay_combined.df=read.csv("ebay_combined.csv",header = TRUE,stringsAsFactors =
FALSE,sep = ",")
ebay_combined.df=ebay_combined.df[,-c(1)]
# Remove character string between < >
ebay_combined.df$text=genX(ebay_combined.df$text, " <", ">")

# Create document corpus with tweet text
ebay_combined.df.corpus<- Corpus(VectorSource(ebay_combined.df$text))

#####convert to Lowercase
ebay_combined.df.corpus <- tm_map(ebay_combined.df.corpus,
content_transformer(stri_trans_tolower))

#####Remove the links (URLs)
removeURL <- function(x) gsub("http[^[:space:]]*", "", x)
ebay_combined.df.corpus <- tm_map(ebay_combined.df.corpus,
content_transformer(removeURL))
```

```
#####Remove anything except the english language and space
removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)
ebay_combined.df.corpus <- tm_map(ebay_combined.df.corpus,
content_transformer(removeNumPunct))

#####Remove Stopwords
myStopWords<- c((stopwords('english')),c("rt", "use", "used", "via",
"amp","should","go","then","what","when","where","than","ebay"))
ebay_combined.df.corpus <- tm_map(ebay_combined.df.corpus,
removeWords,myStopWords)

#####Remove Single letter words
removeSingle <- function(x) gsub(" . ", " ", x)
ebay_combined.df.corpus <- tm_map(ebay_combined.df.corpus,
content_transformer(removeSingle))

#####Remove Extra Whitespaces
ebay_combined.df.corpus<- tm_map(ebay_combined.df.corpus, stripWhitespace)
#####keep a copy of "amazon.corpus" for stem completion later
ebay_combined.df.corpus.copy<- ebay_combined.df.corpus
save(ebay_combined.df.corpus.copy, file = "ebay_combined.df.corpus.copy.RData")



#Cleaning shop clues  data

sc_combined.df=read.csv("sc_combined.csv",header = TRUE,stringsAsFactors =
FALSE,sep = ",")
sc_combined.df=sc_combined.df[,-c(1)]
# Remove character string between < >
sc_combined.df$text=genX(sc_combined.df$text, " <", ">")

# Create document corpus with tweet text
sc_combined.df.corpus<- Corpus(VectorSource(sc_combined.df$text))

#####convert to Lowercase
sc_combined.df.corpus <- tm_map(sc_combined.df.corpus,
content_transformer(stri_trans_tolower))

#####Remove the links (URLs)
removeURL <- function(x) gsub("http[^[:space:]]*", "", x)
sc_combined.df.corpus <- tm_map(sc_combined.df.corpus,
content_transformer(removeURL))

#####Remove anything except the english language and space
```

```
removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)
sc_combined.df.corpus <- tm_map(sc_combined.df.corpus,
content_transformer(removeNumPunct))

#####Remove Stopwords
myStopWords<- c((stopwords('english')),c("rt", "use", "used", "via",
"amp","should","go","then","what","when","where","than","shopClues","ShopClues"))
sc_combined.df.corpus <- tm_map(sc_combined.df.corpus, removeWords,myStopWords)

#####Remove Single letter words
removeSingle <- function(x) gsub(" . ", " ", x)
sc_combined.df.corpus <- tm_map(sc_combined.df.corpus,
content_transformer(removeSingle))

#####Remove Extra Whitespaces
sc_combined.df.corpus<- tm_map(sc_combined.df.corpus, stripWhitespace)
#####keep a copy of "amazon.corpus" for stem completion later
sc_combined.df.corpus.copy<- sc_combined.df.corpus
save(sc_combined.df.corpus.copy, file = "sc_combined.df.corpus.copy.RData")
```

Appendix C

step3_amazon_stem_completion.r

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
```

```
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")

load("amazon_combined.corpus.copy.RData")
amazon_combined.corpus = amazon_combined.corpus.copy

#####Stem words in the corpus
amazon_combined.corpus=tm_map(amazon_combined.corpus, stemDocument)

writeLines(strwrap(amazon_combined.corpus[[250]]$content,60))

##### inspect the first 100 documents (tweets)
##### The code below is used for to make text fit for paper width
##### put the output to a file for inspection
#put the output back to

#####The above was used for our analysis,as this would print all the 8107 values so for documenting shake we will look up only first 5  lines
# for (i in c(1:5, 8107)) {
#   cat(paste0("[", i, "] "))
#   writeLines(strwrap(as.character(amazon_combined_twitter.corpus[[i]]), 60) )
# }

#####Function to correct/complete the text after stemming

stemCompletion2 <- function(x, dictionary) {
  x <- unlist(strsplit(as.character(x), " "))
  # Unexpectedly, stemCompletion completes an empty string to
  # a word in dictionary. Remove empty string to avoid above issue.
  x <- x[x != ""]
  x <- stemCompletion(x, dictionary=dictionary)
  x <- paste(x, sep="", collapse=" ")
  PlainTextDocument(stripWhitespace(x))
}

#####Stem Complete and Display the same tweet above with the completed and corrected text.
amazon_combined.corpus=lapply(amazon_combined.corpus,stemCompletion2,
dictionary=amazon_combined.corpus.copy)

##convert the list to Corpus again
```

```
amazon_combined.corpus= Corpus(VectorSource(amazon_combined.corpus))

sink("stemwords_amazon_combined_after.txt")
for (i in c(1:3000, 10394)) {
  cat(paste0("[", i, "] "))
  #put the output to a file for inspection
  writeLines(strwrap(as.character(amazon_combined.corpus[[i]]), 60) )
}
sink()

#####Correcting mis-spelt words

replaceWord <- function(corpus, oldword, newword)
{
  tm_map(corpus, gsub, pattern=oldword, replacement=newword)
}

amazon_combined.corpus<- replaceWord(amazon_combined.corpus, "amazon", "")
amazon_combined.corpus<- replaceWord(amazon_combined.corpus, "helpfacts", "help facts")
amazon_combined.corpus<- replaceWord(amazon_combined.corpus, "amazonin", "")
amazon_combined.corpus<- replaceWord(amazon_combined.corpus, "amazonhelp", "")
amazon_combined.corpus<- replaceWord(amazon_combined.corpus, "experi", "experiment")
amazon_combined.corpus<- replaceWord(amazon_combined.corpus, "backhvnt", "back havent")
amazon_combined.corpus<- replaceWord(amazon_combined.corpus, "recvd", "received")
amazon_combined.corpus<- replaceWord(amazon_combined.corpus, "delivary", "delivery")
amazon_combined.corpus<- replaceWord(amazon_combined.corpus, "definitaly", "definitely")
amazon_combined.corpus<- replaceWord(amazon_combined.corpus, "referandearn", "refer and earn")
amazon_combined.corpus<- replaceWord(amazon_combined.corpus, "festiva", "festival")
amazon_combined.corpus<- replaceWord(amazon_combined.corpus, "deliveries", "delivery")
#####remove unwanted words
amazon_combined.corpus <- tm_map(amazon_combined.corpus, removeWords, c("also", "article", "Article", "download", "google", "figure","fig", "groups","Google", "however","high", "human", "levels","larger", "may", "number","shown", "study", "studies", "this","using", "two", "the", "Scholar","pubmedncbi", "PubMedNCBI","view", "View", "the", "biol","via", "image", "doi", "one",
"analysis","how","which","when","there","amazonin","the","th","for","then","when"))


sink("stemwords_amazon_combined_after1.txt")
for (i in c(1:3000, 10394)) {
  cat(paste0("[", i, "] "))
  #put the output to a file for inspection
  writeLines(strwrap(as.character(amazon_combined.corpus[[i]]), 60) )
}
sink()

#####save the cleaned corpus for future epidemics:is very important else we have to start from begining
amazon_combined.corpus.cleaned = amazon_combined.corpus
save.image(file="amazon_combined.corpus.cleaned.RData")
```

Appendix D

step3_ebay_stem_completion.r

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
```

```
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")

load("ebay_combined.df.corpus.copy.RData")
ebay_combined.df.corpus = ebay_combined.df.corpus.copy

#####Stem words in the corpus
ebay_combined.df.corpus=tm_map(ebay_combined.df.corpus, stemDocument)

writeLines(strwrap(ebay_combined.df.corpus[[250]]$content,60))

stemCompletion2 <- function(x, dictionary) {
  x <- unlist(strsplit(as.character(x), " "))
  # Unexpectedly, stemCompletion completes an empty string to
  # a word in dictionary. Remove empty string to avoid above issue.
  x <- x[x != ""]
  x <- stemCompletion(x, dictionary=dictionary)
  x <- paste(x, sep="", collapse=" ")
  PlainTextDocument(stripWhitespace(x))
}
```

#####Stem Complete and Display the same tweet above with the completed and corrected text.

```
ebay_combined.df.corpus=lapply(ebay_combined.df.corpus,stemCompletion2,
dictionary=ebay_combined.df.corpus.copy)

##convert the list to Corpus again
ebay_combined.df.corpus= Corpus(VectorSource(ebay_combined.df.corpus))

sink("stemwords_ebay_combined_after.txt")
for (i in c(1:3000, 11408)) {
  cat(paste0("[", i, "] "))
  #put the output to a file for inspection
  writeLines(strwrap(as.character(ebay_combined.df.corpus[[i]]), 60) )
}
sink()
```

#####Correcting mis-spelt words

```
replaceWord <- function(corpus, oldword, newword)
{
  tm_map(corpus, gsub, pattern=oldword, replacement=newword)
}

ebay_combined.df.corpus<- replaceWord(ebay_combined.df.corpus, "ebay", "")
ebay_combined.df.corpus<- replaceWord(ebay_combined.df.corpus, "ebayindia", "")
ebay_combined.df.corpus<- replaceWord(ebay_combined.df.corpus, "deliv", "delivery")
ebay_combined.df.corpus<- replaceWord(ebay_combined.df.corpus, "delivering",
"delivery")
```

```
#####remove unwanted words
ebay_combined.df.corpus <- tm_map(ebay_combined.df.corpus, removeWords, c("also", "article", "Article", "download", "google", "figure","fig", "groups","Google", "however","high", "human", "levels","larger", "may", "number","shown", "study", "studies", "this","using", "two", "the", "Scholar","pubmedncbi", "PubMedNCBI","view", "View", "the", "biol","via", "image", "doi", "one",

"analysis","how","which","when","there","myntra","the","th","for","then","when","httpst
codfvdjzyy"))



sink("stemwords_myntra_combined_after1.txt")
for (i in c(1:100, 5047)) {
  cat(paste0("[", i, "] "))
  #put the output to a file for inspection
  writeLines(strwrap(as.character(myntra_combined.corpus[[i]]), 60) )
}
sink()

#####save the cleaned corpus for future epidemics:is very important else we have to start
from begining
ebay_combined.df.corpus.cleaned = ebay_combined.df.corpus
save.image(file="ebay_combined.df.corpus.cleaned.RData")
```

Appendix E

step3_flipkart_stem_completion.r

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")
```

```
load("flipkart_handle.df.corpus.copy.RData")
flipkart_handle.df.corpus = flipkart_handle.df.corpus.copy

#####Stem words in the corpus
flipkart_handle.df.corpus=tm_map(flipkart_handle.df.corpus, stemDocument)




stemCompletion2 <- function(x, dictionary) {
  x <- unlist(strsplit(as.character(x), " "))
  # Unexpectedly, stemCompletion completes an empty string to
  # a word in dictionary. Remove empty string to avoid above issue.
  x <- x[x != ""]
  x <- stemCompletion(x, dictionary=dictionary)
  x <- paste(x, sep="", collapse=" ")
  PlainTextDocument(stripWhitespace(x))
}

#####Stem Complete and Display the same tweet above with the completed and
corrected text.
flipkart_handle.df.corpus=lapply(flipkart_handle.df.corpus,stemCompletion2,
dictionary=flipkart_handle.df.corpus.copy)

##convert the list to Corpus again
flipkart_handle.df.corpus= Corpus(VectorSource(flipkart_handle.df.corpus))

sink("stemwords_flipkart_combined_after.txt")
for (i in c(1:2400, 10000)) {
  cat(paste0("[", i, "] "))
  #put the output to a file for inspection
  writeLines(strwrap(as.character(flipkart_handle.df.corpus[[i]]), 60) )
}
sink()

#####Correcting mis-spelt words

replaceWord <- function(corpus, oldword, newword)
{
  tm_map(corpus, gsub, pattern=oldword, replacement=newword)
}

flipkart_handle.df.corpus<- replaceWord(flipkart_handle.df.corpus, "flipkart", "")
flipkart_handle.df.corpus<- replaceWord(flipkart_handle.df.corpus, "deliveries",
"delivery")

#####remove unwanted words
flipkart_handle.df.corpus <- tm_map(flipkart_handle.df.corpus, removeWords, c("also",
"article", "Article", "download", "google", "figure","fig", "groups","Google",
"however","high", "human", "levels","larger", "may", "number","shown", "study",
"studies", "this","using", "two", "the", "Scholar","pubmedncbi", "PubMedNCBI","view",
"View", "the", "biol","via", "image", "doi", "one",

"analysis","how","which","when","there","flipkart","the","th","for","then","when"))



sink("stemwords_aflipkart_combined_after1.txt")
for (i in c(1:2400, 10000)) {
  cat(paste0("[", i, "] "))
  #put the output to a file for inspection
  writeLines(strwrap(as.character(flipkart_handle.df.corpus[[i]]), 60) )
}
sink()

#####save the cleaned corpus for future epidemics:is very important else we have to start from begining
flipkart_handle.df.corpus.cleaned = flipkart_handle.df.corpus
save.image(file="flipkart_handle.df.corpus.cleaned.RData")
```

Appendix F

step3_sc_stem_completion.r

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
```

```
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")
load("sc_combined.df.corpus.copy.RData")

sc_combined.df.corpus = sc_combined.df.corpus.copy

#####Stem words in the corpus
sc_combined.df.corpus=tm_map(sc_combined.df.corpus, stemDocument)

writeLines(strwrap(ebay_combined.df.corpus[[250]]$content,60))

stemCompletion2 <- function(x, dictionary) {
  x <- unlist(strsplit(as.character(x), " "))
  # Unexpectedly, stemCompletion completes an empty string to
  # a word in dictionary. Remove empty string to avoid above issue.
  x <- x[x != ""]
  x <- stemCompletion(x, dictionary=dictionary)
  x <- paste(x, sep="", collapse=" ")
  PlainTextDocument(stripWhitespace(x))
}

#####Stem Complete and Display the same tweet above with the completed and corrected text.
sc_combined.df.corpus=lapply(sc_combined.df.corpus,stemCompletion2,
dictionary=sc_combined.df.corpus.copy)
```

```
##convert the list to Corpus again
sc_combined.df.corpus= Corpus(VectorSource(sc_combined.df.corpus))

sink("stemwords_sc_combined_after.txt")
for (i in c(1:3000, 6924)) {
  cat(paste0("[", i, "] "))
  #put the output to a file for inspection
  writeLines(strwrap(as.character(sc_combined.df.corpus[[i]]), 60) )
}
sink()

#####Correcting mis-spelt words

replaceWord <- function(corpus, oldword, newword)
{
  tm_map(corpus, gsub, pattern=oldword, replacement=newword)
}

sc_combined.df.corpus<- replaceWord(sc_combined.df.corpus, "shopclu", "")
sc_combined.df.corpus<- replaceWord(sc_combined.df.corpus, "deliveryplease", "delivery please")
sc_combined.df.corpus<- replaceWord(sc_combined.df.corpus, "deliv", "delivery")
sc_combined.df.corpus<- replaceWord(sc_combined.df.corpus, "delivering", "delivery")
sc_combined.df.corpus<- replaceWord(sc_combined.df.corpus, "deliveryi", "delivery")
sc_combined.df.corpus<- replaceWord(sc_combined.df.corpus, "deliverybut", "delivery")
sc_combined.df.corpus<- replaceWord(sc_combined.df.corpus, "deliver", "delivery")
sc_combined.df.corpus<- replaceWord(sc_combined.df.corpus, "deliveryy", "delivery")

#####remove unwanted words
```

```
sc_combined.df.corpus <- tm_map(sc_combined.df.corpus, removeWords, c("also", "article", "Article", "download", "google", "figure","fig", "groups","Google", "however","high", "human", "levels","larger", "may", "number","shown", "study", "studies", "this","using", "two", "the", "Scholar","pubmedncbi", "PubMedNCBI","view", "View", "the", "biol","via", "image", "doi", "one",

"analysis","how","which","when","there","myntra","the","th","for","then","when","httpst
codfvdjzyy"))




#####save the cleaned corpus for future epidemics:is very important else we have to start from begining
sc_combined.df.corpus.cleaned = sc_combined.df.corpus
save.image(file="sc_combined.df.corpus.cleaned.RData")
```

Appendix G

step3_snapdeal_stem_completion.r

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")

load("snapdeal_combined.df.corpus.copy.RData")
snapdeal_combined.corpus = snapdeal_combined.df.corpus.copy

#####Stem words in the corpus
snapdeal_combined.corpus=tm_map(snapdeal_combined.corpus, stemDocument)

writeLines(strwrap(snapdeal_combined.corpus[[250]]$content,60))

stemCompletion2 <- function(x, dictionary) {
  x <- unlist(strsplit(as.character(x), " "))
  # Unexpectedly, stemCompletion completes an empty string to
  # a word in dictionary. Remove empty string to avoid above issue.
  x <- x[x != ""]
  x <- stemCompletion(x, dictionary=dictionary)
  x <- paste(x, sep="", collapse=" ")
  PlainTextDocument(stripWhitespace(x))
}

#####Stem Complete and Display the same tweet above with the completed and corrected text.
snapdeal_combined.corpus=lapply(snapdeal_combined.corpus,stemCompletion2,
dictionary=snapdeal_combined.df.corpus.copy)

##convert the list to Corpus again
snapdeal_combined.corpus= Corpus(VectorSource(snapdeal_combined.corpus))

sink("stemwords_snapdeal_combined_after.txt")
for (i in c(1:2400, 10392)) {
  cat(paste0("[", i, "] "))
  #put the output to a file for inspection
  writeLines(strwrap(as.character(snapdeal_combined.corpus[[i]]), 60) )
}
sink()

#####Correcting mis-spelt words

replaceWord <- function(corpus, oldword, newword)
{
  tm_map(corpus, gsub, pattern=oldword, replacement=newword)
}

snapdeal_combined.corpus<-                    replaceWord(snapdeal_combined.corpus,
"secretsashwinsanghionsnapdeal", "secret ashwinsanghi on snapdeal")
snapdeal_combined.corpus<- replaceWord(snapdeal_combined.corpus, "snapdeal", "")
snapdeal_combined.corpus<- replaceWord(snapdeal_combined.corpus, "snapdealhelp", "")
snapdeal_combined.corpus<- replaceWord(snapdeal_combined.corpus, "locationnot", "location not")
snapdeal_combined.corpus<- replaceWord(snapdeal_combined.corpus, "itemreturn", "item return")
snapdeal_combined.corpus<- replaceWord(snapdeal_combined.corpus, "orderyour", "order your")
snapdeal_combined.corpus<- replaceWord(snapdeal_combined.corpus, "customerservic", "customer service")
snapdeal_combined.corpus<- replaceWord(snapdeal_combined.corpus, "returnreplace", "return replace")
snapdeal_combined.corpus<- replaceWord(snapdeal_combined.corpus, "respons", "response")
snapdeal_combined.corpus<- replaceWord(snapdeal_combined.corpus, "st", "sent")
snapdeal_combined.corpus<- replaceWord(snapdeal_combined.corpus, "accusations", "acquisition")
snapdeal_combined.corpus<- replaceWord(snapdeal_combined.corpus, "deliv", "delivery")
snapdeal_combined.corpus<- replaceWord(snapdeal_combined.corpus, "dearzindagithisfriday", "")
snapdeal_combined.corpus<- replaceWord(snapdeal_combined.corpus, "deliveries", "delivery")

#####remove unwanted words
snapdeal_combined.corpus <- tm_map(snapdeal_combined.corpus, removeWords, c("also", "article", "Article", "download", "google", "figure","fig", "groups","Google", "however","high", "human", "levels","larger", "may", "number","shown", "study", "studies", "this","using", "two", "the", "Scholar","pubmedncbi", "PubMedNCBI","view", "View", "the", "biol","via", "image", "doi", "one",
"analysis","how","which","when","there","snapdeal","the","th","for","then","when"))
```

```
#####save the cleaned corpus for future epidemics:is very important else we have to start from begining
snapdeal_combined.corpus.cleaned = snapdeal_combined.corpus
save.image(file="snapdeal_combined.corpus.cleaned.RData")
```

Appendix H

step4_amazon_wordcloud.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
```

```
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")

load("amazon_combined.corpus.cleaned.RData")



#####Find the terms used most frequently
amazon_combined.corpus<- replaceWord(amazon_combined.corpus, "in", "")
amazon_combined.corpus<- replaceWord(amazon_combined.corpus, "get", "")

amazon_combined.corpus.ptd <- tm_map(amazon_combined.corpus,
PlainTextDocument)
tdm<- TermDocumentMatrix(amazon_combined.corpus.ptd, control= list(wordLengths=
c(1, Inf)))
freq.terms <- findFreqTerms(tdm, lowfreq = 200)
term.freq <- rowSums(as.matrix(tdm))
term.freq <- subset(term.freq, term.freq > 200)
df <- data.frame(term = names(term.freq), freq= term.freq)

#####plotting the graph of frequent terms
ggplot(df, aes(reorder(term, freq),freq)) + theme_bw() + geom_bar(stat = "identity") +
coord_flip() +labs(list(title="Term Frequency Chart", x="Terms", y="Term
Counts"))+theme(text = element_text(size=10),axis.text.x = element_text(angle=90,
vjust=1))
```

```
word.freq <-sort(rowSums(as.matrix(tdm)), decreasing= F)
pal<- brewer.pal(8, "Dark2")
wordcloud(words = names(word.freq), freq = word.freq, min.freq = 2, random.order = F,
colors = pal, max.words = 1000)

save.image(file="amazon.wordcloud.RData")
```

Appendix I

step4_ebay_wordcloud.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
```

```
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")

load("ebay_combined.df.corpus.cleaned.RData")



#####Find the terms used most frequently
ebay_combined.corpus.ptd <- tm_map(ebay_combined.df.corpus, PlainTextDocument)
tdm<- TermDocumentMatrix(ebay_combined.corpus.ptd, control= list(wordLengths=
c(1, Inf)))
freq.terms <- findFreqTerms(tdm, lowfreq = 200)
term.freq <- rowSums(as.matrix(tdm))
term.freq <- subset(term.freq, term.freq > 200)
df <- data.frame(term = names(term.freq), freq= term.freq)

#####plotting the graph of frequent terms
ggplot(df, aes(reorder(term, freq),freq)) + theme_bw() + geom_bar(stat = "identity") +
coord_flip() +labs(list(title="Term Frequency Chart of Ebay", x="Terms", y="Term
Counts"))+theme(text = element_text(size=10),axis.text.x = element_text(angle=90,
vjust=1))



#####calculate the frequency of words and sort it by frequency and setting up the
Wordcloud
```

```
word.freq <-sort(rowSums(as.matrix(tdm)), decreasing= F)
pal<- brewer.pal(8, "Dark2")
wordcloud(words = names(word.freq), freq = word.freq, min.freq = 2, random.order = F,
colors = pal, max.words = 1000)

save.image(file="ebay.wordcloud.RData")
```

Appendix J

step4_flipkart_wordcloud.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")

load("flipkart_handle.df.corpus.cleaned.RData")



#####Find the terms used most frequently
flipkart_handle.corpus.ptd <- tm_map(flipkart_handle.df.corpus, PlainTextDocument)
tdm<- TermDocumentMatrix(flipkart_handle.corpus.ptd, control= list(wordLengths= c(1,
Inf)))
freq.terms <- findFreqTerms(tdm, lowfreq = 200)
term.freq <- rowSums(as.matrix(tdm))
term.freq <- subset(term.freq, term.freq > 200)
df <- data.frame(term = names(term.freq), freq= term.freq)

#####plotting the graph of frequent terms
ggplot(df, aes(reorder(term, freq),freq)) + theme_bw() + geom_bar(stat = "identity") +
coord_flip() +labs(list(title="Term Frequency Chart For Flipkart", x="Terms", y="Term
Counts"))+theme(text = element_text(size=10),axis.text.x = element_text(angle=90,
vjust=1))



#####calculate the frequency of words and sort it by frequency and setting up the
Wordcloud
word.freq <-sort(rowSums(as.matrix(tdm)), decreasing= F)
pal<- brewer.pal(8, "Dark2")
```

```
wordcloud(words = names(word.freq), freq = word.freq, min.freq = 2, random.order = F,
colors = pal, max.words = 1000)

save.image(file="flipkart.wordcloud.RData")
```

Appendix K

step4_sc_wordcloud.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")

load("sc_combined.df.corpus.cleaned.RData")



#####Find the terms used most frequently
sc_combined.corpus.ptd <- tm_map(sc_combined.df.corpus, PlainTextDocument)
tdm<- TermDocumentMatrix(sc_combined.corpus.ptd, control= list(wordLengths= c(1,
Inf)))
freq.terms <- findFreqTerms(tdm, lowfreq = 200)
term.freq <- rowSums(as.matrix(tdm))
term.freq <- subset(term.freq, term.freq > 200)
df <- data.frame(term = names(term.freq), freq= term.freq)

#####plotting the graph of frequent terms
ggplot(df, aes(reorder(term, freq),freq)) + theme_bw() + geom_bar(stat = "identity") +
coord_flip() +labs(list(title="Term Frequency Chart of Shop Clues", x="Terms",
y="Term Counts"))+theme(text = element_text(size=10),axis.text.x =
element_text(angle=90, vjust=1))



#####calculate the frequency of words and sort it by frequency and setting up the
Wordcloud
word.freq <-sort(rowSums(as.matrix(tdm)), decreasing= F)
pal<- brewer.pal(8, "Dark2")
wordcloud(words = names(word.freq), freq = word.freq, min.freq = 2, random.order = F,
colors = pal, max.words = 1000)
save.image(file="sc.wordcloud.RData")
```

Appendix L

step4_snapdeal_wprdcloud.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")
```

```
load("snapdeal_combined.corpus.cleaned.RData")



#####Find the terms used most frequently
snapdeal_combined.corpus.ptd <- tm_map(snapdeal_combined.corpus, PlainTextDocument)
tdm<- TermDocumentMatrix(snapdeal_combined.corpus.ptd, control= list(wordLengths= c(1, Inf)))
freq.terms <- findFreqTerms(tdm, lowfreq = 200)
term.freq <- rowSums(as.matrix(tdm))
term.freq <- subset(term.freq, term.freq > 200)
df <- data.frame(term = names(term.freq), freq= term.freq)

#####plotting the graph of frequent terms
ggplot(df, aes(reorder(term, freq),freq)) + theme_bw() + geom_bar(stat = "identity") + coord_flip() +labs(list(title="Term Frequency Chart for Snapdeal", x="Terms", y="Term Counts"))+theme(text = element_text(size=10),axis.text.x = element_text(angle=90, vjust=1))



#####calculate the frequency of words and sort it by frequency and setting up the Wordcloud
word.freq <-sort(rowSums(as.matrix(tdm)), decreasing= F)
pal<- brewer.pal(8, "Dark2")
wordcloud(words = names(word.freq), freq = word.freq, min.freq = 2, random.order = F, colors = pal, max.words = 1000)

save.image(file="snapdeal.wordcloud.RData")
```

Appendix M

step5_amazon_word_corelation.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")
load("amazon.wordcloud.RData")
```

```
###### Identify and plot word correlations. For example - help
WordCorr_help <- apply_as_df(amazon_combined.corpus[1:10395], word_cor, word =
"help", r=.20)
plot(WordCorr_help)
qheat(vect2df(WordCorr_help[[1]], "help", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()

###### Identify and plot word correlations. For example - order
WordCorr_order <- apply_as_df(amazon_combined.corpus[1:10395], word_cor, word =
"order", r=.25)
plot(WordCorr_order)
qheat(vect2df(WordCorr_order[[1]], "order", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()

###### Identify and plot word correlations. For example - sign
WordCorr_amazonsign <- apply_as_df(amazon_combined.corpus[1:10395], word_cor,
word = "sign", r=.25)
plot(WordCorr_amazonsign)
qheat(vect2df(WordCorr_amazonsign[[1]], "sign", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()


###### Identify and plot word correlations. For example - copies
WordCorr_copies <- apply_as_df(amazon_combined.corpus[1:10395], word_cor, word =
"copies", r=.25)
plot(WordCorr_copies)
qheat(vect2df(WordCorr_copies[[1]], "copies", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()
```

```
###### Identify and plot word correlations. For example - fraud
WordCorr_fraud <- apply_as_df(amazon_combined.corpus[1:10395], word_cor, word =
"fraud", r=.25)
plot(WordCorr_fraud)
qheat(vect2df(WordCorr_fraud[[1]], "fraud", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()




###### Find Messages with word - help
df <- data.frame(text=unlist(sapply(amazon_combined.corpus.ptd, '[', "content")),
stringsAsFactors=FALSE)
head(unique(df[grep("help", df$text), ]), n=5)

###### Find Messages with word - order
head(unique(df[grep("order", df$text), ]), n=5)

###### Find Messages with word - sign
head(unique(df[grep("sign", df$text), ]), n=5)

###### Find Messages with word - copies
head(unique(df[grep("copies", df$text), ]), n=5)
###### Find Messages with word - fraud
head(unique(df[grep("fraud", df$text), ]), n=5)


save.image(file="amazon.word_corelation.RData")
```

Appendix N

step5_ebay_word_corelation.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")
load("ebay.wordcloud.RData")
```

```
###### Identify and plot word correlations. For example - game
WordCorr_game<- apply_as_df(ebay_combined.df.corpus[1:11409], word_cor, word = "game", r=.25)
plot(WordCorr_game)
qheat(vect2df(WordCorr_game[[1]], "game", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()

###### Identify and plot word correlations. For example - new
WordCorr_new<- apply_as_df(ebay_combined.df.corpus[1:11409], word_cor, word = "new", r=.25)
plot(WordCorr_new)
qheat(vect2df(WordCorr_new[[1]], "new", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()

###### Identify and plot word correlations. For example - india
WordCorr_india<- apply_as_df(ebay_combined.df.corpus[1:11409], word_cor, word = "india", r=.25)
plot(WordCorr_india)
qheat(vect2df(WordCorr_india[[1]], "india", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()

###### Identify and plot word correlations. For example - deal
WordCorr_deal<- apply_as_df(ebay_combined.df.corpus[1:11409], word_cor, word = "deal", r=.25)
plot(WordCorr_deal)
qheat(vect2df(WordCorr_deal[[1]], "deal", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()
```

```
###### Find Messages with word - game
df <- data.frame(text=unlist(sapply(ebay_combined.corpus.ptd, '[', "content")),
stringsAsFactors=FALSE)
head(unique(df[grep("game", df$text), ]), n=5)

head(unique(df[grep("new", df$text), ]), n=5)

head(unique(df[grep("india", df$text), ]), n=5)

head(unique(df[grep("deal", df$text), ]), n=5)


save.image(file="ebay.word_corelation.RData")
```

Appendix O

step5_flipkar_word_corelation.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
```

```
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")
load("flipkart.wordcloud.RData")


###### Identify and plot word correlations. For example - redmi
WordCorr_redmi <- apply_as_df(flipkart_handle.df.corpus[1:10000], word_cor, word =
"redmi", r=.25)
plot(WordCorr_redmi)
qheat(vect2df(WordCorr_redmi[[1]], "redmi", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()

###### Identify and plot word correlations. For example - beard
WordCorr_beard <- apply_as_df(flipkart_handle.df.corpus[1:10000], word_cor, word =
"beard", r=.25)
plot(WordCorr_beard)
qheat(vect2df(WordCorr_beard[[1]], "beard", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()
```

```
###### Identify and plot word correlations. For example - love
WordCorr_love <- apply_as_df(flipkart_handle.df.corpus[1:10000], word_cor, word =
"love", r=.25)
plot(WordCorr_love)
qheat(vect2df(WordCorr_love[[1]], "love", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()



###### Identify and plot word correlations. For example - delivery
WordCorr_delivery <- apply_as_df(flipkart_handle.df.corpus[1:10000], word_cor, word
= "delivery", r=.25)
plot(WordCorr_delivery)
qheat(vect2df(WordCorr_delivery[[1]], "delivery", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()



###### Find Messages with word - redmi
df <- data.frame(text=unlist(sapply(flipkart_handle.corpus.ptd, '[', "content")),
stringsAsFactors=FALSE)
head(unique(df[grep("redmi", df$text), ]), n=5)

###### Find Messages with word - beard
head(unique(df[grep("beard", df$text), ]), n=5)

###### Find Messages with word - love
head(unique(df[grep("love", df$text), ]), n=5)

###### Find Messages with word - delivery
head(unique(df[grep("delivery", df$text), ]), n=5)
save.image(file="flipkart.word_corelation.RData")
```

Appendix P

step5_sc_word_corelation.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")
load("sc.wordcloud.RData")
```

```
###### Identify and plot word correlations. For example - intex
WordCorr_intex<- apply_as_df(sc_combined.df.corpus[1:6924], word_cor, word =
"intex", r=.25)
plot(WordCorr_intex)
qheat(vect2df(WordCorr_intex[[1]], "intex", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()

###### Identify and plot word correlations. For example - bad
WordCorr_bad<- apply_as_df(sc_combined.df.corpus[1:6924], word_cor, word = "bad",
r=.25)
plot(WordCorr_bad)
qheat(vect2df(WordCorr_bad[[1]], "bad", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()

###### Identify and plot word correlations. For example - marshmallow
WordCorr_marshmallow<- apply_as_df(sc_combined.df.corpus[1:6924], word_cor, word
= "marshmallow", r=.25)
plot(WordCorr_marshmallow)
qheat(vect2df(WordCorr_marshmallow[[1]], "marshmallow", "cor"), values=TRUE,
high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()
###### Find Messages with word - intex
df <- data.frame(text=unlist(sapply(sc_combined.corpus.ptd, '[', "content")),
stringsAsFactors=FALSE)
head(unique(df[grep("intex", df$text), ]), n=5)
head(unique(df[grep("bad", df$text), ]), n=5)
head(unique(df[grep("marshmallow", df$text), ]), n=5)
save.image(file="sc.word_corelation.RData")
```

Appendix Q

step5_snapeal_word_corelation.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")
load("snapdeal.wordcloud.RData")
```

```
###### Identify and plot word correlations. For example - order
WordCorr_order<- apply_as_df(snapdeal_combined.corpus[1:10392], word_cor, word =
"order", r=.20)
plot(WordCorr_order)
qheat(vect2df(WordCorr_order[[1]], "order", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()

###### Identify and plot word correlations. For example - help
WordCorr_help<- apply_as_df(snapdeal_combined.corpus[1:10392], word_cor, word =
"help", r=.10)
plot(WordCorr_help)
qheat(vect2df(WordCorr_help[[1]], "help", "cor"), values=TRUE, high="red",
    digits=2, order.by ="cor", plot = FALSE) + coord_flip()



###### Find Messages with word - help
df <- data.frame(text=unlist(sapply(snapdeal_combined.corpus.ptd, '[', "content")),
stringsAsFactors=FALSE)
head(unique(df[grep("help", df$text), ]), n=5)

head(unique(df[grep("order", df$text), ]), n=5)


save.image(file="snapdeal.word_corelation.RData")
```

Appendix R

step6_amazon_association_topicmodeling.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")
```

```
load("amazon.word_corelation.RData")
amazon_combined.df=read.csv("amazon_combined.csv",header =
TRUE,stringsAsFactors = FALSE,sep = ",")
amazon_combined.df=amazon_combined.df[,-c(1)]

findAssocs(tdm, "help", 0.2)
findAssocs(tdm, "order", 0.2)
findAssocs(tdm, "sign", 0.2)
findAssocs(tdm, "copies", 0.2)
findAssocs(tdm, "fraud", 0.2)

##### Topic Modelling to identify latent/hidden topics
#### Create document-term
dtm <- as.DocumentTermMatrix(tdm)

#### dtm may contain 0 values,we need to clean it
raw.sum=apply(dtm,1,FUN=sum)

####  delete all raws which are all 0 doing
dtm=dtm[raw.sum!=0,]

####collapse matrix by summing over columns
freq <- colSums(as.matrix(dtm))

####length should be total number of terms
length(freq)

####create sort order (descending)
ord <- order(freq,decreasing=TRUE)
```

```
####List all terms in decreasing order of freq and write to disk
#freq[ord]
#for documenting purpose only the first 10 elements were shown.
head(freq[ord],10)
write.csv(freq[ord],"amazon_word_freq.csv")
amazon_combined.df.copy=amazon_combined.df



####Set parameters for Gibbs sampling
burnin <- 4000
iter <- 2000
thin <- 500
seed <-list(2003,5,63,100001,765)
nstart <- 5
best <- TRUE

####Number of topics
k <- 4
######Run LDA using Gibbs sampling
ldaOut <-LDA(dtm,k, method="Gibbs", control=list(nstart=nstart, seed = seed, best=best,
burnin = burnin, iter = iter, thin=thin))

####write out results
ldaOut.topics <- as.matrix(topics(ldaOut))
write.csv(ldaOut.topics,file=paste("LDAGibbs",k,"amazon_DocsToTopics.csv"))

###### find top 6 terms in each topic
ldaOut.terms <- as.matrix(terms(ldaOut,6))
ldaOut.terms
write.csv(ldaOut.terms,file=paste("LDAGibbs",k,"Amazon_TopicsToTerms.csv"))
```

```
######probabilities associated with each topic assignment
topicProbabilities <- as.data.frame(ldaOut@gamma)
head(topicProbabilities,6)
write.csv(topicProbabilities,file=paste("LDAGibbs",k,"Amazon_TopicProbabilities.csv")
)

######Find relative importance of top 2 topics
topic1ToTopic2 <- lapply(1:nrow(dtm),function(x)
  sort(topicProbabilities[x,])[k]/sort(topicProbabilities[x,])[k-1])

######Find relative importance of second and third most important topics
topic2ToTopic3 <- lapply(1:nrow(dtm),function(x)
  sort(topicProbabilities[x,])[k-1]/sort(topicProbabilities[x,])[k-2])

######write to file
t12df=as.data.frame(topic1ToTopic2)
t12df[1,1:6]# To check its contents
write.csv(topic1ToTopic2,file=paste("LDAGibbs",k,"Amazon_Topic1ToTopic2.csv"))
#save it for complete analysis
t23df=as.data.frame(topic2ToTopic3)
t23df[1,1:6]# To check its contents
write.csv(topic2ToTopic3,file=paste("LDAGibbs",k,"Amazon_Topic2ToTopic3.csv"))
#save it for complete analysis

#k means clustering
#
# Let us create a Term-Document matrix with words having sparsity less than 99
percentile
#
```

```
# Remove sparse terms
tdm2<-removeSparseTerms(tdm,sparse=0.98)
m2<-as.matrix(tdm2)

#Cluster terms
distmatrix=dist(scale(m2))
fit=hclust(distmatrix,method="ward.D2")
plot(fit,cex=0.9)

groups <- cutree(fit, k=4)
groups
#cut the tree
rect.hclust(fit,k=4)



save.image(file="amazon_association_topicmodelling.RData")
```

Appendix S

step6_ebay_association_topicmodeling.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")

load("ebay.word_corelation.RData")
ebay_combined.df=read.csv("ebay_combined.csv",header = TRUE,stringsAsFactors =
FALSE,sep = ",")
ebay_combined.df=ebay_combined.df[,-c(1)]

findAssocs(tdm, "game", 0.20)
findAssocs(tdm, "india", 0.2)
findAssocs(tdm, "deal", 0.2)

##### Topic Modelling to identify latent/hidden topics
#### Create document-term
dtm <- as.DocumentTermMatrix(tdm)
```

```
#### dtm may contain 0 values,we need to clean it
raw.sum=apply(dtm,1,FUN=sum)

####  delete all raws which are all 0 doing
dtm=dtm[raw.sum!=0,]

####collapse matrix by summing over columns
freq <- colSums(as.matrix(dtm))

####length should be total number of terms
length(freq)

####create sort order (descending)
ord <- order(freq,decreasing=TRUE)

####List all terms in decreasing order of freq and write to disk
#freq[ord]
#for documenting purpose only the first 10 elements were shown.
head(freq[ord],10)
write.csv(freq[ord],"snapdeal_word_freq.csv")




####Set parameters for Gibbs sampling
burnin <- 4000
iter <- 2000
thin <- 500
seed <-list(2003,5,63,100001,765)
nstart <- 5
best <- TRUE
```

```
####Number of topics
k <- 4
######Run LDA using Gibbs sampling
ldaOut <-LDA(dtm,k, method="Gibbs", control=list(nstart=nstart, seed = seed, best=best,
burnin = burnin, iter = iter, thin=thin))

####write out results
ldaOut.topics <- as.matrix(topics(ldaOut))
write.csv(ldaOut.topics,file=paste("LDAGibbs",k,"snapdealt_DocsToTopics.csv"))

###### find top 6 terms in each topic
ldaOut.terms <- as.matrix(terms(ldaOut,6))
ldaOut.terms
write.csv(ldaOut.terms,file=paste("LDAGibbs",k,"snapdeal_TopicsToTerms.csv"))

######probabilities associated with each topic assignment
topicProbabilities <- as.data.frame(ldaOut@gamma)
head(topicProbabilities,6)
write.csv(topicProbabilities,file=paste("LDAGibbs",k,"snapdeal_TopicProbabilities.csv")
)

######Find relative importance of top 2 topics
topic1ToTopic2 <- lapply(1:nrow(dtm),function(x)
  sort(topicProbabilities[x,])[k]/sort(topicProbabilities[x,])[k-1])

######Find relative importance of second and third most important topics
topic2ToTopic3 <- lapply(1:nrow(dtm),function(x)
  sort(topicProbabilities[x,])[k-1]/sort(topicProbabilities[x,])[k-2])
```

```
######write to file
t12df=as.data.frame(topic1ToTopic2)
t12df[1,1:6]# To check its contents
write.csv(topic1ToTopic2,file=paste("LDAGibbs",k,"snapdeal_Topic1ToTopic2.csv"))
#save it for complete analysis
t23df=as.data.frame(topic2ToTopic3)
t23df[1,1:6]# To check its contents
write.csv(topic2ToTopic3,file=paste("LDAGibbs",k,"snapdeal_Topic2ToTopic3.csv"))
#save it for complete analysis


#k means clustering
#
# Let us create a Term-Document matrix with words having sparsity less than 99 percentile
#
# Remove sparse terms
tdm2<-removeSparseTerms(tdm,sparse=0.98)
m2<-as.matrix(tdm2)

#Cluster terms
distmatrix=dist(scale(m2))
fit=hclust(distmatrix,method="ward.D2")
plot(fit,cex=0.9)

#cut the tree
rect.hclust(fit,k=4)

save.image(file="ebay_association_topicmodelling.RData")
```

Appendix T

step6_flipkart_association_topicmodeling.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")
```

```
load("flipkart.word_corelation.RData")
flipkart_handle.df=read.csv("flipkart_handle.csv",header = TRUE,stringsAsFactors =
FALSE,sep = ",")
flipkart_handle.df=flipkart_handle.df[,-c(1)]

findAssocs(tdm, "redmi", 0.2)
findAssocs(tdm, "beard", 0.2)
findAssocs(tdm, "love", 0.2)
findAssocs(tdm, "delivery", 0.2)

##### Topic Modelling to identify latent/hidden topics
#### Create document-term
dtm <- as.DocumentTermMatrix(tdm)

#### dtm may contain 0 values,we need to clean it
raw.sum=apply(dtm,1,FUN=sum)

####  delete all raws which are all 0 doing
dtm=dtm[raw.sum!=0,]

####collapse matrix by summing over columns
freq <- colSums(as.matrix(dtm))

####length should be total number of terms
length(freq)

####create sort order (descending)
ord <- order(freq,decreasing=TRUE)

####List all terms in decreasing order of freq and write to disk
```

```
#freq[ord]
#for documenting purpose only the first 10 elements were shown.
head(freq[ord],10)
write.csv(freq[ord],"flipkart_word_freq.csv")




####Set parameters for Gibbs sampling
burnin <- 4000
iter <- 2000
thin <- 500
seed <-list(2003,5,63,100001,765)
nstart <- 5
best <- TRUE

####Number of topics
k <- 4
######Run LDA using Gibbs sampling
ldaOut <-LDA(dtm,k, method="Gibbs", control=list(nstart=nstart, seed = seed, best=best,
burnin = burnin, iter = iter, thin=thin))

####write out results
ldaOut.topics <- as.matrix(topics(ldaOut))
write.csv(ldaOut.topics,file=paste("LDAGibbs",k,"flipkart_DocsToTopics.csv"))

###### find top 6 terms in each topic
ldaOut.terms <- as.matrix(terms(ldaOut,6))
ldaOut.terms
write.csv(ldaOut.terms,file=paste("LDAGibbs",k,"flipkart_TopicsToTerms.csv"))
```

```
######probabilities associated with each topic assignment
topicProbabilities <- as.data.frame(ldaOut@gamma)
head(topicProbabilities,6)
write.csv(topicProbabilities,file=paste("LDAGibbs",k,"flipkart_TopicProbabilities.csv"))

######Find relative importance of top 2 topics
topic1ToTopic2 <- lapply(1:nrow(dtm),function(x)
  sort(topicProbabilities[x,])[k]/sort(topicProbabilities[x,])[k-1])

######Find relative importance of second and third most important topics
topic2ToTopic3 <- lapply(1:nrow(dtm),function(x)
  sort(topicProbabilities[x,])[k-1]/sort(topicProbabilities[x,])[k-2])

######write to file
t12df=as.data.frame(topic1ToTopic2)
t12df[1,1:6]# To check its contents
write.csv(topic1ToTopic2,file=paste("LDAGibbs",k,"flipkart_Topic1ToTopic2.csv"))
#save it for complete analysis
t23df=as.data.frame(topic2ToTopic3)
t23df[1,1:6]# To check its contents
write.csv(topic2ToTopic3,file=paste("LDAGibbs",k,"flipkart_Topic2ToTopic3.csv"))
#save it for complete analysis

#k means clustering
#
# Let us create a Term-Document matrix with words having sparsity less than 99 percentile
#
# Remove sparse terms
tdm2<-removeSparseTerms(tdm,sparse=0.98)
```

```
m2<-as.matrix(tdm2)

#Cluster terms
distmatrix=dist(scale(m2))
fit=hclust(distmatrix,method="ward.D2")
plot(fit,cex=0.9)

#cut the tree
rect.hclust(fit,k=4)

save.image(file="flipkart_association_topicmodelling.RData")
```

Appendix U

step6_shopclues_association_topicmodeling.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
```

```
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")

load("sc.word_corelation.RData")

sc_combined.df=read.csv("sc_combined.csv",header = TRUE,stringsAsFactors =
FALSE,sep = ",")
sc_combined.df=sc_combined.df[,-c(1)]

findAssocs(tdm, "intex", 0.20)
findAssocs(tdm, "bad", 0.2)
findAssocs(tdm, "marshmallow", 0.2)

##### Topic Modelling to identify latent/hidden topics
#### Create document-term
dtm <- as.DocumentTermMatrix(tdm)

#### dtm may contain 0 values,we need to clean it
raw.sum=apply(dtm,1,FUN=sum)

####  delete all raws which are all 0 doing
dtm=dtm[raw.sum!=0,]
```

```
####collapse matrix by summing over columns
freq <- colSums(as.matrix(dtm))

####length should be total number of terms
length(freq)

####create sort order (descending)
ord <- order(freq,decreasing=TRUE)

####List all terms in decreasing order of freq and write to disk
#freq[ord]
#for documenting purpose only the first 10 elements were shown.
head(freq[ord],10)
write.csv(freq[ord],"snapdeal_word_freq.csv")



####Set parameters for Gibbs sampling
burnin <- 4000
iter <- 2000
thin <- 500
seed <-list(2003,5,63,100001,765)
nstart <- 5
best <- TRUE

####Number of topics
k <- 4
######Run LDA using Gibbs sampling
```

```
ldaOut <-LDA(dtm,k, method="Gibbs", control=list(nstart=nstart, seed = seed, best=best,
burnin = burnin, iter = iter, thin=thin))

####write out results
ldaOut.topics <- as.matrix(topics(ldaOut))
write.csv(ldaOut.topics,file=paste("LDAGibbs",k,"sc_DocsToTopics.csv"))

###### find top 6 terms in each topic
ldaOut.terms <- as.matrix(terms(ldaOut,6))
ldaOut.terms
write.csv(ldaOut.terms,file=paste("LDAGibbs",k,"snapdeal_TopicsToTerms.csv"))

######probabilities associated with each topic assignment
topicProbabilities <- as.data.frame(ldaOut@gamma)
head(topicProbabilities,6)
write.csv(topicProbabilities,file=paste("LDAGibbs",k,"snapdeal_TopicProbabilities.csv")
)

######Find relative importance of top 2 topics
topic1ToTopic2 <- lapply(1:nrow(dtm),function(x)
  sort(topicProbabilities[x,])[k]/sort(topicProbabilities[x,])[k-1])

######Find relative importance of second and third most important topics
topic2ToTopic3 <- lapply(1:nrow(dtm),function(x)
  sort(topicProbabilities[x,])[k-1]/sort(topicProbabilities[x,])[k-2])

######write to file
t12df=as.data.frame(topic1ToTopic2)
t12df[1,1:6]# To check its contents
```

```
write.csv(topic1ToTopic2,file=paste("LDAGibbs",k,"snapdeal_Topic1ToTopic2.csv"))
#save it for complete analysis
t23df=as.data.frame(topic2ToTopic3)
t23df[1,1:6]# To check its contents
write.csv(topic2ToTopic3,file=paste("LDAGibbs",k,"snapdeal_Topic2ToTopic3.csv"))
#save it for complete analysis



#k means clustering
#
# Let us create a Term-Document matrix with words having sparsity less than 99 percentile
#
# Remove sparse terms
tdm2<-removeSparseTerms(tdm,sparse=0.98)
m2<-as.matrix(tdm2)

#Cluster terms
distmatrix=dist(scale(m2))
fit=hclust(distmatrix,method="ward.D2")
plot(fit,cex=0.9)

#cut the tree
rect.hclust(fit,k=8)

save.image(file="sc_association_topicmodelling.RData")
```

Appendix V

step6_snapdeal_association_topicmodeling.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")
```

```
load("snapdeal.word_corelation.RData")
snapdeal_combined.df=read.csv("snapdeal_combined.csv",header =
TRUE,stringsAsFactors = FALSE,sep = ",")
snapdeal_combined.df=snapdeal_combined.df[,-c(1)]

findAssocs(tdm, "help", 0.1)
findAssocs(tdm, "order", 0.1)

##### Topic Modelling to identify latent/hidden topics
#### Create document-term
dtm <- as.DocumentTermMatrix(tdm)

#### dtm may contain 0 values,we need to clean it
raw.sum=apply(dtm,1,FUN=sum)

####  delete all raws which are all 0 doing
dtm=dtm[raw.sum!=0,]

####collapse matrix by summing over columns
freq <- colSums(as.matrix(dtm))

####length should be total number of terms
length(freq)

####create sort order (descending)
ord <- order(freq,decreasing=TRUE)

####List all terms in decreasing order of freq and write to disk
#freq[ord]
#for documenting purpose only the first 10 elements were shown.
```

```
head(freq[ord],10)
write.csv(freq[ord],"snapdeal_word_freq.csv")




####Set parameters for Gibbs sampling
burnin <- 4000
iter <- 2000
thin <- 500
seed <-list(2003,5,63,100001,765)
nstart <- 5
best <- TRUE

####Number of topics
k <- 4
######Run LDA using Gibbs sampling
ldaOut <-LDA(dtm,k, method="Gibbs", control=list(nstart=nstart, seed = seed, best=best,
burnin = burnin, iter = iter, thin=thin))

####write out results
ldaOut.topics <- as.matrix(topics(ldaOut))
write.csv(ldaOut.topics,file=paste("LDAGibbs",k,"snapdealt_DocsToTopics.csv"))

###### find top 6 terms in each topic
ldaOut.terms <- as.matrix(terms(ldaOut,6))
ldaOut.terms
write.csv(ldaOut.terms,file=paste("LDAGibbs",k,"snapdeal_TopicsToTerms.csv"))

######probabilities associated with each topic assignment
topicProbabilities <- as.data.frame(ldaOut@gamma)
```

```
head(topicProbabilities,6)
write.csv(topicProbabilities,file=paste("LDAGibbs",k,"snapdeal_TopicProbabilities.csv")
)

######Find relative importance of top 2 topics
topic1ToTopic2 <- lapply(1:nrow(dtm),function(x)
  sort(topicProbabilities[x,])[k]/sort(topicProbabilities[x,])[k-1])

######Find relative importance of second and third most important topics
topic2ToTopic3 <- lapply(1:nrow(dtm),function(x)
  sort(topicProbabilities[x,])[k-1]/sort(topicProbabilities[x,])[k-2])

######write to file
t12df=as.data.frame(topic1ToTopic2)
t12df[1,1:6]# To check its contents
write.csv(topic1ToTopic2,file=paste("LDAGibbs",k,"snapdeal_Topic1ToTopic2.csv"))
#save it for complete analysis
t23df=as.data.frame(topic2ToTopic3)
t23df[1,1:6]# To check its contents
write.csv(topic2ToTopic3,file=paste("LDAGibbs",k,"snapdeal_Topic2ToTopic3.csv"))
#save it for complete analysis



#k means clustering
#
# Let us create a Term-Document matrix with words having sparsity less than 99
percentile
#
# Remove sparse terms
tdm2<-removeSparseTerms(tdm,sparse=0.98)
```

```
m2<-as.matrix(tdm2)

#Cluster terms
distmatrix=dist(scale(m2))
fit=hclust(distmatrix,method="ward.D2")
plot(fit,cex=0.9)

#cut the tree
rect.hclust(fit,k=4)

save.image(file="snapdeal_association_topicmodelling.RData")
```

Appendix W

step7_amazon_sentimentanalysis.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
```

```
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
library(Rstem)
library(ggplot2)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")

load("amazon_association_topicmodelling.RData")

sentiments <- polarity(amazon_combined.df$text)
sentiments <- data.frame(sentiments$all$polarity)
sentiments[["polarity"]] <- cut(sentiments[[ "sentiments.all.polarity"]], c(-5,0.0,5),
labels = c("negative","positive"))
table(sentiments$polarity)

#Sentiment Plot by date

sentiments$score<- 0
sentiments$score[sentiments$polarity == "positive"]<-1
sentiments$score[sentiments$polarity == "negative"]<- -1
sentiments$date <-as.IDate(amazon_combined.df$created)
result = aggregate(score ~ date, data = sentiments, sum)
plot(result,type='b')
```

```
#stream grapg

Data<-data.frame(sentiments$polarity)
colnames(Data)[1] <- "polarity"
Data$Date <- as.Date(amazon_combined.df$created)
Data$text <- NULL
Data$Count <- 1

graphdata <- aggregate(Count ~ polarity +
as.character.Date(Data$Date),data=Data,FUN=length)
colnames(graphdata)[2] <- "Date"
str(graphdata)
write.csv(graphdata,file="amazon_sentiments_graphdata.csv")

#stream graph

graphdata %>%
  streamgraph(polarity, Count, Date) %>%
  sg_axis_x(20) %>%
  sg_axis_x(1, "Date","%d-%b") %>%
  sg_legend(show=TRUE, label="Polarity: ")



#detail analysis
install_url("http://cran.r-project.org/src/contrib/Archive/sentiment/sentiment_0.2.tar.gz")
library(sentiment)
AmazonTweetsClassEmo =
classify_emotion(amazon_combined.df$text,algorithm="bayes", prior=1.0)
AmazonTweetsClassEmo
```

```
AmazonEmotion = AmazonTweetsClassEmo[,7]
AmazonEmotion[is.na(AmazonEmotion)] = "unknown"
head(AmazonEmotion,20)

plotSentiments1<- function (sentiment_dataframe,title) {

  ggplot(sentiment_dataframe, aes(x=emotion)) +
    geom_bar(aes(y=..count.., fill=emotion)) +

    scale_fill_brewer(palette="Dark2") +

    ggtitle(title) +

    theme(legend.position='right') + ylab('Number of Tweets') +
    xlab('Emotion Categories')

}

plotSentiments2 <- function (sentiment_dataframe,title) {

  library(ggplot2)

  ggplot(sentiment_dataframe, aes(x=polarity)) +

    geom_bar(aes(y=..count.., fill=polarity)) +

    scale_fill_brewer(palette="RdGy") +

    ggtitle(title) +
  theme(legend.position='right') + ylab('Number of Tweets') +
  xlab('Polarity Categories')

}

AmazonTweetsClassPol =
classify_polarity(amazon_combined.df$text,algorithm="bayes")
head(AmazonTweetsClassPol,20)
AmazonPol = AmazonTweetsClassPol[,4]

AmazonSentimentDataFrame =
data.frame(text=amazon_combined.df$text,emotion=AmazonEmotion,
polarity=AmazonPol, stringsAsFactors=FALSE)
AmazonSentimentDataFrame%>%filter(!emotion %in% c("unknown")) -
>amazon.filter.df

amazon.filter.df = within(amazon.filter.df, emotion <-factor(emotion,
levels=names(sort(table(emotion),decreasing=TRUE))))
plotSentiments1(amazon.filter.df, 'Sentiment Analysis of Tweets on Amazon by
Emotions')
write.csv(amazon.filter.df,file = "amazon_filter_sentiment_by_emotion.csv")
#Amazon wordcloud with emotions
#####calculate the frequency of words and sort it by frequency and setting up the
Wordcloud
removeCustomeWords <- function (TweetsCleaned) {

 for(i in 1:length(TweetsCleaned)){

  TweetsCleaned[i] <- tryCatch({
   stpword.vector <- readLines("stop-word-list.csv")[1]
```

```
    TweetsCleaned[i] = removeWords(TweetsCleaned[i],stpword.vector)

    TweetsCleaned[i]

   }, error=function(cond) {

    TweetsCleaned[i]

   }, warning=function(cond) {

    TweetsCleaned[i]

   })

  }

  return(TweetsCleaned)

}
getWordCloud <- function
(sentiment_dataframe, TweetsCleaned, Emotion) {

  emos = levels(factor(sentiment_dataframe$emotion))

  n_emos = length(emos)

  emo.docs = rep("", n_emos)

  TweetsCleaned = removeCustomeWords(TweetsCleaned)
  for (i in 1:n_emos){

    emo.docs[i] = paste(TweetsCleaned[Emotion ==
                                      emos[i]], collapse=" ")

  }

  corpus = Corpus(VectorSource(emo.docs))

  tdm = TermDocumentMatrix(corpus)

  tdm = as.matrix(tdm)

  colnames(tdm) = emos

  require(wordcloud)

  suppressWarnings(comparison.cloud(tdm, colors =
                                    brewer.pal(n_emos, "Dark2"), scale = c(3,.5), random.order =
                                    FALSE, title.size = 1.5))

}
rem.some.punct <- function(txt, notpunct=NULL){
  punctstr <- "[]!\"#$%&'()*/+,.:;<=>?@[^_`{|}~-]"
  rempunct <- gsub(paste0("",notpunct), "", punctstr)
  gsub(rempunct, "", txt)}
```

```
getWordCloud(AmazonSentimentDataFrame,rem.some.punct(amazon_combined.df$text)
,AmazonEmotion)
save.image("amazon_sentiment_analysis.RData")
```

Appendix X

step7_ebay_sentimentanalysis.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")

load("ebay_association_topicmodelling.RData")
rem.some.punct <- function(txt, notpunct=NULL){
  punctstr <- "[]!\"#$%&'()*/+,.:;<=>?@[^_`{|}~-]"
  rempunct <- gsub(paste0("",notpunct), "", punctstr)
  gsub(rempunct, "", txt)}

sentiments <- polarity(rem.some.punct(ebay_combined.df$text))
sentiments <- data.frame(sentiments$all$polarity)
sentiments[["polarity"]] <- cut(sentiments[[ "sentiments.all.polarity"]], c(-5,0.0,5),
labels = c("negative","positive"))
table(sentiments$polarity)

#Sentiment Plot by date

sentiments$score<- 0
sentiments$score[sentiments$polarity == "positive"]<-1
sentiments$score[sentiments$polarity == "negative"]<- -1
sentiments$date <-as.IDate(ebay_combined.df$created)
result = aggregate(score ~ date, data = sentiments, sum)
plot(result,type='l')

#stream grapg

Data<-data.frame(sentiments$polarity)
colnames(Data)[1] <- "polarity"
Data$Date <- as.Date(ebay_combined.df$created)
Data$text <- NULL
```

```
Data$Count <- 1

graphdata <- aggregate(Count ~ polarity + as.character.Date(Data$Date),data=Data,FUN=length)
colnames(graphdata)[2] <- "Date"
str(graphdata)
str(graphdata)
write.csv(graphdata,file="ebay_sentiments_graphdata.csv")



#stream graph

 graphdata %>%
  streamgraph(polarity, Count, Date) %>%
  sg_axis_x(20) %>%
  sg_axis_x(1, "Date","%d-%b") %>%
  sg_legend(show=TRUE, label="Polarity: ")



 install_url("http://cran.r-project.org/src/contrib/Archive/sentiment/sentiment_0.2.tar.gz")
 library(sentiment)
 ebaytweetsemo = classify_emotion(ebay_combined.df$text,algorithm="bayes", prior=1.0)
 ebaytweetsemotion = ebaytweetsemo[,7]
 ebaytweetsemotion[is.na(ebaytweetsemotion)] = "unknown"
 head(ebaytweetsemotion,20)

plotSentiments1<- function (sentiment_dataframe,title) {

 ggplot(sentiment_dataframe, aes(x=emotion)) +
  geom_bar(aes(y=..count.., fill=emotion)) +

  scale_fill_brewer(palette="Dark2") +

  ggtitle(title) +

  theme(legend.position='right') + ylab('Number of Tweets') +
  xlab('Emotion Categories')

}

plotSentiments2 <- function (sentiment_dataframe,title) {

 library(ggplot2)

 ggplot(sentiment_dataframe, aes(x=polarity)) +

  geom_bar(aes(y=..count.., fill=polarity)) +

  scale_fill_brewer(palette="RdGy") +

  ggtitle(title) +

  theme(legend.position='right') + ylab('Number of Tweets') +
  xlab('Polarity Categories')

}

EbayTweetsClassPol = classify_polarity(ebay_combined.df$text,algorithm="bayes")
head(EbayTweetsClassPol,20)
```

```
ebayPol = EbayTweetsClassPol[,4]

ebaySentimentDataFrame = data.frame(text=ebay_combined.df$text,emotion=ebaytweetsemotion, polarity=ebayPol, stringsAsFactors=FALSE)
ebaySentimentDataFrame%>%filter(!emotion %in% c("unknown")) ->ebay.filter.df

ebay.filter.df = within(ebay.filter.df, emotion <-factor(emotion, levels=names(sort(table(emotion),decreasing=TRUE))))
plotSentiments1(ebay.filter.df, 'Sentiment Analysis of Tweets on Twitter about Ebay')
write.csv(ebay.filter.df,file = "ebay_filter_sentiment_by_emotion.csv")


#myntra wordcloud with emotions
#####calculate the frequency of words and sort it by frequency and setting up the Wordcloud
removeCustomeWords <- function (TweetsCleaned) {

 for(i in 1:length(TweetsCleaned)){

  TweetsCleaned[i] <- tryCatch({

   stpword.vector <- readLines("stop-word-list.csv")[1]

   TweetsCleaned[i] = removeWords(TweetsCleaned[i],stpword.vector)

   TweetsCleaned[i]

  }, error=function(cond) {
      TweetsCleaned[i]

    }, warning=function(cond) {

      TweetsCleaned[i]

    })

  }

  return(TweetsCleaned)

}
getWordCloud <- function
(sentiment_dataframe, TweetsCleaned, Emotion) {

  emos = levels(factor(sentiment_dataframe$emotion))

  n_emos = length(emos)

  emo.docs = rep("", n_emos)

  TweetsCleaned = removeCustomeWords(TweetsCleaned)



  for (i in 1:n_emos){

    emo.docs[i] = paste(TweetsCleaned[Emotion ==
                                   emos[i]], collapse=" ")
```

```
 }

 corpus = Corpus(VectorSource(emo.docs))

 tdm = TermDocumentMatrix(corpus)

 tdm = as.matrix(tdm)

 colnames(tdm) = emos

 require(wordcloud)

 suppressWarnings(comparison.cloud(tdm, colors =
                    brewer.pal(n_emos, "Dark2"), scale = c(3,.5), random.order =
                    FALSE, title.size = 1.5))

}

getWordCloud(ebaySentimentDataFrame,rem.some.punct(ebay_combined.df$text),ebayt
weetsemotion)
save.image("ebay_sentiment_analysis.RData")
```

Appendix Y

step7_flipkart_sentimentanalysis.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")

load("flipkart_association_topicmodelling.RData")
rem.some.punct <- function(txt, notpunct=NULL){
  punctstr <- "[]!\"#$%&'()*/+,.:;<=>?@[^_`{|}~-]"
  rempunct <- gsub(paste0("",notpunct), "", punctstr)
  gsub(rempunct, "", txt)}

sentiments <- polarity(rem.some.punct(flipkart_handle.df$text))
sentiments <- data.frame(sentiments$all$polarity)
```

```
sentiments[["polarity"]] <- cut(sentiments[[ "sentiments.all.polarity"]], c(-5,0.0,5), labels = c("negative","positive"))
table(sentiments$polarity)

#Sentiment Plot by date

sentiments$score<- 0
sentiments$score[sentiments$polarity == "positive"]<-1
sentiments$score[sentiments$polarity == "negative"]<- -1
sentiments$date <-as.IDate(flipkart_handle.df$created)
result = aggregate(score ~ date, data = sentiments, sum)
plot(result,type='l')

#stream grapg

Data<-data.frame(sentiments$polarity)
colnames(Data)[1] <- "polarity"
Data$Date <- as.Date(flipkart_handle.df$created)
Data$text <- NULL
Data$Count <- 1

graphdata <- aggregate(Count ~ polarity + as.character.Date(Data$Date),data=Data,FUN=length)
colnames(graphdata)[2] <- "Date"
str(graphdata)
str(graphdata)
write.csv(graphdata,file="flipkart_sentiments_graphdata.csv")



#stream graph
```

```
graphdata %>%
  streamgraph(polarity, Count, Date) %>%
  sg_axis_x(20) %>%
  sg_axis_x(1, "Date","%d-%b") %>%
  sg_legend(show=TRUE, label="Polarity: ")

#detail analysis
install_url("http://cran.r-project.org/src/contrib/Archive/sentiment/sentiment_0.2.tar.gz")
library(sentiment)
flipkarttweetsemo = classify_emotion(flipkart_handle.df$text,algorithm="bayes",
prior=1.0)
flipkarttweetsemo
flipkartemotion = flipkarttweetsemo[,7]
flipkartemotion[is.na(flipkartemotion)] = "unknown"
head(flipkartemotion,20)

plotSentiments1<- function (sentiment_dataframe,title) {

  ggplot(sentiment_dataframe, aes(x=emotion)) +
    geom_bar(aes(y=..count.., fill=emotion)) +

    scale_fill_brewer(palette="Dark2") +

    ggtitle(title) +

    theme(legend.position='right') + ylab('Number of Tweets') +
    xlab('Emotion Categories')

}
```

```
plotSentiments2 <- function (sentiment_dataframe,title) {

  library(ggplot2)

  ggplot(sentiment_dataframe, aes(x=polarity)) +

    geom_bar(aes(y=..count.., fill=polarity)) +

    scale_fill_brewer(palette="RdGy") +

    ggtitle(title) +

    theme(legend.position='right') + ylab('Number of Tweets') +
    xlab('Polarity Categories')

}

flipkartTweetsClassPol = classify_polarity(flipkart_handle.df$text,algorithm="bayes")
head(flipkartTweetsClassPol,20)
flipkartPol = flipkartTweetsClassPol[,4]

FkipkartSentimentDataFrame =
data.frame(text=flipkart_handle.df$text,emotion=flipkartemotion, polarity=flipkartPol,
stringsAsFactors=FALSE)
FkipkartSentimentDataFrame%>%filter(!emotion %in% c("unknown")) -
>flipkart.filter.df

flipkart.filter.df = within(flipkart.filter.df, emotion <-factor(emotion,
levels=names(sort(table(emotion),decreasing=TRUE))))
```

```
plotSentiments1(flipkart.filter.df, 'Sentiment Analysis of Tweets on Twitter about
Flipkart')
write.csv(flipkart.filter.df,file = "flipkart_filter_sentiment_by_emotion.csv")

#flipkart wordcloud with emotions
#####calculate the frequency of words and sort it by frequency and setting up the
Wordcloud
removeCustomeWords <- function (TweetsCleaned) {

  for(i in 1:length(TweetsCleaned)){

    TweetsCleaned[i] <- tryCatch({

      stpword.vector <- readLines("stop-word-list.csv")[1]

      TweetsCleaned[i] = removeWords(TweetsCleaned[i],stpword.vector)

    }, error=function(cond) {

      TweetsCleaned[i]

    }, warning=function(cond) {

      TweetsCleaned[i]

    })

  }

  return(TweetsCleaned)
```

```
}
getWordCloud <- function
(sentiment_dataframe, TweetsCleaned, Emotion) {

  emos = levels(factor(sentiment_dataframe$emotion))

  n_emos = length(emos)

  emo.docs = rep("", n_emos)

  TweetsCleaned = removeCustomeWords(TweetsCleaned)



  for (i in 1:n_emos){

    emo.docs[i] = paste(TweetsCleaned[Emotion ==
                                      emos[i]], collapse=" ")

  }

  corpus = Corpus(VectorSource(emo.docs))

  tdm = TermDocumentMatrix(corpus)

  tdm = as.matrix(tdm)

  colnames(tdm) = emos
  require(wordcloud)

  suppressWarnings(comparison.cloud(tdm, colors =
                          brewer.pal(n_emos, "Dark2"), scale = c(3,.5), random.order =
                          FALSE, title.size = 1.5))

}

getWordCloud(FkipkartSentimentDataFrame,rem.some.punct(flipkart_handle.df$text),fli
pkartemotion)


save.image("flipkart_sentiment_analysis.RData")
```

Appendix Z

step7_sc_sentimentanalysis.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
```

```
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")

load("sc_association_topicmodelling.RData")
rem.some.punct <- function(txt, notpunct=NULL){
  punctstr <- "[]!\"#$%&'()*/+,.:;<=>?@[^_`{|}~-]"
  rempunct <- gsub(paste0("",notpunct), "", punctstr)
  gsub(rempunct, "", txt)}

  sentiments <- polarity(rem.some.punct(sc_combined.df$text))
  sentiments <- data.frame(sentiments$all$polarity)
  sentiments[["polarity"]] <- cut(sentiments[[ "sentiments.all.polarity"]], c(-5,0.0,5),
labels = c("negative","positive"))
  table(sentiments$polarity)

#Sentiment Plot by date

sentiments$score<- 0
sentiments$score[sentiments$polarity == "positive"]<-1
sentiments$score[sentiments$polarity == "negative"]<- -1
```

```
sentiments$date <-as.IDate(sc_combined.df$created)
result = aggregate(score ~ date, data = sentiments, sum)
plot(result,type='l')

#stream grapg

Data<-data.frame(sentiments$polarity)
colnames(Data)[1] <- "polarity"
Data$Date <- as.Date(sc_combined.df$created)
Data$text <- NULL
Data$Count <- 1

graphdata <- aggregate(Count ~ polarity + 
as.character.Date(Data$Date),data=Data,FUN=length)
colnames(graphdata)[2] <- "Date"
str(graphdata)
str(graphdata)
write.csv(graphdata,file="sc_sentiments_graphdata.csv")


#stream graph

graphdata %>%
  streamgraph(polarity, Count, Date) %>%
  sg_axis_x(20) %>%
  sg_axis_x(1, "Date","%d-%b") %>%
  sg_legend(show=TRUE, label="Polarity: ")


install_url("http://cran.r-project.org/src/contrib/Archive/sentiment/sentiment_0.2.tar.gz")
```

```
library(sentiment)
sctweetsemo = classify_emotion(sc_combined.df$text,algorithm="bayes", prior=1.0)
sctweetsemotion = sctweetsemo[,7]
sctweetsemotion[is.na(sctweetsemotion)] = "unknown"
head(sctweetsemotion,20)

plotSentiments1<- function (sentiment_dataframe,title) {

  ggplot(sentiment_dataframe, aes(x=emotion)) +
    geom_bar(aes(y=..count.., fill=emotion)) +

    scale_fill_brewer(palette="Dark2") +

    ggtitle(title) +

    theme(legend.position='right') + ylab('Number of Tweets') +
    xlab('Emotion Categories')

}

plotSentiments2 <- function (sentiment_dataframe,title) {

  library(ggplot2)

  ggplot(sentiment_dataframe, aes(x=polarity)) +

    geom_bar(aes(y=..count.., fill=polarity)) +

    scale_fill_brewer(palette="RdGy") +
  ggtitle(title) +

  theme(legend.position='right') + ylab('Number of Tweets') +
  xlab('Polarity Categories')

}

scTweetsClassPol = classify_polarity(sc_combined.df$text,algorithm="bayes")
head(scTweetsClassPol,20)
scPol = scTweetsClassPol[,4]

scSentimentDataFrame = data.frame(text=sc_combined.df$text,emotion=sctweetsemotion, polarity=scPol, stringsAsFactors=FALSE)
scSentimentDataFrame%>%filter(!emotion %in% c("unknown")) ->sc.filter.df

sc.filter.df = within(sc.filter.df, emotion <-factor(emotion, levels=names(sort(table(emotion),decreasing=TRUE))))
plotSentiments1(sc.filter.df, 'Sentiment Analysis of Tweets on Twitter about Shop Clues')
write.csv(sc.filter.df,file = "sc_filter_sentiment_by_emotion.csv")


#myntra wordcloud with emotions
#####calculate the frequency of words and sort it by frequency and setting up the Wordcloud
removeCustomeWords <- function (TweetsCleaned) {

 for(i in 1:length(TweetsCleaned)){

  TweetsCleaned[i] <- tryCatch({
    stpword.vector <- readLines("stop-word-list.csv")[1]

    TweetsCleaned[i] = removeWords(TweetsCleaned[i],stpword.vector)

    TweetsCleaned[i]

  }, error=function(cond) {

    TweetsCleaned[i]

  }, warning=function(cond) {

    TweetsCleaned[i]

  })

 }

 return(TweetsCleaned)

}
getWordCloud <- function
(sentiment_dataframe, TweetsCleaned, Emotion) {

 emos = levels(factor(sentiment_dataframe$emotion))

 n_emos = length(emos)

 emo.docs = rep("", n_emos)
 TweetsCleaned = removeCustomeWords(TweetsCleaned)



 for (i in 1:n_emos){

   emo.docs[i] = paste(TweetsCleaned[Emotion ==
                                   emos[i]], collapse=" ")

 }

 corpus = Corpus(VectorSource(emo.docs))
   tdm = TermDocumentMatrix(corpus)
   tdm = as.matrix(tdm)
   colnames(tdm) = emos

 require(wordcloud)
   suppressWarnings(comparison.cloud(tdm, colors =
                                 brewer.pal(n_emos, "Dark2"), scale = c(3,.5), random.order =
                                 FALSE, title.size = 1.5))

}
getWordCloud(scSentimentDataFrame,rem.some.punct(sc_combined.df$text),sctweetse
motion)

save.image("sc_sentiment_analysis.RData")
```

Appendix A1

step7_snapdeal_sentimentanalysis.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")
```

```
load("snapdeal_association_topicmodelling.RData")
rem.some.punct <- function(txt, notpunct=NULL){
  punctstr <- "[]!\"#$%&'()*/+,.:;<=>?@[^_`{|}~-]"
  rempunct <- gsub(paste0("",notpunct), "", punctstr)
  gsub(rempunct, "", txt)}

sentiments <- polarity(rem.some.punct(snapdeal_combined.df$text))
sentiments <- data.frame(sentiments$all$polarity)
sentiments[["polarity"]] <- cut(sentiments[[ "sentiments.all.polarity"]], c(-5,0.0,5),
labels = c("negative","positive"))
table(sentiments$polarity)

#Sentiment Plot by date

sentiments$score<- 0
sentiments$score[sentiments$polarity == "positive"]<-1
sentiments$score[sentiments$polarity == "negative"]<- -1
sentiments$date <-as.IDate(snapdeal_combined.df$created)
result = aggregate(score ~ date, data = sentiments, sum)
plot(result,type='l')

#stream grapg

Data<-data.frame(sentiments$polarity)
colnames(Data)[1] <- "polarity"
Data$Date <- as.Date(snapdeal_combined.df$created)
Data$text <- NULL
Data$Count <- 1
```

```
graphdata <- aggregate(Count ~ polarity +
as.character.Date(Data$Date),data=Data,FUN=length)
colnames(graphdata)[2] <- "Date"
str(graphdata)
str(graphdata)
write.csv(graphdata,file="snapdeal_sentiments_graphdata.csv")


#stream graph

graphdata %>%
  streamgraph(polarity, Count, Date) %>%
  sg_axis_x(20) %>%
  sg_axis_x(1, "Date","%d-%b") %>%
  sg_legend(show=TRUE, label="Polarity: ")


#detail analysis
install_url("http://cran.r-project.org/src/contrib/Archive/sentiment/sentiment_0.2.tar.gz")
library(sentiment)
snapdealtweetsemo = classify_emotion(snapdeal_combined.df$text,algorithm="bayes",
prior=1.0)
snapdealemotion = snapdealtweetsemo[,7]
snapdealemotion[is.na(snapdealemotion)] = "unknown"
head(snapdealemotion,20)

plotSentiments1<- function (sentiment_dataframe,title) {

  ggplot(sentiment_dataframe, aes(x=emotion)) +
    geom_bar(aes(y=..count.., fill=emotion)) +
    scale_fill_brewer(palette="Dark2") +

    ggtitle(title) +

    theme(legend.position='right') + ylab('Number of Tweets') +
    xlab('Emotion Categories')

}

plotSentiments2 <- function (sentiment_dataframe,title) {

  library(ggplot2)

  ggplot(sentiment_dataframe, aes(x=polarity)) +

    geom_bar(aes(y=..count.., fill=polarity)) +

    scale_fill_brewer(palette="RdGy") +

    ggtitle(title) +

    theme(legend.position='right') + ylab('Number of Tweets') +
    xlab('Polarity Categories')

}

snapdealTweetsClassPol                                    =
classify_polarity(snapdeal_combined.df$text,algorithm="bayes")
head(snapdealTweetsClassPol,20)
```

```
snapdealPol = snapdealTweetsClassPol[,4]

snapdealSentimentDataFrame = data.frame(text=snapdeal_combined.df$text,emotion=snapdealemotion, polarity=snapdealPol, stringsAsFactors=FALSE)
snapdealSentimentDataFrame%>%filter(!emotion %in% c("unknown")) ->snapdeal.filter.df

snapdeal.filter.df = within(snapdeal.filter.df, emotion <-factor(emotion, levels=names(sort(table(emotion),decreasing=TRUE))))
plotSentiments1(snapdeal.filter.df, 'Sentiment Analysis of Tweets on Twitter about snapdeal')
write.csv(snapdeal.filter.df,file = "snapdeal_filter_sentiment_by_emotion.csv")

#snapdeal wordcloud with emotions
#####calculate the frequency of words and sort it by frequency and setting up the Wordcloud
removeCustomeWords <- function (TweetsCleaned) {

  for(i in 1:length(TweetsCleaned)){

    TweetsCleaned[i] <- tryCatch({

      stpword.vector <- readLines("stop-word-list.csv")[1]

      TweetsCleaned[i] = removeWords(TweetsCleaned[i],stpword.vector)
      TweetsCleaned[i]

    }, error=function(cond) {
```

```
    TweetsCleaned[i]

  }, warning=function(cond) {

    TweetsCleaned[i]

  })

 }

 return(TweetsCleaned)

}
getWordCloud <- function
(sentiment_dataframe, TweetsCleaned, Emotion) {

 emos = levels(factor(sentiment_dataframe$emotion))

 n_emos = length(emos)

 emo.docs = rep("", n_emos)

 TweetsCleaned = removeCustomeWords(TweetsCleaned)



 for (i in 1:n_emos){

  emo.docs[i] = paste(TweetsCleaned[Emotion ==
                      emos[i]], collapse=" ")
 }

 corpus = Corpus(VectorSource(emo.docs))

 tdm = TermDocumentMatrix(corpus)

 tdm = as.matrix(tdm)

 colnames(tdm) = emos

 require(wordcloud)

 suppressWarnings(comparison.cloud(tdm, colors =
                   brewer.pal(n_emos, "Dark2"), scale = c(3,.5), random.order =
                   FALSE, title.size = 1.5))

}

getWordCloud(snapdealSentimentDataFrame,rem.some.punct(snapdeal_combined.df$tex
t),snapdealemotion)
save.image("snapdeal_sentiment_analysis.RData")
```

Appendix B1

step8_sentiment_comaprison.R

```
rm(list=ls())
require(twitteR)
require(RCurl)
require(tm)
library(qdap)
library(ROAuth)
library(chron)
library(SnowballC)
library(ggplot2)
library(RColorBrewer)
library(wordcloud)
library(topicmodels)
library(data.table)
library(stringi)
library(rmarkdown)
#install.packages("bitops")
library(devtools)
#install_github('sentiment140', 'okugami79')
library(zoo)
#install_github("hrbrmstr/streamgraph")
# install_github('sentiment140', 'okugami79') before using sentiment pkg
library(sentiment)
library(reshape2)
library(qdap)
library(dplyr)
library(streamgraph)
setwd("C:\\AWBackup\\PGPBA\\PGPBA-GreatLakes\\Modules\\capstoneproject\\data")

amazon_sentiment.df = read.csv("amazon_sentiments_graphdata.csv",header = TRUE,sep = ",",stringsAsFactors = FALSE)
names(amazon_sentiment.df)[1]="Brand"
amazon_sentiment.df$Brand="amazon"

flipkart_sentiment.df = read.csv("flipkart_sentiments_graphdata.csv",header = TRUE,sep = ",",stringsAsFactors = FALSE)
names(flipkart_sentiment.df)[1]="Brand"
```

```
flipkart_sentiment.df$Brand="flipkart"

snapdeal_sentiment.df = read.csv("snapdeal_sentiments_graphdata.csv",header =
TRUE,sep = ",",stringsAsFactors = FALSE)
names(snapdeal_sentiment.df)[1]="Brand"
snapdeal_sentiment.df$Brand="snapdeal"

ebay_sentiment.df = read.csv("ebay_sentiments_graphdata.csv",header = TRUE,sep =
",",stringsAsFactors = FALSE)
names(ebay_sentiment.df)[1]="Brand"
ebay_sentiment.df$Brand="ebay"

sc_sentiment.df = read.csv("sc_sentiments_graphdata.csv",header = TRUE,sep =
",",stringsAsFactors = FALSE)
names(sc_sentiment.df)[1]="Brand"
sc_sentiment.df$Brand="shop clues"

sentiment.combined.df =
rbind(amazon_sentiment.df,snapdeal_sentiment.df,flipkart_sentiment.df,ebay_sentiment.
df,sc_sentiment.df)
write.csv(sentiment.combined.df,"sentiment.combined.csv")
sentiment.combined.df%>%group_by(Brand,polarity)%>%summarise(totalpolarity=sum(
Count))->summaryplot.polarity
write.csv(summaryplot.polarity,"summaryplot.polarity.csv")
sentiment.combined.df%>%group_by(Date)->sentiment.groupby.date

write.csv(sentiment.groupby.date,"sentiment.groupby.date.csv")

amazon_combined.df = read.csv("amazon_combined.csv",header =
TRUE,stringsAsFactors = FALSE)
```

```
flipkart_combined.df = read.csv("flipkart_handle.csv",header = TRUE,stringsAsFactors =
FALSE)
snapdeal_combined.df = read.csv("snapdeal_combined.csv",header =
TRUE,stringsAsFactors = FALSE)
ebay_combined.df=read.csv("ebay_combined.csv",header = TRUE,stringsAsFactors =
FALSE)
sc_combined.df=read.csv("sc_combined.csv",header = TRUE,stringsAsFactors =
FALSE)

amazon_combined.df$brand=c("amazon")
flipkart_combined.df$brand=c("flipkart")
snapdeal_combined.df$brand=c("snapdeal")
ebay_combined.df$brand=c("ebay")
sc_combined.df$brand=c("shop clues")

combined.tweets.df=rbind(amazon_combined.df,flipkart_combined.df,snapdeal_combine
d.df,ebay_combined.df,sc_combined.df)

combined.tweets.df$created = as.IDate(combined.tweets.df$created)

# # select top retweeted tweets
# table(combined.tweets.df$retweetCount)
# selected <- which(combined.tweets.df$retweetCount >= 1500)
# # plot them
# dates <- strptime(combined.tweets.df$created, format="%Y-%m-%d")
# plot(x=dates, y=combined.tweets.df$retweetCount, type="l", col="grey",xlab="Date",
ylab="Times retweeted")
# colors <- rainbow(10)[1:length(selected)]
# points(dates[selected], combined.tweets.df$retweetCount[selected],pch=19, col=colors)
# text(dates[selected],
combined.tweets.df$retweetCount[selected],combined.tweets.df$text[selected],
col=colors, cex=0.5,srt=45)

amazon_snetiment_byemotion.df =
read.csv("amazon_filter_sentiment_by_emotion.csv",header = TRUE,stringsAsFactors =
FALSE)
names(amazon_snetiment_byemotion.df)[1]="Brand"
amazon_snetiment_byemotion.df$brand=c("amazon")
flipkart_snetiment_byemotion.df =
read.csv("flipkart_filter_sentiment_by_emotion.csv",header = TRUE,stringsAsFactors =
FALSE)
names(flipkart_snetiment_byemotion.df)[1]="Brand"
flipkart_snetiment_byemotion.df$brand=c("flipkart")
snapdeal_snetiment_byemotion.df =
read.csv("snapdeal_filter_sentiment_by_emotion.csv",header = TRUE,stringsAsFactors =
FALSE)
names(snapdeal_snetiment_byemotion.df)[1]="Brand"
snapdeal_snetiment_byemotion.df$brand=c("snapdeal")
ebay_snetiment_byemotion.df=read.csv("ebay_filter_sentiment_by_emotion.csv",header
= TRUE,stringsAsFactors = FALSE)
names(ebay_snetiment_byemotion.df)[1]="Brand"
ebay_snetiment_byemotion.df$brand=c("ebay")
sc_snetiment_byemotion.df=read.csv("sc_filter_sentiment_by_emotion.csv",header =
TRUE,stringsAsFactors = FALSE)
names(sc_snetiment_byemotion.df)[1]="Brand"
sc_snetiment_byemotion.df$brand=c("shop clues")
combined_sentiments.df=rbind(amazon_snetiment_byemotion.df,flipkart_snetiment_bye
motion.df,snapdeal_snetiment_byemotion.df,ebay_snetiment_byemotion.df,sc_snetiment
_byemotion.df)
```

```
combined_sentiments.df=combined_sentiments.df[,-1]
combined_sentiments.df=combined_sentiments.df[,-1]
combined_sentiments.df=combined_sentiments.df[,-2]
combined_sentiments.df%>%group_by(emotion,brand)%>%summarise(totalemotion=n()
)->summarise_by_emotion.df
write.csv(summarise_by_emotion.df,"summarise_by_emotion.csv")

summarise_by_emotion.df%>%group_by(brand,emotion)%>%summarise(total=totalemo
tion)->temp_combined.total_emotion#summary_emotion_group_by_brandAndEmotion
write.csv(temp_combined.total_emotion,"temp_combined.total_emotion.csv")

write.csv(combined.tweets.df,"combined.tweets.csv")
combined.tweets.df%>%filter(!replyToSN                                    %in%
c("amazonIN","AmazonHelp","Flipkart","flipkartsupport","snapdeal","ebay","Ebay","Sh
opClues")) ->test.filter.df
test.filter.df%>%filter(!is.na(test.filter.df$replyToSN))->test.filter.clean.df
test.filter.clean.df%>% group_by(brand,created)%>% summarise(n = n())->summarize.df
names(summarize.df)[3]=c("replycount")
write.csv(summarize.df,"summarize.df.replycount.csv")



combined.tweets.df%>%filter(replyToSN                                     %in%
c("amazonIN","AmazonHelp","Flipkart","flipkartsupport","snapdeal","ebay","Ebay","Sh
opClues","ebayindia")) ->test.filter.queries.df
test.filter.queries.df%>%filter(!is.na(test.filter.queries.df$replyToSN))-
>test.filter.queries.clean.df
test.filter.queries.clean.df%>%   group_by(brand,created)%>%   summarise(n   =   n())-
>summarize.querie.df
names(summarize.querie.df)[3]=c("querycount")
write.csv(summarize.querie.df,"summarize.df.querycount.csv")
```

```
library(dplyr)
inner_join(summarize.df, summarize.querie.df)->querresponse.df
querresponse.df=querresponse.df[-c(4),]
querresponse.df$responserate=
(querresponse.df$replycount/querresponse.df$querycount)*100
write.csv(querresponse.df,"querresponse.csv")

querresponse.df%>%group_by(brand)%>%summarise(sumreply=sum(replycount),sumqu
ery=sum(querycount))->querygroupbybrand.df
querygroupbybrand.df$responserate =
(querygroupbybrand.df$sumreply/querygroupbybrand.df$sumquery)*100
write.csv(querygroupbybrand.df,"querygroupbybrand.csv")

#Total tweets count
combined.tweets.df%>%group_by(brand)%>%summarise(totaltweets=n())

#Total favorite count
combined.tweets.df%>%group_by(brand)%>%summarise(totalfavcount=sum(favoriteCo
unt))


#Total retweet count
combined.tweets.df%>%group_by(brand)%>%summarise(totalretweetcount=sum(retwee
tCount))

consumer_key = 'vitqhIlWGWfRUOUP8JIixb7OB'
consumer_secret = 'gJrNUmTl99eU3bTpUQTnBhO1zs3TST1qluwSjyozfkTm4FOGjW'
access_token = '3306555647-NpzZt0Af4T06E5ea9EWlVQeGRfE4wF4fvQBOBMg'
access_token_secret = '5Y4ojfc0LO8UtHmjrGVdhMlsF9AjkjWIxluBmofH45v2y'
```

```
setup_twitter_oauth(consumer_key,consumer_secret,access_token, access_token_secret)

#####Retrieve user info and followers
amazonuser=getUser("@amazonIN")
amazonuser.df=amazonuser$toDataFrame()

flipkartuser=getUser("@Flipkart")
flipkartuser.df=flipkartuser$toDataFrame()

snapdealuser=getUser("@snapdeal")
snapdealuser.df=snapdealuser$toDataFrame()

Ebayuser=getUser("@ebayindia")
Ebayuser.df=Ebayuser$toDataFrame()

scuser=getUser("@ShopClues")
scuser.df=scuser$toDataFrame()

combined.users.brandpercention.df=rbind(amazonuser.df,flipkartuser.df,snapdealuser.df,
Ebayuser.df,scuser.df)
write.csv(combined.users.brandpercention.df,"combined.users.brandpercention.csv")
save.image("sentiment_comparison.RData")
```