Final_Project_Rishabh_taneja

INTRODUCTION

A score is either 1 (for positive) or 0 (for negative)

The sentences come from three different websites/fields:imdb.com amazon.com yelp.com

For each website, there exist 500 positives and 500 negative sentences. Those were selected randomly for larger datasets of reviews. We attempted to select sentences that have a clearly positive or negative connotation, the goal was for no neutral sentences to be selected.

Amazon: contains reviews and scores for products sold on amazon.com in the cell phones and accessories category

IMDb: refers to the IMDb movie review sentiment

Yelp: refers to the dataset from the Yelp dataset challenge from which we extracted the restaurant reviews. IMDB: Learning word vectors for sentiment analysis. (n.d.). Retrieved from https://dl.acm.org/citation.cfm?id=2002491

amazon: Understanding rating dimensions with review text. (n.d.). Retrieved from https://dl.acm.org/citation.cfm?id=2507163

yelp: Yelp dataset challenge http://www.yelp.com/dataset_challenge. This dataset was created for the Paper 'From Group to Individual Labels using Deep Features', Kotzias et. al,. KDD 2015.

My dataset doesn't have any specific predefined labels. It is a text file which contains sentences labelled with positive or negative sentiment, extracted from reviews of products, movies, and restaurants.

FOCUS - The focus is on the sentiment analysis, and by using the ratings 0 or 1 with negative or positive sentiment. GOAL - Finally, would able to conclude which movie, product or a restaurant is good based on the sentiments. Additionally, doing the n-gram analysis while trying to answer "Whether n-gram analysis is important in every text analysis?"

PREPARATION

```
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 3.5.3

## -- Attaching packages ---------------------------------------------------
------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0       v purrr   0.3.0
## v tibble  2.0.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0

## Warning: package 'tidyr' was built under R version 3.5.3

## Warning: package 'dplyr' was built under R version 3.5.3

## Warning: package 'stringr' was built under R version 3.5.3

## -- Conflicts ---------------------------------------------------------------
------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(gmodels)  # Crosstable
library(tm)

## Loading required package: NLP

##
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':
##
##     annotate

library(wordcloud)

## Loading required package: RColorBrewer

library(e1071)
library(ggplot2)

library(rvest)

## Warning: package 'rvest' was built under R version 3.5.3

## Loading required package: xml2

##
## Attaching package: 'rvest'

## The following object is masked from 'package:purrr':
##
##     pluck

## The following object is masked from 'package:readr':
##
##     guess_encoding

library(stringr)
library(dplyr)
```

```r
library(colorRamps)
require(SnowballC)

## Loading required package: SnowballC

require(tidyr)
require(gridExtra)

## Loading required package: gridExtra

## Warning: package 'gridExtra' was built under R version 3.5.3

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine

require(tidytext)

## Loading required package: tidytext

## Warning: package 'tidytext' was built under R version 3.5.3

require(RColorBrewer)

#reading the text files, using auto detection to find the text file directly
from the path specified.

data.path <- "C:\\Users\\Rishabh Taneja\\Documents\\sentiment labelled
sentences\\sentiment labelled sentences"
txt.files <- list.files(path=data.path, full.names = T, recursive = T)

# Filtering description files
txt.files <- txt.files[str_detect(txt.files, "(amazon|imdb|yelp)")]

#function to load the data and create a data frame for the same
LoadData <- function(file_list) {
    tables <- lapply(file_list, ReadFile)
    data.frame <- do.call(rbind, tables)

    # Transform activity to lables, factorising the numbers
    data.frame$sentiment <- factor(data.frame$sentiment, levels = c(0, 1),
                                   labels = c("Negative","Positive"))

    return(data.frame)
}

#function to read the data and separating the text using the delimiter.
ReadFile <- function(file_name) {
```

```
    # Read txt file
    table <- read_delim(file_name,
                        delim = "\t",
                        col_names = c("text", "sentiment"),
                        quote = "")

    # Adding columns by parsing filenames and directories
    file.name <- str_split(file_name, "/", simplify = TRUE)[2]
    table['source']  <- str_split(file.name, "_", simplify = TRUE)[1]

    return(table)
}

data.frame <- LoadData(txt.files)

## Parsed with column specification:
## cols(
##   text = col_character(),
##   sentiment = col_double()
## )

## Parsed with column specification:
## cols(
##   text = col_character(),
##   sentiment = col_double()
## )
## Parsed with column specification:
## cols(
##   text = col_character(),
##   sentiment = col_double()
## )

# Shuffle rows in dataframe
set.seed(1985)
data.frame <- data.frame[order(runif(n=3000)),]
```

The above code is used to detect all the text files from the location and to read them into the system. Various functions have been created so that it is easy to read the file in a separate variable and then convert it into a tabular form.

VISUALISATION

```
str(data.frame)

## Classes 'tbl_df', 'tbl' and 'data.frame':    3000 obs. of  3 variables:
##  $ text     : chr  "It has everything I need and I couldn't ask for more." "This movie is terrible.  " "It seems completely secure, both holding on to my belt, and keeping the iPhone inside." "I love this cable - it allows me to connect any mini-USB device to my PC." ...
##  $ sentiment: Factor w/ 2 levels "Negative","Positive": 2 1 2 2 2 1 1 1 2
```

```
2 ...
##  $ source   : chr  "amazon" "imdb" "amazon" "amazon" ...
```

```r
summary(data.frame)
```

```
##      text              sentiment       source         
##  Length:3000        Negative:1500   Length:3000       
##  Class :character   Positive:1500   Class :character  
##  Mode  :character                   Mode  :character
```

```r
head(data.frame)
```

```
## # A tibble: 6 x 3
##   text                                                       sentiment 
source
##   <chr>                                                      <fct>     <chr>
## 1 It has everything I need and I couldn't ask for more.    Positive  
amazon
## 2 "This movie is terrible.  "                                Negative  imdb
## 3 It seems completely secure, both holding on to my belt,~ Positive  
amazon
## 4 I love this cable - it allows me to connect any mini-US~ Positive  
amazon
## 5 "I didn't realize how wonderful the short really is unt~ Positive  imdb
## 6 The commercials are the most misleading.                 Negative  
amazon
```

As we can see, the dataframe is evenly devided between 1500 positive and 1500 negative texts from 3 different sources. I have tried to showcase the first 6 rows from the data. We can see that the sentiment column has been converted from the 0 and 1 format to the negative and positive format respectively. The source column has the source from where the text has been taken into consideration.

```r
#creating corpus vector
corpus <- Corpus(VectorSource(data.frame$text))

#cleaning
#converting all the text into lower case
clean.corpus <- tm_map(corpus, content_transformer(tolower))

## Warning in tm_map.SimpleCorpus(corpus, content_transformer(tolower)):
## transformation drops documents

#removing all the numbers from the text data
clean.corpus <- tm_map(clean.corpus, removeNumbers)

## Warning in tm_map.SimpleCorpus(clean.corpus, removeNumbers):
transformation
## drops documents
```

```
#eliminating all the stop words, explicitly mentioning the language as English
clean.corpus <- tm_map(clean.corpus, removeWords, stopwords("english"))

## Warning in tm_map.SimpleCorpus(clean.corpus, removeWords,
## stopwords("english")): transformation drops documents

#getting rid of the punctuations
clean.corpus <- tm_map(clean.corpus, removePunctuation)

## Warning in tm_map.SimpleCorpus(clean.corpus, removePunctuation):
## transformation drops documents

#finally removing the extra white space
clean.corpus <- tm_map(clean.corpus, stripWhitespace)

## Warning in tm_map.SimpleCorpus(clean.corpus, stripWhitespace):
## transformation drops documents

# Take a look again to first five corpus
inspect(clean.corpus[1:5])

## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 5
##
## [1]  everything need ask
## [2]  movie terrible
## [3]  seems completely secure holding belt keeping iphone inside
## [4]  love cable allows connect miniusb device pc
## [5]  realize wonderful short really last two scenes
```

Since the data is very dirty, I have performed various data cleaning techniques to remove the punctuations, numbers, stopwords and the extra white space. The cleaned data has been stored into a clean.corpus variable.

```
#creating document term matrix
clean.corpus.dtm <- DocumentTermMatrix(clean.corpus)

#creating training and test datasets to work on. 80% for train, 20% for test.
n <- nrow(data.frame)
raw.text.train <- data.frame[1:round(.8 * n),]
raw.text.test  <- data.frame[(round(.8 * n)+1):n,]

nn <- length(clean.corpus)
clean.corpus.train <- clean.corpus[1:round(.8 * nn)]
clean.corpus.test  <- clean.corpus[(round(.8 * nn)+1):nn]

nnn <- nrow(clean.corpus.dtm)
clean.corpus.dtm.train <- clean.corpus.dtm[1:round(.8 * nnn),]
clean.corpus.dtm.test  <- clean.corpus.dtm[(round(.8 * nnn)+1):nnn,]
```

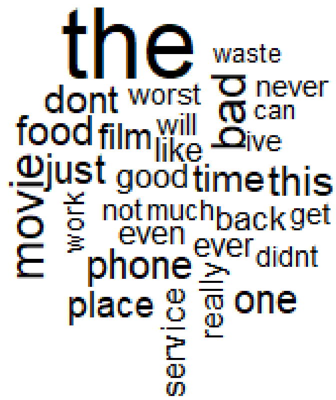
We now need to split the data into a training dataset and test dataset. We'll divide the data into two portions: 80 percent for training and 20 percent for testing. Since we already shuffled the dataframe in the first step, we need not worry about the randomness in the data.

```r
wordcloud(clean.corpus.train, min.freq = 30, random.order = FALSE)
```


```r
positive <- subset(raw.text.train, sentiment == "Positive")
negative <- subset(raw.text.train, sentiment == "Negative")
```

Here is a visual representation of the cleaned corpus data, with minimum frequency of the word set to 30. We can see that from the wordcloud, the words 'great','good','movie' and 'phone' have been used the most among all. Whereas, words like 'ear', 'still', 'everything' and'see' seems to be used the least. I have also created the subset for the positive and negative sentiments using the raw(original) dataset to train the data.

```r
wordcloud(negative$text, max.words = 30, scale = c(3, 0.5))

## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation):
## transformation drops documents

## Warning in tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x,
## tm::stopwords())): transformation drops documents
```



Here is the visual representation of the negative texts taken from the raw data itself. The word 'the' is used the most, whereas words like 'get', 'not', 'worst','waste' and 'can' seem to be used the least.

NOTE - The subset is generated using the raw data and NOT the cleaned data so there will be a lot of stopwords too.

```
wordcloud(positive$text, max.words = 30, scale = c(3, 0.5))

## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation):
## transformation drops documents

## Warning in tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x,
## tm::stopwords())): transformation drops documents
```

Here is the visual representation of the positive texts taken from the raw data itself. The words ‘good’,‘the’ and ‘great’ are used the most, whereas words like ‘recomment’, ‘excellent’, ‘made’,‘better’ and ‘time’ seem to be used the least. NOTE - The subset is generated using the raw data and NOT the cleaned data so there will be a lot of stopwords too.

```
#creating corpus vector
corpus_negative <- Corpus(VectorSource(negative))

#cleaning
#converting all the text into lower case
clean.corpus_negative <- tm_map(corpus_negative,
content_transformer(tolower))

## Warning in tm_map.SimpleCorpus(corpus_negative,
## content_transformer(tolower)): transformation drops documents

#removing all the numbers from the text data
clean.corpus_negative <- tm_map(clean.corpus_negative, removeNumbers)

## Warning in tm_map.SimpleCorpus(clean.corpus_negative, removeNumbers):
## transformation drops documents

#eliminating all the stop words, explicitly mentioning the language as
English
clean.corpus_negative <- tm_map(clean.corpus_negative, removeWords,
stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(clean.corpus_negative, removeWords,
## stopwords("english")): transformation drops documents
```

```
#getting rid of the punctuations
clean.corpus_negative <- tm_map(clean.corpus_negative, removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(clean.corpus_negative, removePunctuation):
## transformation drops documents
```

```
#finally removing the extra white space
clean.corpus_negative <- tm_map(clean.corpus_negative, stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(clean.corpus_negative, stripWhitespace):
## transformation drops documents
```

```
#Not inspecting the data since it is too big
```

To see the difference between the sentiments from the raw data and the cleaned data, I have performed additional cleaning steps for negative sentiments from the raw text. This will help me get rid of the stopwords which were hindering my analysis.

```
#creating corpus vector
corpus_positive <- Corpus(VectorSource(positive))

#cleaning
#converting all the text into lower case
clean.corpus_positive <- tm_map(corpus_positive,
content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(corpus_positive,
## content_transformer(tolower)): transformation drops documents
```

```
#removing all the numbers from the text data
clean.corpus_positive <- tm_map(clean.corpus_positive, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(clean.corpus_positive, removeNumbers):
## transformation drops documents
```

```
#eliminating all the stop words, explicitly mentioning the language as
English
clean.corpus_positive <- tm_map(clean.corpus_positive, removeWords,
stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(clean.corpus_positive, removeWords,
## stopwords("english")): transformation drops documents
```

```
#getting rid of the punctuations
clean.corpus_positive <- tm_map(clean.corpus_positive, removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(clean.corpus_positive, removePunctuation):
## transformation drops documents
```

```r
#finally removing the extra white space
clean.corpus_positive <- tm_map(clean.corpus_positive, stripWhitespace)

## Warning in tm_map.SimpleCorpus(clean.corpus_positive, stripWhitespace):
## transformation drops documents

#Not inspecting the data since it is too big
```

To see the difference between the sentiments from the raw data and the cleaned data, I have performed additional cleaning steps for positive sentiments from the raw text. This will help me get rid of the stopwords which were hindering my analysis.

```r
#wordcloud for the clean nagative sentiment data
wordcloud(clean.corpus_negative, max.words = 30, scale = c(3, 0.5),
random.order = FALSE)
```

Here is the wordcloud visualisation of the negative cleaned sentiments. Notice that the top 3 words which are used are amazon, yelp and imdb. Surprisingly, we can assume that the customers might be making use of these words, however, they are not helping me analyse anything regarding the negativeness from the customers. So we will make sure to handle this in the next steps.

```r
#wordcloud for the clean nagative sentiment data
wordcloud(clean.corpus_positive, max.words = 30, scale = c(3, 0.5),
random.order = FALSE)
```

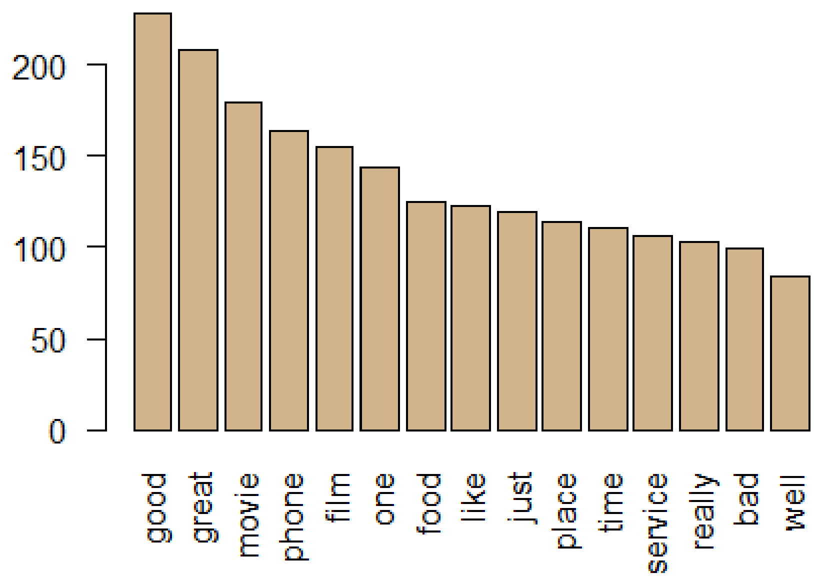
Here is the wordcloud visualisation of the negative cleaned sentiments. Notice that the top 3 words which are used are amazon, yelp and imdb additional to good,great and best. Surprisingly, we can assume that the customers might be making use of these words, however, they are not helping me analyse anything regarding the positiveness from the customers. So we will make sure to handle this in the next steps.

```
#generating clean matrix for the corpus data
clean_matrix <- as.matrix(clean.corpus.dtm)
clean_matrix[1:5, 20:25]

##     Terms
## Docs last realize really scenes short two
##    1    0       0      0      0     0   0
##    2    0       0      0      0     0   0
##    3    0       0      0      0     0   0
##    4    0       0      0      0     0   0
##    5    1       1      1      1     1   1
```

The above is the bag of wards taken from the document. We first converted the document term matrix as a normal matrix. We then showcased the first 5 rows and the 20-25th columns. The 0s represent that the word has not been used in that row and on the other hand, 1 represent that the corresponding word has been used in that row.

```
#calculating the frequency of the words
term_frequency <- colSums(clean_matrix)

#sorting the words and frequency in decreasing order
```

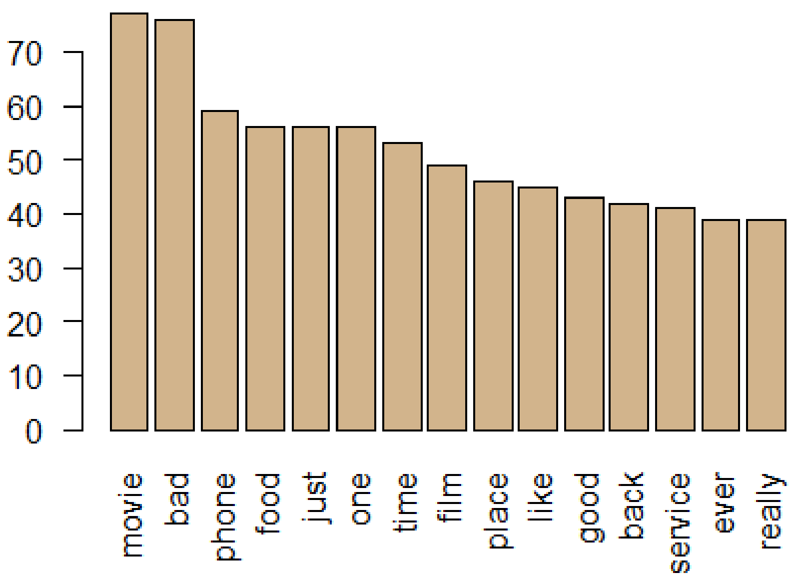
```
term_frequency <- sort(term_frequency, decreasing = T)
term_frequency[1:10]

##   good great movie phone  film   one  food  like  just place
##    228   208   179   164   155   144   125   123   119   114

#showcasing top 15 words
barplot(term_frequency[1:15], col = "tan", las = 2)
```



Here is the visual representation of the words against its count value. The word 'good' being the most used word and 'place' being the least.

```
#getting rid of yelp,amazon and imdb from the sentiments
clean.corpus_negative_updated <- tm_map(clean.corpus_negative, removeWords,
c("yelp","amazon","imdb"))

## Warning in tm_map.SimpleCorpus(clean.corpus_negative, removeWords,
## c("yelp", : transformation drops documents

clean.corpus_positive_updated <- tm_map(clean.corpus_positive, removeWords,
c("yelp","amazon","imdb"))

## Warning in tm_map.SimpleCorpus(clean.corpus_positive, removeWords,
## c("yelp", : transformation drops documents

#creating document term matrix
clean.corpus_negative.dtm <-
DocumentTermMatrix(clean.corpus_negative_updated)
```

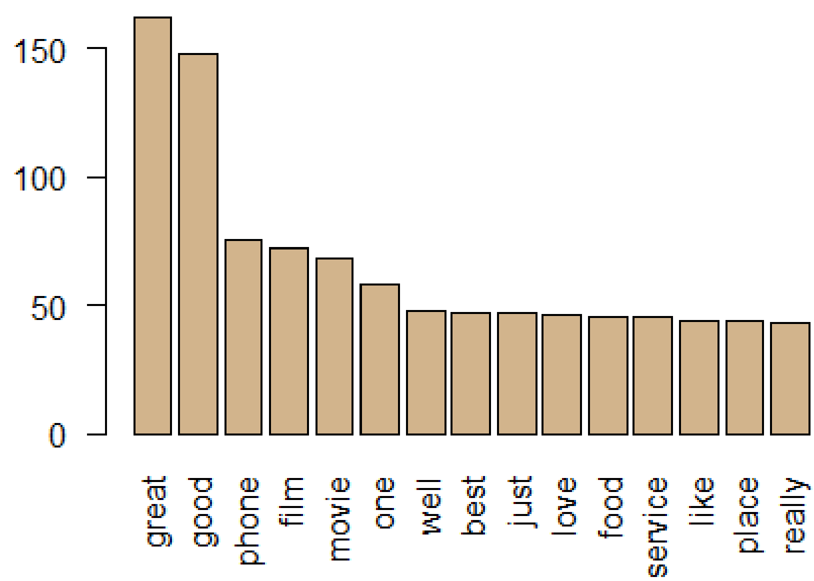
```
clean_matrix_negative <- as.matrix(clean.corpus_negative.dtm)

#generating term frequncy matrix for the negative sentiments
term_frequency_negative <- colSums(clean_matrix_negative)
term_frequency_negative <- sort(term_frequency_negative, decreasing = T)
term_frequency_negative[1:10]

## movie   bad phone  food  just   one  time  film place  like
##    77    76    59    56    56    56    53    49    46    45

barplot(term_frequency_negative[1:15], col = "tan", las = 2)
```



As we noticed that the words yelp, amazon and imdb were being used in the sentiments section, which were not useful in anyway, I performed some further cleaning and removed them from the term document matrix so that I can only see the useful words. The above bar graph tells me that the words 'movie' and 'bad' have been used the most. Maybe customers were trying to correlate the specific movie as bad.

```
#creating document term matrix
clean.corpus_positive.dtm <-
DocumentTermMatrix(clean.corpus_positive_updated)

clean_matrix_positive <- as.matrix(clean.corpus_positive.dtm)

#generating term frequency for the postive sentiments
term_frequency_positive <- colSums(clean_matrix_positive)
```

```
term_frequency_positive <- sort(term_frequency_positive, decreasing = T)

term_frequency_positive[1:10]

## great   good phone  film movie   one  well  best  just  love
##   162    148    75    72    68    58    48    47    47    46

#displaying top 15 words
barplot(term_frequency_positive[1:15], col = "tan", las = 2)
```



As we noticed that the words yelp, amazon and imdb were being used in the sentiments section, which were not useful in anyway, I performed some further cleaning and removed them from the term document matrix so that I can only see the useful words. The above bar graph tells me that the words 'great' and 'good' have been used the most. Maybe customers were trying to correlate the specific restaurant, film, movie or a product were good.

```
freq.terms <- findFreqTerms(clean.corpus.dtm, lowfreq = 20)
freq.terms

##  [1] "everything"   "movie"        "terrible"     "love"
##  [5] "really"       "two"          "wonderful"    "time"
##  [9] "worth"        "first"        "get"          "great"
## [13] "little"       "now"          "also"         "bad"
## [17] "will"         "characters"   "money"        "one"
## [21] "waste"        "nice"         "people"       "service"
## [25] "item"         "better"       "never"        "film"
```

```
##  [29] "good"         "battery"     "buy"         "right"
##  [33] "well"         "enough"      "product"     "case"
##  [37] "ear"          "minutes"     "made"        "place"
##  [41] "however"      "like"        "real"        "since"
##  [45] "even"         "best"        "ever"        "phone"
##  [49] "price"        "story"       "make"        "see"
##  [53] "can"          "take"        "camera"      "used"
##  [57] "awful"        "food"        "thing"       "fine"
##  [61] "disappointed" "way"         "easy"        "use"
##  [65] "horrible"     "say"         "just"        "times"
##  [69] "got"          "works"       "loved"       "movies"
##  [73] "seen"         "years"       "comfortable" "headset"
##  [77] "piece"        "back"        "going"       "watching"
##  [81] "pretty"       "vegas"       "came"        "restaurant"
##  [85] "excellent"    "definitely"  "recommend"   "look"
##  [89] "far"          "many"        "anyone"      "lot"
##  [93] "another"      "character"   "nothing"     "quality"
##  [97] "always"       "around"      "found"       "screen"
## [101] "sound"        "every"       "acting"      "script"
## [105] "much"         "think"       "give"        "amazing"
## [109] "highly"       "experience"  "life"        "awesome"
## [113] "poor"         "worst"       "still"       "friendly"
## [117] "happy"        "worked"      "know"        "new"
## [121] "quite"        "delicious"   "absolutely"  "want"
## [125] "long"         "work"        "funny"       "probably"
## [129] "went"         "plot"        "films"       "car"
## [133] "bought"
```

Just for some analysis, I tried to find the words from the term doctument matrix which has lowfreq upto 20. This will help me know which words were highly used and which were not.

```
#finding the correlation among the words within the corpus data
findAssocs(clean.corpus.dtm, c("great","movie","good","food","bad") ,
corlimit=0.1)

## $great
##     works colleague      pics  desserts
##      0.14      0.13      0.13      0.11
##
## $movie
##      beginning    fascinating           duet           june      endearing
##           0.17           0.15           0.15           0.15           0.14
##        familys        latched       girolamo          titta         vision
##           0.14           0.14           0.14           0.14           0.14
##      astronaut     astronauts      considers           ussr          coach
##           0.14           0.14           0.14           0.14           0.14
##        columbo        peaking          dodge         makers        planned
##           0.14           0.14           0.14           0.14           0.14
##         québec     restrained        stratus          angel        curtain
```

```
##          0.14          0.14          0.14          0.14          0.14
##       edition excellentangel      funniest           hes           ive
##          0.14          0.14          0.14          0.14          0.14
##           lid         scamp         yelps       shelves       special
##          0.14          0.14          0.14          0.14          0.13
##        scared      occupied         spent   pretentious      business
##          0.12          0.12          0.12          0.11          0.11
##        visual           art          fear
##          0.10          0.10          0.10
##
## $good
## cancellation   yearsgreat       laughs       watson     angelina
##         0.13         0.13         0.13         0.13         0.13
##        cameo        elias   koteasjack   nakedbilly          ole
##         0.13         0.13         0.13         0.13         0.13
##      palance         sven      thorsen       prices        value
##         0.13         0.13         0.13         0.11         0.10
##      appears
##         0.10
##
## $food
## connoisseur  difference        luke       sever  despicable     service
##        0.26        0.18        0.17        0.17        0.17        0.15
##   delicious     wasting   selection overwhelmed   preparing     typical
##        0.13        0.12        0.12        0.12        0.12        0.12
##         omg       eaten
##        0.12        0.11
##
## $bad
##        plain       acting      crayons      haggis       handle
##         0.20         0.19         0.18        0.18         0.18
##      painted storytelling      strokes      backed     bordered
##         0.18         0.18         0.18        0.18         0.18
##       flakes   needlessly      repeats  slowmotion    stupidity
##         0.18         0.18         0.18        0.18         0.18
##     vehicles  connoisseur   cheesiness   unethical         bold
##         0.18         0.18         0.18        0.18         0.12
##      crafted          ugly  difference    involved    direction
##         0.12         0.12         0.12        0.12         0.12
##         corn     normally          box  continuity  directorial
##         0.12         0.12         0.12        0.12         0.12
##     contains        proud      writing      script         idea
##         0.12         0.12         0.11        0.11         0.10
##       reason
##         0.10
```

This is an important step which helps us to know the correlation between the words in the dataset. I tried to find the correlation between 'great' and 'movie'. All the words associated with 'great','movie' and their limit of correlation is mentioned. For example, the word great is associated with 'works', 'colleague','pics' and 'desserts'. We can sort of conclude

that maybe the customers are trying to convey that the works is great, pics are great or the desserts are great.

N-gram Analysis

```
library(tau)

## Warning: package 'tau' was built under R version 3.5.3

##
## Attaching package: 'tau'

## The following object is masked from 'package:readr':
##
##     tokenize

library(data.table)

## Warning: package 'data.table' was built under R version 3.5.3

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose

# given a string vector and size of ngrams this function returns word ngrams
with corresponding frequencies
createNgram <-function(stringVector, ngramSize){

  ngram <- data.table()

  ng <- textcnt(stringVector, method = "string", n=ngramSize, tolower =
FALSE)

  if(ngramSize==1){
    ngram <- data.table(w1 = names(ng), freq = unclass(ng),
length=nchar(names(ng)))
  }
  else {
    ngram <- data.table(w1w2 = names(ng), freq = unclass(ng),
length=nchar(names(ng)))
  }
  return(ngram)
}
```

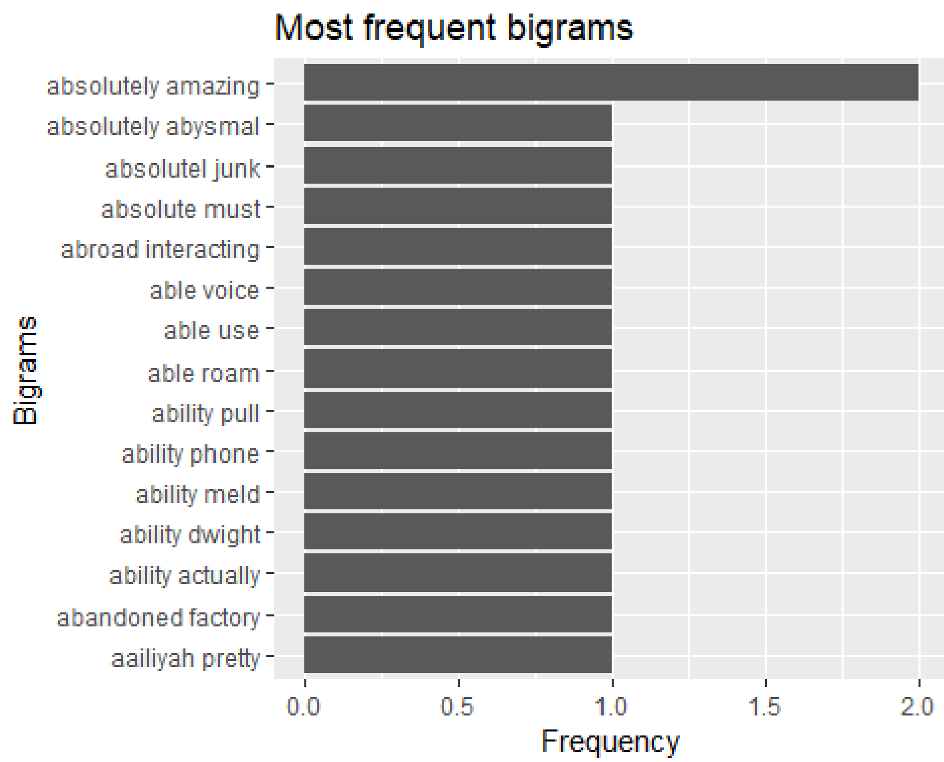
```r
#creating the n-gram and storing it in the variable, for bigram n=2
res <- createNgram(clean.corpus, 2)
res[1:20]
```

```
##                      w1w2 freq length
##  1:       aailiyah pretty    1     15
##  2:     abandoned factory    1     17
##  3:      ability actually    1     16
##  4:        ability dwight    1     14
##  5:          ability meld    1     12
##  6:         ability phone    1     13
##  7:          ability pull    1     12
##  8:             able roam    1      9
##  9:              able use    1      8
## 10:            able voice    1     10
## 11:    abroad interacting    1     18
## 12:         absolute must    1     13
## 13:        absolutel junk    1     14
## 14:    absolutely abysmal    1     18
## 15:    absolutely amazing    2     18
## 16: absolutely appalling    1     20
## 17:       absolutely back    1     15
## 18:       absolutely clue    1     15
## 19: absolutely delicious    1     20
## 20: absolutely flatlined    1     20
```

```r
#color palette
pal=brewer.pal(8,"Blues")
pal=pal[-(1:3)]

#creating a wordcloud for the result
wordcloud(res$w1w2,res$freq,max.words=50,scale=c(2.5,.20),random.order =
F,rot.per=.5,vfont=c("sans serif","plain"),colors=pal)
```

```
## Warning in wordcloud(res$w1w2, res$freq, max.words = 50, scale = c(2.5, :
## movies ever could not be fit on page. It will not be plotted.

## Warning in wordcloud(res$w1w2, res$freq, max.words = 50, scale = c(2.5, :
## well made could not be fit on page. It will not be plotted.

## Warning in wordcloud(res$w1w2, res$freq, max.words = 50, scale = c(2.5, :
## works well could not be fit on page. It will not be plotted.

## Warning in wordcloud(res$w1w2, res$freq, max.words = 50, scale = c(2.5, :
## years ago could not be fit on page. It will not be plotted.
```


```r
#creating a data frame for the result since ggplot takes data frames only
freq.df1 <- data.frame(res)

#plotting the graph of the bigram and frequency
ggplot(head(freq.df1,15), aes(reorder(w1w2,freq), freq)) +
  geom_bar(stat = "identity") + coord_flip() +
  xlab("Bigrams") + ylab("Frequency") +
  ggtitle("Most frequent bigrams")
```



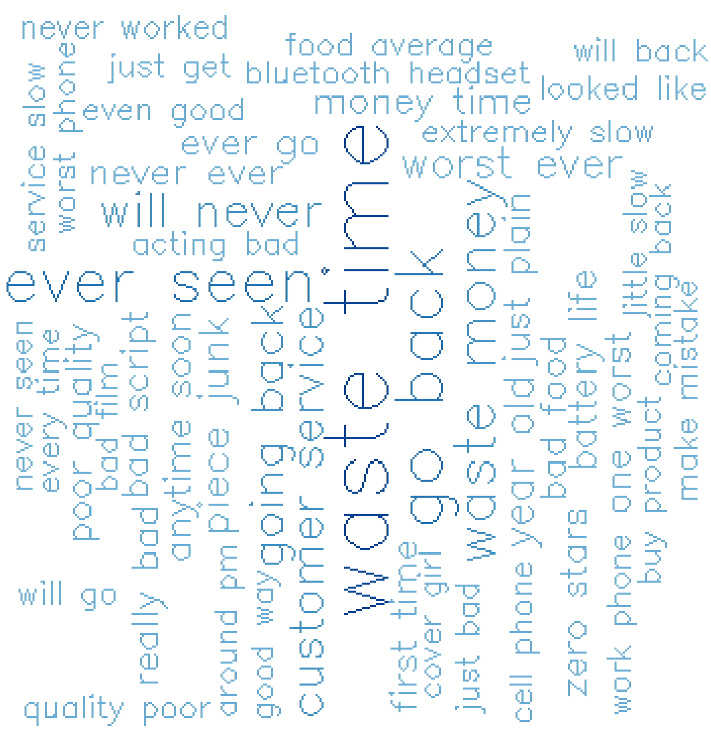
From the above results we can notice that the words are not used together more than once apart from a few rare cases where the frequency shows upto 2 or 3. This maybe because of the data. When we have a limited amount of data, the relationship between words get restricted. Although, word cloud shows that a few combinations like 'works great','customer service' have been used quite heavily.

```
#the n-gram analysis for the negative sentiments
res_negative <- createNgram(clean.corpus_negative_updated, 2)
res_negative[1:20]

##                      w1w2 freq length
##  1:     abandoned factory    1     17
##  2:             abhor bad    1      9
##  3:       able recognizes    1     15
##  4:         abound months    1     13
##  5:       absolutely clue    1     15
##  6:  absolutely flatlined    1     20
##  7:     absolutely flavor    1     17
##  8:    absolutely nothing    1     18
##  9:   absolutely suspense    1     19
## 10:     absolutely warmth    1     17
## 11:         abysmal sadly    1     13
## 12:       accept anything    1     15
## 13:        accessory good    1     14
## 14: accidentally activate    1     21
## 15:    accidentally touch    1     18
## 16:  accolades especially    1     20
```

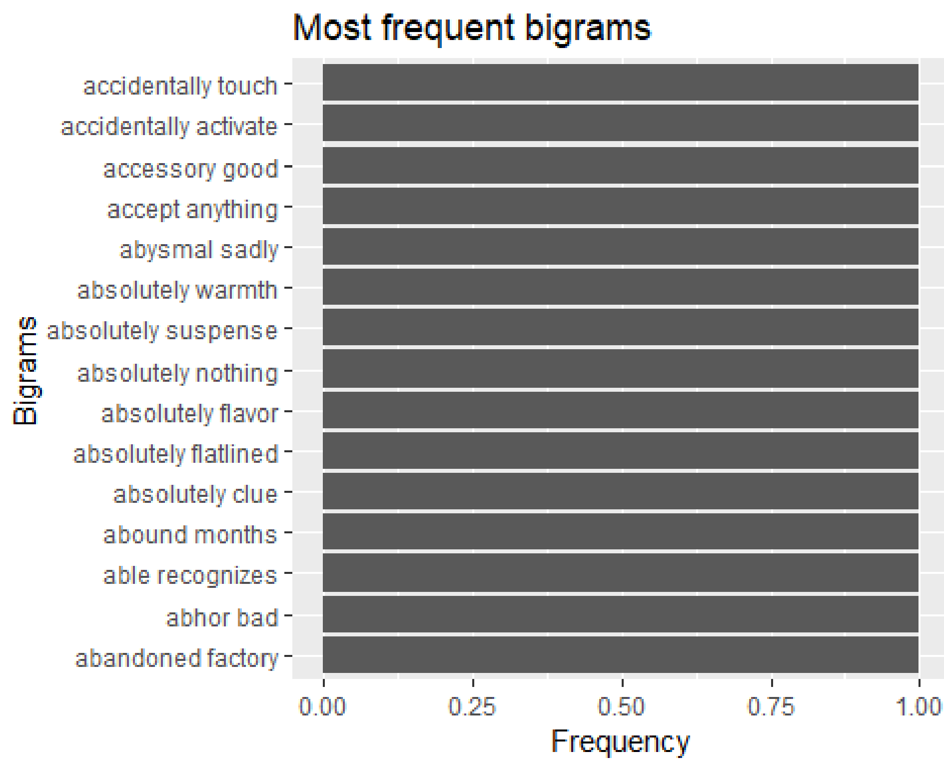
```
## 17:       accountant know    1     15
## 18:    accurately defined    1     18
## 19:         accused murder    1     14
## 20:  acknowledged another    1     20
```

```r
pal=brewer.pal(8,"Blues")
pal=pal[-(1:3)]

#word cloud for the negative sentiments with max words being 50.
wordcloud(res_negative$w1w2,res_negative$freq,max.words=50,scale=c(3,.50),random.order = F,rot.per=.5,vfont=c("sans serif","plain"),colors=pal)
```

```
## Warning in wordcloud(res_negative$w1w2, res_negative$freq, max.words =
## 50, : waited waited could not be fit on page. It will not be plotted.
```




```r
freq.df_negative <- data.frame(res_negative)

#plotting the graph for negative sentiment bigrams for top 15
ggplot(head(freq.df_negative,15), aes(reorder(w1w2,freq), freq)) +
  geom_bar(stat = "identity") + coord_flip() +
  xlab("Bigrams") + ylab("Frequency") +
  ggtitle("Most frequent bigrams")
```



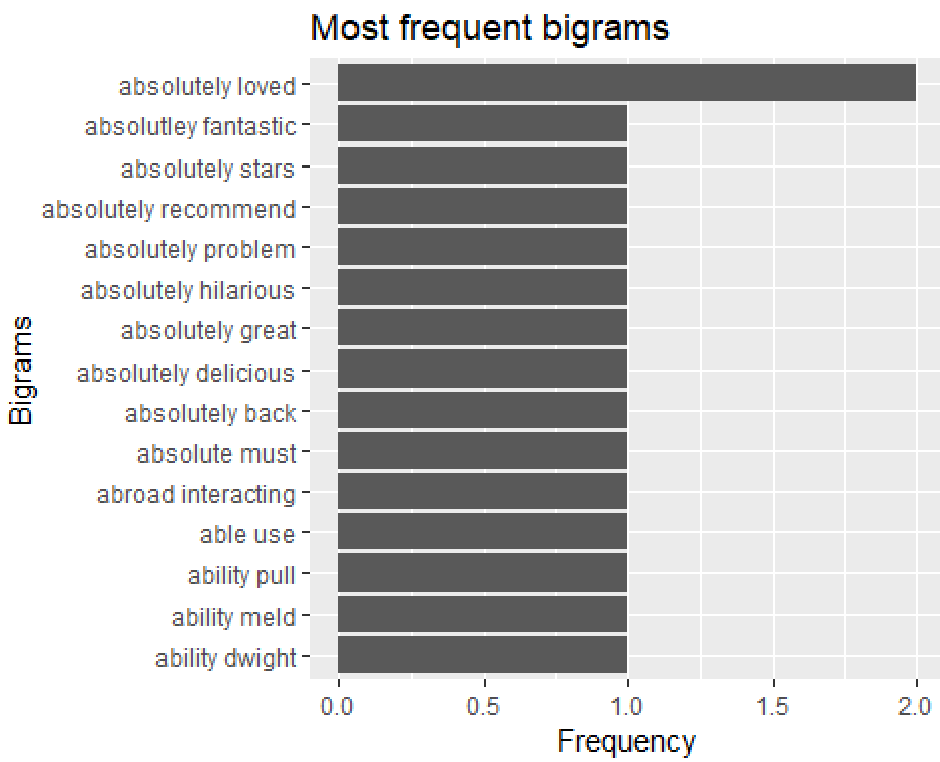
The above results shows the top 15 negative sentiment bigrams which has frequency as 1. This means that the specific combinations have only been used once throughout the data set. Wordcloud shows interesting combinations popping up like, 'waste time','waste money','go back' which tells us a lot about the customers sentiments behind a specific movie, restaurant or a product on amazon.

```
library(igraph)

## Warning: package 'igraph' was built under R version 3.5.3

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:purrr':
##
##     compose, simplify

## The following object is masked from 'package:tidyr':
##
##     crossing

## The following object is masked from 'package:tibble':
##
##     as_data_frame
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

#results for positive sentiments for bigrams
res_positive<- createNgram(clean.corpus_positive_updated, 2)
res_positive[1:20]

##                     w1w2 freq length
##  1:       ability dwight    1     14
##  2:         ability meld    1     12
##  3:         ability pull    1     12
##  4:             able use    1      8
##  5:   abroad interacting    1     18
##  6:        absolute must    1     13
##  7:      absolutely back    1     15
##  8: absolutely delicious    1     20
##  9:     absolutely great    1     16
## 10: absolutely hilarious    1     20
## 11:     absolutely loved    2     16
## 12:   absolutely problem    1     18
## 13: absolutely recommend    1     20
## 14:     absolutely stars    1     16
## 15: absolutley fantastic    1     20
## 16:        academy award    1     13
## 17:         access phone    1     12
## 18:    accessable lastly    1     17
## 19:     accessible films    1     16
## 20:   accessing internet    1     18

#color palette
pal=brewer.pal(8,"Blues")
pal=pal[-(1:3)]

#generating a word cloud for the positive sentiment bigrams
wordcloud(res_positive$w1w2,res_positive$freq,max.words=50,scale=c(3,.34),ran
dom.order = F,rot.per=.5,vfont=c("sans serif","plain"),colors=pal)

## Warning in wordcloud(res_positive$w1w2, res_positive$freq, max.words =
## 50, : works well could not be fit on page. It will not be plotted.

## Warning in wordcloud(res_positive$w1w2, res_positive$freq, max.words =
## 50, : fits comfortably could not be fit on page. It will not be plotted.

## Warning in wordcloud(res_positive$w1w2, res_positive$freq, max.words =
## 50, : good prices could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(res_positive$w1w2, res_positive$freq, max.words =
## 50, : good time could not be fit on page. It will not be plotted.

## Warning in wordcloud(res_positive$w1w2, res_positive$freq, max.words =
## 50, : great reception could not be fit on page. It will not be plotted.

## Warning in wordcloud(res_positive$w1w2, res_positive$freq, max.words =
## 50, : happy purchase could not be fit on page. It will not be plotted.

## Warning in wordcloud(res_positive$w1w2, res_positive$freq, max.words =
## 50, : place good could not be fit on page. It will not be plotted.

## Warning in wordcloud(res_positive$w1w2, res_positive$freq, max.words =
## 50, : sound effects could not be fit on page. It will not be plotted.
```


```r
freq.df_positive <- data.frame(res_positive)

#creating a plot for the bigrams for positive sentiments
ggplot(head(freq.df_positive,15), aes(reorder(w1w2,freq), freq)) +
  geom_bar(stat = "identity") + coord_flip() +
  xlab("Bigrams") + ylab("Frequency") +
  ggtitle("Most frequent bigrams")
```




```
#separating the bigrams into two words
bigram_net <- freq.df1 %>%
        separate(w1w2,c( "word1", "word2"), sep = " ") %>%
        count(word1,word2, sort = TRUE)
bigram_net

## # A tibble: 13,799 x 3
##    word1     word2        n
##    <chr>     <chr>    <int>
##  1 aailiyah  pretty       1
##  2 abandoned factory      1
##  3 ability   actually     1
##  4 ability   dwight       1
##  5 ability   meld         1
##  6 ability   phone        1
##  7 ability   pull         1
##  8 able      roam         1
##  9 able      use          1
## 10 able      voice        1
## # ... with 13,789 more rows

relation_bigram <- bigram_net %>%
  filter(word2 == "bad") %>%
  count(word1, sort = TRUE)
relation_bigram
```

```
## # A tibble: 63 x 2
##    word1          n
##    <chr>      <int>
##  1 acting         1
##  2 also           1
##  3 anything       1
##  4 apologize      1
##  5 backed         1
##  6 bad            1
##  7 beyond         1
##  8 book           1
##  9 buttons        1
## 10 camerawork     1
## # ... with 53 more rows

AFINN <- get_sentiments("afinn")

words_with_bad <- bigram_net %>%
  filter(word1 == "bad") %>%
  inner_join(AFINN, by = c(word2 = "word")) %>%
  count(word2, score, sort = TRUE)
words_with_bad

## # A tibble: 5 x 3
##   word2 score     n
##   <chr> <int> <int>
## 1 bad      -3     1
## 2 fit       1     1
## 3 kind      2     1
## 4 lost     -3     1
## 5 pay      -1     1

words_with_good <- bigram_net %>%
  filter(word1 == "good") %>%
  inner_join(AFINN, by = c(word2 = "word")) %>%
  count(word2, score, sort = TRUE)
words_with_good

## # A tibble: 8 x 3
##   word2     score     n
##   <chr>     <int> <int>
## 1 bargain       2     1
## 2 free          1     1
## 3 friendly      2     1
## 4 glad          3     1
## 5 humorous      2     1
## 6 nice          3     1
## 7 pretty        1     1
## 8 problems     -2     1
```

Since the correlation limit did not give me a clear idea of the relationship between the words, I performed a N-gram analysis. The results were quite interesting on its own. Even though the data is limited, the combinations of words were just used once throughout, apart from a few rare ones. I could find out that the combinations like “works great”,“sounds good”,“waste time”,“go back”,“customer service” were used widely. This may also indicate that every customer had a unique way of writing their reviews about the restaurant, a movie or a product on amazon. This might actaully help in analysing the reviews made by actaul people and the ones made by bots. Overall, N-gram analysis helped me to identify the uniqueness among the 2 words which are used together in the text, some seems to be meaningful and some not. Get sentiments function helped me analyse the sentiments of the words associated with it and tried to interpret the words associated with the first word being ‘bad’ (followed by ‘bad’) and followed by ‘good’. The words associated with ‘good’ mostly have positive sentiment score where highest being the 3 for ‘glad’ and ‘nice’ and lowest for the ‘problems’ as -2. The scores show the weight and impact of those words in the reviews.

```
#To reduce the number of features, I will eliminate any words that appear in
less than three texts, or less than about 0.1 percent of records in the
training data.
freq.terms_modeling <- findFreqTerms(clean.corpus.dtm.train, 3)

#creating a training dataset
clean.corpus.dtm.freq.train <- DocumentTermMatrix(clean.corpus.train,
list(dictionary = freq.terms))

#creating a test dataset
clean.corpus.dtm.freq.test  <- DocumentTermMatrix(clean.corpus.test,
list(dictionary = freq.terms))

#The naive Bayes classifier is typically trained on data with categorical
features. This poses a problem since the cells in the sparse matrix indicate
a count of the times a word appears in a message. We should change this to a
factor variable that simply indicates yes or no depending on whether the word
appears at all.

convert_counts <- function(x) {
    x <- ifelse(x > 0, 1, 0)
    x <- factor(x, levels = c(0, 1), labels = c("No", "Yes"))
    return(x)
}

clean.corpus.dtm.freq.train <- apply(clean.corpus.dtm.freq.train, MARGIN = 2,
convert_counts)
clean.corpus.dtm.freq.test  <- apply(clean.corpus.dtm.freq.test, MARGIN = 2,
convert_counts)

# Constructing model and making prediction
text.classifer <- naiveBayes(clean.corpus.dtm.freq.train,
```

```
raw.text.train$sentiment)
text.pred <- predict(text.classifer, clean.corpus.dtm.freq.test)

t <- CrossTable(text.pred, raw.text.test$sentiment,
           prop.chisq = FALSE,
           prop.t = FALSE,
           dnn = c('predicted', 'actual'))

##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  600
##
##
##              | actual
##    predicted |  Negative |  Positive | Row Total |
## -------------|-----------|-----------|-----------|
##     Negative |       246 |       104 |       350 |
##              |     0.703 |     0.297 |     0.583 |
##              |     0.837 |     0.340 |           |
## -------------|-----------|-----------|-----------|
##     Positive |        48 |       202 |       250 |
##              |     0.192 |     0.808 |     0.417 |
##              |     0.163 |     0.660 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       294 |       306 |       600 |
##              |     0.490 |     0.510 |           |
## -------------|-----------|-----------|-----------|
##
##

#accuracy = (TP + TN)/(TP + TN + FP + FN)
library(scales)

## Warning: package 'scales' was built under R version 3.5.3

##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
accuracy <- sum(diag(t$t))/sum(t$t)
percent(accuracy)
```

```
## [1] "74.7%"
```

CONCLUSION

From the visualisations, I can make some assumptions and not concrete results since I have taken just a small sample for now. So, the words which appeared in the wordcloud of the cleaned data represents a lot of positive words in bold and abundance. This means that the customers were quite happy with the service in the restuarant, or the product was really good on Amazon or maybe the movie was really good. The bag of words and the term matrix helped us know the words used in the row and how many times it is being used. This can help us analyse more accurately and get the results when sentiments are considered.

I also analysed the difference between the cleaned sentiments and the sentiments taken from raw text data. With the correlation among specific words with the text data, we can get to know their relationship and would help us analyse our sentiments more in detail. For example, in case, good movie and bad movie both comes up together in both positive and negative analysis, the correlation factor would help us compare the limit of correlation and our conclusion would be better for the one combination which will have a higher correlation limit.

Since the correlation limit did not give me a clear idea of the relationship between the words, I performed a N-gram analysis. The results were quite interesting on its own. Even though the data is limited, the combinations of words were just used once throughout, apart from a few rare ones. I could find out that the combinations like "works great","sounds good","waste time","go back","customer service" were used widely. This may also indicate that every customer had a unique way of writing their reviews about the restaurant, a movie or a product on amazon. This might actaully help in analysing the reviews made by actaul people and the ones made by bots. Overall, N-gram analysis helped me to identify the uniqueness among the 2 words which are used together in the text, some seems to be meaningful and some not. Get sentiments function helped me analyse the sentiments of the words associated with it and tried to interpret the words associated with the first word being 'bad' (followed by 'bad').

I finally performed the Naive bayes classification and predicted the output. To reduce the number of features, we will eliminate any words that appear in less than three texts, or less than about 0.1 percent of records in the training data. The results of the Naive bayes was quite interesting where, True Positive being 0.66 and True negative as 0.837. The accuracy turned out to be 74.7% which is quite good for data we have. We can improve the accuracy of the model by setting Laplace = 1. This means that the words which have not appreared in either Positive or Negative texts to have an indisputable impact in the classification model will get fixed.