Predict Review Rating Using Sentiment Analysis by R

Eyad Alkronz

eyadalkronz@gmail.com

9/30/2021

Overview

The computational process of automatically determining what feelings a writer is expressing in text is known as sentiment analysis. Sentiment is frequently characterized as a binary contrast (positive vs. negative), but it can also be more nuanced, such as recognizing the exact emotion expressed by an author (like fear, joy or anger).

How does it work?

1. Make or find a list of terms that have a strong positive or negative connotation.
2. Count how many positive and negative terms there are in the text.
3. Examine the proportion of positive and negative words. Positive sentiment is indicated by a large number of positive words and a small number of negative words, whereas negative emotion is shown by a large number of negative words and a small number of positive words.

The Data

The data used for the project is the TripAdvisor Hotel Review Dataset, A dataset for TripAdvisor Hotel Review, crawled from Tripadvisor , it consisting of 20k reviews and data avilable on Kaggle on this link https://www.kaggle.com/andrewmvd/trip-advisor-hotel-reviews

Load Data

in this section check if data exists , if not then download it from url ,and then add id for each row in data.frame, this id represent reviewID ,and create new column with name newRating to represent Negative or positive , positive when rating equal three,four or five and negative when rating equal one or two

Looking at the first few rows of the “data”, we can see the features which are “id”, “Review”, “Rating”, “newRating”, Each row represents a single customer review . id: represent the review-ID , Review: contains the customer review as text , Rating: contains the customer rating [0,5] , newRating: contains the customer rating [0,1] , zero represent negative impact and one represent positive ,

```
data%>% select( Rating,id,newRating )%>%head()
```

```
##   Rating id newRating
## 1      4  1         1
## 2      2  2         0
## 3      3  3         1
## 4      5  4         1
## 5      5  5         1
## 6      5  6         1
```

A summary of the data can confirm that there are no missing values.

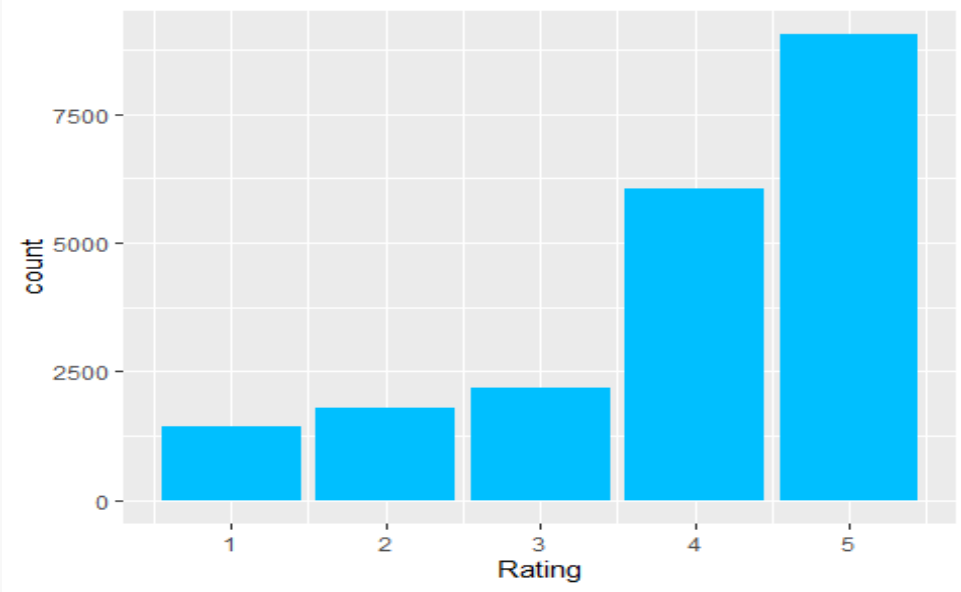
```
##     Review              Rating            id          newRating
##  Length:20491       Min.   :1.000   Min.   :    1   Min.   :0.0000
##  Class :character   1st Qu.:3.000   1st Qu.: 5124   1st Qu.:1.0000
##  Mode  :character   Median :4.000   Median :10246   Median :1.0000
##                     Mean   :3.952   Mean   :10246   Mean   :0.8432
##                     3rd Qu.:5.000   3rd Qu.:15368   3rd Qu.:1.0000
##                     Max.   :5.000   Max.   :20491   Max.   :1.0000
```

Exploratory Data Analysis

in this section we visualisation and transformation to explore data in a systematic way,Check for data quality; confirm meaning and prevalence of missing values ,Understand univariate relationships between variables ,Perform an initial assessment on what variables to include and what transformations need to be done on them.

Visualizing distributions of Rating

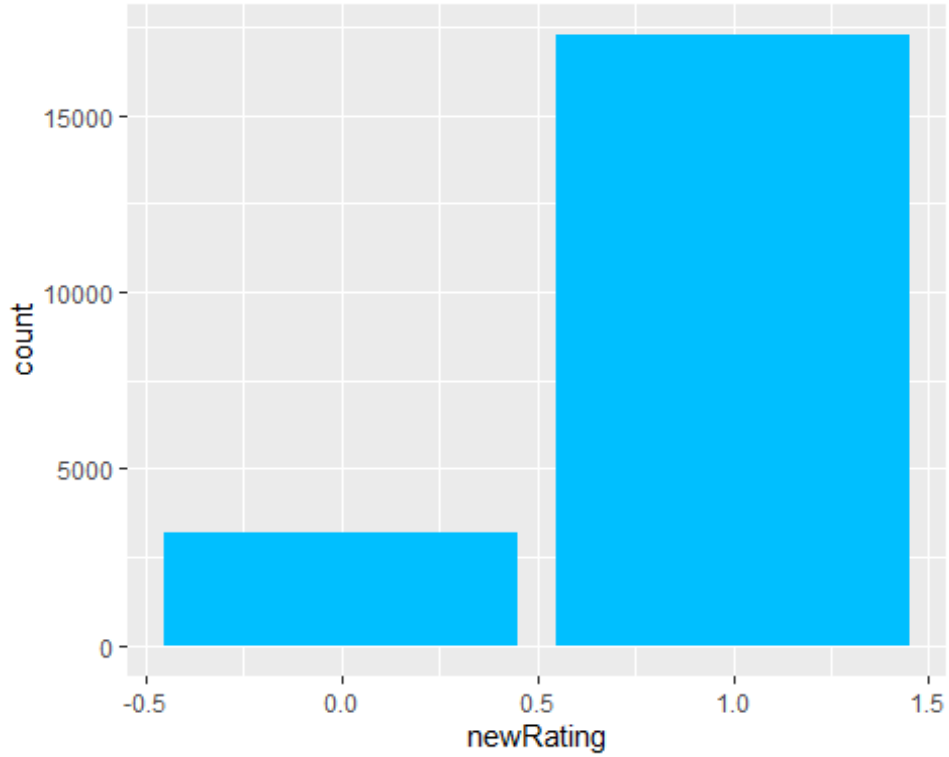
```
#group by rate Plot

data %>%
  ggplot(aes(x= Rating ))+
  geom_bar(fill="deepskyblu
```




```
e")
```

Visualizing distributions of Rating as Negative equal zero or positive equal one

```
 #group by new rate Plot
data %>%
  ggplot(aes(x= newRating ))+
  geom_bar( fill="deepskyblue" )
```

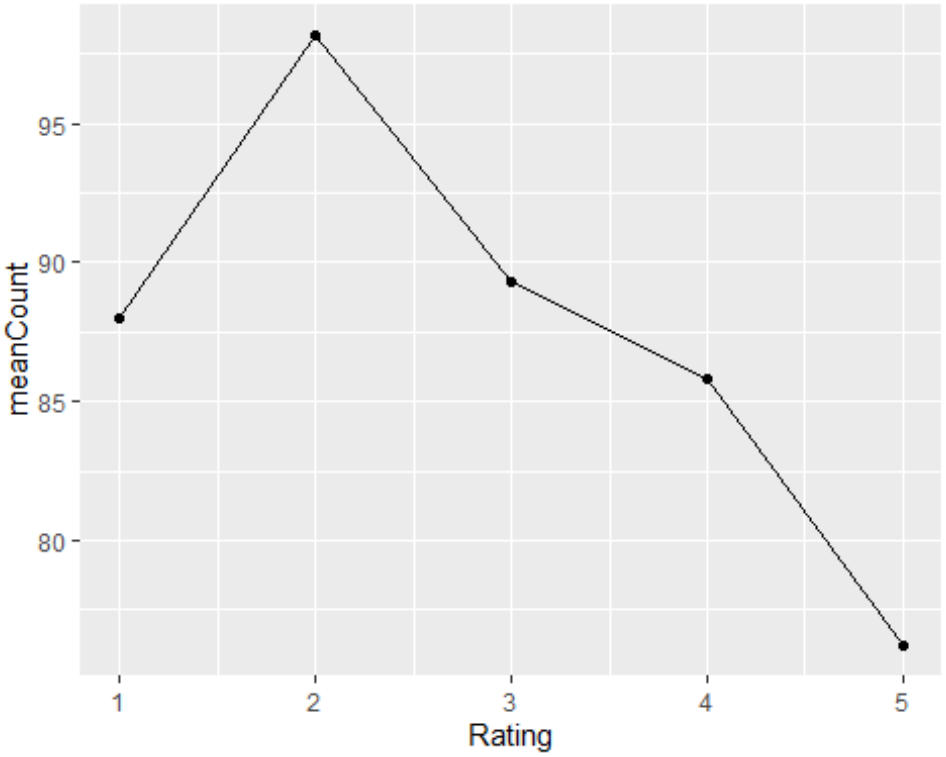


Data Preparation

The first step, creating or finding a word list (also called a lexicon), is generally the most time-consuming , and then count the number of word in each Sentence

Visualizing relation between Rating and WordCount

```
 #groub by new rate Plot
data%>% group_by(Rating) %>%
  summarize(meanCount = mean(WordCount)) %>%
  ggplot(aes(x = Rating , y = meanCount   ))+
  geom_point() +
  geom_line()
```



Insight : Higher Rated Reviews tend to have less words while, lower rated reviews have very high word count

Sentiment Analysis

in this analysis , used Bing lexicon for sentiment analysis ,The bing lexicon categorizes words in a binary fashion into positive and negative categories.

```
 bing <- get_sentiments("bing") %>%
  select(word, sentiment)%>%
  mutate(value = ifelse(sentiment == "negative" , 0 , 1) )

bing%>% count(sentiment)

## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 negative   4781
## 2 positive   2005
```

Now , join our Review words(tokens) list with bing words and assign value for each words to its value that offered by bing

```
## # A tibble: 6 x 4
## # Groups:   id [1]
##      id newRating value word
```

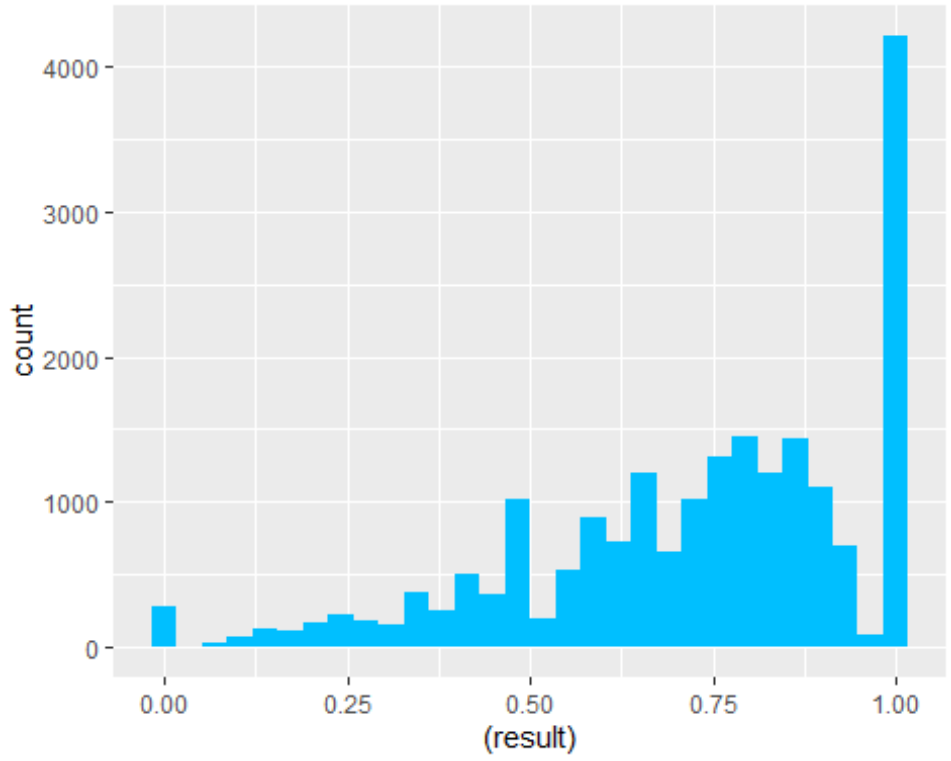
```
##   <int>     <dbl> <dbl> <chr>
## 1     1         1 1     nice
## 2     1         1 0.643 hotel
## 3     1         1 0     expensive
## 4     1         1 0.643 parking
## 5     1         1 0.643 deal
## 6     1         1 0.643 stay
```

Now calculate value for each sentences and create new column with name result , result = average of values for all words in this sentence

```
predicted_rating_per_sentences <-  Review_words%>%
  group_by(id) %>%
  summarize(result = (mean(value)))
```

plot histogram for predicted rating per sentences

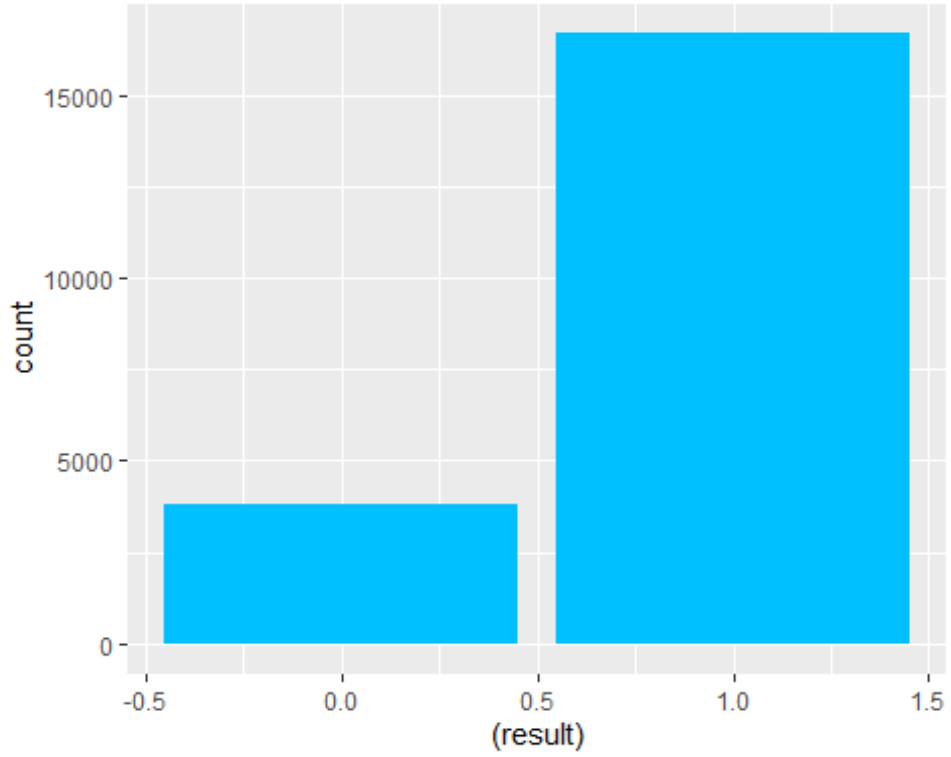
```
predicted_rating_per_sentences %>% ggplot(aes((result))) +
  geom_histogram(fill="deepskyblue")

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



After applying round() function on result

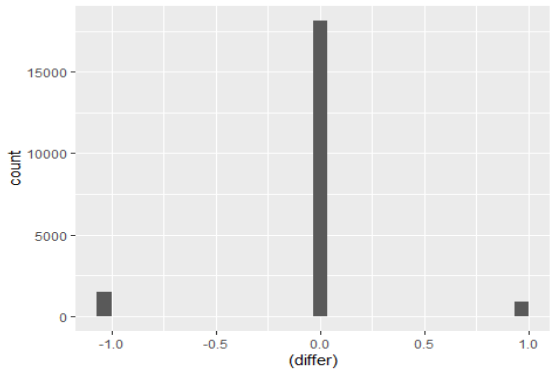
```
predicted_rating_per_sentences <- predicted_rating_per_sentences%>%
  mutate(result  = round(result))

predicted_rating_per_sentences%>%
```

```
  ggplot(aes((result))) +
  geom_bar(fill="deepskyblue")
```



join predicted rating per sentences with original Data and create new column with name differ to represent deference between predicted Rating and original Rating

```
#join predicted_rating_per_sentences with original Data
data <- predicted_rating_per_sentences %>% left_join(data ,by = "id")%>%
  mutate(differ = result - newRating )

#plot histogram to show differ
data %>% ggplot(aes((differ))) +
  geom_histogram()
```



Calculate Accurecy

```
successCount <- data %>% filter(differ == 0) %>% count()%>%pull
failCout<- data %>% filter(differ != 0) %>% count()%>%pull
totalCount <- data%>%count()%>%pull

accuracy <- successCount / totalCount

print(paste("Accuracy = ",accuracy))

## [1] "Accuracy =  0.885217900541701"
```

Conclusion

The overall aim is to use Sentiment Analysis technique over user review to predict rating is positive or negative , I found the accuracy after implementing these technique of Sentiment Analysis equal to 0.885 ,Sentiment analysis is not perfect, and as with any automatic analysis of language, you will have errors in your results. It also cannot tell you why a writer is feeling a certain way. However, it can be useful to quickly summarize some qualities of text, especially if you have so much text that a human reader cannot analyze all of it.

Appendix

Environment

Operating System:

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          4
## minor          1.1
## year           2021
## month          08
## day            10
## svn rev        80725
## language       R
## version.string R version 4.1.1 (2021-08-10)
## nickname       Kick Things
```