satujoda-ss3203-finalProject

Shanmukha Srivathsav Satujoda

May 3, 2019

#Scraping function to scrape The Godfather movie reviews, rating and review date from metacritic.com  
collect\_data\_metacritic <- function(url){  
 #Reading the url  
 meta\_god <- read\_html(url)  
   
 #Getting the Rating of the user   
 rating\_data<- html\_nodes(meta\_god,'.indiv') %>% html\_text() %>% as.data.frame()  
 colnames(rating\_data)[1] <- "Rating"  
   
 #Getting the review of the user  
 review\_data <- html\_nodes(meta\_god,'.review\_body') %>% html\_text() %>% as.data.frame()  
 review\_data <- as.data.frame(review\_data[-1,])  
 colnames(review\_data)[1] <- "Review"  
   
 #Getting the Review Date for reviews  
 review\_date <- html\_nodes(meta\_god, '.date')  
 review\_date <- html\_nodes(meta\_god, '.date') %>% html\_text() %>% as.data.frame()  
 colnames(review\_date)[1] <- "reviewDate"  
   
 #Creating arbirary id for joining  
 createID <- function(df){  
 df$id = 0  
 for(x in 1:nrow(df)){  
 df$id[x] = x  
 }  
 return(df)  
 }  
 rating\_data <- createID(rating\_data)  
 review\_data <- createID(review\_data)  
 review\_date <- createID(review\_date)  
   
 #join using ids to create a dataframe  
   
 data<- left\_join(rating\_data, review\_data, by='id') %>% left\_join(., review\_date, by='id')  
 data<- data[,-c(2)]  
 return(data)  
   
}

#There are 4 pages of reviews, so storing all the page urls   
url\_page0 = "https://www.metacritic.com/movie/the-godfather/user-reviews?sort-by=date&num\_items=100&page=0"  
url\_page1 = "https://www.metacritic.com/movie/the-godfather/user-reviews?sort-by=date&num\_items=100&page=1"  
url\_page2 = "https://www.metacritic.com/movie/the-godfather/user-reviews?sort-by=date&num\_items=100&page=2"  
url\_page3 = "https://www.metacritic.com/movie/the-godfather/user-reviews?sort-by=date&num\_items=100&page=3"  
  
#Creating temporary data frames for the each scraped data from the above mentioned pages  
#which are used later to form the entire data set  
temp\_data1 <- collect\_data\_metacritic(url\_page0)  
temp\_data2 <- collect\_data\_metacritic(url\_page1)  
temp\_data3 <- collect\_data\_metacritic(url\_page2)  
temp\_data4 <- collect\_data\_metacritic(url\_page3)  
  
  
#Combining data from different web pages to a single dataframe  
temp\_data <- rbind(temp\_data1,temp\_data2,temp\_data3,temp\_data4)

#This function is used for preprocessig the data:  
preprocessing<-function(df){  
 #direct conversion to numeric data is replacing the rating with level rather than the actual rating  
 df$Rating <- as.numeric(as.character(df$Rating))  
 df$Review <- (as.character(df$Review))  
   
 #Removing the special characters, leading, trailing white spaces  
 for(x in 1:nrow(df)){  
 df$Review[x] <- trimws(gsub("\r?\n|\r", " ", temp\_data$Review[x]))  
 df$Review[x] <- str\_replace\_all(df$Review[x], "[^[:alnum:]]", " ")  
 #Merging multiple white spaces to one white space  
 #NODE EXPLANATION  
 #--------------------------------------------------------------------------------  
 #(?<= look behind to see if there is:  
 # --------------------------------------------------------------------------------  
 #[\s] any character of: whitespace (\n, \r, \t, \f, and " ")  
 # --------------------------------------------------------------------------------  
 #) end of look-behind  
 #--------------------------------------------------------------------------------  
 #\s\* whitespace (\n, \r, \t, \f, and " ") (0 or  
 # more times (matching the most amount  
 # possible))  
 #--------------------------------------------------------------------------------  
 # | OR  
 #--------------------------------------------------------------------------------  
 # ^ the beginning of the string  
 #--------------------------------------------------------------------------------  
 #\s+ whitespace (\n, \r, \t, \f, and " ") (1 or  
 # more times (matching the most amount  
 # possible))  
 #--------------------------------------------------------------------------------  
 #$ before an optional \n, and the end of the  
 # string  
 df$Review[x] <- gsub("(?<=[\\s])\\s\*|^\\s+|\\s+$", "", df$Review[x], perl=TRUE)  
   
   
 }  
 return(df)  
   
}

#Finally applying the preprocessing function on the data   
metacritic\_data<-preprocessing(temp\_data)

#Converting date to date format  
metacritic\_data$reviewDate<- as.character(as.Date(metacritic\_data$reviewDate,format='%B %d, %Y'))  
  
#Extracting the year and adding a new column  
a = ymd(metacritic\_data$reviewDate)  
metacritic\_data$rating\_year = year(a)

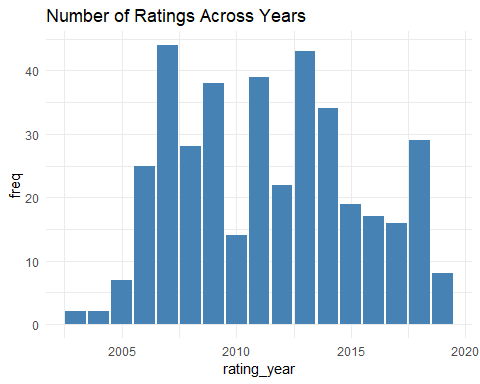
#Saving the processed data as a CSV  
write.csv(metacritic\_data, "metacritic\_data.csv")

# Visualizations

#Number of reviews across the years  
#To obtain this we group by year and calculate the frequency  
number\_of\_reviews\_across\_years <- metacritic\_data%>%  
 group\_by(rating\_year)%>%  
 summarise(freq=n())  
  
head(number\_of\_reviews\_across\_years)

## # A tibble: 6 x 2  
## rating\_year freq  
## <int> <int>  
## 1 2003 2  
## 2 2004 2  
## 3 2005 7  
## 4 2006 25  
## 5 2007 44  
## 6 2008 28

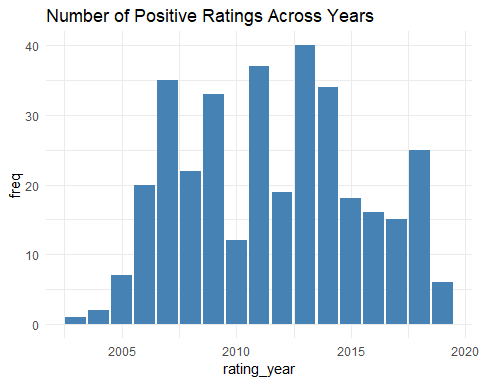
#Visualizing the above data as a histogram  
p<-ggplot(data=number\_of\_reviews\_across\_years, aes(x=rating\_year, y=freq)) +  
 geom\_bar(stat="identity", fill="steelblue")+  
 theme\_minimal()+labs(title = "Number of Ratings Across Years")  
p



#Considering a rating above 7.5 to be positve we want to see how the positive reviews are distributed over time  
positive\_rating<-metacritic\_data%>%filter(Rating>7.5)  
positive\_rating\_per\_year <- positive\_rating%>%   
 group\_by(rating\_year)%>%  
 summarise(freq=n())  
  
head(positive\_rating\_per\_year)

## # A tibble: 6 x 2  
## rating\_year freq  
## <int> <int>  
## 1 2003 1  
## 2 2004 2  
## 3 2005 7  
## 4 2006 20  
## 5 2007 35  
## 6 2008 22

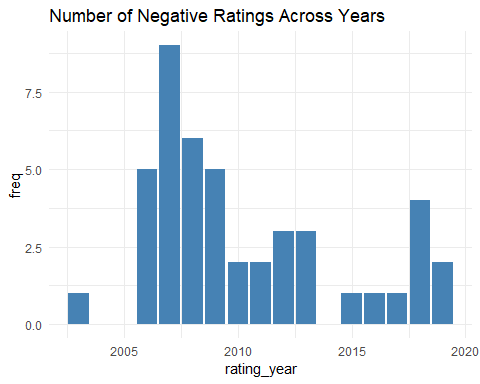
#Visualising over a bar plot  
ggplot(data=positive\_rating\_per\_year, aes(x=rating\_year, y=freq)) +  
 geom\_bar(stat="identity", fill="steelblue")+  
 theme\_minimal()+labs(title = "Number of Positive Ratings Across Years")



#Any rating below 7.5 is considered to be negative, after defining this metric we can see the distribution of the   
#negative reviews  
negative\_ratings<-metacritic\_data%>%filter(Rating<=7.5)  
negative\_ratings\_per\_year <- negative\_ratings%>%   
 group\_by(rating\_year)%>%  
 summarise(freq=n())  
  
head(negative\_ratings\_per\_year)

## # A tibble: 6 x 2  
## rating\_year freq  
## <int> <int>  
## 1 2003 1  
## 2 2006 5  
## 3 2007 9  
## 4 2008 6  
## 5 2009 5  
## 6 2010 2

#Visualising the above data  
ggplot(data=negative\_ratings\_per\_year, aes(x=rating\_year, y=freq)) +  
 geom\_bar(stat="identity", fill="steelblue")+  
 theme\_minimal()+labs(title = "Number of Negative Ratings Across Years")



# Sentiment Anlysis

#Unnesting all the reviews as a column Word  
meta<-metacritic\_data %>% unnest\_tokens(word, Review)  
  
#Removing all the stop words and counting the frequency  
meta\_non\_stop<-meta%>%  
 anti\_join(stop\_words) %>%  
 count(word, sort = TRUE)

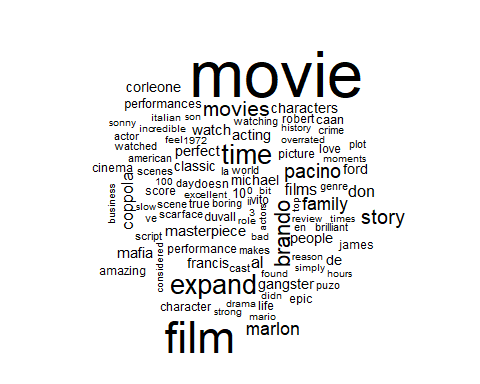
## Joining, by = "word"

#top ten words  
top\_10 <-head(meta\_non\_stop,10)  
top\_10

## # A tibble: 10 x 2  
## word n  
## <chr> <int>  
## 1 movie 597  
## 2 film 389  
## 3 godfather 361  
## 4 time 206  
## 5 expand 203  
## 6 brando 151  
## 7 movies 139  
## 8 pacino 132  
## 9 story 125  
## 10 family 107

#Creating the wordcloud from 100  
meta\_non\_stop %>%with(wordcloud(word, n, max.words = 100))

## Warning in wordcloud(word, n, max.words = 100): godfather could not be fit  
## on page. It will not be plotted.



#Creating a tible  
meta.tbl<-metacritic\_data  
  
#Adding the bigram token in unnest\_tokes  
meta\_bi <-meta.tbl %>%  
 unnest\_tokens(bigram, Review, token = "ngrams", n = 2)  
  
#Removing the stop words  
meta\_bi\_separated <- meta\_bi %>%  
 separate(bigram, c("word1", "word2"), sep = " ")  
meta\_bi\_filtered <- meta\_bi\_separated %>%  
 filter(!word1 %in% stop\_words$word) %>%  
 filter(!word2 %in% stop\_words$word)  
  
#finding the top 10 bigrams  
counted\_bigrams<-meta\_bi\_filtered %>%  
 count(word1, word2, sort = TRUE)  
top\_10\_bigrams<-head(counted\_bigrams,10)  
top\_10\_bigrams

## # A tibble: 10 x 3  
## word1 word2 n  
## <chr> <chr> <int>  
## 1 marlon brando 102  
## 2 al pacino 92  
## 3 francis ford 63  
## 4 ford coppola 62  
## 5 james caan 39  
## 6 robert duvall 35  
## 7 don vito 31  
## 8 brando al 24  
## 9 mario puzo 22  
## 10 vito corleone 22

#As expected the most common bigrams are names of people or actors in the movie

## Joining, by = "word"

## Warning: Column `word` joining factor and character vector, coercing into  
## character vector

## Joining, by = "word"

## Warning: Column `word` joining factor and character vector, coercing into  
## character vector

## # A tibble: 6 x 4  
## word negative positive sentiment  
## <chr> <dbl> <dbl> <dbl>  
## 1 bad 1 0 -1  
## 2 classic 0 1 1  
## 3 die 2 0 -2  
## 4 difficult 1 0 -1  
## 5 disappointed 1 0 -1  
## 6 exciting 0 2 2

## [1] "The final sentiment score of the entire review is: "

## [1] -4

#The final sentiment score of the entire review  
print("The final sentiment score of the entire review is: ")

## [1] "The final sentiment score of the entire review is: "

final\_sentiment\_score

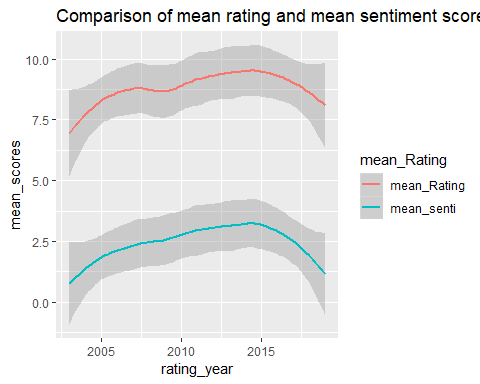
## [1] -4

#Calculating the sentiment score using the sentmentr package  
metacritic\_data$SentimentScore = (sentiment(metacritic\_data$Review)$sentiment)\*10  
  
#Calculating the mean sentiment score across all the years  
mean\_senti = aggregate(metacritic\_data$SentimentScore , by=list(rating\_year= metacritic\_data$rating\_year), FUN= mean)   
  
#Calculating the mean rating across all years  
mean\_rating = aggregate(metacritic\_data$Rating , by=list(rating\_year= metacritic\_data$rating\_year), FUN= mean)  
  
colnames(mean\_rating)[2] <- "mean\_Rating"  
colnames(mean\_senti)[2] <- "mean\_senti"  
  
#Data frame which contains the mean rating and mean sentiment score  
mean\_rating\_senti = merge(mean\_rating,mean\_senti)  
mean\_rating\_senti

## rating\_year mean\_Rating mean\_senti  
## 1 2003 5.000000 -0.6171131  
## 2 2004 10.000000 3.4471456  
## 3 2005 9.428571 1.9252662  
## 4 2006 8.480000 1.8034862  
## 5 2007 8.022727 2.5627122  
## 6 2008 8.285714 2.3968623  
## 7 2009 9.210526 1.9030215  
## 8 2010 8.714286 2.9392325  
## 9 2011 9.512821 3.5705942  
## 10 2012 9.000000 3.0141120  
## 11 2013 9.279070 2.7075065  
## 12 2014 9.676471 3.2401413  
## 13 2015 9.684211 2.3648600  
## 14 2016 9.176471 4.5736292  
## 15 2017 9.312500 2.4338207  
## 16 2018 8.586207 1.8477428  
## 17 2019 8.000000 0.9127173

mean\_rating\_senti\_long <- mean\_rating\_senti %>% gather(mean\_Rating, mean\_senti, -c(rating\_year))

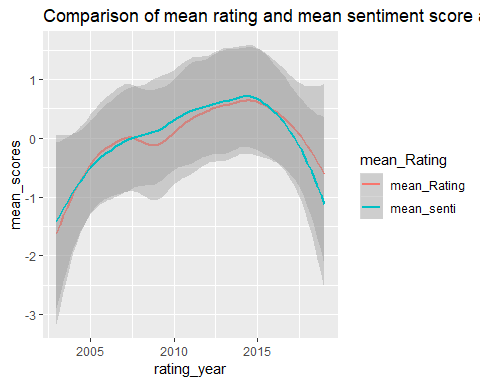
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



## Warning: attributes are not identical across measure variables;  
## they will be dropped

#Comparision between mean rating and mean sentiment score after normalizing  
g <- ggplot(data = mean\_rating\_senti\_long\_normalized, aes(x = rating\_year, y = mean\_senti, color = mean\_Rating))+geom\_smooth()+  
 labs(title = "Comparison of mean rating and mean sentiment score across years", y = "mean\_scores")  
g

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



#Final comparision between the mean rating and mean sentiment score:  
#The final sentiment score of the entire review  
print("The final sentiment score of the entire review calculated manually is: ")

## [1] "The final sentiment score of the entire review calculated manually is: "

final\_sentiment\_score

## [1] -4

print("The final sentiment score of the entire review calculated using sentimentr is: ")

## [1] "The final sentiment score of the entire review calculated using sentimentr is: "

metacritic\_data$SentimentScore[1]

## [1] -3.097187

print("The user rating is: ")

## [1] "The user rating is: "

metacritic\_data$Rating[1]

## [1] 3