Movie Recommendation Engine (group 10)

Qianyun Zhang, Yi Liu, Danmo Wang, Zehao Wang, Zhibo Wan

April,13,2016

Introduction

In this project, we attempt to use the user-based Collaborative Filtering approach to build a basic movie recommendation engine and analyze the information of the movies.There are 4 parts of the this presentation:

- The Dataset
- The user-based Collaborative Filtering approach
- Movie Recommendation Engine
- Shiny App: Movie Analysis

The Dataset

We firstly use the dataset “movie.csv”.In order to keep the recommender simple, we select the 5000 movies with most reviews based on ASIN and then compare the result with OMDB data, filtering out ASIN point to the same movies.

Data Processing

```
knitr::opts_chunk$set(echo = TRUE)

library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(hexbin)

##Parsing the helpfuness votes

movies.raw=read.csv("~/Desktop/project4/moviescsv.csv")

movies.raw=movies.raw%>%
  separate(review_helpfulness,
           c("helpful.v", "total.v"), sep = "/",
           remove=FALSE)
```

```
## Warning: Too few values at 12 locations: 1106516, 1137005, 1552153,
## 1832938, 2207608, 3171568, 3569707, 4228944, 5610576, 5805793, 6592106,
## 7462411
```

```
movies.raw=movies.raw%>%mutate(review_h=as.numeric(helpful.v)/as.numeric(total.v))
sample_n(movies.raw, 3)
```

```
##         product_productid  review_userid review_helpfulness helpful.v
## 2415603        B0002KVUKW A2S14TSRH6CMKP                1/3         1
## 1460575      6302403596.0 A2V29IO9JQQF5P                0/1         0
## 1350799        B00008AOVN A2WQJB9PQ8PCET                0/0         0
##         total.v review_score  review_h
## 2415603       3            3 0.3333333
## 1460575       1            3 0.0000000
## 1350799       0            4       NaN
```

```
##Compute some user summaries
user.table=movies.raw%>%
  #sample_n(100000)%>%
  group_by(review_userid)%>%
  summarize(
    user.count=n(),
    UReview_ave=mean(review_score, na.rm=T),
    UReview_read=mean(as.numeric(total.v), na.rm=T),
    UReview_help=mean(review_h, na.rm=T)
  )
head(user.table, n=3)
```

```
## Source: local data frame [3 x 5]
##
##        review_userid user.count UReview_ave UReview_read UReview_help
##               (fctr)      (int)       (dbl)        (dbl)        (dbl)
## 1                            12        4.25     2.833333    0.8785714
## 2 #oc-R1FQ7AEZE601ZD          1        1.00     8.000000    0.3750000
## 3 #oc-R2ZIMCXX9A2H0D          1        1.00    44.000000    0.2727273
```

```
##compute some movie summaries
product.table=movies.raw%>%
  #sample_n(100000)%>%
  group_by(product_productid)%>%
  summarize(
    prod.count=n(),
    PReview_read=sum(as.numeric(total.v)),
    PReview_ave=mean(review_score, na.rm=F)
  )

head(product.table, n=3)
```

```
## Source: local data frame [3 x 4]
##
##   product_productid prod.count PReview_read PReview_ave
##              (fctr)      (int)        (dbl)       (dbl)
## 1                           12           34        4.25
## 2        000500005X          3           43        5.00
## 3        000500411X          1            2        5.00
```

```
product.table_sort=product.table[with(product.table,order(-product.table$prod.count))
,]

product_5000=head(product.table_sort,n=5000)
```

OMDB

```
library(rvest)
library(tidyr)
library(devtools)
# Install omdbapi
devtools::install_github("hrbrmstr/omdbapi")

library(omdbapi)
library(pbapply)
library(dplyr)
library(stringr)
# Example 1, not found in OMDB
# ASIN.inq="000500005X"
# Example 2, found in OMDB
ASIN.list=product_5000$product_productid

#######below is data clean part#####################################################
#############
ASIN.str=toString(ASIN.list)
ASIN.str.left=ASIN.str

pivot=1
ASIN.GoodList=c()

while(toString(pivot)!="NA")
{
  pos=str_locate(ASIN.str.left,",")
  pivot=pos[1]
  ASIN.tmp=substr(ASIN.str.left,1,pivot-1)


    if(toString(as.numeric(ASIN.tmp))!="NA")
    {
      ASIN.tmp=substr(ASIN.tmp,1,str_length(ASIN.tmp)-2)
    }

  ASIN.GoodList=c(ASIN.GoodList,ASIN.tmp)
  ASIN.str.left=substring(ASIN.str.left,pivot+2)

}

ASIN.GoodList=ASIN.GoodList[1:499]

####below is feature


features_name=c("Rated","Type")
```

```
feature_table=NULL


for(i in 1:length(ASIN.GoodList))
{
movie1=NULL
movie1.title=NULL

ASIN.inq=ASIN.GoodList[i] # this movie's title has a "("

movie1<-tryCatch( {html(paste("http://www.amazon.com/exec/obidos/ASIN/", ASIN.inq, se
p=""))},error=function(e){})


if(is.null(movie1)){next}

movie1.title=
  movie1 %>%
  html_node("title") %>%
  html_text()

movie1.title=strsplit(movie1.title, ": ")[[1]][2]
movie1.title=strsplit(movie1.title, " \\[")[[1]][1]
movie1.title=strsplit(movie1.title, " \\(")[[1]][1]

movie1.title=substr(movie1.title,1,45)



tryCatch({omdb.entry=search_by_title(movie1.title)},error=function(e){})

if(length(omdb.entry)==0){next}


movie_feature=find_by_id(omdb.entry$imdbID[1], include_tomatoes=T)

tmp_row=c(ASIN.inq,movie1.title)
feature_list=names(movie_feature)

for(j in 1:length(features_name))
{
  index=match(features_name[j],feature_list)
  tmp_row=c(tmp_row,movie_feature[index])

}
feature_table=rbind(feature_table,tmp_row)
}
```

Algorithms we have tried

We have tryed the Movie Recommendation System of “Beer Dataset”, but unfortunately it doesn’t fit well to the movie dataset. It puts the information of user-based onto the item-based.To illustrate it, For movie 1,if all the ratings of users are 10,and for movie 2, all the ratings are 1,their correlation is 1,which is not true.

Slope One(item-based algorithm)

Item-based collaborative filtering of purchase statistics [edit]

We are not always given ratings: when the users provide only binary data (the item was purchased or not), then Slope One and other rating-based algorithm do not apply[citation needed]. Examples of binary item-based collaborative filtering include Amazon's item-to-item patented algorithm[12] which computes the cosine between binary vectors representing the purchases in a user-item matrix.

Being arguably simpler than even Slope One, the Item-to-Item algorithm offers an interesting point of reference. Let us consider an example.

Sample purchase statistics

Customer	Item 1	Item 2	Item 3
John	Bought it	Didn't buy it	Bought it
Mark	Didn't buy it	Bought it	Bought it
Lucy	Didn't buy it	Bought it	Didn't buy it

In this case, the cosine between items 1 and 2 is:

$$\frac{(1,0,0)\cdot(0,1,1)}{\|(1,0,0)\|\|(0,1,1)\|}=0,$$

The cosine between items 1 and 3 is:

$$\frac{(1,0,0)\cdot(1,1,0)}{\|(1,0,0)\|\|(1,1,0)\|}=\frac{1}{\sqrt{2}},$$

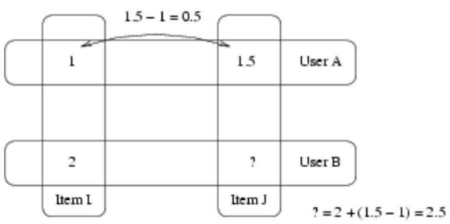
Whereas the cosine between items 2 and 3 is:

$$\frac{(0,1,1)\cdot(1,1,0)}{\|(0,1,1)\|\|(1,1,0)\|}=\frac{1}{2}.$$

Hence, a user visiting item 1 would receive item 3 as a recommendation, a user visiting item 2 would receive item 3 as a recommendation, and finally, a user visiting item 3 would receive item 1 (and then item 2) as a recommendation. The model uses a single parameter per pair of item (the cosine) to make the recommendation. Hence, if there are *n* items, up to *n(n-1)/2* cosines need to be computed and stored.

Slope one collaborative filtering for rated resources [edit]

To drastically reduce overfitting, improve performance and ease implementation, the **Slope One** family of easily implemented Item-based Rating-Based collaborative filtering algorithms was proposed. Essentially, instead of using linear regression from one item's ratings to another item's ratings ($f(x)=ax+b$), it uses a simpler form of regression with a single free parameter ($f(x)=x+b$). The free parameter is then simply the average difference between the two items' ratings. It was shown to be much more accurate than linear regression in some instances,[1] and it takes half the storage or less.



Example:

1. User A gave a 1 to Item I and an 1.5 to Item J.
2. User B gave a 2 to Item I.
3. How do you think User B rated Item J?
4. The Slope One answer is to say 2.5 (1.5-1+2=2.5).

For a more realistic example, consider the following table.

Sample rating database

Customer	Item A	Item B	Item C
John	5	3	2
Mark	3	4	Didn't rate it
Lucy	Didn't rate it	2	5

In this case, the average difference in ratings between item B and A is (2+(-1))/2=0.5. Hence, on average, item A is rated above item B by 0.5. Similarly, the average difference between item C and A is 3. Hence, if we attempt to predict the rating of Lucy for item A using her rating for item B, we get 2+0.5 = 2.5. Similarly, if we try to predict her rating for item A using her rating of item C, we get 5+3=8.

If a user rated several items, the predictions are simply combined using a weighted average where a good choice for the weight is the number of users having rated both items. In the above example, we would predict the following rating for Lucy on item A:

$$\frac{2\times 2.5+1\times 8}{2+1}=\frac{13}{3}=4.33$$

Hence, given *n* items, to implement Slope One, all that is needed is to compute and store the average differences and the number of common ratings for each of the n^2 pairs of items.

Since there are too many Users,and it’s really complicted to add all the missing-values of the ratings.

The user-based Collaborative Filtering approach

The User-Based Collaborative Filtering approach groups users according to their preferences, and then recommends an item that a similar user in the same group viewed or liked.
For example, if user 1 liked movie A, B and C, and if user 2 liked movie A and B, then movie C might make a good recommendation to user 2.
Hence in this post, We will use User-Based Collaborative Filtering based on "Cosine Similarity" Algorithm to generate a top-5 recommendation list for users Then given your UserID, we identify the most similar users and then return the movies that are similar to the movies already liked by the user.

Cosine Similarity

The cosine of two vectors can be derived by using the Euclidean dot product formula:

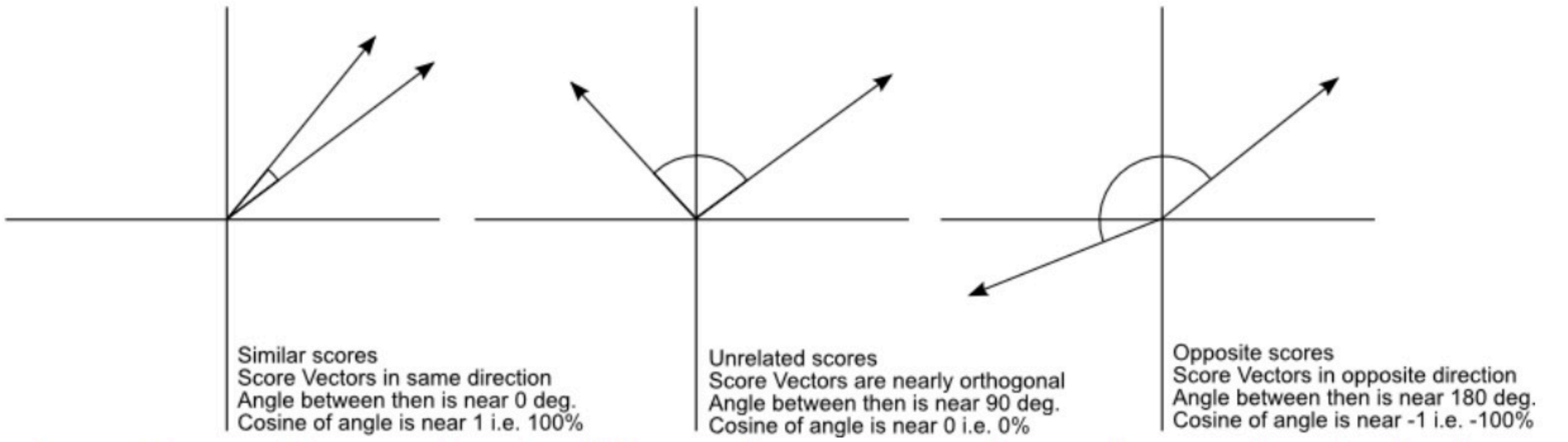
$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta$$

Given two vectors of attributes, *A* and *B*, the cosine similarity, *cos(θ)*, is represented using a dot product and magnitude as

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

, where A_i and B_i are components of vector A and B respectively.

In this case, A and B represents the vector of all the movie ratings from two Users.
For example



The Cosine Similarity values for different documents, 1 (same direction), 0 (90 deg.), -1 (opposite directions).

We reshape the data and construct a big matrix where the columns are the product IDs and the rows are the User IDs

```
library("plyr")
library("reshape2")
setwd("E:/W4249")
data<-read.csv("moviescsv.csv")
matrix<-acast(data[1:200,],review_userid~product_productid,value.var="review_score")
matrix[is.na(matrix)]<-0
similarity <- function(u1, u2){
```

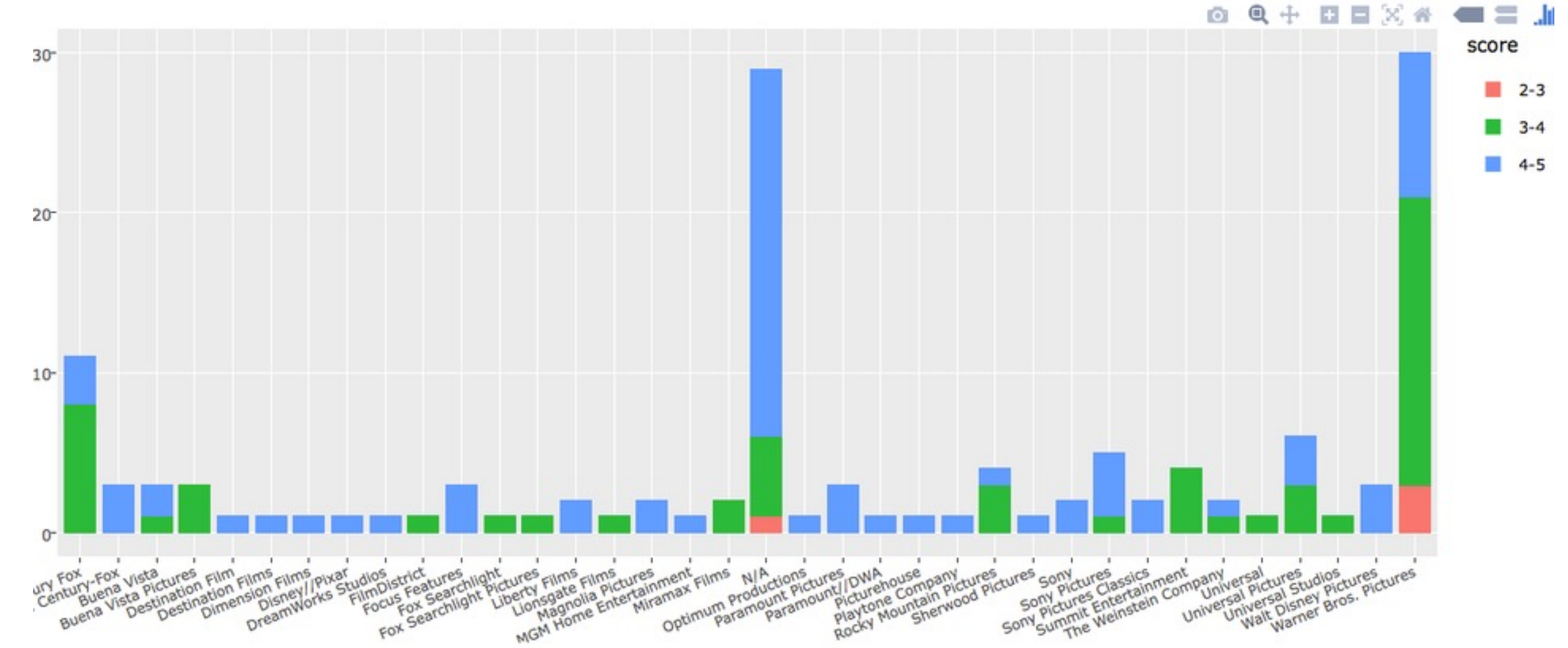
```
  c1<-matrix[which(rownames(matrix)==u1),]
  c2<-matrix[which(rownames(matrix)==u2),]
  corr<-crossprod(c1,c2)/(norm(as.matrix(c1),"f")*norm(as.matrix(c2),"f"))
  corr
}
user.pairs <- expand.grid(user1=rownames(matrix), user2=rownames(matrix))
user.pairs <- subset(user.pairs, user1!=user2)
results <- ddply(user.pairs, .(user1, user2), function(x) {
  #b1 <- beer_name_to_id(x$beer1)
  #b2 <- beer_name_to_id(x$beer2)
  c("sim"=similarity(x$user1, x$user2))
}, .progress="text")

recursivefunction<-function(myid,b,n=5,j=1){
  c1<-matrix[which(rownames(matrix)==myid),]
  c2<-matrix[which(rownames(matrix)==b[1]),]
  for(i in 1:length(c1)){
    if (c1[i]==0 & c2[i]>1){
      if(!(names(c1)[i] %in% movie)){
        movie[j]=names(c1)[i]
        j=j+1
      }
    }
  }
  if(j<n+1){
    if(length(b)==0){
      return(movie)
    }
    else{
      b<-b[-1]
      recursivefunction(myid,b,n=5,j)
    }
  }
  else{
    return(movie)
  }
}

find_similarity_movie<-function(myid,n=5){
  similar <- subset(results, user1==myid)
  similar <- similar[order(-similar$sim),]
  b<-similar[,2]
  movie<-rep(NA,n)
  recursivefunction(myid,b,n=5,j=1)
}
```

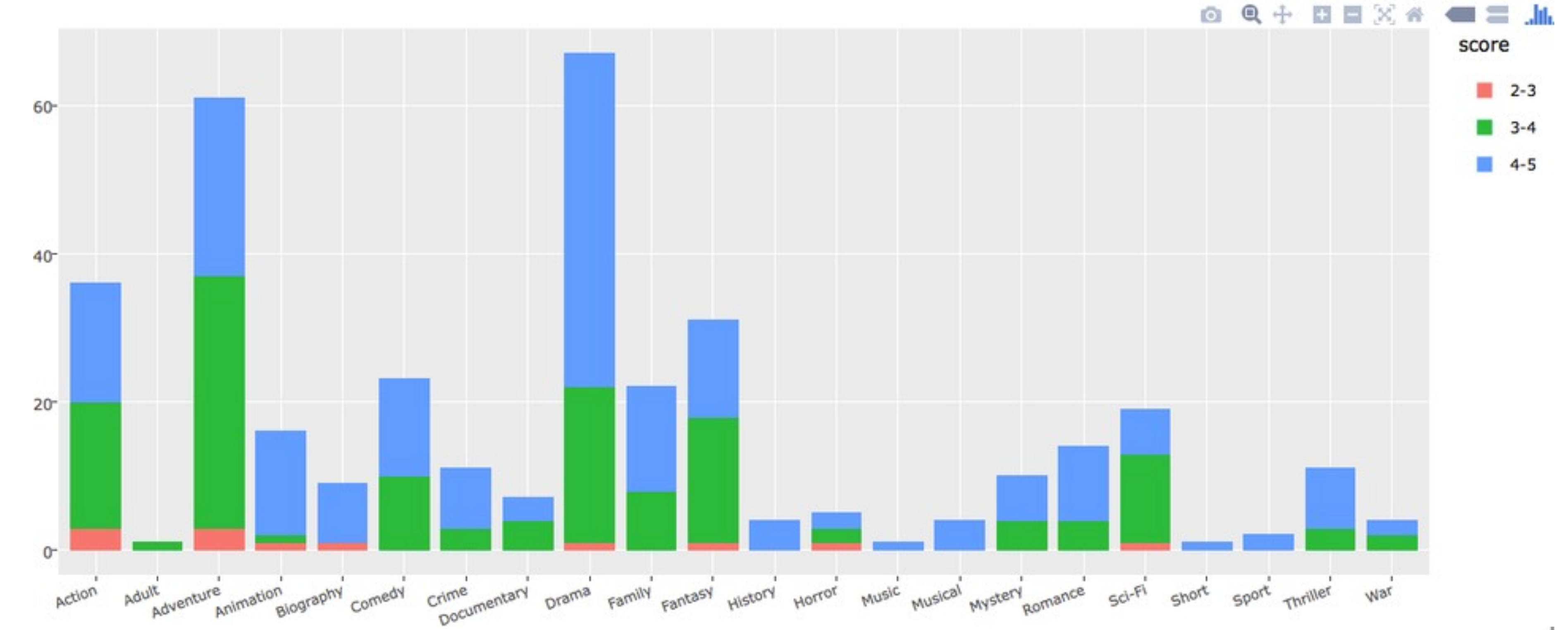
Movie Recommendation Engine

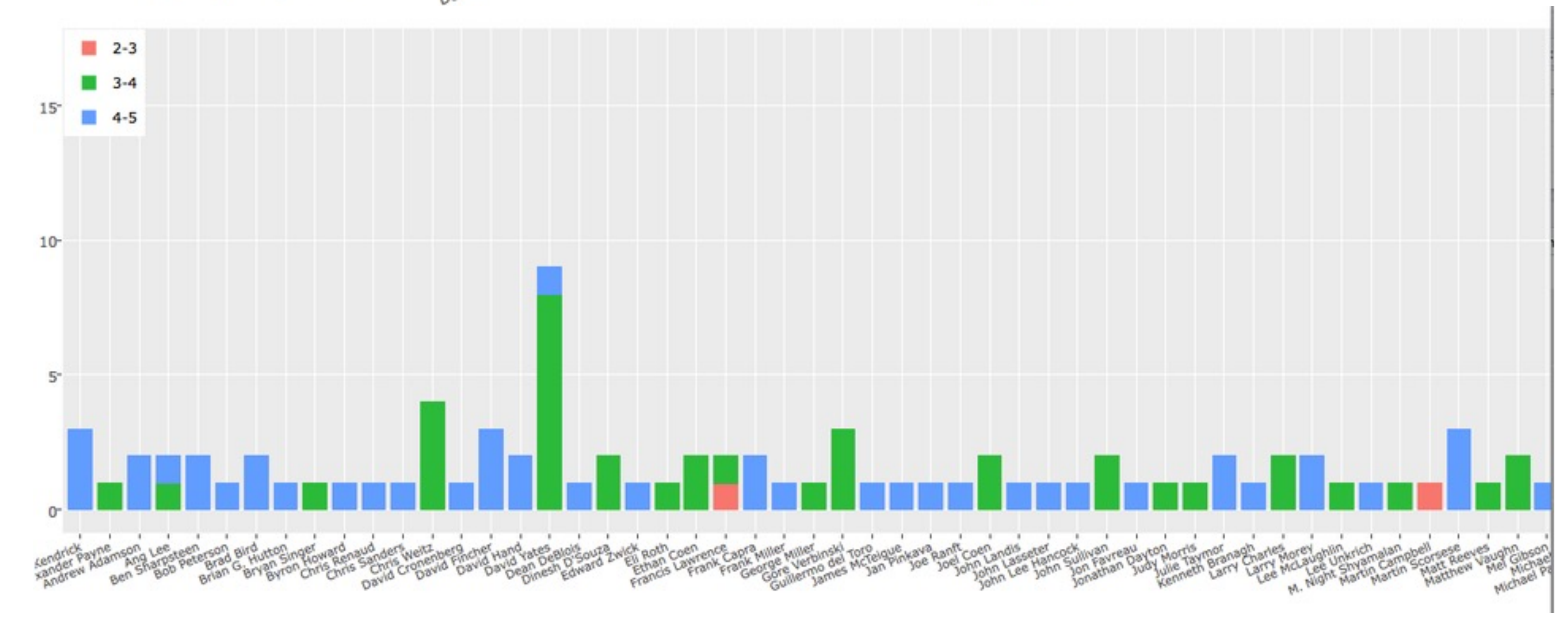
```
> find_similarity_movie("A103KNDW8GN92L",n=5)
[1] "B00005JMXX" "B0002NIAZW" "B003BZXHZG" "B000AXWHS0" "B00005JM0B"
> find_similarity_movie("A103QX7NUHBOUF")
[1] "B00005JMXX" "B0002NIAZW" "B003BZXHZG" "630472229X" "B00004RR8Z"
> find_similarity_movie("A103W7ZPKGOCC9")
[1] "B00005JMXX" "B0002NIAZW" "B003BZXHZG" "B00004XPQM" "B00005JLZK"
```

Movie Analysis

We analyzed the relationship between director names, Genre, Production industry and PReview_average scores, counts.


score
2-3
3-4
4-5
0
10
20
30
Century-Fox
Buena Vista
Buena Vista Pictures
Destination Film
Destination Films
Dimension Films
Disney//Pixar
DreamWorks Studios
FilmDistrict
Focus Features
Fox Searchlight
Fox Searchlight Pictures
Liberty Films
Lionsgate Films
Magnolia Pictures
MGM Home Entertainment
Miramax Films
N/A
Optimum Productions
Paramount Pictures
Paramount//DWA
Picturehouse
Playtone Company
Rocky Mountain Pictures
Sherwood Pictures
Sony
Sony Pictures
Sony Pictures Classics
Summit Entertainment
The Weinstein Company
Universal
Universal Pictures
Universal Studios
Walt Disney Pictures
Warner Bros. Pictures


score
2-3
3-4
4-5
0
20
40
60
Action
Adult
Adventure
Animation
Biography
Comedy
Crime
Documentary
Drama
Family
Fantasy
History
Horror
Music
Musical
Mystery
Romance
Sci-Fi
Short
Sport
Thriller
War


2-3
3-4
4-5
0
5
10
15
Andrew Adamson
Ang Lee
Ben Sharpsteen
Bob Peterson
Brad Bird
Brian G. Hutton
Bryan Singer
Byron Howard
Chris Renaud
Chris Sanders
Chris Weitz
David Cronenberg
David Fincher
David Hand
David Yates
Dean DeBlois
Dinesh D'Souza
Edward Zwick
Eli Roth
Ethan Coen
Francis Lawrence
Frank Capra
Frank Miller
George Miller
Gore Verbinski
Guillermo del Toro
James McTeigue
Jan Pinkava
Joe Ranft
Joel Coen
John Landis
John Lasseter
John Lee Hancock
John Sullivan
Jon Favreau
Jonathan Dayton
Judy Morris
Julie Taymor
Kenneth Branagh
Larry Charles
Larry Morey
Lee McLaughlin
Lee Unkrich
M. Night Shyamalan
Martin Campbell
Martin Scorsese
Matt Reeves
Matthew Vaughn
Mel Gibson

Then we select Geere, Year, PReview_average, Awards, Production Industry of the data and make a shiny app. We don't choose reviews because we have selected the 5000 movies with most reviews so it may not be distinct.

For example

Movie_Name	PReview_ave	Year	Awards	Production	Genre
close encounters of	4.466666667	1977	Won 1 Oscar. Another 13 wins & 3	Sony Pictures Releasing	Drama, Sci-Fi
emmet otter's jug-b	4.52	1977	Nominated for 4 Primetime Emmy	N/A	Comedy, Drama, Family
dark city	4.43003413	1998	9 wins & 14 nominations.	York	Drama, Fantasy, Sci-Fi
national lampoon's	4.507653061	1989	N/A	Warner Bros. Pictures	Comedy
forbidden planet	4.608017817	1956	Nominated for 1 Oscar. Another 1	MGM Home Entertainment	Action, Adventure, Family
braveheart	4.618110236	1995	Won 5 Oscars. Another 26 wins &	Paramount Pictures	Biography, Drama, History
ben-hur	4.812182741	1959	Won 11 Oscars. Another 16 wins &	MGM	Adventure, Drama, War
bram stoker's dracu	3.916167665	1992	Won 3 Oscars. Another 12 wins &	Columbia Pictures	Fantasy, Horror, Romance
monty python and t	4.659025788	1975	2 wins & 2 nominations.	Almi Cinema 5	Adventure, Comedy, Fantasy
grave of the fireflies	4.71810089	1988	3 wins.	Shinchosha Company	Animation, Drama, War
lonesome dove	4.611801242	1989	Won 2 Golden Globes. Another 16	N/A	Adventure, Drama, Western
snow white and the	4.564356436	1937	Nominated for 1 Oscar. Another 1	N/A	Animation, Family, Fantasy
the band wagon	4.680115274	1953	Nominated for 3 Oscars. Another	MGM Home Entertainment	Comedy, Musical, Romance
reservoir dogs	4.464174455	1992	9 wins & 15 nominations.	Miramax Films	Crime, Thriller
pinocchio	4.56097561	1940	Won 2 Oscars. Another 3 wins.	RKO	Animation, Family, Fantasy
terminator	4.305555556	1991	Won 4 Oscars. Another 20 wins &	TriStar Pictures	Action, Sci-Fi
it's a wonderful life	4.752669039	1946	Nominated for 5 Oscars. Another	Liberty Films	Drama, Family, Fantasy
baraka	4.660660661	1992	1 win & 1 nomination.	Magidson Films	Documentary
white christmas	4.75257732	1954	N/A	Paramount Pictures	Comedy, Musical, Romance
apollo 13	4.443298969	1995	Won 2 Oscars. Another 24 wins &	Universal Pictures	Adventure, Drama, History
seven	4.559006211	2008	5 wins & 6 nominations.	Sony Pictures	Drama, Romance
pulp fiction	4.606315789	1994	Won 1 Oscar. Another 54 wins & 6	Miramax Films	Crime, Drama
toy story	4.65625	1995	Nominated for 3 Oscars. Another	Buena Vista	Animation, Adventure, Comedy
alien 3	4.362745098	1992	N/A	N/A	Documentary
independence day	3.806153846	1996	Won 1 Oscar. Another 31 wins & 3	20th Century Fox	Action, Adventure, Sci-Fi
ten commandments	4.552693208	1956	Won 1 Oscar. Another 5 wins & 8	Paramount Pictures	Adventure, Biography, Drama
predator	4.198083067	1987	Nominated for 1 Oscar. Another 3	20th Century Fox	Action, Horror, Sci-Fi
being john malkovic	4.154676259	1999	Nominated for 3 Oscars. Another	Gramercy Pictures	Comedy, Drama, Fantasy
the princess and the	4.119241192	2009	Nominated for 3 Oscars. Another	Walt Disney Pictures	Animation, Family, Fantasy

Here are the codes of Shiny UI ande Server
Shiny UI

```
library(shiny)
library(plotly)
library(dplyr)
library(splitstackshape)
library(ggplot2)
library(shinydashboard)
# Define UI for application that draws a histogram
shinyUI(fluidPage(titlePanel("Movie Movie Movie"),

                            # Sidebar with a selector input for neighborhood
                            sidebarLayout(position="left",
                                          sidebarPanel(
                                            conditionalPanel(condition="input.Panels=
=1",
                                                             helpText("Give the overa
ll analysis"),
                                                             br(),
                                                             selectInput("Plot", "Ana
```

```
lysis",
                                                                                     c("Year", "A
wards"))
                                            ),
                                            conditionalPanel(condition="input.Panels=
=2",
                                                             helpText("Choose the gen
re"),
                                                             hr(),
                                                             selectInput("genre", "Ge
nre (a movie can have multiple genres)",
                                                                         c( "Action",
"Adventure", "Animation", "Biography", "Comedy",
                                                                            "Crime",
"Documentary", "Drama", "Family", "Fantasy", "History",
                                                                            "Horror",
"Music", "Musical", "Mystery", "Romance", "Sci-Fi",
                                                                            "Short",
"Sport", "Thriller", "War", "Western"))

                                            ),
                                            conditionalPanel(condition="input.Panels=
=3",
                                                             helpText("Choose the gen
re"),
                                                             hr()
                                            )
                                          ),
                                          mainPanel(
                                            tabsetPanel(type="pill",

                                                        tabPanel("Analysis", br(),plo
tlyOutput("distplot1") , value=1),

                                                        tabPanel("Genre Analysis", br
(), plotlyOutput("distplot2"), value=2),
                                                        tabPanel("engine", br(), help
Text("Choose the genre"), value=3),

                                                        id = "Panels"
                                            )
                                          )




                   )
))
```

Shiny Server

```r
library(shiny)

# Define server logic required to draw a histogram
shinyServer(function(input, output) {
  genre <- read.csv("genre.csv")
  test <- read.csv("test.csv")
  output$distplot1 <- renderPlotly(
    if (input$Plot >0){
      if (input$Plot =="Year"){
        year <- test %>%
          group_by(Year) %>%
          summarize(
            count = n())
        plot_ly(year, x = Year, y = count) %>% layout(title = "number of movies")
      }
      else if (input$Plot == "Awards"){
        plot_ly(data = test, x = Year, y = PReview_ave, color = awards, text=paste("T
itle:", test$Movie_Name), mode = "markers")
      }

    })

    output$distplot2 <- renderPlotly(
      if(input$genre >0){
        genre$Genre <- as.character(genre$Genre)
        drama <- genre[which(genre$Genre == as.character(input$genre)),]
        plot_ly(drama, x = Year, y =PReview_ave,color = awards,text=paste("Title:", d
rama$Movie_Name), mode = "markers" ,colors=c("#f03b20","#7fcdbb"))

      }

    )

})
```