Intent_HQ_Markdown_Euclidean_Big_Dataset_Genome

Top 10 most similar movies

The objective is to take a movie title as input and respond with the top-K most similar movies using the MovieLens https://grouplens.org/datasets/movielens/ dataset

For this exercise I'll be using the **Big MovieLens Latest** Dataset.

```
rm(list = ls())


library(dplyr)
library(knn.covertree)
library(tm)

#Reading the datasets
path = "C:/Users/pedro.e.sequera/Desktop/Intent HQ - Movie Similarity/ml-latest/"

links = read.csv(paste0(path,"links.csv"),header = T)
movies = read.csv(paste0(path,"movies.csv"),header = T)
ratings = read.csv(paste0(path,"ratings.csv"),header = T)
genome = read.csv(paste0(path,"genome-scores.csv"),header=T)
genome_tags = read.csv(paste0(path,"genome-tags.csv"),header=T) #14M rows
```

Below are some extracts of each of the CSV files. For this exercise we will use the tag genome data which encodes how strongly movies exhibit particular properties represented by tags (atmospheric, thought-provoking, realistic, etc.)

RATINGS

```
head(ratings)
```

```
##   userId movieId rating  timestamp
## 1      1     307    3.5 1256677221
## 2      1     481    3.5 1256677456
## 3      1    1091    1.5 1256677471
## 4      1    1257    4.5 1256677460
## 5      1    1449    4.5 1256677264
## 6      1    1590    2.5 1256677236
```

MOVIES

```
head(movies)
```

```
##   movieId                              title
## 1       1                   Toy Story (1995)
## 2       2                     Jumanji (1995)
## 3       3            Grumpier Old Men (1995)
## 4       4           Waiting to Exhale (1995)
## 5       5 Father of the Bride Part II (1995)
## 6       6                        Heat (1995)
##                                        genres
## 1 Adventure|Animation|Children|Comedy|Fantasy
## 2                  Adventure|Children|Fantasy
## 3                              Comedy|Romance
## 4                        Comedy|Drama|Romance
## 5                                      Comedy
## 6                       Action|Crime|Thriller
```

TAGS GENOME

```
head(genome)
```

```
##   movieId tagId relevance
## 1       1     1   0.02900
## 2       1     2   0.02375
## 3       1     3   0.05425
## 4       1     4   0.06875
## 5       1     5   0.16000
## 6       1     6   0.19525
```

LINKS

```
head(links)
```

```
##   movieId imdbId tmdbId
## 1       1 114709    862
## 2       2 113497   8844
## 3       3 113228  15602
## 4       4 114885  31357
## 5       5 113041  11862
## 6       6 113277    949
```

Methodology

The methodology used here to identify the 10 most similar movies ot each movie is the **k-Nearest Neighbor (K-NN)**. For that, we will take advantage of the **find_knn** function from the *knn.covertree* package in **R**

Recommenders identify items that are similar to each other or online users with similar preferences. In order to do this, the recommender needs a framework with which to first compare users or items, and then identify those that are most similar to each other. This is where the k-Nearest Neighbor (k-NN) algorithm comes to use.

k-NN is a machine learning algorithm that inputs items or users as data points in a feature space. It then seeks to solve the following query: given (N) objects in the feature space, find the (k) most similar objects or neighbors. The k-NN algorithm then identifies the (k) most similar objects by locating the data points that are closest in proximity to the (N) objects.

To locate data points to the items we are testing, k-NN must employ some type of metric to measure the distance between the data points in the feature space. This **distance metric** is the key method a recommender uses to identify items or users that are similar to each other.

Final results are based on the **Euclidean** distance metric

Preprocessing

The input for the **find_nn** algorithm is a data frame where each row represents a unique *movieId* and the columns are the **ratings**, **genre** and **tag** attributes. Each dataset will be processed independently and then joined by *movieID*

RATINGS Dataset

For each *movieID* we will estimate the average rating by user and the number of users that rated that movie. We will then normalize each vector on a min-max scale

```
#Writing a normalizing function
 ##the normalization function is created
 nor <-function(x) { (x -min(x))/(max(x)-min(x))   }

ratings.matrix = ratings %>%
  group_by(movieId) %>%
  summarise(rating_avg = mean(rating),rating_number = length(rating)) %>%
  mutate(rating_avg_norm = nor(rating_avg),rating_number_norm = nor(rating_number)) %>%
  select(one_of(c("movieId","rating_avg_norm","rating_number_norm")))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
head(ratings.matrix)
```

```
## # A tibble: 6 x 3
##   movieId rating_avg_norm rating_number_norm
##     <int>           <dbl>              <dbl>
## 1       1           0.753             0.699
## 2       2           0.610             0.277
## 3       3           0.594             0.159
## 4       4           0.528             0.0305
## 5       5           0.573             0.158
## 6       6           0.743             0.293
```

GENRE

For genres we will create a Document Term Matrix, where each row is a *movieID* and each column is an indicator column for each genre, where 1 indicates that the movie corresponds to a specific genre and 0 if it doesn't

```
genre.list = strsplit(as.character(movies$genres),split="\\|")


docs <- as.VCorpus(genre.list)

docs <- tm_map(docs, PlainTextDocument)

dtm = DocumentTermMatrix(docs)

dtm.matrix = as.matrix(dtm) #9742 22

#removing redudant variables and renaming
genre.matrix = cbind.data.frame(data.frame(movies$movieId),dtm.matrix[,-1])
names(genre.matrix)[1] = "movieID"
names(genre.matrix)[15] = "Not Listed"
rownames(genre.matrix) = NULL


head(genre.matrix)
```

```
##   movieID action adventure animation children comedy crime documentary drama
## 1       1      0         1         1        1      1     0           0     0
## 2       2      0         1         0        1      0     0           0     0
## 3       3      0         0         0        0      1     0           0     0
## 4       4      0         0         0        0      1     0           0     1
## 5       5      0         0         0        0      1     0           0     0
## 6       6      1         0         0        0      0     1           0     0
##   fantasy film-noir genres horror imax Not Listed musical mystery romance
## 1       1         0      0      0    0          0       0       0       0
## 2       1         0      0      0    0          0       0       0       0
## 3       0         0      0      0    0          0       0       0       1
## 4       0         0      0      0    0          0       0       0       1
## 5       0         0      0      0    0          0       0       0       0
## 6       0         0      0      0    0          0       0       0       0
##   sci-fi thriller war western
## 1      0        0   0       0
## 2      0        0   0       0
## 3      0        0   0       0
## 4      0        0   0       0
## 5      0        0   0       0
## 6      0        1   0       0
```

Tag Genome

Reshaping the Genome Score table into wide format

```
#Reshaping into wide format
tags.matrix = reshape(genome,idvar = "movieId",timevar="tagId",direction="wide" )
names(tags.matrix) = c("movieID",paste(genome_tags$tag,"_Relevance",sep=""))




rownames(tags.matrix) = NULL


head(tags.matrix)[,30:40]
```

```
##   adventure_Relevance affectionate_Relevance afi 100_Relevance
## 1              0.9070                0.60850           0.03700
## 2              0.9780                0.12375           0.01100
## 3              0.3205                0.09725           0.01275
## 4              0.1465                0.10475           0.01375
## 5              0.1525                0.10200           0.01200
## 6              0.0885                0.22025           0.02225
##   afi 100 (laughs)_Relevance afi 100 (movie quotes)_Relevance africa_Relevance
## 1                    0.23675                          0.30075          0.06200
## 2                    0.07875                          0.06850          0.07525
## 3                    0.03400                          0.05950          0.05925
## 4                    0.03500                          0.04875          0.03325
## 5                    0.03350                          0.07025          0.02575
## 6                    0.02900                          0.15025          0.01000
##   afterlife_Relevance aging_Relevance aids_Relevance airplane_Relevance
## 1             0.03450         0.27325        0.11775            0.00750
## 2             0.05600         0.23675        0.06175            0.01875
## 3             0.03050         0.15925        0.05575            0.02975
## 4             0.02300         0.07700        0.11075            0.00900
## 5             0.03525         0.05500        0.04100            0.02525
## 6             0.03325         0.06250        0.09850            0.01325
##   airport_Relevance
## 1           0.02200
## 2           0.02725
## 3           0.04100
## 4           0.01300
## 5           0.05175
## 6           0.10350
```

Joining all Datasets into a single Data Frame

Genre has **58,098** movieIDs
Ratings has **53,889** movieIDs
Tags has **13,176** movieIDs

```
final.df.stage = left_join(genre.matrix,ratings.matrix,by=c("movieID" = "movieId"))

final.df = left_join(final.df.stage,tags.matrix,by="movieID") #9742 by 1766

#Replacing NAs with 0's
final.df[is.na(final.df)] = 0

head(final.df)[,1:30]
```

```
##   movieID action adventure animation children comedy crime documentary drama
## 1       1      0         1         1        1      1     0           0     0
## 2       2      0         1         0        1      0     0           0     0
## 3       3      0         0         0        0      1     0           0     0
## 4       4      0         0         0        0      1     0           0     1
## 5       5      0         0         0        0      1     0           0     0
## 6       6      1         0         0        0      0     1           0     0
##   fantasy film-noir genres horror imax Not Listed musical mystery romance
## 1       1         0      0      0    0          0       0       0       0
## 2       1         0      0      0    0          0       0       0       0
## 3       0         0      0      0    0          0       0       0       1
## 4       0         0      0      0    0          0       0       0       1
## 5       0         0      0      0    0          0       0       0       0
## 6       0         0      0      0    0          0       0       0       0
##   sci-fi thriller war western rating_avg_norm rating_number_norm 007_Relevance
## 1      0        0   0       0       0.7525888         0.69866732       0.02900
## 2      0        0   0       0       0.6103518         0.27696484       0.03625
## 3      0        0   0       0       0.5942181         0.15902365       0.04150
## 4      0        0   0       0       0.5276756         0.03049042       0.03350
## 5      0        0   0       0       0.5727313         0.15789098       0.04050
## 6      0        1   0       0       0.7431580         0.29267944       0.02925
##   007 (series)_Relevance 18th century_Relevance 1920s_Relevance 1930s_Relevance
## 1                0.02375                0.05425         0.06875         0.16000
## 2                0.03625                0.08275         0.08175         0.10200
## 3                0.04950                0.03000         0.09525         0.04525
## 4                0.03675                0.04275         0.02625         0.05250
## 5                0.05175                0.03600         0.04625         0.05500
## 6                0.02575                0.02700         0.03450         0.06825
##   1950s_Relevance
## 1         0.19525
## 2         0.06900
## 3         0.05925
## 4         0.03025
## 5         0.08000
## 6         0.04675
```

Creating the Top 10 most similar movies

The function estimates the pairwise **Cosine** distance between all movieIDs and then sorts from lower to higher distances. Additional processing is needed to translate the matric from indices to titles

```
p = find_knn(final.df[,-1],10)


# #Creating table to translate index into title
temp = left_join(final.df,movies,by = c("movieID" = "movieId")) %>%
   select(one_of(c("movieID","title")))

#The function output yields a matrix with the indices
top10_nn =p$index

#Retrieving the distances
top10_nn_dist = p$dist

#translating movieIDs into Titles
top10_nn_title = top10_nn


title <- function(x) {
  return(temp[x,2])
}

top10_nn_title = as.data.frame(apply(top10_nn, 2, title))



#Final datafame
top10_nn_final = cbind.data.frame(data.frame(temp[,2]),top10_nn_title)
names(top10_nn_final) = c("Movie Title",paste0("top",(1:10)))

top10_nn_final_dist = cbind.data.frame(data.frame(temp[,2]),top10_nn_dist)
names(top10_nn_final_dist) = names(top10_nn_final)

t(head(top10_nn_final))
```

```
##             1
## Movie Title "Toy Story (1995)"
## top1        "Monsters, Inc. (2001)"
## top2        "Toy Story 2 (1999)"
## top3        "Bug's Life, A (1998)"
## top4        "Toy Story 3 (2010)"
## top5        "Finding Nemo (2003)"
## top6        "Ice Age (2002)"
## top7        "Ratatouille (2007)"
## top8        "Incredibles, The (2004)"
## top9        "Shrek (2001)"
## top10       "Antz (1998)"
##             2
## Movie Title "Jumanji (1995)"
## top1        "Honey, I Shrunk the Kids (1989)"
## top2        "Borrowers, The (1997)"
## top3        "Escape to Witch Mountain (1975)"
## top4        "Mighty Joe Young (1998)"
## top5        "Zathura (2005)"
## top6        "Gnome-Mobile, The (1967)"
## top7        "Water Horse: Legend of the Deep, The (2007)"
## top8        "Dinotopia (2002)"
## top9        "Spiderwick Chronicles, The (2008)"
## top10       "Jumanji: Welcome to the Jungle (2017)"
##             3
## Movie Title "Grumpier Old Men (1995)"
## top1        "Three Men and a Little Lady (1990)"
## top2        "Arthur 2: On the Rocks (1988)"
## top3        "My Girl 2 (1994)"
## top4        "Meet the Fockers (2004)"
## top5        "Look Who's Talking Too (1990)"
## top6        "Look Who's Talking (1989)"
## top7        "Evening Star, The (1996)"
## top8        "Grumpy Old Men (1993)"
## top9        "Barbershop 2: Back in Business (2004)"
## top10       "Smile Like Yours, A (1997)"
##             4
## Movie Title "Waiting to Exhale (1995)"
## top1        "How Stella Got Her Groove Back (1998)"
## top2        "Moonlight and Valentino (1995)"
## top3        "Tyler Perry's Diary of a Mad Black Woman (2005)"
## top4        "How to Make an American Quilt (1995)"
## top5        "Mr. Wonderful (1993)"
## top6        "Mirror Has Two Faces, The (1996)"
## top7        "Heartburn (1986)"
## top8        "First Wives Club, The (1996)"
## top9        "Catch and Release (2006)"
## top10       "Something Borrowed (2011)"
##             5
## Movie Title "Father of the Bride Part II (1995)"
## top1        "Father of the Bride (1991)"
## top2        "Grumpier Old Men (1995)"
## top3        "My Big Fat Greek Wedding 2 (2016)"
## top4        "Three Men and a Little Lady (1990)"
## top5        "Meet the Fockers (2004)"
## top6        "Son of Flubber (1963)"
## top7        "Made in America (1993)"
## top8        "Delivery Man (2013)"
## top9        "Game Plan, The (2007)"
## top10       "For Richer or Poorer (1997)"
##             6
## Movie Title "Heat (1995)"
## top1        "Town, The (2010)"
## top2        "Collateral (2004)"
## top3        "Getaway, The (1972)"
## top4        "End of Watch (2012)"
## top5        "Way of the Gun, The (2000)"
## top6        "Bullitt (1968)"
## top7        "French Connection, The (1971)"
## top8        "Thief (1981)"
## top9        "Carlito's Way (1993)"
## top10       "Training Day (2001)"
```

Writing the final function and Displaying the Output

```
top10_nn_function = function(title){
  top10_vector = top10_nn_final[which(top10_nn_final$`Movie Title` %in% title),2:11]
  top10_dist = data.frame(top10_nn_dist[which(top10_nn_final$`Movie Title` %in% title),])

  top10.df = cbind.data.frame(c(t(top10_vector)),c(top10_dist))
  names(top10.df) = c("Top 10 Movie Titles","Euclidean Distance")
  rownames(top10.df) = paste0("top",(1:10))
  return(top10.df)

}
```

Displaying the output for "Sense and Sensibility (1995)"

```
top10_nn_function("Sense and Sensibility (1995)")
```

```
##             Top 10 Movie Titles Euclidean Distance
## top1    Pride & Prejudice (2005)           2.070794
## top2  Pride and Prejudice (1995)           2.308703
## top3           Persuasion (1995)           2.431412
## top4                 Emma (1996)           2.704907
## top5  Room with a View, A (1986)           2.724740
## top6           Persuasion (2007)           2.782876
## top7           Jane Eyre (2006)            2.902337
## top8        North & South (2004)           2.962152
## top9                 Emma (2009)           2.991557
## top10           Jane Eyre (2011)           3.040281
```