DATA 607 - Final Project

Chitrarth Kaushik

5/6/2020

DATA

Data was extracted from https://grouplens.org/datasets/movielens/25m/. The datasets contains 25 million ratings for 62,000 movies by 162,000 users. Given the huge nature of data and the resources implications it would have had, the ratings data was brought to a more manageable size by applying following filters:

1. Movies released after 2015 were considered
2. Movies which have at least 20 ratings were considered
3. Users who have rated at least 100 movies were considered

Subsequent to applying above filters we were left with 784 unique users and 1640 movies. This gave us a dataset that was manageable given the resource constraints.

The ultimate aim of this exercise is to build recommender engine using IBCF and UBCF methods and thereafter perform a comparison in the accuracy of the two recommenders built using two different methods.

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------
- tidyverse 1.3.0 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

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

```
library(recommenderlab)
```

```
## Warning: package 'recommenderlab' was built under R version 3.6.3
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

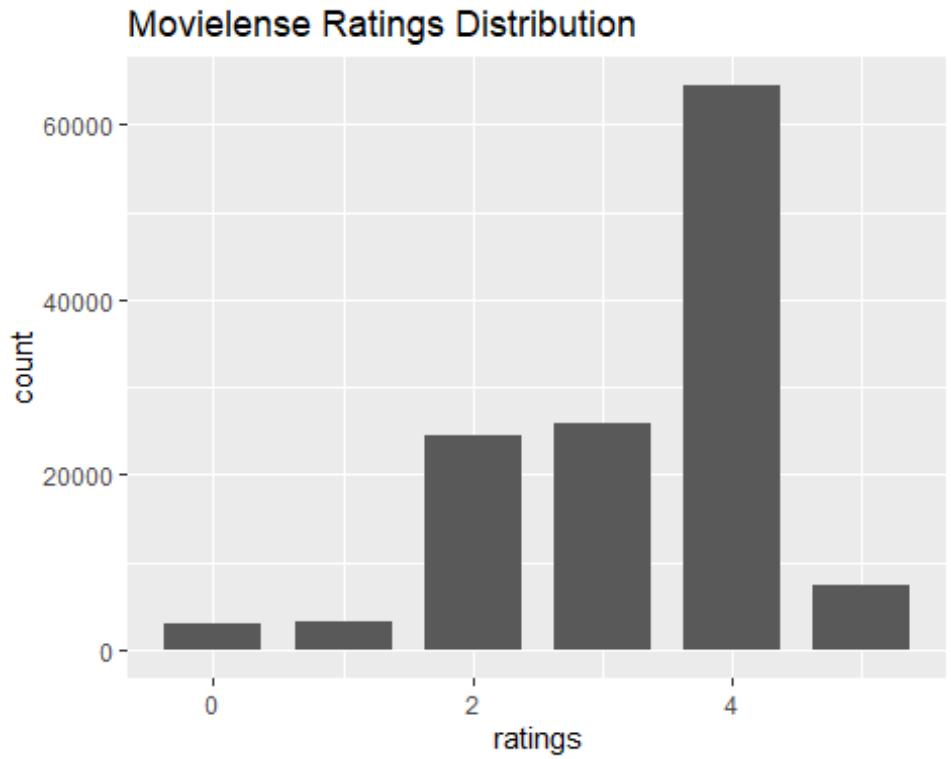
```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loading required package: arules

## Warning: package 'arules' was built under R version 3.6.3

##
## Attaching package: 'arules'

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

## The following objects are masked from 'package:base':
##
##     abbreviate, write

## Loading required package: proxy

## Warning: package 'proxy' was built under R version 3.6.3

##
## Attaching package: 'proxy'

## The following object is masked from 'package:Matrix':
##
##     as.matrix

## The following objects are masked from 'package:stats':
##
##     as.dist, dist

## The following object is masked from 'package:base':
##
##     as.matrix

## Loading required package: registry

## Registered S3 methods overwritten by 'registry':
##   method               from
##   print.registry_field proxy
##   print.registry_entry proxy

url <- http://files.grouplens.org/datasets/movielens/ml-25m.zip

td <- tempdir()

print(td)

tf <- tempfile(tmpdir=td, fileext=".zip")

download.file(url, tf)
```

```r
# to see files:

#unzip(tf, list = TRUE)

## read the data

movie_dat <- read_csv(unz(tf, "ml-25m/movies.csv"))

## Parsed with column specification:
## cols(
##   movieId = col_double(),
##   title = col_character(),
##   genres = col_character()
## )

ratings_dat <- read_csv(unz(tf, "ml-25m/ratings.csv"))

## Parsed with column specification:
## cols(
##   userId = col_double(),
##   movieId = col_double(),
##   rating = col_double(),
##   timestamp = col_double()
## )

#filtering movies which have been rated by at least 20 users
sum_movie<-ratings_dat  %>% group_by(movieId) %>%
tally()%>%arrange(desc(n))%>%subset(n>=20, -n)

#merging filtered movie list with original dataset
movies_data_sub <- right_join(movie_dat, sum_movie, by = "movieId")


#filtering movies released in last decade
movie_titles <- movies_data_sub %>%
  extract(title, c("title", "year"), regex = "(.*)\\s\\((\\d+)\\)", convert =
TRUE)%>%subset(year>2015, movieId)

#merging the filtered list with the original file
ratings_dat_sub <- right_join(ratings_dat, movie_titles, by =
"movieId")%>%arrange(userId)%>%select(userId, movieId, rating)

#selecting the users who have given at least 100 ratings
sub_user<-
ratings_dat_sub%>%group_by(userId)%>%tally()%>%arrange(desc(n))%>%subset(n>=1
00)

#merging these user id with the origina dataset
ratings_dat_sub1<-right_join(ratings_dat_sub, sub_user, by =
"userId")%>%select(-n)
```

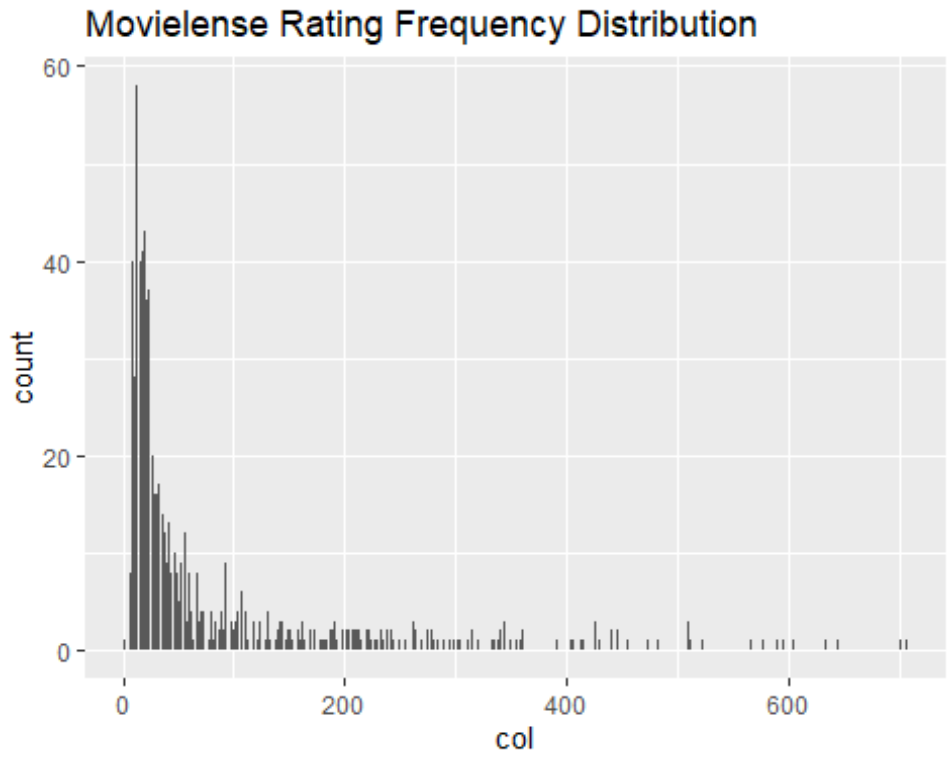
```r
#merging title in place of movieID
ratings_dat_tit<-left_join(ratings_dat_sub1,movie_dat,
by="movieId")%>%select(-movieId, -genres)

#converting from long to wide format
ratings_mat<-spread(ratings_dat_tit, title,
rating)%>%mutate(userId=c(1:784))%>%select(-userId)

ratings_mat1<-spread(ratings_dat_tit, title,
rating)%>%mutate(userId=c(1:784))

#coercing into a real rating matrix so that methods of recommenderlab package
can be applied
rating_real<-as(as.matrix(ratings_mat),"realRatingMatrix")
rating_real1<-as(as.matrix(ratings_mat1),"realRatingMatrix")

#exploring the ratings data by user and by movie

dim(rating_real)
```

```
## [1]  784 1640
```

```r
data.frame(ratings = round(getRatings(rating_real), digits=0))%>%
  ggplot(aes(ratings)) + geom_bar(width = 0.75) +
    labs(title = 'Movielense Ratings Distribution')
```



```
data.frame(col<-colCounts(rating_real))%>%
  ggplot(aes(col)) + geom_bar(width = 0.75) + labs(title= 'Movielense Rating
Frequency Distribution')
```



##Building recommender system using IBCF and UBCF methods

```
library(recommenderlab)
#preparing IBCF based recommender
norm_rating_real<-normalize(rating_real, method="center", row=TRUE)

n_rating_eval <- evaluationScheme(norm_rating_real, method="split",
train=0.8, given=5, goodRating=3)


IBCF_rating <- Recommender(getData(n_rating_eval, "train"), "IBCF",
                           param=list(normalize = NULL ,method="cosine"))

names(getModel(IBCF_rating))

## [1] "description"          "sim"                  "k"
## [4] "method"               "normalize"            "normalize_sim_matrix"
## [7] "alpha"                "na_as_zero"           "verbose"

getModel(IBCF_rating)$k

## [1] 30
```

```r
#Building UBCF based recommender

set.seed(123)
n_rating_eval_UB <- rating_real %>%
  evaluationScheme(method = 'split', # single train/test split
                   train = 0.8, # proportion of rows to train.
                   given = 5, # shouldn't keep n rec. items >
min(rowCounts(movie_r))
                   goodRating = 3) # for binary classifier analysis.

#Building a Recommender System with R by Gorakala and Usuelli. Ch.4 pp 84
model_params <- list(method = "cosine",
                     nn = 10, # find each user's 10 most similar users.
                     sample = FALSE, # already did this.
                     normalize = "center")

model1 <- getData(n_rating_eval_UB, "train") %>% #only fit on the 80%
training data.
  Recommender(method = "UBCF", parameter = model_params)

library(recommenderlab)
#accuracy comparison between user based collaborative filtering and item
based collaborative filtering
#predict items for unknown data
pred <- predict(IBCF_rating, getData(n_rating_eval, "known"), type="ratings",
n=5)

test_error_IB<- calcPredictionAccuracy(pred, getData(n_rating_eval,
"unknown"), byUser=FALSE, goodRating=3, given=5)

test_error_IB

##      RMSE       MSE       MAE
## 1.0767210 1.1593280 0.7909015

model1_pred <- predict(model1, getData(n_rating_eval_UB, "known"), type =
"ratings")

test_error_UB <- calcPredictionAccuracy(model1_pred,
getData(n_rating_eval_UB, "unknown"), byUser = FALSE)

test_error_UB

##      RMSE       MSE       MAE
## 0.8910806 0.7940247 0.6808167
```

##Conclusion

As can be seen from the prediciton accuracy results that UBCF recommender is significantly more accurate that the IBCF based recommender given the data that has been

used. We will have to test and compare the accuracy of the two recommenders using other datasets.