Hands-On Exercise: Implementing a Basic Recommender

In this Hands-On Exercise, you will build a simple recommender system in R using the techniques you have just learned. There are 7 sections. Please ensure you complete each section before you move the next one.

Used Packages

We will build recommender systems using "`recommenderlab`", which is an R package for collaborative filtering.

```
# install.packages("recommenderlab")
library(recommenderlab)
library(ggplot2)
```

Load Data

Like many other R packages, `recommenderlab` contains some datasets that can be used to play around with the functions:

Jester5k, MSWeb, and MovieLense

In this lab, we will use the MovieLense dataset; the data is about movies. The table contains the ratings that the users give to movies. Let's load the data and take a look at it:

```
set.seed(1)
data_package <- data(package = "recommenderlab")
data_package$results[, "Item"]
## [1] "Jester5k"   "MSWeb"      "MovieLense"
data(MovieLense)
class(MovieLense)
## [1] "realRatingMatrix"
## attr(,"package")
## [1] "recommenderlab"
```

Lab 1. Computing the Similarity Matrix

a) Determine how similar the first four USERS are with each other

```
similarity_users <- similarity(MovieLense[1:4, ],
                               method = "cosine",
                               which = "users")
```

b) Convert similarity_users class into a matrix and visualize it

```
as.matrix(similarity_users)
##            1          2          3          4
## 1 0.00000000 0.16893670 0.03827203 0.06634975
## 2 0.16893670 0.00000000 0.09706862 0.15310468
## 3 0.03827203 0.09706862 0.00000000 0.33343036
## 4 0.06634975 0.15310468 0.33343036 0.00000000
image(as.matrix(similarity_users), main = "User similarity")
```

c) Examine the image and ensure you understand what it illustrates

d) Compute and visualize the similarity between the first four ITEMS

```
similarity_items <- similarity(MovieLense[, 1:4], method =
                                "cosine", which = "items")
as.matrix(similarity_items)
##          Toy Story(1995) GoldenEye(1995) Four Rooms(1995) Get Shorty(1995)
## Toy Story(1995)   0.0000000     0.4023822       0.3302448        0.4549379
## GoldenEye(1995)   0.4023822     0.0000000       0.2730692        0.5025708
## Four Rooms(1995)  0.3302448     0.2730692       0.0000000        0.3248664
## Get Shorty(1995)  0.4549379     0.5025708       0.3248664        0.0000000

image(as.matrix(similarity_items), main = "Item similarity")
```

e) Examine the image and ensure you understand what it illustrates.

Lab 2. Recommendation Models.

a) Display the model applicable to the objects of type `realRatingMatrix` using `recommenderRegistry$get_entries`:

```
recommender_models <- recommenderRegistry$get_entries(dataType = "realRatingM
atrix")

names(recommender_models)
```

b) How many models are listed?

c) Describe these models

```
lapply(recommender_models, "[[", "description")
```

d) We plan to use IBCF and UBCF. Check the parameters of these two models.

```
recommender_models$IBCF_realRatingMatrix$parameters
```

e) List the parameters of these two model. What do they mean?

Lab 3. Data Exploration

a) Initial exploration of data types and dimensions

```
dim(MovieLense)
## [1]  943 1664
slotNames(MovieLense)
## [1] "data"      "normalize"
class(MovieLense@data)
## [1] "dgCMatrix"
## attr(,"package")
## [1] "Matrix"
dim(MovieLense@data)
## [1]  943 1664
```

b) Exploring values of ratings

```
vector_ratings <- as.vector(MovieLense@data)
unique(vector_ratings) # what are unique values of ratings
## [1] 5 4 0 3 1 2
table_ratings <- table(vector_ratings)#what is the count of each rating value
table_ratings
## vector_ratings
##       0       1       2       3       4       5
## 1469760    6059   11307   27002   33947   21077
```

c) Visualize the rating:

```
vector_ratings<-vector_ratings[vector_ratings !=0] #rating == 0 are NA values
vector_ratings <- factor(vector_ratings)

qplot(vector_ratings) +  ggtitle("Distribution of the ratings")
```

d) Examine the image and ensure you understand what it illustrates

e) Exploring viewings of movies:

```
views_per_movie <- colCounts(MovieLense) # count views for each movie
table_views <- data.frame(movie = names(views_per_movie),
                          views = views_per_movie) # create dataframe of views
table_views <- table_views[order(table_views$views,
                           decreasing = TRUE), ] # sort by number of views

ggplot(table_views[1:6, ], aes(x = movie, y = views)) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
ggtitle("Number of views of the top movies")
```

f) What are the top movies listed?

g) Exploring average ratings:

```
average_ratings <- colMeans(MovieLense)

qplot(average_ratings) +
  stat_bin(binwidth = 0.1) +
  ggtitle("Distribution of the average movie rating")
```

Lab 4. Data Preparation

In this lab exercise, the goal will be to prepare the data set to be used in our recommender models. In these exercises we will accomplish this via the following:

1. Selection of relevant data
2. Normalization of the data

Selection of Relevant data:

The reason we explored the data was to get a sense of how and in what ways the data might influence a recommendation. A recommendation based on a small set of data points will not be very good. Now, when we explored the data previously, we noticed the following:

- The movies that were viewed only a few times might result in biased ratings because of insufficient data
- Ratings from users who rated only a few movies might also result in biased ratings.

Exercise 4.1:

We need to determine the minimum number of users per movie and vice versa.

a) Define `ratings_movies` to contain the matrix (users who have rated at least 50 movies + movies that have been watched at least 100 times)
b) Using the same approach as we did in the previous section, visualize the top 2 percent of users and movies in the new matrix
c) Build the heat map, what do you observe?
d) Visualize the distribution of the average rating by user

Exercise 4.2:

Normalizing the data:

a) Visualize the normalized matrix using the prebuilt normalize function
b) Why is it colored?

c) Why are there still some lines that are more blue and some that are more red?
d) Why is normalization important here?

Exercise 4.3:

Binarizing the data: Using the binarize function binarize for these two options

a) Option 1: define a matrix equal to 1 if the movie has been watched
b) Visualize 5% of users and movies using quantile by building a heatmap
c) Option 2: define a matrix equal to 1 if the cell has a rating above the threshold (3)
d) Visualize option 2 using the same approach and build the heat map.

In this lab exercise, we have prepared the data to perform recommendations. In the next exercises, we will build collaborative filtering models.

Lab 5. ITEM-based Collaborative Filtering

The starting point is a rating matrix in which rows correspond to users and columns correspond to items.

Defining the training and test sets

The two sets are as follows:

- Training set: This set includes users from which the model learns
- Test set: This set includes users to whom we recommend movies

Use `ratings_movies` from the previous exercises. This is the subset of MovieLense users who have rated at least 50 movies and movies that have been rated at least 100 times.

First, we randomly define the `which_train` vector that is TRUE for users in the training set and FALSE for the others. We will set the probability in the training set as 80 percent:

```
which_train <- sample(x = c(TRUE, FALSE), size = nrow(ratings_movies),replace
= TRUE, prob = c(0.8, 0.2))

head(which_train)
```

Exercise 5.1:

Defining training/test sets

a) Define the training and test sets

b) Recommend items to each user, and use the k-fold:

- Split the users randomly into five groups
- Use a group as a test set and the other groups as training sets
- Repeat it for each group

Exercise 5.2:

Building the recommendation model

The function to build models is recommender and its inputs are as follows:

- Data: This is the training set
- Method: This is the name of the technique
- Parameters: These are some optional parameters of the technique

The model is called IBCF, which stands for item-based collaborative filtering. Let's take a look at its parameters:

```
recommender_models <- recommenderRegistry$get_entries(dataType ="realRatingMa
trix")

recommender_models$IBCF_realRatingMatrix$parameters
```

Parameters	Default
k	30
method	Cosine
normalize	center
normalize_sim_matrix	FALSE
alpha	0.5
na_as_zero	FALSE
minRating	NA

a) Build the recommender model using default values
b) How many users does the model learn with?
c) Explore the recommender model using `getModel`
   - Describe the model
   - Structure of the similarity matrix
   - What are the dimensions?
d) Build a distribution chart

e) Which movies have the most elements (top 6)?

Exercise 5.3:

Applying recommender system on the dataset:

Define `n_recommended` that specifies the number of items to recommend to each user. This section will show you the most popular approach to computing a weighted sum:

```
n_recommended <- 6 # the number of items to recommend to each user
```

For each user, the algorithm extracts its rated movies. For each movie, it identifies all its similar items, starting from the similarity matrix. Then, the algorithm ranks each similar item in this way:

- Extract the user rating of each purchase associated with this item. The rating is used as a weight.
- Extract the similarity of the item with each purchase associated with this item.
- Multiply each weight with the related similarity.
- Sum everything up. Then, the algorithm identifies the top *n* recommendations:

a) Produce the object that contains the recommendations using `n_recommend`
b) What is the structure?
c) What are the slots obtained with `slotNames(recommendation object obtained in (a))`?
d) What are the recommendations for the first user?
e) Extract the movie names for the recommendation

Hint

- **items:** *This is the list with the indices of the recommended items for each user*
- **itemLabels:** *This is the name of the items*
- **n:** *This is the number of recommendations*

f) Define a matrix with the recommendations for each user
g) Visualize the recommendations for the first four users
h) Identify the most recommended movies

Lab 6. USER-based collaborative filtering

Exercise 6.1:

Defining training/test sets

a) Define the training and test sets
b) Recommend items to each user, and use the k-fold:
- Split the users randomly into five groups
- Use a group as a test set and the other groups as training sets
- Repeat it for each group

Exercise 6.2:

Building the recommendation system:

The R command to build the model is the same as the previous lab. Now, the technique is called UBCF:

```
recommender_models <- recommenderRegistry$get_entries(dataType ="realRatingMa
trix")

recommender_models$UBCF_realRatingMatrix$parameters
```

Parameters	Default
method	Cosine
Nn	25
sample	FALSE
normalize	center
minRating	NA

Some relevant parameters are:

- **method:** This shows how to compute the similarity between users
- **nn:** This shows the number of similar users

a) Build a recommender model leaving the parameters to their defaults
b) Extract some details about the model using `getModel`

Exercise 6.3:

Applying recommender system on the dataset:

a) Determine the top six recommendations for each new user
b) Define a matrix with the recommendations to the test set users
c) Examine the first four users
d) What are the top titles?