DATA-643, Final Project

Mohamed Elmoudni
Shazia Khan
Senthil Dhanapal

Contents

Introduction

Recommendation systems are composed of filtering algorithms that aim to predict a rating a user would assign to a given item. Recommender systems have become increasingly important across many platforms such as movies (Netflix), restaurants (Yelp), friends (Facebook and Twitter), and music (Pandora and Spotify).

Our final project will be about recommending books based on user ratings. The dataset is called Book-Crossings. The dataset is a book ratings dataset compiled by Cai-Nicolas Ziegler based on data from bookcrossing.com. It contains 1.1 million ratings of 270,000 books by 90,000 users. The ratings are on a scale from 1 to 10.

Our project plan will be based on the below high level steps:

1- Data acquisition
2- Data Exploration and Preparation
3- Model creation
4- Model Selection
5- Prediction
6- Model tuning

Purpose

The goal of the project is to build a recommendation system for books based on user ratings. A user is asked to rate a fixed number of books from our dataset and based on the user's rating for these selected books and ratings given to them by other individuals, our recommendation system recommends other books from our dataset that matches user's interest based on ratings.

Data acquisition

The dataset is a snapshot of www.BookCrossing.com. The motto of the site is "If you love your books, let them go!". The members of this website are located all over the world. A member registers their book and labels it and receives a book ID. The book is then shared with other members privately or publicly. The member can trace the book from one member to another and from one location to another as it travels around the world. The site mentions that at a given point they had 850,000 active users with seven million books which are traveling around 130 countries!

a. `Data source Identification`

b. `Data collection and storing`

c. `Data merging`

Data Exploration

Data samples

Table 1: USERS

User.ID	Location	Age
1	nyc, new york, usa	NULL
2	stockton, california, usa	18
3	moscow, yukon territory, russia	NULL
4	porto, v.n.gaia, portugal	17
5	farnborough, hants, united kingdom	NULL
6	santa monica, california, usa	61
7	washington, dc, usa	NULL
8	timmins, ontario, canada	NULL
9	germantown, tennessee, usa	NULL
10	albacete, wisconsin, spain	26

ISBN	Book.Title	Book
0195153448	Classical Mythology	Mark
0002005018	Clara Callan	Richa
0060973129	Decision in Normandy	Carlo
0374157065	Flu: The Story of the Great Influenza Pandemic of 1918 and the Search for the Virus That Caused It	Gina
0393045218	The Mummies of Urumchi	E. J.
0399135782	The Kitchen God's Wife	Amy
0425176428	What If?: The World's Foremost Military Historians Imagine What Might Have Been	Robe
0671870432	PLEADING GUILTY	Scott
0679425608	Under the Black Flag: The Romance and the Reality of Life Among the Pirates	David
074322678X	Where You'll Find Me: And Other Stories	Ann

Table 3: RATING

User.ID	ISBN	Book.Rating
276725	034545104X	0
276726	0155061224	5
276727	0446520802	0
276729	052165615X	3
276729	0521795028	6
276733	2080674722	0
276736	3257224281	8
276737	0600570967	6
276744	038550120X	7
276745	342310538	10

Data Fields

Table 4: USERS Columns

USERS
User.ID
Location
Age

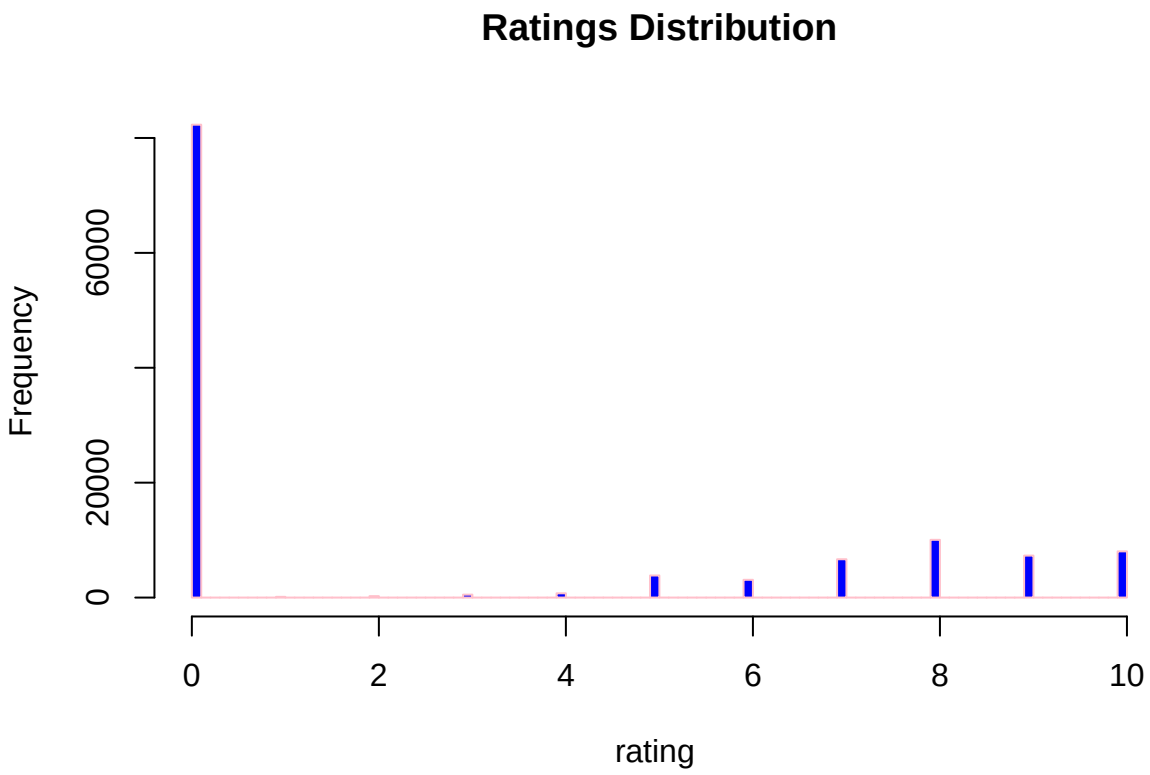
Table 5: BOOKS Columns

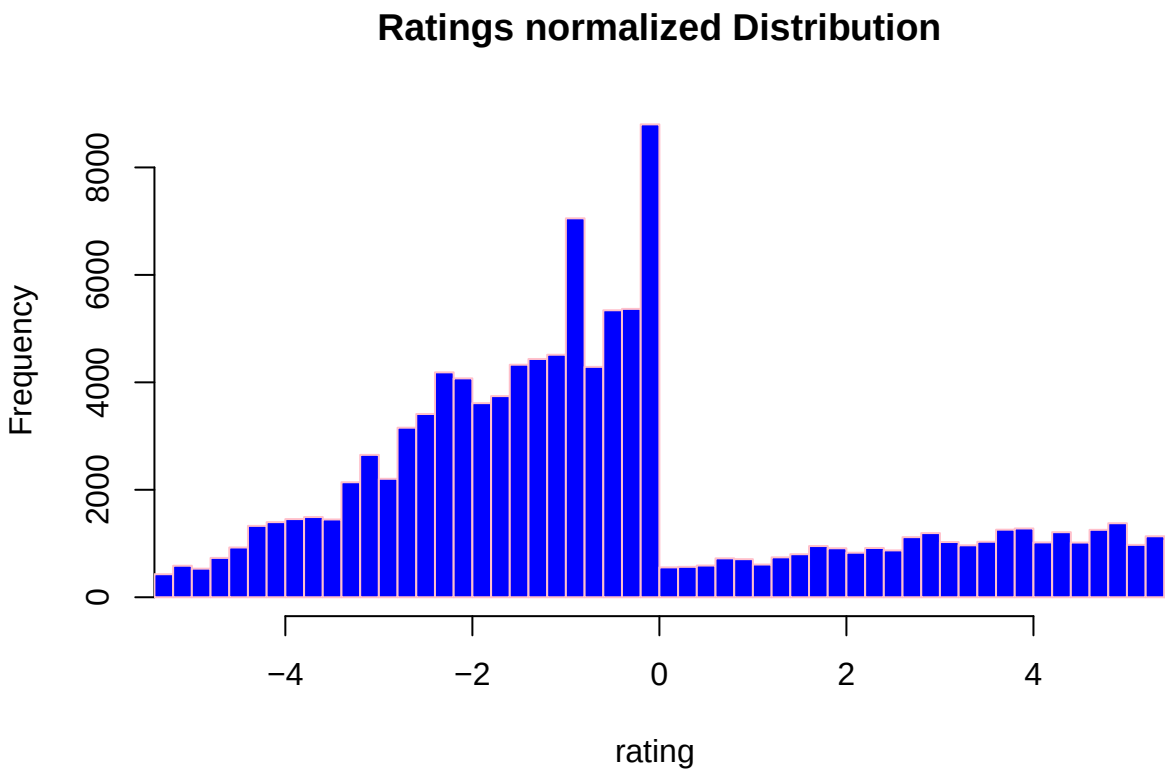
names.books.
ISBN
Book.Title
Book.Author
Year.Of.Publication
Publisher
Image.URL.S
Image.URL.M
Image.URL.L

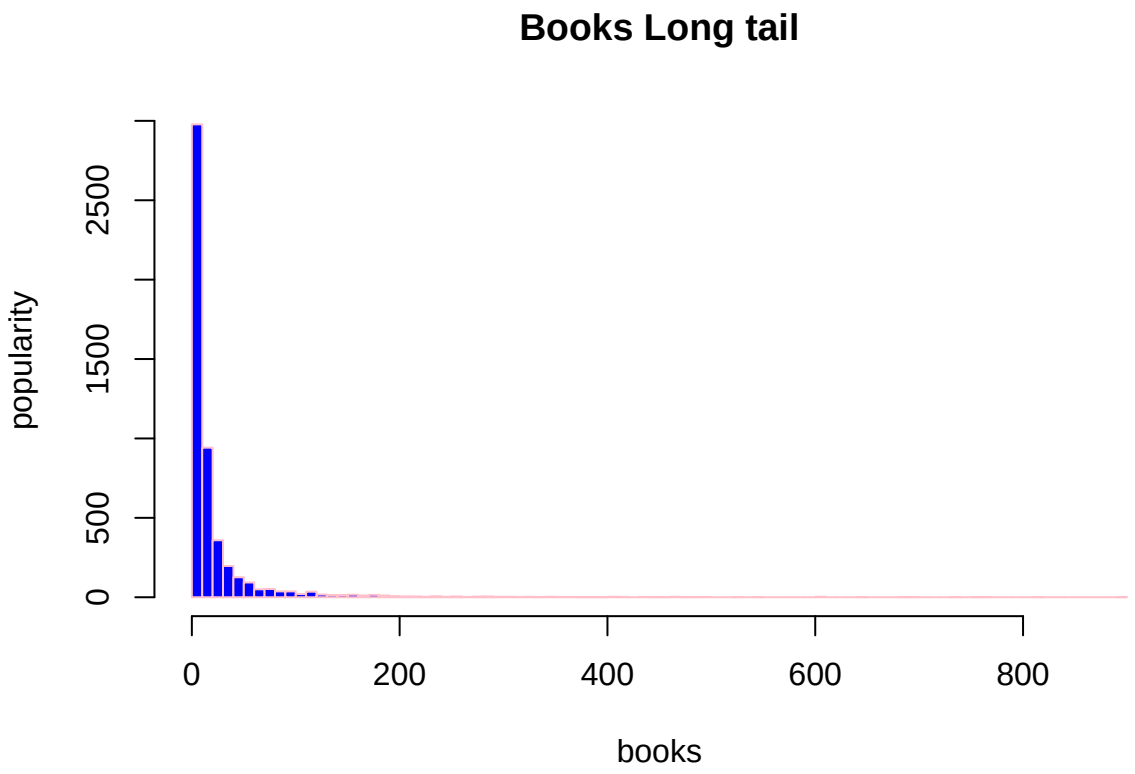
Table 6: RATING Columns

names.rating.
User.ID
ISBN
Book.Rating

Plots


Ratings Distribution
Frequency
0
20000
60000
0
2
4
6
8
10
rating


Ratings normalized Distribution
Frequency
0
2000
4000
6000
8000
−4
−2
0
2
4
rating



Methods

- We used Sparklyr Package to interface with Spark
- We used the Alternating Least Squares (ALS) matrix factorization to reduce our rating matrix.
- We used Collaborative Filtering.

Data Preparation

Looking at the data, there will be alot of cleaning and tidying to do before we can use it for creating to create models for recommender systems. The BX-Books.csv and BX-Ratings.csv files are properly delimited by semicolon and text is qualified by double quotes but the BX-Users.csv file does not seem to be in this format.The cell content is split into multiple columns as seen in the csv Excel format. We will have to make sure that all the users in ratings file are in users file just as we will check for books in ratings file are listed in book file.

Therefore, our data preparation methodology follows the below approach:

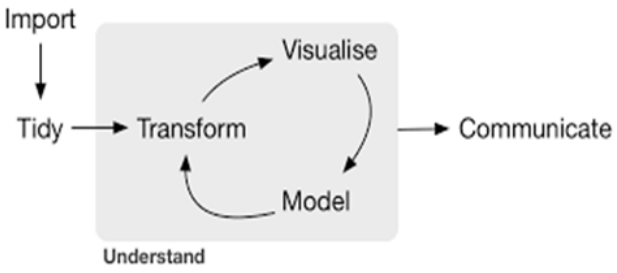


Figure 1:

Data Import and Cleansing Process

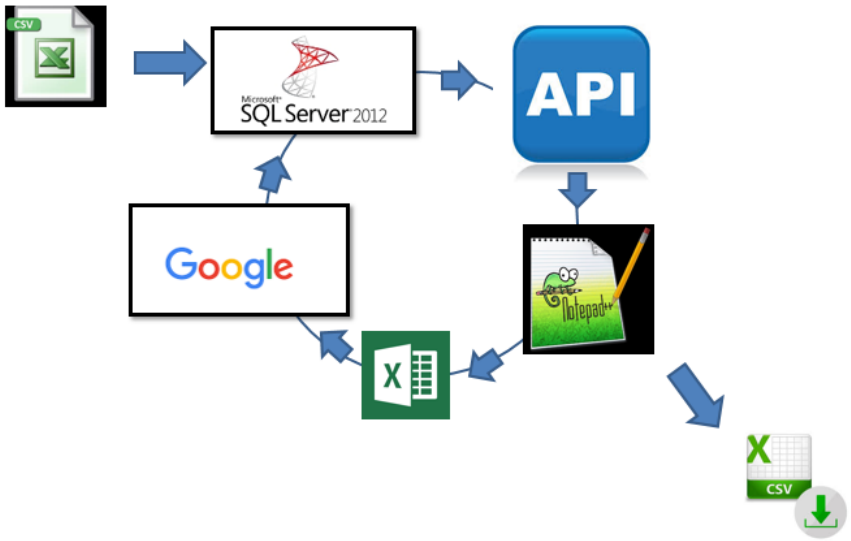


Figure 2:

SQL Code used for data cleansing and adding surrogate key. Added as comments.

Model Creation

Since our real rating matrix is a sparse matrix as it has 89890 rows and 212931 columns with only 965290 ratings; which is about 0.00522 %, we chose to use collaborative filtering along with ALS for matrix factorization.

```
##                   Length Class      Mode
## item.factors      11     data.frame list
## user.factors      11     data.frame list
## data               2     spark_jobj environment
## ml.options         6     ml_options list
## model.parameters   2     -none-     list
## .call              9     -none-     call
## .model             2     spark_jobj environment
```

Model Evaluation

We use RMSE (root mean square error) to evaluate our model against regularization

```
## [1] 1.348361
```

```
## [1] 1.459424
```

And with regularization learning rate of 0.1, our RMSE is 1.348361; however, with .2, our RMSE is 1.459424

Conclusion

The project presented a number of challenges. Given, the data size, we had to use Spark specifically Sparklyr package to process the data. All our non-Spark attempts failed and resulted in insufficient memory errors. The data cleaning and preparation process was another challenge that took almost 60% of the project time. Spark on Databricks presented limited options as we could not load Recommenderlab package as it requires the latest version of R that Spark on Databricks did not provide. Finally, in the positive note, the combination of ALS and Spark has delivered robust and fast solution that enabled us to successfully process a large dataset.

Appendix A: DATA643 Final Project R Code

```
### Explore raw data

library(recommenderlab)
library(cluster)
library(knitr)

setwd("C:/CUNY/Courses/IS-643/Final/data")


users = read.csv("BX-Users.csv",sep = ";")
books = read.csv("BX-Books.csv",sep = ";")
rating = read.csv("Bx-Book-Ratings.csv",sep = ";",header = T)




### Display raw data samples

kable(head(users, 10), caption = "USERS")
kable(head(books,10), caption = "BOOKS")
kable(head(rating, 10), caption = "RATING")

### Show columns from raw data


#str(users) # 140291 obs. of  3 variables, "User.ID"  "Location" "Age"
USERS1 <- data.frame(names(users))
USERS1$USERS<- USERS1$names.users.
USERS1$names.users.<- NULL
kable(USERS1, caption = 'USERS Columns')

BOOKS1 <- data.frame(names(books))
BOOKS1$BOOKS<- BOOKS1$names.BOOKS.
BOOKS1$names.BOOKS.<- NULL
kable(BOOKS1, caption = 'BOOKS Columns')


RATING1 <- data.frame(names(rating))
RATING1$RATING<- RATING1$names.RATING.
RATING1$names.RATING.<- NULL
kable(RATING1, caption = 'RATING  Columns')

### Display Plots
ratings <- read.csv("C:/CUNY/Courses/IS-643/Final/data/Bx-Book-Ratings.csv", sep=";",header = T)


ratings = subset(ratings,ratings$Rating > 0)
RatingMatrix = as(rating,"realRatingMatrix")
RatingMatrix = RatingMatrix[rowCounts(RatingMatrix)>10,colCounts(RatingMatrix)>10]
```

```r
hist(getRatings(RatingMatrix), breaks = 100, col = "blue", border = "pink", xlim= range(0:10),
     main = "Ratings Distribution", xlab = 'rating ')

hist(getRatings(normalize(RatingMatrix)), breaks = 100, col = "blue", border = "pink", xlim= range(-5:5)
     main = "Ratings normalized Distribution", xlab = 'rating ')



# We can get the values of how many books each user has rated and mean rating of each book


hist(rowCounts(RatingMatrix), breaks = 100, col = "blue", border = "pink",
     main = "Books Long tail", xlab = 'books', ylab='popularity'
                )


### Code for recommendation using cleansed dataset generated from raw data

library(sparklyr)
library(dplyr)
library(randomForest)
library(magrittr)
library(methods)

sc <- spark_connect(master = "local")

ratings_df <- read.csv("C:/CUNY/Courses/IS-643/Final/Presentation/Senthil/BxBookRating.csv", head=T)
ratings_df <- ratings_df[,c(1,3,4)]
ratings_df <- ratings_df[,c('UserID','BookID','Rating')]
colnames(ratings_df)[1:3] <- c("user","item","rating")




book_rating <- copy_to(sc, ratings_df, overwrite = T)

## Create the recommendation model using sparklyr using Alternate Least Square method

model <- ml_als_factorization(book_rating, rating.column = "rating", user.column = "user",
                              item.column = "item", rank = 10L, regularization.parameter = 0.1,
                              iter.max = 10L, ml.options = ml_options())

summary(model)

## Evaluate model by predicting the ratings on the full dataset and compare the values using RMSE value

predictions <- model$.model %>%
  invoke("transform", spark_dataframe(book_rating)) %>%
  collect()

sqrt(mean(with(predictions, prediction-rating)^2))

######### using learning rate of .2
```

```
### Adjust the learning rate slightly to see if it increases or decreases RMSE

model <- ml_als_factorization(book_rating, rating.column = "rating", user.column = "user",
                              item.column = "item", rank = 10L, regularization.parameter = 0.2,
                              iter.max = 10L, ml.options = ml_options())

predictions <- model$.model %>%
  invoke("transform", spark_dataframe(book_rating)) %>%
  collect()

sqrt(mean(with(predictions, prediction-rating)^2))

spark_disconnect(sc)

##
##
```