Recommender Systems Project Report

Raji Wareez Olalekan Student Number: **436965**

Md Imran Pavel Student Number: **429266**

Project: Book Recommendation

* Our project is a direct implementation of standard collaborative filtering recommendation approach. We haven't managed to add any new additional functionality/GUI to our project.

Introduction

In this report, there are 3 more following parts. In the next part, the project and methodology of how it works will be described in general. That chapter will explain what is our project, how collaborative filtering works and what formulas we have used to generate recommended rating for an unrated item in short. A short description of our data and issues we've faced during development will be given here as well. We'll also explain how to run, the outputs of the system in this section. The next part gives a short description of the code. Finally, in the "Conclusion" part we will mention some of our ideas that can be further developed from this point.

Our Project

This project is a book recommendation system. In our data, we have numerical ratings for books provided by users. Data is available online here:
https://www.kaggle.com/zygmunt/goodbooks-10k

ratings.csv contains ratings, book id and user id. books.csv contains the original title of the book and multiple other attributes (authors, isbn, frequency of rating 1, 2, 3 and so on). We didn't need to use the other attributes. Data contains 10000 books and 53424 users. However, we removed duplicates and filtered out some data. We didn't change our data file. Every time the project is run these filtering steps will be taken and we will use 6247 books and 16206 users to demonstrate the recommendation system. We filtered using the following checks:

1. Keep books that have been rated by at least 100 users

2. Keep users that have at least 10 books

Once filtered we create the content matrix from the filtered data. The following is a screenshot of the first 5 rows of the content matrix.

```
In [10]: contentMatrix.head()
```

Out[10]:

book_id	1	2	3	4	5	6	7	8	9	10	...	9938	9960	9963	9964	9966	9976	9977	9981	9990	9994
user_id																					
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
23	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
24	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
27	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
35	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 6247 columns

Once we have done this far we implement the collaborative filtering based on the data in the content matrix.

In collaborative filtering approach, to calculate rating for an item of a target user the following steps are taken:

1. Find users that are eligible to be a similar user. A user can be a similar user if both of the followings are true (we have not considered recursive collaborative filtering in our system):
    a. A user can be a similar user if he has rated at least one item that have been rated by the target user as well.
    b. A user can be a similar user if he has also rated the unrated item of the target user.
2. If a user can be a similar user, similarity is calculated using pearson correlation:

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

3. Once similarities has been calculated between the target user and all possible similar users, from those available similar users top 50 similar users are selected to calculate rating for that particular unrated item. Rating has been calculated using the following prediction function:

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}$$

In our recommender system, for a user, for each unrated item of that user steps 1 to 3 described above are repeated and a rating is generated for each unrated item.

However, if for an unrated item there is no similar user found against target user (meaning no one has rated at least one item same as the target user or has rated at least one item as the target user, but has not rated that particular unrated item), then that unrated item will not get any rating on that run of the system. If content matrix gets updated enough after some recommendations generation, then the a, b checks in step 1 will be true at some point and rating will be generated for that unrated item of that user as well later on.

The worst case scenario is for a target user no recommendations is generated. This will happen if for that target user for none of the unrated items there is a similar user. However, this hasn't happened in our system. We've ran and tested it many times.

Initially when we developed the system using Jupyter Notebook, we only removed duplicate entries from our data. We didn't reduce the size of data using additional 2 checks mentioned previously. It was working fine in Jupyter Notebook with all the 10000 books and 53424 users. But if we don't reduce the size of data using those checks, it throws memory error when we run the project from terminal. If you have enough RAM in your PC you can actually skip the filtering and run the system using the whole data. If you want to do that, you can comment out the lines 12 – 25 in BookRecommender.py and then run it.

The code and data are available in this repository: https://github.com/imran-pavel/BookRecommender

To run the code you have to keep the files BookRecommender.py, ratings.csv and books.csv in the same directory. Then navigate to that directory using terminal. After that run the command "python BookRecommender.py"

```
D:\Study Materials\Final Year\Recommender Systems\Project\BookRecommender>python BookRecommender.py
```

Once you run it, it'll take couple of seconds to read the csv files, filter and create the content matrix behind the scenes. Next the following prompt will show up:

```
Choose from one of the following options (Enter an option number):
1. Randomly select a user and show recommended books for him
2. Enter a user id and show recommended books for him
3. Quit
```

From here on, if you press 1 and enter, the system will randomly select a user, calculate ratings for his unrated items and show the top 5 most recommended books for that user. If you press 2 a list will show up of available users and you'll have to choose one user id from that list, input it and press enter and the system will proceed as option 1 from that point with that user. Option 3 is self explanatory. If you enter some invalid user id, system will fail.

Once you start running, until recommendation is shown the screen will look like this:

```
For User 30314 Selected items so far: 1     Unrated Item 14 of 6228   Checked similarity on users: 2/7
For User 30314 Selected items so far: 1     Unrated Item 14 of 6228   Checked similarity on users: 3/7
For User 30314 Selected items so far: 1     Unrated Item 14 of 6228   Checked similarity on users: 4/7
For User 30314 Selected items so far: 1     Unrated Item 14 of 6228   Checked similarity on users: 5/7
For User 30314 Selected items so far: 1     Unrated Item 14 of 6228   Checked similarity on users: 6/7
For User 30314 Selected items so far: 1     Unrated Item 14 of 6228   Checked similarity on users: 7/7
For User 30314 Selected items so far: 1     Unrated Item 14 of 6228   Checked similarity on users: 7/7
For User 30314 Selected items so far: 1     Unrated Item 15 of 6228   Checked similarity on users: 1/3
For User 30314 Selected items so far: 1     Unrated Item 15 of 6228   Checked similarity on users: 2/3
For User 30314 Selected items so far: 1     Unrated Item 15 of 6228   Checked similarity on users: 3/3
For User 30314 Selected items so far: 1     Unrated Item 15 of 6228   Checked similarity on users: 3/3
For User 30314 Selected items so far: 1     Unrated Item 16 of 6228   Checked similarity on users: 1/7
```

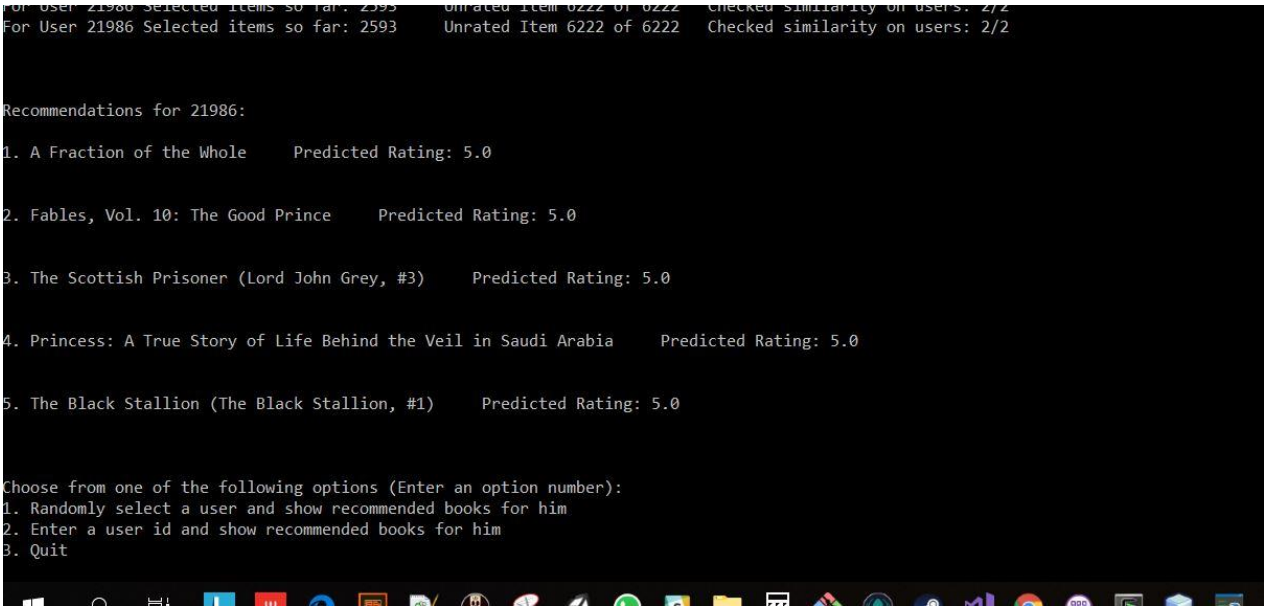
This is to see how much of the calculations is left.

“For User ‘X’ Selected items so far: N Unrated Item A of B Checked similarity on users: C/D”

It means, we are calculating recommendations for X, new ratings have been found for N items. The user has B unrated items and we’ve checked on so far A of them. Content matrix for X in this run is not rich enough to calculate ratings for A – N items. So those items were not selected. For each A there are D similar users and we’ve calculated similarity on C of them at that very moment.

Now depending on how many unrated items a user has and how many similar user are there when we calculate rating for that unrated item, the time it takes to finish one run varies. For some users it takes around 35-40 mins, for some users it can 1.5 – 2 hours or even more to generate recommendations. For example, for user 27503 it takes about 30 mins to show recommendations. But for 51553 it will take significantly longer time.

Following is one of the recommendation output for a user:



Code

Importing the necessary files and dropping duplicate ratings:

```
import pandas as pd
import numpy as np
import math
from random import *


allRatings = pd.read_csv("ratings.csv")
books = pd.read_csv("books.csv")
allRatings = allRatings.drop_duplicates(subset=['book_id', 'user_id'])
```

Filtering is done in the following parts of the code. Comment these lines if you don't want to reduce the size of the data. You should have enough RAM otherwise there will be memory error.

```
minNumberOfRatingsForEachBook = 100         # Leaves us with 6247 book items
minNumberOfRatingsForEachUser = 10          # Leaves us with 16206 users


numberOfRatingsOfEachBook = allRatings["book_id"].value_counts()
eligibleBookItems = numberOfRatingsOfEachBook[numberOfRatingsOfEachBook >= minNumberOfRatingsForEachBook].index
filteringCondition = allRatings["book_id"].apply(lambda x: x in eligibleBookItems)
allRatings = allRatings[filteringCondition]


numberOfRatingsOfEachUser = allRatings["user_id"].value_counts()
eligibleUsers = numberOfRatingsOfEachUser[numberOfRatingsOfEachUser >= minNumberOfRatingsForEachUser].index
filteringCondition = allRatings["user_id"].apply(lambda x: x in eligibleUsers)
allRatings = allRatings[filteringCondition]
```

Keeping the list of all available users in userIDs. getNullAndNonNullColumnsLists will return what books have been and not been rated by a user.

```
userIDs = allRatings['user_id']
userIDs.drop_duplicates(inplace=True)
userIDs = userIDs.sort_values()
userIDs = userIDs.reset_index(drop=True)


def getNullAndNonNullColumnsLists(user, dataFrame):    # G
    nonNullIndexes = []
    nullIndexes = []
    bookIDs = contentMatrix.columns

    for book in bookIDs:
        if pd.isnull(dataFrame.loc[user, book]) == False:
            nonNullIndexes.append(book)
        else:
            nullIndexes.append(book)
    return (nonNullIndexes, nullIndexes)
```

The next part contains the recommendBooks function and the while loop that takes the input. That function is a bit too long to explain here. It is briefly commented in code. This function executes the functionalities of collaborative filtering method.

Conclusion

We have some ideas regarding rectifying recommendations from the current point.

One of them is embedding prioritization in final recommendations before selecting top 5s. We also have authors data. The idea is to figure out which authors' books are more well rated by user than the others. Than sort the generated recommendations based on their author preferences.

Another is to filter the recommended books by doing content based filtering against already highest rated books by the user, then recommend the top 5 books. This approach would likely be more close to user preference, but in our data we don't have contents of the books.