Recommender System on Books based on User Ratings

Abhilash Rajaram(rajaram2@buffalo.edu), Apurva Saha(apurvasa@buffalo.edu), Khushal Arora(khushala@buffalo.edu)
University at Buffalo
Buffalo, NY, 14214

I. Introduction

With data everywhere it was only a matter of time where people keen on observing patterns to learn a meaningful inference from, the method to endorse an item giving it a preference over others. This thought process gave rise to the Recommender System.

Recommender system is defined as a decision making strategy for users under complex information environments. In a very general way, recommender systems are algorithms aimed at suggesting relevant items to users (items being movies to watch, text to read, products to buy or anything else depending on industries).There are different types of recommender systems and in this project, we use collaborative filtering with neighborhood-based methods to make recommendations.

As large OTT conglomerates like Netflix , HBO Max, Prime, Youtube are gaining more and more popularity, the usage of books has been left somewhat behind. Our idea was to look for a way to create a system which reaches out for the not so avid readers as well.

II. Conceptual Approach

There are two major approaches to develop recommendation systems, one is “User Based Collaborative Filtering” and the other is “ Item Based Collaborative Filtering”.

User Based Collaborative Filtering methods (UBCF) :

It is based on finding similar users and finding the items those users have liked but that other users haven’t tried yet. UBCF for recommender systems are usually based on the previous “user and item interactions” to help predict/provide some new recommendations.

Item Based Collaborative Filtering methods (IBCF) :

In this case, we will find similar products to the one the user has bought and we will recommend those products that are similar to those which have rated as best. IBCF for recommender systems are usually based on the previous “item to item interactions” to help predict/provide some new recommendations.

Without the utility matrix making recommendations will be nearly impossible to make. So the interaction that can be done is we can ask the user to rate the Item based on how much he preferred it on a defined scale (1 to 5, 1 to 10 etc.).

Similarity: The way we can recommend items based on user item interaction is by considering how similar two people are before making any predictions. This similarity measure is calculated using a few methods.

1. Jaccard Similarity:

   simJaccard(X , Y) = |X ∩ Y|/|X ∪ Y|,
   where X and Y rating vectors of users X and Y respectively

   In this type of similarity we simply look at the items the users have in common among the total union of items they have rated. The problem of this method is that it doesn't always capture the right intuition from the ratings matrix.
   This is mostly because the Jaccard formula doesn't take the rating values into consideration. This can be used when we don't have a lot of data to work with.
   We can consider the data as a binary matrix with only 2 classes, i.e. ('0' and '1'). In such cases jaccard similarity gives us an idea of the items that are being used more frequently and we don't care as much about the user preferences.

2. Cosine Similarity:

   simCosine(x, y) = (x · y) / (||x||.||y||),
   where x, and y are the user rating vectors and || . || represents the l2 norm of the vectors

   Here we take the dot product of the two rating vectors and divide that by the magnitudes of the two vectors. For this similarity we can't ignore the missing ratings so we consider the default ratings as 'zero'(0). But, then an issue arises as the cosine distance considers these zero values as the users having disliked the item and inherently takes them as 'negative' ratings as in that the user does not prefer them.
   This may cause a problem when 2 users who have similar likes but sparse items in common but the same user may not have such strong likes with another but has more items rated in common. In this case the cosine distance may not differ by much and show both the distances as quite similar.

3. Pearson Similarity:

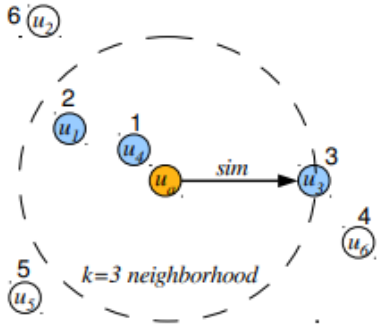
$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where x(i) and y(i) are the ratings from the ratings vector of users x and y and xbar and ybar are the average ratings for the users.

Using the Pearson correlation is helpful because we can reduce the effect of the user bias somewhat since we are taking the average of the user and subtracting it from the given rating. This helps when some users are more conservative with their ratings or some other person may be more free with their ratings always giving higher ratings.
This helps catch a more stronger intuition of similarity from the rating matrix.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	?	4.0	4.0	2.0	1.0	2.0	?	?
u_2	3.0	?	?	?	5.0	1.0	?	?
u_3	3.0	?	?	3.0	2.0	2.0	?	3.0
u_4	4.0	?	?	2.0	1.0	1.0	2.0	4.0
u_5	1.0	1.0	?	?	?	?	?	1.0
u_6	?	1.0	?	?	1.0	1.0	?	1.0
u_a	?	?	4.0	3.0	?	1.0	?	5.0
$\hat{r}_a$	3.5	4.0			1.3		2.0	

(a)



(b)

After finding out the similarities between the users we need to get the predictions so that we can make some recommendations based on it.

To make the predictions we consider the 'k' nearest neighbours from the user and fill in the missing ratings depending on those nearest neighbour ratings. We can see it represented in both the tabular (plot (a)) and graphical (plot (b)) form. This can have two approaches-

1) **r^aj = (1/ |N (a)|) * ∑ i∈N(a) (rij)**

N(a) is the neighbourhood of the user U with a = 0,1,2,....k
r^aj is the rating being predicted
∑ i∈N(a) (rij) is the sum of the ratings for the item that the neighbours have rated

This method just takes the average of the ratings for the item whose rating is being predicted from the ratings of the nearest 'k' specified neighbours.
But this method does not take into account the similarity between the users. Two user may have some strong common likes but may not have much similarity considering all the item set.
To rectify this we have another method

2) **r^aj = (1/ ∑ i∈N(a) (sai)) * (∑ i∈N(a) (sai * rij))**

sai is the weight that we consider and is also the similarity between the to users
N(a) is the neighbour with a = 0,1,2….k
r^aj is the rating being predicted

Here we have the weights added to curb the issue mentioned above by adding in the similarity between the two users as weights to make sure the prediction is more intuitively accurate.

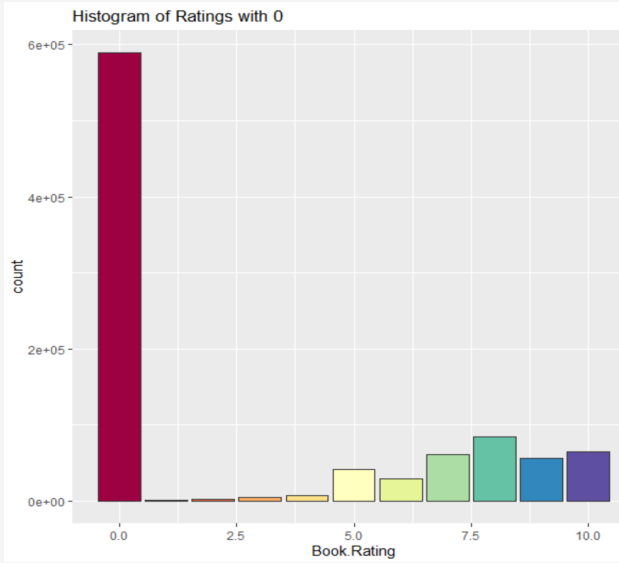
III. Dataset and Preprocessing of Data

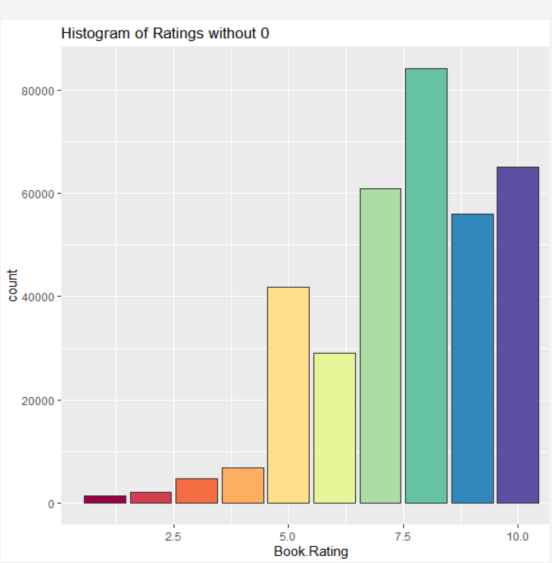
In order to code a recommendation system in R, the first thing that we need is data. Our data was collected by Cai-Nicolas Ziegler in a 4-week crawl (August / September 2004) from the Book-Crossing community with kind permission from Ron Hornbaker, CTO of Humankind Systems. The entire data contains 278,858 users (anonymized but with demographic information) providing 1,048,575 ratings (explicit / implicit) about 271,360 books.

The data consists of 3 datasets.

1) Users:
   Users dataset contains the users and their information. Note that user IDs (`User-ID`) have been anonymized and mapped to integers. Demographic data is provided (`Location`, `Age`) if available. Otherwise, these fields contain NULL-values.
2) Books:
   Books are identified by their respective ISBN. Invalid ISBNs have already been removed from the dataset. Moreover, some content-based information is given (`Book-Title`, `Book-Author`, `Year-Of-Publication`, `Publisher`), obtained from Amazon Web Services. Note that in the case of several authors, only the first is provided. URLs linking to cover images are also given, appearing in three different flavours (`Image-URL-S`, `Image-URL-M`, `Image-URL-L`), i.e., small, medium, large. These URLs point to the Amazon web site.
3) Ratings:
   Ratings dataset contains the book rating information. Book-Rating is either explicit, expressed on a scale from 1-10 (higher values denoting higher appreciation), or implicit, expressed by 0 (indicating the user didn't give ratings to the book).

For data preprocessing, first we find out if there are any missing values. It was found that the ‘Age’ variable contained around 110762 values were missing. Since that is a significant number of values and Age doesn’t contribute to the ratings, it has been removed. Additionally, the image URLs of the books are removed. Finally, all the 3 datasets are merged together, so that the final dataset contains 941112 observations with 8 variables. Our aim is to develop recommendation algorithms through ratings, it can be seen that there are a lot of ratings with 0 values. Therefore, the recommendation algorithms are created for only non-zero rating values.





Users with most Ratings

User.ID	number_of_ratings_per_book
<int64>	<int>
11676	6945
98391	5691
189835	1899
153662	1847
23902	1181
235105	1020

Books with most ratings

A tibble: 6 x 2

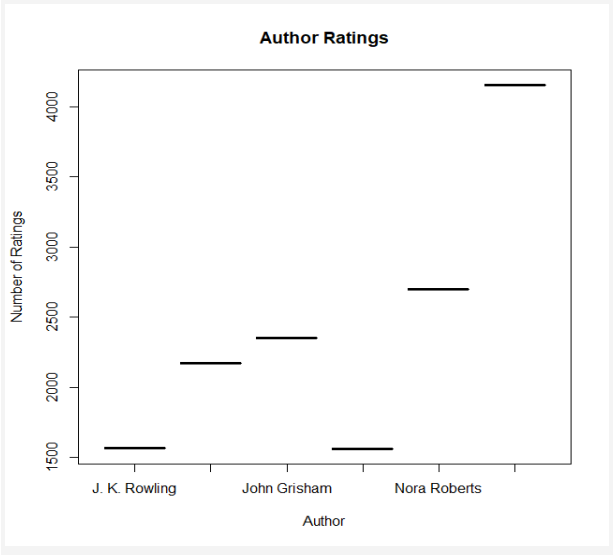
ISBN	number_of_ratings_per_book
<int64>	<int>
316666343	635
971880107	526
385504209	452
312195516	354
60928336	292
59035342	287

Authors with most Ratings

Book.Author	number_of_times_authors_rated
<chr>	<int>
Stephen King	4156
Nora Roberts	2701
John Grisham	2354
James Patterson	2173
J. K. Rowling	1566
Mary Higgins Clark	1560

Publications with most Ratings

	Publisher	number_of_times_publisher_rated
	<chr>	<int>
1	Ballantine Books	11621
2	Pocket	9623
3	Berkley Publishing Group	8501
4	Warner Books	8202
5	Bantam	6751
6	Bantam Books	6693



IV. Computational Approach

The recommenderlab package provides the infrastructure to develop and test recommender algorithms for rating data. With coercion, our preprocessed dataset "new_dats2" is converted into a realRatingMatrix object which stores the data in sparse format (only non-NA values are stored explicitly; NA values are represented by a dot).

Firstly, a Popularity-Based recommender system is created using the creator function Recommender() and method Popular. Then, top-N book predictions for users are made using the predict functions. Then the ISBN for each user is specified, and based on that the book information is fetched.

```
> as(top_rec, "list")
$`1435`
[1] "821749668"  "1551666685" "394534433"  "590251678"  "785281967"
[6] "8423986225" "66210232"   "792727770"  "816157464"

$`3827`
[1] "821749668"  "1551666685" "451208765"  "590251678"  "785281967"
[6] "8423986225" "66210232"   "792727770"  "816157464"

$`11676`
[1] "394534433"  "451208765"  "590251678"  "785281967"  "8423986225"

$`11993`
[1] "821749668"  "1551666685" "394534433"  "451208765"  "590251678"
[6] "8423986225" "66210232"   "792727770"  "816157464"

$`12982`
[1] "821749668"  "1551666685" "394534433"  "451208765"  "590251678"
[6] "785281967"  "66210232"   "792727770"  "816157464"
```

```
> head(Books[Books$ISBN == 394534433, c(1,2)])
          ISBN                                          Book.Title
17876 394534433 Vampire Lestat (Chronicles of the Vampires, 2nd Book)
```

After that, a recommender system based on User-Based collaborative filtering is created using the creator function Recommender() and method UBCF. The similarity measures that are being

used are - Cosine and Jaccard. In the parameter list, the following conditions are specified – the nearest neighbors to be considered is 30 and data is normalized using mean.

The prediction of the ratings is calculated using predict function and type – ratingMatrix. When the ratingMatrix type is used all ratings for all users are predicted. However, since our dataset is huge the similarity measures are returning zero values and the prediction is not happening.

```
> # Using Cosine dissimilarity measure and predicting the user ratings
> as(predict_ratingmatrix_1, "matrix")[,1:10]
      66210232 394534433 451208765 590251678 785281967 792727770
1435        NA        NA         2        NA        NA        NA
3827        NA         4        NA        NA        NA        NA
11676        4        NA        NA        NA        NA         4
11993       NA        NA        NA        NA         5        NA
12982       NA        NA        NA        NA        NA        NA
16795       NA        NA        NA         4        NA        NA
      816157464 821749668 1551666685 8423986225
1435         NA        NA         NA         NA
3827         NA        NA         NA         NA
11676         3         5          5         NA
11993        NA        NA         NA         NA
12982        NA        NA         NA          4
16795        NA        NA         NA         NA
```

```
> # Using Jaccard dissimilarity measure and predicting the user ratings
> as(predict_ratingmatrix_2, "matrix")[,1:10]
      66210232 394534433 451208765 590251678 785281967 792727770
1435        NA        NA         2        NA        NA        NA
3827        NA         4        NA        NA        NA        NA
11676        4        NA        NA        NA        NA         4
11993       NA        NA        NA        NA         5        NA
12982       NA        NA        NA        NA        NA        NA
16795       NA        NA        NA         4        NA        NA
      816157464 821749668 1551666685 8423986225
1435         NA        NA         NA         NA
3827         NA        NA         NA         NA
11676         3         5          5         NA
11993        NA        NA         NA         NA
12982        NA        NA         NA          4
16795        NA        NA         NA         NA
```

Next, another recommender system based on Item-Based collaborative filtering is created using the creator function Recommender() and method IBCF. The similarity measures that are being used are - Cosine and Jaccard. As before the prediction is not happening because of the huge dataset and similarity measures are returning zero values.

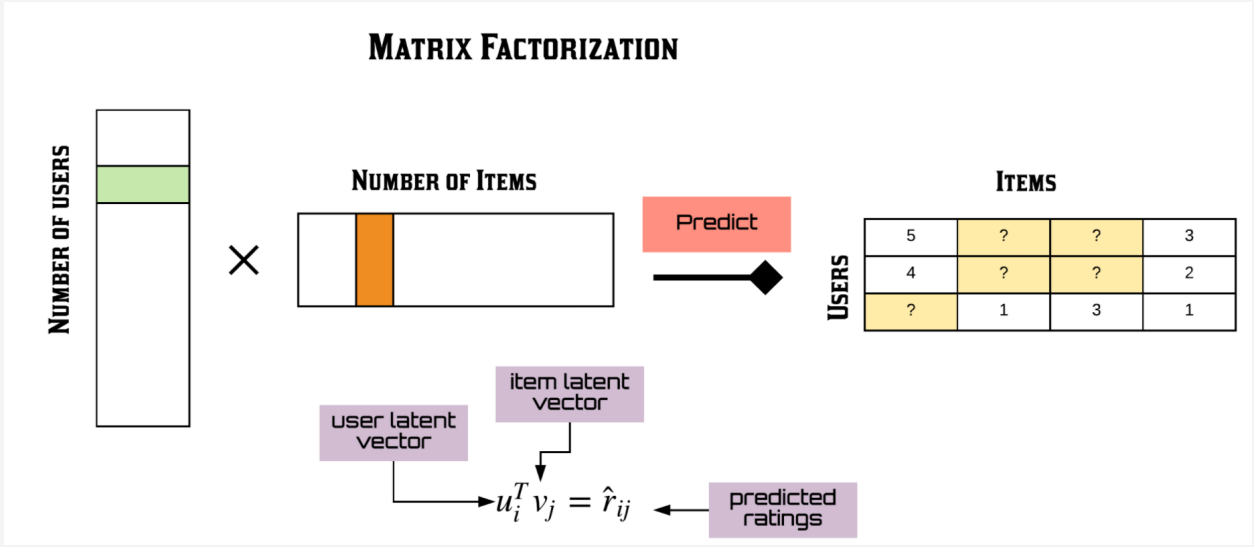
```
> # Using Cosine dissimilarity measure and predicting the item ratings
> as(predict_ratingmatrix_3, "matrix")[,1:10]
      66210232 394534433 451208765 590251678 785281967 792727770
1435        NA        NA        NA        NA        NA        NA
3827        NA        NA        NA        NA        NA        NA
11676     4.25        NA        NA        NA        NA      4.25
11993       NA        NA        NA        NA        NA        NA
12982       NA        NA        NA        NA        NA        NA
16795       NA        NA        NA        NA        NA        NA
      816157464 821749668 1551666685 8423986225
1435         NA        NA         NA         NA
3827         NA        NA         NA         NA
11676       4.5         4          4         NA
11993        NA        NA         NA         NA
12982        NA        NA         NA         NA
16795        NA        NA         NA         NA
```

```
> # Using Jaccard dissimilarity measure and predicting the item ratings
> as(predict_ratingmatrix_4, "matrix")[,1:10]
      66210232 394534433 451208765 590251678 785281967 792727770
1435        NA        NA        NA        NA        NA        NA
3827        NA        NA        NA        NA        NA        NA
11676     4.25        NA        NA        NA        NA      4.25
11993       NA        NA        NA        NA        NA        NA
12982       NA        NA        NA        NA        NA        NA
16795       NA        NA        NA        NA        NA        NA
      816157464 821749668 1551666685 8423986225
1435         NA        NA         NA         NA
3827         NA        NA         NA         NA
11676       4.5         4          4         NA
11993        NA        NA         NA         NA
12982        NA        NA         NA         NA
16795        NA        NA         NA         NA
```

Therefore, the above approaches are limited to only small datasets where the similarity measure can be measured easily.

This leads to the concept of Recommendation using Matrix Factorization for Collaborative Filtering. Our intention is to make the bulkiness of the dataset to create less convolution at the time prediction.

In its natural form, Matrix Factorization characterizes items and users using vectors of factors inferred from item-rating patterns. High correspondence between item and user factors leads to a recommendation.



V. Further Approach

There are different variants of matrix factorization that can be constructed — ranging from the use of side features to the application of Bayesian methods.

1) Vanilla Matrix Factorization:

   A straightforward matrix factorization model maps both users and items to a joint latent factor space of dimensionality D — such that user-item interactions are modeled as inner products in that space.

$$R_{ui} = q_i * p_u$$

2) Matrix Factorization with Biases:

   Similar to Vanilla Matrix Factorization but with biases. The intuition behind this is that some users give higher ratings than others, and some items received higher ratings than others systematically.

$$R_{ui} = q_i * p_u + b + w_i + w_u$$

3) Matrix Factorization with Side Features:

   A common challenge in collaborative filtering is the cold start problem due to its inability to address new items and new users. Or many users are supplying very few ratings,

making the user-item interaction matrix very sparse. A way to relieve this problem is to incorporate additional sources of information about the users, aka side features. These can be user attributes (demographics) and implicit feedback.

$$R_{ui} = q_i * p_u + q_i * t_o + b + w_i + w_u + d_o$$

4) Matrix Factorization with Temporal Features

So far, our matrix factorization models have been static. In reality, item popularity and user preferences change constantly. Therefore, we should account for the temporal effects reflecting the dynamic nature of user-item interactions. To accomplish this, we can add a temporal term that affects user preferences and, therefore, the interaction between users and items.

$$R_{ui}(t) = q_i * p_u + p_u * t_o + p_u(t) + b + w_i + w_u$$

Using any of the above approaches, we can make a better recommendation to each user n-ranked (e.g 5 or 10) books with predicted vote. One thing that needs to be taken into consideration is that, when splitting up the data into training and test sets, you should randomly select (user, book) pairs, not select random users or books.

The whole idea is the model should be able to predict ratings for books users haven't seen, based on the ratings provided for ones you have. If a user is present only in the testing set, the model cannot possibly be basing predictions based on their other ratings

VI. References

1. Charu C Aggarwal : Recommender Systems,Springer.
2. RecommenderLab Tutorials: https://cran.r-project.org/web/packages/recommenderlab/vignettes/recommenderlab.pdf
3. Matrix Factorization with Collaborative Filtering: https://towardsdatascience.com/recommendation-system-series-part-1-an-executive-guide-to-building-recommendation-system-608f83e2630a