Knowledge Discovery and Data Mining - Collaborative recommender system on Goodreads dataset [1]

Mohammad Poul Doust

March 2019

Please refer to corresponding Github Repository.

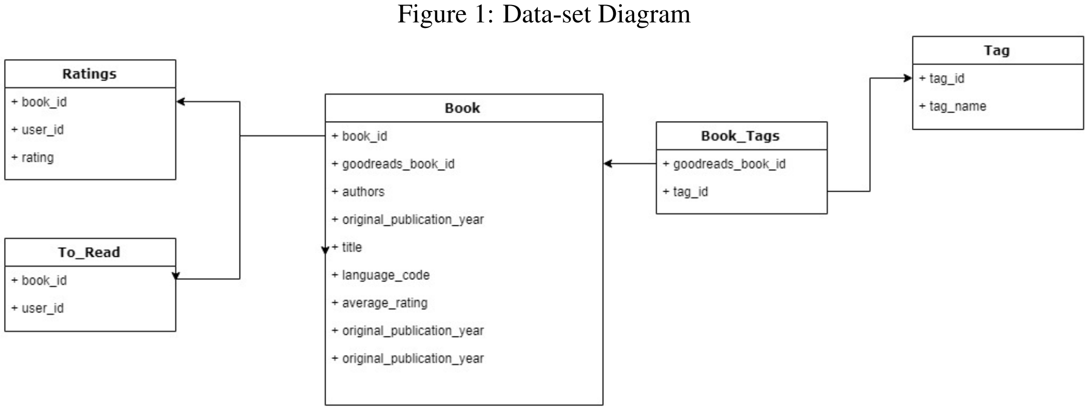
1 Problem

In this report, we will explore a dataset about books preferences and recommendation gathered from websites such as GoodReads. at first we will try to understand the data-set structure. explore each part of it with preliminary statistics and analysis. We will also try to explore some figures about the dataset. afterwards, we will clean the data and prepare it for further analysis to perform recommendation algorithm. Specifically, user-based collaborative filtering. in the end, we will evaluate the model against others with respect to different evaluation metrics.

2 Dataset

The used data-set basically represents users' ratings for books with meta data about the rated books. it was collected from GoodReads website for book readings and recommendation. This data-set is mainly made of five main CSV tables as depicted in the figure below. We will go in details for each part:

Figure 1: Data-set Diagram



2.1 Books

Books table contains information about 10000, each with the corresponding basic information, for instance:

- book_id : book identifier
- goodreads_book_id : book identifier at goodreads website
- authors : book's authors - comma separated
- original_publication_year
- title

```
> dim(books)
[1] 10000    23
> summary(books$id)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1    2501    5000    5000    7500   10000
> summary(books$original_publication_year)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  -1750    1990    2004    1982    2011    2017      21
```

We can also see that Books publication year contains negative years, which should be handled in data preparation phase

2.2 Ratings

Ratings table contains users book's rating (around 981756 rating)

- book_id : book identifier
- user_id : user identifier
- rating : rating for the book

```
> dim(ratings)
[1] 981756      3
> head(ratings)
   book_id user_id rating
1:       1     314      5
2:       1     439      3
3:       1     588      5
4:       1    1169      4
5:       1    1185      4
6:       1    2077      4
```

```
> summary(ratings)
    book_id          user_id          rating
 Min.   :    1   Min.   :    1   Min.   :1.000
 1st Qu.: 2457   1st Qu.:12372   1st Qu.:3.000
 Median : 4921   Median :25077   Median :4.000
 Mean   : 4943   Mean   :25617   Mean   :3.857
 3rd Qu.: 7414   3rd Qu.:38572   3rd Qu.:5.000
 Max.   :10000   Max.   :53424   Max.   :5.000
```

2.3 Tags

This table contains general tags each identified by id. Specifically there are 34252 tags. as depicted in figure below, data cleaning is needed for this table.

```
> dim(tags)
[1] 34252     2
> head(tags)
   tag_id tag_name
1:      0        -
2:      1     --1-
3:      2    --10-
4:      3    --12-
5:      4   --122-
6:      5   --166-
```

```
> summary(tags)
     tag_id        tag_name
 Min.   :    0   Length:34252
 1st Qu.: 8563   Class :character
 Median :17126   Mode  :character
 Mean   :17126
 3rd Qu.:25688
 Max.   :34251
```

2.4 Book Tags

This table contains the tags associated with a book from book table.

2.5 To Read

This table contains books id marked as "to read" by a user id. There are around a million records.

```
> summary(to_read)
    user_id          book_id
 Min.   :    1   Min.   :    1
 1st Qu.:15507   1st Qu.:  360
 Median :27799   Median : 1381
 Mean   :27669   Mean   : 2455
 3rd Qu.:40220   3rd Qu.: 3843
 Max.   :53424   Max.   :10000
```

3 Approach - Analysis

3.1 Data Preparation and Exploration

3.1.1 Ratings

Ratings table reported to have duplicate, we clean the table by removing all duplicates on both (user_id, book_id), when same user rate same movie more than once :

```
Ratings dim before cleaning duplicates:  981756 3
```

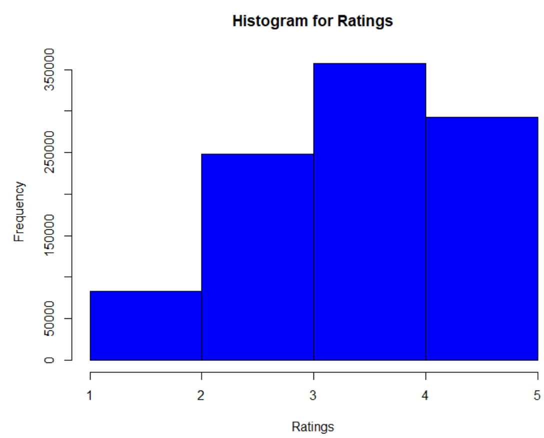
```
> dim(book_tags)
[1] 999912      3
> head(book_tags)
   goodreads_book_id tag_id  count
1:                 1  30574 167697
2:                 1  11305  37174
3:                 1  11557  34173
4:                 1   8717  12986
5:                 1  33114  12716
6:                 1  11743   9954
```

```
> summary(book_tags)
 goodreads_book_id      tag_id          count         
 Min.   :       1   Min.   :    0   Min.   :    -1.0  
 1st Qu.:   46227   1st Qu.: 8067   1st Qu.:     7.0  
 Median :  394841   Median :15808   Median :    15.0  
 Mean   : 5263442   Mean   :16325   Mean   :   208.9  
 3rd Qu.: 9378297   3rd Qu.:24997   3rd Qu.:    40.0  
 Max.   :33288638   Max.   :34251   Max.   :596234.0  
```

```
Ratings dim after cleaning duplicates:  979478 3
```

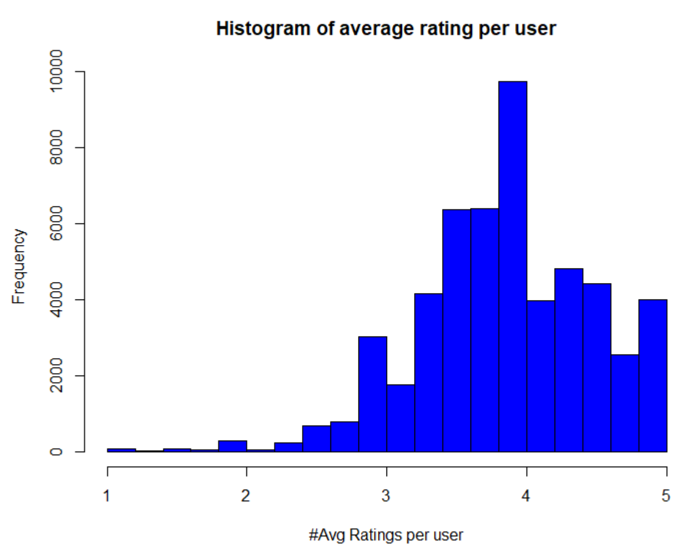
We first start by plotting the histogram of ratings.

```
hist(ratings$rating, main="Histogram for Ratings",
 xlab="Ratings", border="black", col="blue", breaks=5)
```



We can see that rating 4 is the highest rate among others, while rating 1 is the lowest. In general we can notice that most users tend to give high ratings, while low ratings could be interpreted as either low rating or something not rated.

Now we will try to see what is the histogram (frequency) for people having the same mean ratings. this will give us idea about the quantity of people with different mean ratings.



As we can see, there is a relatively low percentage of people with avg rating of 1. which probably represents non active users, or fake users to down-vote ratings. While there is good amount of people with average rating of 5. which could be interpreted as people who only rates books that they like. and do not bother to rate bad books for them, Moreover, those people also might represent a fake up-voters.

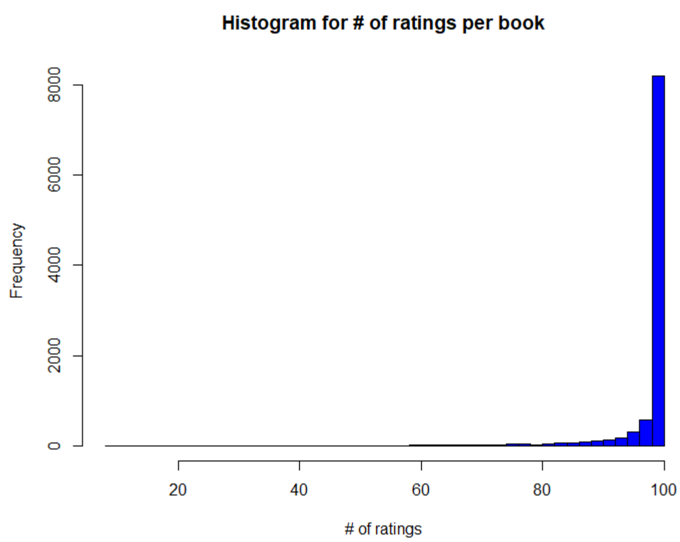
3.1.2 To_Read

To Read table has no duplicates. we can try to see the percentage of books in to_read list of all books list. Moreover, it might be interesting to see the percentage of users that have to read next compared to all users.

```
Percentage of users with to_read to all users:  91.47761 %
Percentage of books in to_read to books:  99.86 %
```

So nearly 92 % of all users have books in their to_read list. Similarly, roughly all books are covered by to_read list of all users.

3.1.3 Book



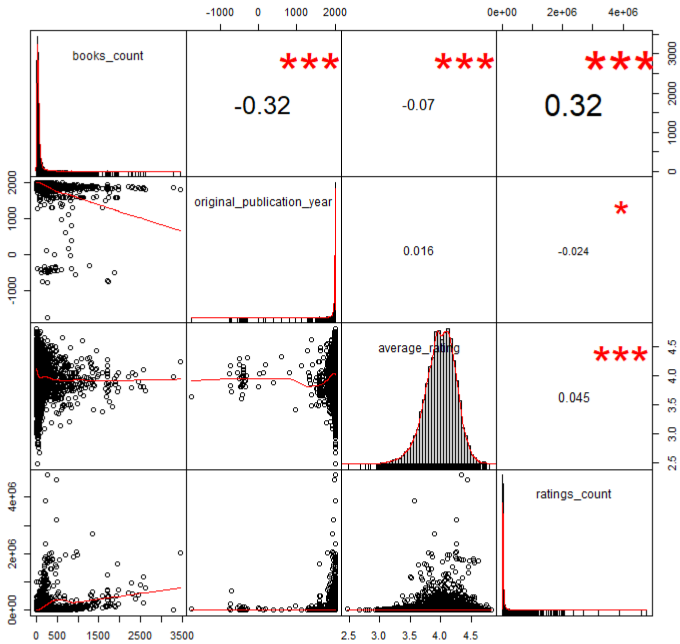
We can see from the figure above that most of books has about 100 ratings. and at least each book has a total of 8 ratings.

3.1.4 Correlation

Since books table contains information about "average_rating" for each book, "ratings_count" it will be interesting to see if there is a correlation between these fields and other book features, such as: "original_publication_year" and "books_count". there are also many text fields that would be interesting to study, like "authors", book tags, language...

```
booksToAnalyze = select(books, books_count,
                        original_publication_year,
                        average_rating, ratings_count)

chart.Correlation(booksToAnalyze, histogram=TRUE,
                  pch=19)
```



We can see clearly, that according this correlation chart, there is no strong correlation between any pair of attributes. maybe if we take other string attributes will get stronger correlation, for example, probably some authors take more rating than others. Moreover, that give us an idea that most of the rating are based on the content of each book. Hence, it might be useful to explore content based approach for this dataset.

3.2 Data Modeling

Mainly, when it comes to Recommendation Systems, we have two options, either to suggest items based on the content of each item, in our case, we have to perform text mining techniques to extract compact information regarding each book (the author, the genres, the title, etc). On the other hand, Collaborative filtering is widely used, the idea is:

- Users with similar tastes probably like similar books (UBCF)
- Items liked by same people will be suggested together (IBCF)

for this study, we will experiment with collaborative filtering, since content-based method require more understanding, cleaning, analyzing and processing. for this, "recommenderlab" library make it easier to apply the algorithm directly and evaluate it in a neat way. but before that, we have to convert our data to the form of "realRatingMatrix" where we have a matrix of row representing users and columns represents books. and since our data will be mostly sparse (a lot of zeros). we will convert to "SparseMatrix" afterwards. and in the end, we convert to realRatingMatrix

```
#Converting the data to a matrix where rows are users,
# columns are books and the values are the ratings
RatingTable = dcast(ratings, formula = user_id ~ book_id,
                    value.var = "rating")

# Convert to sparse matrix representation
SparseTable <- as(RatingTable, "sparseMatrix")

# Convert to realRatingMatrix,
# to be compatible with recommenderlab methods
```

```
RealRatingsTable <- new("realRatingMatrix", data = SparseTable)
```

4 Evaluation and Results

for evaluation we get advantage of recommenderlab built in useful function. Specifically, recommenderlab provides the ability to build an evaluator to evaluate different algorithm against the same data. also it takes care of cross validation.

As depicted below, we first define which algorithms we are willing to test, in our case: Random, Popular and UBCF. [2] [3]

```
algorithms <- list(
        "Random_Items" =
            list(name = "RANDOM", param = list(normalize = "Z-score")),
        "Popular_Items" =
            list(name = "POPULAR", param = list(normalize = "Z-score")),
        "User_based" =
            list(name = "UBCF", param = list(normalize = "Z-score", nn = 50))
        )
```

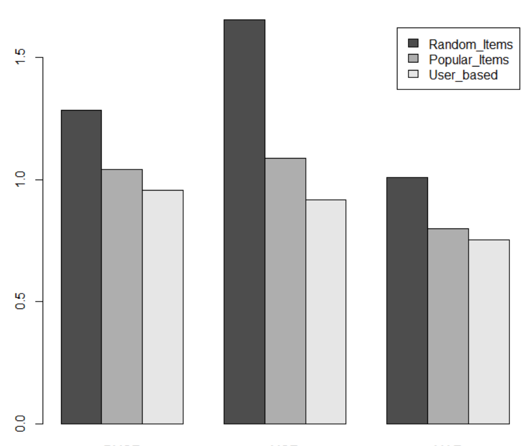
After, we prepare our evaluator scheme by initializing it with the following values:

- RealRatingsTable[1:3000,]: we pass the first 3k rows of our full array of ratings, since passing the whole data would be problematic in case of time and size (53424 x 10000).
- method ="cross-validation": evaluation method to use, we chose k-fold cross-validation method.
- k =3 : number of fold = 3
- given = -1 : choosing -1 indicate that the algorithm will use all ratings (except for one)to learn for prediction, and evaluate using the excluded one.
- goodRating = 5 : threshold to evaluate positive rating

```
scheme <- evaluationScheme(RealRatingsTable[1:3000,],
          method ="cross-validation", k = 3, given = -1, goodRating = 5)
```

Now we can print the evaluation

```
plot(resultsRating)
```



It is clear from the above graph that user based collaborative filter performed best in term of our three evaluation metrics, While random clearly performed the worst:

- Root Mean Square Error (RMSE)
- Mean Squared Error (MSE)
- Mean Absolute Error (MAE)

Method	RMSE	MSE	MAE
Random_Items	1.286	1.655	1.01
Popular_Items	1.043	1.088	0.7993
User_based	0.9576	0.9174	0.7519

5 Why 'GoodReads' Recommendation System

Recommendation systems are very interesting in general. Moreover, it is motivating to explore such database for books recommendation, to see patterns and how users are affected by other factors. Specially, that the dataset is large and versatile which make it perfect candidate for various methods. For instance, content based recommendation system is applicable here. also we could make use of to read table. and many information

6 Conclusion

Overall, this dataset is very large and various which makes it possible to perform wide range of different analysis. it contains duplicates and needs further cleaning and formalizing. in this report, we just performed first step exploration. In addition to build a recommender system and evaluate different methods against this dataset. User-based filtering performed the best between suggested methods, not bad at all. However, this dataset contains good amount of content information. for books. with the ability also to link it with other datasets. Like for example to link each book with corresponding movie (since there are many books that turned into a movie) and that will probably reflect on ratings. also it will be useful to cluster books with different volumes, and see if books has different volumes, would that give any effect on rating ? which volume has the best rating on average.

It would be useful to use user-based collaborative filtering in case we do not know much about the data and it will give good results. but in case we have already meta information and the possibility to go deep into the data. content based recommendation would be more efficient.

References

[1] Zygmunt Zajac. Goodbooks-10k: a new dataset for book recommendations. *FastML*, 2017.

[2] Yanchang Zhao and Yonghua Cen. *Data Mining Applications with R*. Academic Press, Inc., Orlando, FL, USA, 1st edition, 2013.

[3] Kumudini Bhave. Recommender system :movielens.