Recommendation system using collaborative filtering techniques

Smaragdi Benetou
Electrical and Computer Engineering
National and Technical University of Athens
Athens, Greece
el18048@mail.ntua.gr

Theoni Maria Strati
Electrical and Computer Engineering
National and Technical University of Athens
Athens, Greece
el18186@mail.ntua.gr

Margarita Eleni Tsarmpopoulou
Electrical and Computer Engineering
National and Technical University of Athens
Athens, Greece
el18848@mail.ntua.gr

Abstract— The purpose of this paper is to develop a recommendation (collaborative filtering) model for books using the Book-Crossing dataset. A recommender system seeks to predict the "rating" or "preference" a user would give an item. A class of techniques used to formulate recommendations are called Collaborative filtering techniques. The most common collaborative filtering techniques involve using a correlation matrix based on user actions, associating each user with each object. A big problem for this class is scalability as on large datasets the correlation matrix will have 10^7 or more cells. To overcome this kind of problem there are techniques to reduce the dimensions of the correlation matrix.

Keywords—recommender, collaborative, filtering, model, books

I. Introduction

Recommendation systems have become necessary nowadays, as the amount of data available on the internet has increased and multiple users expect personalized recommendations. Collaborative filtering is a method used in recommendation systems to make predictions or recommendations about user preferences for items, based on the preferences of similar users. The users who have similar tastes and preferences in the past are likely to have similar tastes and preferences in the future. In this paper, we are going to apply collaborative filtering in a book recommendation system, using three datasets of user ratings and other information. We will also investigate the impact of various factors, such as the number of books or users included in the analysis.

II. Recommender Systems Overview

Collaborative filtering according to [2] is a type of recommender system using users' previous behavior to predict their preferences. Some of the types of recommender systems are: collaborative filtering, content based, hybrid models, knowledge based, and content-aware.

Collaborative filtering uses either a model-based approach or a neighborhood/memory-based approach. The first takes advantage of the advances in pattern recognition and machine learning by developing models with unsupervised learning which create clusters and then predict results.. On the other hand, the neighborhood approach calculates with various techniques the similarity magnitude between users or items (user-base CF or item based CF) and predicting based on that. This similarity magnitude depends only on users' ratings and not features of items. **Content based filtering** is the technique that takes into account the content and characteristics of the items and recommends to a user the items most similar to their chosen favorite characteristics or to their higher rated items. **Hybrid filtering** fuses these two techniques taking into account both ratings and content.

III. CF with Neighborhood/Memory-Based approach

A. Similarity Metrics

There are various item-based similarity metrics as mentioned in [3] including:

- Cosine similarity:

$$sim_{\cos}(i,j) = \frac{i \cdot j}{\|i\|^2 \| j\|^2}$$

- Adjusted Cosine similarity:

$$sim_{adj\cos}(i,j) = \frac{\sum_{u\in U}(R_{u,i}-\bar{R_u})(R_{u,j}-\bar{R_u})}{\sqrt{\sum_{u\in U}(R_{u,i}-\bar{R_u})^2}\sqrt{\sum_{u\in U}(R_{u,j}-\bar{R_u})^2}}$$

  where R_u is the mean rating of a user, so this metric normalizes the ratings of a user removing bias.

- Pearson correlation:

$$sim_{adj\cos}(i,j) = \frac{\sum_{u\in U}(R_{u,i}-\bar{R_i})(R_{u,j}-\bar{R_j})}{\sqrt{\sum_{u\in U}(R_{u,i}-\bar{R_i})^2}\sqrt{\sum_{u\in U}(R_{u,j}-\bar{R_j})^2}}$$

  instead of removing bias of rating by subtracting a user's mean rating, this metric subtracts the mean rating of all users on that item.

- Spearman Correlation:

$$sim_{adj\cos}(i,j) = \frac{\sum_{u\in U}(k_{u,i}-\bar{k_i})(k_{u,j}-\bar{k_j})}{\sqrt{\sum_{u\in U}(k_{u,i}-\bar{k_i})^2}\sqrt{\sum_{u\in U}(k_{u,j}-\bar{k_j})^2}}$$

  instead of using the rating value, this metric uses the rank a user gives to the item and the mean rank of the item amongst others.

- Euclidean Distance:

$$sim_{adj\cos}(i,j) = \frac{\sum_{u\in U}(k_{u,i}-\bar{k_i})(k_{u,j}-\bar{k_j})}{\sqrt{\sum_{u\in U}(k_{u,i}-\bar{k_i})^2}\sqrt{\sum_{u\in U}(k_{u,j}-\bar{k_j})^2}}$$

- Mean Squared Distance:

$$sim_{adj\cos}(i,j) = \frac{\sum_{u\in U}(k_{u,i}-\bar{k_i})(k_{u,j}-\bar{k_j})}{\sqrt{\sum_{u\in U}(k_{u,i}-\bar{k_i})^2}\sqrt{\sum_{u\in U}(k_{u,j}-\bar{k_j})^2}}$$

Similarity between two users can be calculated with the metrics above by using u as an item variable and i,j as two users.

B. Prediction Approaches

After calculating the item similarity matrices the recommendation prediction can be calculated with various methods according to [3] such as:

- Weighted Average:

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u(i)} sim(i,j) r_{uj}}{\sum_{j \in N_u(i)} |sim(i,j)|}$$

  is the weighted average of ratings on other items by the specific user with weights being the similarity of the target item and the other items.

- Mean centering:

$$\hat{r}_{ui} = \bar{r_i} + \frac{\sum_{j \in N_u(i)} sim(i,j)(r_{uj} - \bar{r_j})}{\sum_{j \in N_u(i)} |sim(i,j)|}$$

  is the weighted average of ratings on other items by the specific user with the same weights as in weighted average, but here the mean rating of each movie is subtracted from its rating by the user and the predicted rating is the weighted average plus the mean rating of the target movie.

C. Performance Metrics

On the test set we can evaluate the correctness of the prediction using metrics such as:

- Mean absolute error:

$$Error_{mae} = \frac{\sum_{i=1}^{N} |r_i - \hat{r_i}|}{N}$$

- Root mean squared error:

$$Error_{rmse} = \sqrt{\frac{\sum_{i=1}^{N} (r_i - \hat{r_i})^2}{N}}$$

IV. CF with Model-Based approach

Since the number of items in a big dataset is enormous and a user has rated only a small fraction of them, the user-item matrix has many empty cells. Thus the model-based approach aims to reduce the dimensions of this matrix and compact the amount of information by factorizing it as proposed in [4]. One of the many methods of matrix factorization is SVD and others include PCA, NMF, Autoencoders etc. Of the two matrices that occur by factorization, one is used to reduce the dimension of items in categories of similarity and the other to reduce the dimension of users. The kth rank of a matrix A is calculated with this formula [6] :

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

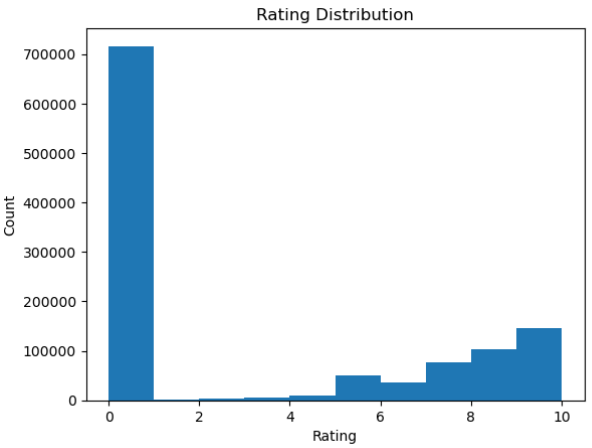
V. Implementation choices and details

A. Data Visualization and Statistics

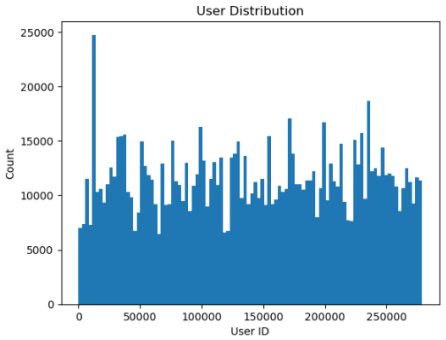
We explore the Book-Crossing Dataset [1] using data visualization and statistics. We start by reading the dataset files for users, ratings, books, and preprocessed data. We then print out some basic information about the dataset, such as the number of books, users, and ratings, which are 271379, 278858 and 1149780 respectively.

We also print out some summary statistics for the ratings data.

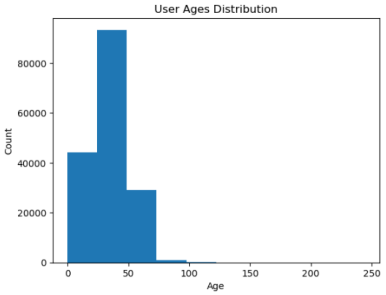
We then proceed to visualize the dataset using various plots. Firstly, we plot the distribution of book ratings, which shows that the majority of ratings are in the range of 0-10.



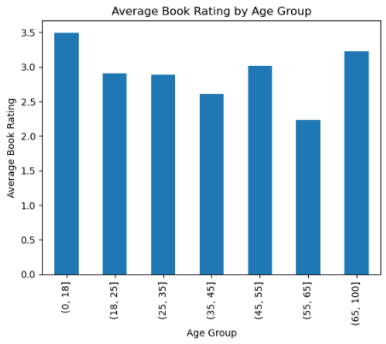
We also plot the distribution of users, which shows that the number of users who rated books varies widely.



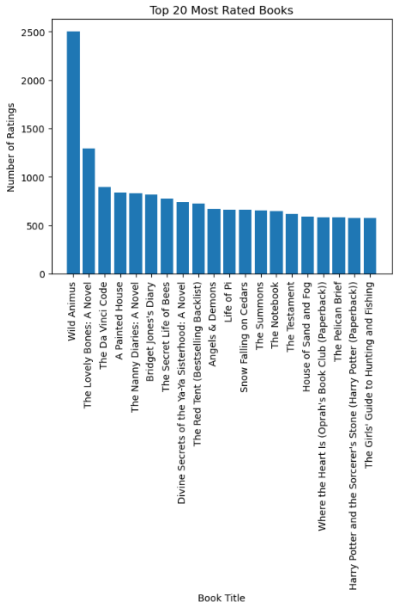
We then plot the distribution of user ages, which shows that most users are in the age range of 20-40.



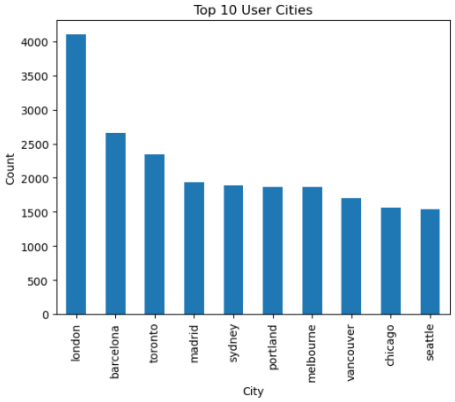
Next, we investigate the distribution of book ratings by age group, which reveals that the youngest and oldest age groups tend to rate books more positively than the middle aged groups.

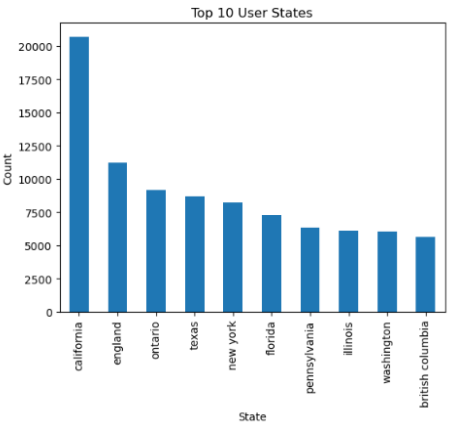


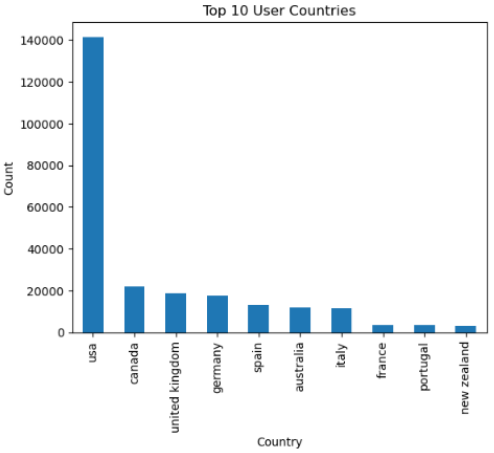
We also investigate the distribution of book ratings by author, which reveals the top 20 most rated authors.



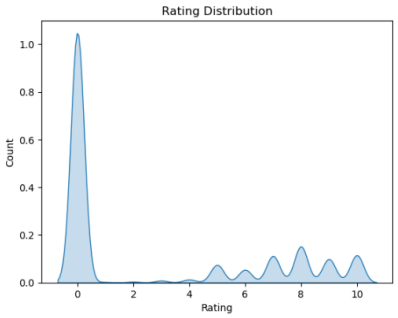
Further, we investigate the distribution of users by city, state, and country, which shows the top 10 user cities, states, and countries.







Next, we use a KDE plot to visualize the rating distribution and a scatter plot to visualize the relationship between book rating and user age.

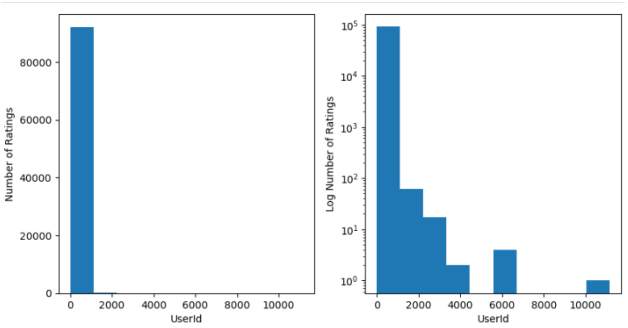


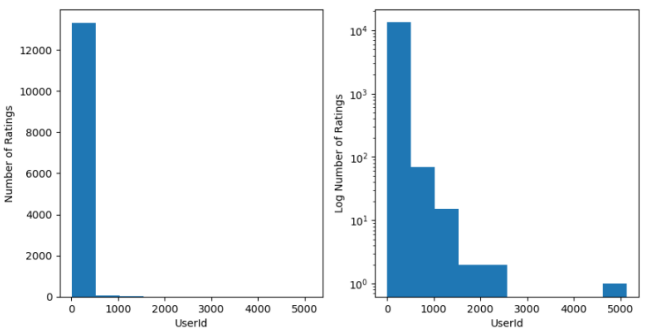
Overall, the data visualization and statistics provide valuable insights into the Book-Crossing Dataset and help us understand the patterns and trends in the data.

B. *Data Preprocessing*

The section on Data Preprocessing involves cleaning and preparing the data for analysis. The first step was using the most significant categories of the dataset, such as User-IDs, ISBNs and Book Ratings.

To better understand the data, the number of ratings for each user and book is computed. Histograms are then plotted to visualize the distribution of ratings across users and books.





Determining the minimum number of ratings is an important step in preparing the data for analysis. In order to do this, we calculated the mean and median number of ratings per user and per object. The mean number of ratings per user was found to be 11.20, while the median number of ratings per user was only 1.00. Similarly, the mean number of ratings per object was 3.82, and the median number of ratings per object was 1.00.

These results suggest that a large portion of the data may consist of users and objects with only a few ratings. Therefore, using a minimum number of ratings as a threshold for inclusion in the analysis can help ensure that only the most reliable and relevant data is used.

Based on the mean and median number of ratings per user and per object, the decision is made to filter out books with less than 10 ratings and users having rated less than 5 times. We don't choose 11 minimum number of ratings per user because we may exclude a large proportion of users from our analysis. We also don't choose 4 minimum number of ratings per object because it could result in a sparse dataset. Both of those options could lead to biased results.

Two pivot tables are created with the book and user ratings preprocessed data, filled with zero instead of missing values and calculating the normalized values in order to remove bias and improve dataset quality.

Finally, the data is normalized by subtracting the mean rating from each rating in the pivot tables. The number of ratings for each user and book is computed again to plot histograms in order to visualize the new distribution of ratings across users and books and compare to the preprocessed one.

C. *Definition of training/test sets*

In order to evaluate the performance of the recommendation system, we need to split the data into training and test sets. The training set is used to fit the model, while the test set is used to evaluate its performance. First, we shuffled the data and then we randomly split our preprocessed data into 80% training and 20% test sets. Additionally, we used K-Fold Cross Validation technique to check the consistency of the model by splitting the data into multiple subsets or folds (in this case, 5 folds) and using one fold as the test set and the remaining folds as the training set.

D. *Collaborative Filtering Model*

Due to the large amount of data SVD is applied on the pivot table created by the users as indexes and the books as columns [5]. Predictions were generated with the weighted average prediction approach and both inner product and cosine similarity for the calculation of the similarity between two users on the reduced matrix. The mean squared error evaluation metric was used on the test set predictions and targets.

The cluster used consisted of two machines of 4 CPUs each and 8GB RAM. On the cluster Apache Spark was installed and ran alongside Yarn with HDFS distributed file system. Using PySpark the data was loaded o a dataframe and SVD was calculated on the pivot matrix using MLLib.

REFERENCES

[1] http://www2.informatik.uni-freiburg.de/~cziegler/BX/
[2] https://www.iteratorshq.com/blog/collaborative-filtering-in-recommender-systems/
[3] Collaborative Filtering in Recommender Systems: Technicalities, Challenges, Applications, and Research Trends
[4] https://realpython.com/build-recommendation-engine-collaborative-filtering/
[5] https://jaketae.github.io/study/svd/
[6] https://web.stanford.edu/class/cs168/l/l9.pdf