American University of Armenia

Zaven & Sonia Akian College of Science and Engineering

CS 251/340, Machine Learning Course Project

Book Recommendations System

Team: Lilit Yerknapeshyan

Spring, 2022

Abstract

In this project I tried to implement a book recommender system. The system was coded using a dataset of users and their ratings from goodreads.com. The book recommender system uses the collaborative filtering approach. When *User A* wants to be recommended *n* books, the system, based on item-item/user-user similarity recommends a new book. I also made it possible to narrow the recommendation list further by providing the desired category and author.

Keywords: Recommender System, Collaborative Filtering, User-based, Similarity

Contents

Chapter 1

Introduction

1.1 Problem Setting, Project Motivation and Description

The goal of this project is firstly to create a book recommender system. I think, while a lot of datasets for movies (Netflix, Movielens) or songs have been explored previously to understand how recommendation engine works for those applications and what are the scopes of future improvement, book recommendation engines have been relatively less explored. So, this is a good opportunity to analyse books dataset, try to find useful patterns for book recommendations.

Recommendation systems are a heavily researched topic, and we stumble upon them daily. Netflix, Spotify, Amazon all use some technique to recommend their users new content or product. Recommender systems are critical in these industries as they can generate a huge amount of income when they are efficient or also be a way to stand out significantly from competitors. As a proof of the importance of recommender systems, a few years ago, Netflix organised a challenge (the "Netflix prize") where the goal was to produce a recommender system that performs better than its own algorithm with a prize of 1 million dollars to win.

Broadly speaking there are two types of recommenders: content-based and collaborative. There also exists a third one which is a hybrid of the previous two. Collaborative methods for recommender systems are methods that are based solely on the past interactions recorded between users and items in order to produce new recommendations. Unlike collaborative methods that only rely on the user-item interactions, content-based approaches use additional information about users and/or items. If we consider the example of a movies recommender system, this additional information can be, for example, the age, the sex, the job or any other personal information for users as well as the category, the author of the book (other feature of the item being recommended).

For each of these methods there are multitude of algorithms and mathematical models to use: kNN, Matrix Factorization, Classification, Regression.

I chose this as my project, as I think building a recommender system is a productive way to learn about this important topic that was not covered during our class. This also introduced me to different programming and mathematical models that I did not know of.

In the system that I implemented collaborative filtering is used. Mainly I compared baseline algorithms, neighbourhood methods and matrix factorization methods. Chose the one that worked best with my dataset. To narrow down the recommendation list, I added the possibility to specify the category of the book. As a result, non-personalized

recommender function was also written, which can recommend a book based on the desired category to a new user with no history.

1.1.1 The Structure of the Project Paper

This Project Paper consists of 5 sections: Introduction, Data Pre-processing, Performance Measurement, Algorithms and Models, Results and Conclusion, Appendices. In the 2nd section I will describe the dataset I used, what kind of processing and analysis I performed on that dataset and what was the final look of my data. I will also give what performance metrics will be used to evaluate whether the chosen models work well or not. In the 3rd section, I will talk about the algorithms and models that I have tested and used for my project and mention which one was chosen based on the performance metrics given in the second section. In the 4th section, I will describe different experiments I have done with the dataset, the models and I will show the obtained results. In the 5th section, I will leave some theoretical knowledge behind the models used in the code and also the link to my project code.

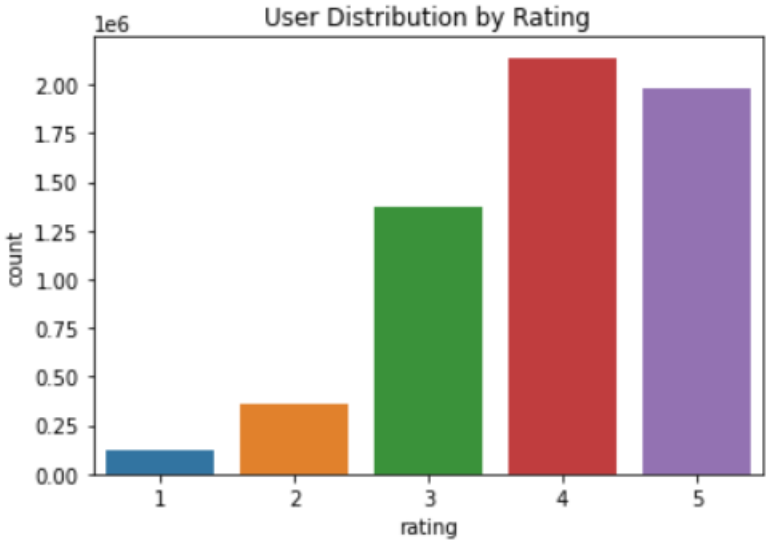
Chapter 2

Data and Pre-processing, Performance Measurement

2.1 Dataset

My dataset consisted of five csv files:

1. ***ratings.csv*** - contains user_ids, book_ids and ratings. It has almost 6,000,000 observations.



2. ***books.csv*** - has metadata for each book (goodreads IDs, authors, title, original publishing year, ratings count, number of ratings from 1 to 5, etc.). The raw dataset has 23 columns and 10,000 entries.

3. ***book_tags.csv*** - contains tags/shelves/genres assigned by users to books. Tags in this file are represented by their IDs. The file has 999912 observations

4. ***tags.csv*** - translates tag IDs to names. The file contains 34252 observations

5. ***to_read.csv*** - provides IDs of the books marked "to read" by each user, as user_id, book_id pairs.

For my project I used the first four, so we can disregard the to_read.csv.

2.2 Feature Selection and Pre-processing

Let's go through the dataset one by one.

1. For ratings.csv dataset I visualised the data by computing empirical cumulative distribution function. I did this for ratings per each user and ratings per each book. I filtered the books that have received less than 3000 ratings and users that have rated less than 50 books. As a result, I have smaller dataset that can be used for creating predictions.
2. I combine the rest of the datasets: books, book tags and tags. The resulting data frame looks like this.

	goodreads_book_id	tag_id	count	tag_name	title
0	1	30574	167697	to-read	Harry Potter and the Half-Blood Prince (Harry ...
1	1	11305	37174	fantasy	Harry Potter and the Half-Blood Prince (Harry ...
2	1	11557	34173	favorites	Harry Potter and the Half-Blood Prince (Harry ...
3	1	8717	12986	currently-reading	Harry Potter and the Half-Blood Prince (Harry ...
4	1	33114	12716	young-adult	Harry Potter and the Half-Blood Prince (Harry ...

3. I clean the tag table by trying to combine the tag names that have the same meaning (different spelling for example, f.e. ya, young adult, etc). I obtain the following categories: Favorites, Children's book, Fiction, Non-Fiction, Young-Adult (with different genres), Science, History, Crime & Mystery, Audio Books, Ebooks. This also minimizes the dataset and helps with better visualization.
4. As I am using mainly collaborative filtering, the features I user are user id, book id and ratings. I also use categories for further filtering.

2.3 Performance Measurement

I took first 100000 samples of my cleaned ratings data and tested several models on it. I have also done grid search for some models. In the end, I chose a model that based on RMSE score. The model with the smallest score was chosen.

Chapter 3

Algorithms and Models

3.1 Algorithms and Models

In the end, I used matrix factorization algorithm called Singular Value Decomposition. As it performed the best based on RMSE score. It predicts the rating for unrated items, based on the following equation:

$$r_{ui} = \mu + b_u + b_i + q_i^T p_u$$

If user u is unknown, then the bias b_u and the factors p_u are assumed to be zero. The same applies for item i with b_i and q_i.

The following parameters were chosen:

1. n_factors – The number of factors. 20
2. n_epochs – The number of iteration of the SGD procedure. 50
3. biased (bool) – Whether to use baselines (or biases). True
4. lr_all – The learning rate for all parameters. 0.005.
5. reg_all – The regularization term for all parameters. 0.05.

Then, I wrote a recommend function, which takes user_id, category, and number of recommended books as parameters. Using the predicted ratings dataset, the categories data frame I recommend books that the user is more likely to rate higher.

Chapter 4

Results and Conclusion

4.1 Experiments and Results

I tried out several the following models:

	Model	Paramters
3	BaselineOnly	-
4	SVD	n_factors=20, n_epochs = 30, biased=False
0	KNNWithMeans	k=40, name: cosine, user_based: True,min_suppo...
1	KNNBasic	k=40, name: cosine, user_based: True,min_suppo...
5	NMF	n_factors=20, n_epochs = 30, biased = True
2	Normal Predictor	-

I also tried Grid Search for SVD and NMF. The following parameter values were given:

```
parameters_svd = {'n_factors': [20,30,50], 'n_epochs': [50,100,200], 'lr_all': [0.005],'reg_all': [0.05], 'biased': [True, False]}
```

```
parameters_nmf = {'n_factors': [20,30,50], 'n_epochs': [20, 30,40,50],'biased': [False, True]}
```

Below are the RMSE results of different models without grid search and with grid search respectively:

	Model	Paramters	RMSE
3	BaselineOnly	-	0.912593
4	SVD	n_factors=20, n_epochs = 30, biased=False	0.927444
0	KNNWithMeans	k=40, name: cosine, user_based: True,min_suppo...	0.932154
1	KNNBasic	k=40, name: cosine, user_based: True,min_suppo...	0.932154
5	NMF	n_factors=20, n_epochs = 30, biased = True	1.292027
2	Normal Predictor	-	1.418263

	Model	Paramters	RMSE
0	SVD	{'rmse': {'n_factors': 20, 'n_epochs': 50, 'lr...	0.91003
1	NMF	{'rmse': {'n_factors': 20, 'n_epochs': 50, 'bi...	0.98869

4.2 Conclusion

I built a collaborative recommender system using SVD. I checked the RMSE scores on different models, training them on the 100000 samples of my ratings dataset. Chose the best one, which was SVD with the parameters shown above. Trained the model on the 60 percent of the rest of the data and tested it on the remaining 40 percent. Predictions were turned into a dataset and used in the recommend function. Recommend function finds the user from the predictions dataset and recommends top n books with the highest ratings. It is also possible to have further filtering based on category.

4.3 Further Work that can be done etc

The recommend function needs a lot of improvement. Because my prediction dataset has only 40 percent of the users' predicted ratings, often there are not enough books for recommending to the user, especially if we are also filtering based on category. Category search can also be improved, because of we do not write a word that exactly matches at least one category, the program thinks that no books with such category exists.

Chapter 5

Appendices

5.1 A. Mathematical Miscellany

In this section I included further mathematical explanation of the SVD model. Recall that the equation used to predict the rating was the following:

$$r_{ui\,pred} = \mu + b_u + b_i + q_i^T q_u$$

To estimate the unknown, we minimize the following equation:

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - r_{ui\,pred})^2 + \lambda({b_i}^2 + {b_u}^2 + \|q_i\|^2 + \|p_u\|^2)$$

The minimization is performed by a stochastic gradient descent:

$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda b_u)$$
$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda_{bi})$$
$$p_u \leftarrow p_u + \gamma(e_{ui} q_i - \lambda p_u)$$
$$p_i \leftarrow p_i + \gamma(e_{ui} p_u - \lambda q_i)$$

where,

$$e_{ui} = r_{ui} - r_{ui\,pred}$$

5.2 B. Algorithm Implementation in Python

The code was sent via email. The packages needed are numpy, pandas, seaborn, matplotlib.pyplot, scikit-surprise package.