Book Recommender

Book recommendation system using Deep Learning

Adelina Kildeeva
B21-DS-02
Innopolis University
a.kildeeva@innopolis.university

Oksana Konovalova
B21-DS-02
Innopolis University
o.konovalova@innopolis.university

1. Project topic

Recommendation systems can be found in almost every area of life, from shopping on marketplaces to choosing a TV show for the weekend. Such systems make it possible to assess user preferences and offer them good options. And while there are a large number of different sites where users can discover new songs or films, it is quite difficult to find quality book recommendations. We wanted to study the topic of recommendations in more detail, understand the work of various methods and create our own recommendation system for books. The goal of our project is to study and develop a hybrid book recommendation system.

This report describes the process of studying the topic and developing the project. The remainder of the report organized as follows. Section 2 contains a structure description of the GitHub repository and link to it. Sections 3 and 4 provide the dataset description and some findings from the data. Section 5 proposes the detailed description of each point of project development. Section 6 presents the main results of the project, model evaluations, and artifacts. Finally, sections 7 and 8 consist of the project timeline and individual contributions of each team members.

2. Link to the GitHub repository

The GitHub repository has the following structure: README file, data (raw and interim), models, jupyter notebooks with the main points of the project, technical report, references, and the telegram bot with book recommendations. The README file shortly explains the project flow and how to use the recommender system. Data folder contains the original dataset and transformed data (train/test data). Models folder contains pre-trained model for user-based recommendations.

Link to the GitHub repository: https://github.com/ksko02/book_recommender/tree/main

3. Dataset description

To train and evaluate our systems, we used the open goodbooks-10k dataset [1], consisting of 10000 books, 53424 users and more than 6 million ratings. The dataset contains 3 files: ratings.csv with book ratings from various users, books.csv with books metadata, to_read.csv with books marked “to read” by each user. After analyzing the data, it was decided to use only ratings.csv and books.csv files. The data is clean and prepared for work.

4. Data Analysis

Data analysis for each file in the dataset was carried out to determine the evaluation metrics of our system and possible problems in the future.

4.1. Ratings.csv

Data is cleaned, each user rated from 8 to 200 books. In average, each user rated 112 books. Books have different number of ratings (from 8 to 22806).

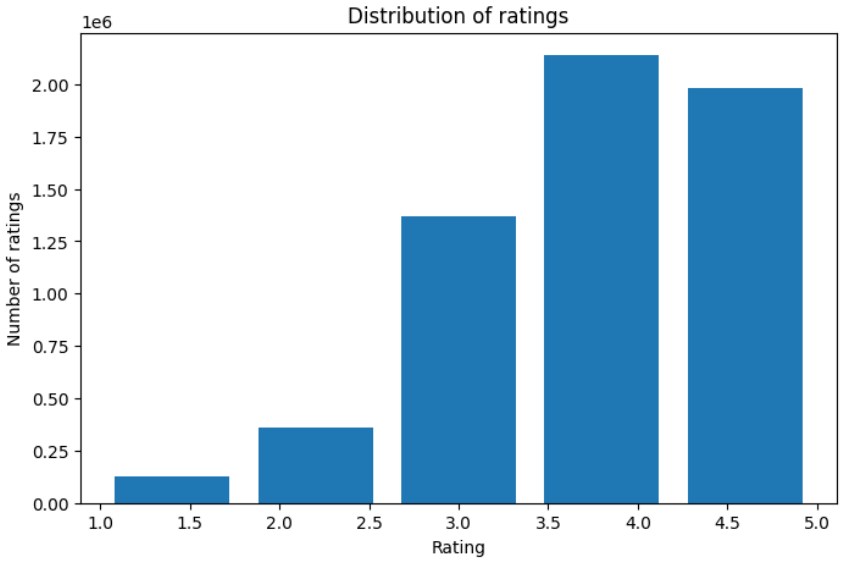

Figure 1.

Most ratings range from 4 to 5 as can be seen from figure 1.

4.2. Books.csv

Some books do not have original publication year. Moreover, null values can be found in isbn, original_title, and language_code columns but they will not be used in our project.

There are authors with a large number of books in the dataset. Additionally, there can be multiple entries of the specific books. For example, the green mile has 7 entries: the full book and 6 books representing each part of the book. Some book series as Harry Potter or the Hunger games have "boxset" entry with all book parts. These features of the dataset can affect the work of the content-based model because the model will recommend the most similar books and may produce recommendations with books of the same author or already read books in other representations.

Further we will work with book_id, author, original_publication_year, title, average_rating, and image_url columns.

4.3. To_read.csv

The data from to_read.csv can be used to evaluate the precision@k and recall@k metrics. However, each user has different number of books marked as "to read". More than 5000 users have not marked any book. The range of marked books for each user is quite large (from 1 to 117). Such scattering of data and the presence of users without marked books greatly influence the quality of precision and recall metrics. It was decided not to use this data to evaluate the models.

5. Methodology

5.0. Metrics for evaluation

To measure recommender system performance, we chose the rooted mean squared error (RMSE) metric.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

RMSE is typically used to evaluate regression problems where the output (a predicted scalar value) is compared with the true scalar value output for a given data point. We aim to create systems that predict a user's rating for a particular book, therefore, this metric will allow to evaluate the performance of the system.

The main disadvantage of the RMSE metric is that RMSE might penalize a method that does well for high ratings and badly for others. Since we are primarily interested in predicting good scores, for some models we separately estimated RMSE on “high-rated” data containing only 4 and 5 rates.

5.1. User-based Collaborative Filtering

At the very beginning of the project development, we created a simple collaborative filtering system. User-Based Collaborative Filtering is a recommendation system technology that suggests products based on the preferences of users similar to the target user. The solution is presented in the form of an algorithm that calculates cosine similarity between users and based on this predicts whether a user will like a particular book or not.

The algorithm is trained on a small amount of data, since matrix calculation takes a lot of memory. The RMSE of this system is 3.536. The main goal of this model was to understand how the recommendation system works, so we further focused on creating better systems.

5.2. Content-based Recommendation System

Content-based Recommender System try to recommend items that are similar to those that a user liked in the past. To build this system we used books metadata and TF-IDF vectorizer to compute cosine similarity between books.

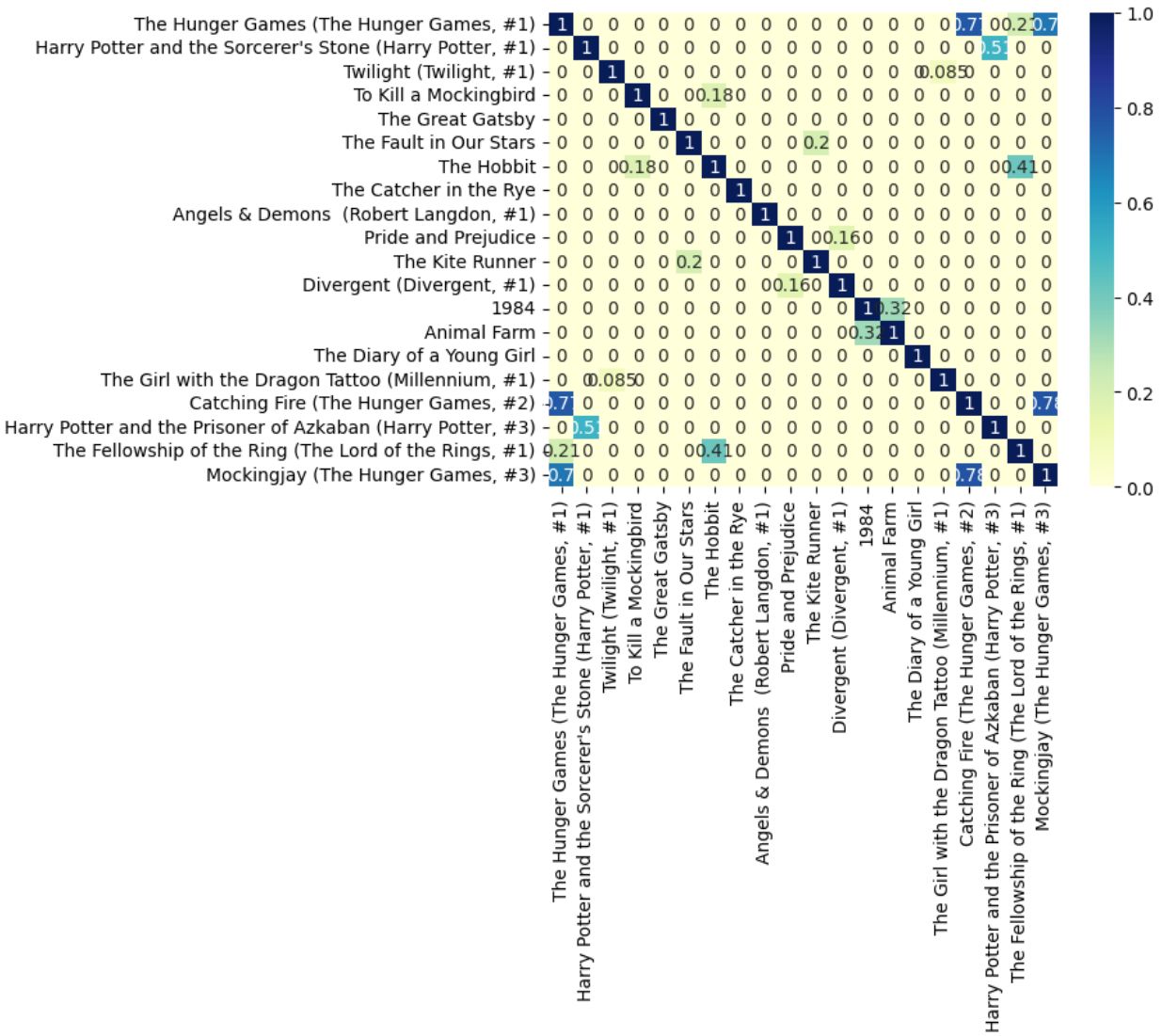

Figure 2. 20x20 book similarity matrix

Figure 2 represents part of the similarity matrix. We can see the correlation between the book series (the Hunger Games, Harry Potter, The Lord of the Rings) and the books of the same author (1984 and Animal Farm).

To compute the predicted rating for book m we use user's ratings for book 1, …, n, similarity matrix and combine them using a formula,

$$R_m = \frac{\sum_{i=1}^{n} S_{i,m} \cdot R_i}{\sum_{i=1}^{n} S_{i,m}} \quad if \sum_{i=1}^{n} S_{i,m} > \beta \; else \; 3$$

Where $S_{i,m}$ is the similarity between books i and m, R_i is the rating of the book i given by user, and β is the hyperparameter to tune (threshold of the similarities). If the sum of book similarity coefficients is less than β, we consider that there is not enough data to evaluate the book and give it an average of 3 since the user can potentially both like and dislike the book.

5.3. User-based Collaborative Filtering using Deep Learning

We implemented user-based collaborative filtering using Deep Learning [2]. The architecture of the neural network is a type of collaborative filtering model commonly used in recommendation systems. Specifically, it's an example of a matrix factorization model. Figure 3 shows the breakdown of its components.

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
book_input (InputLayer)	[(None, 1)]	0	[]
user_input (InputLayer)	[(None, 1)]	0	[]
embedding_1 (Embedding)	(None, 1, 10)	100010	['book_input[0][0]']
embedding (Embedding)	(None, 1, 10)	534250	['user_input[0][0]']
flatten_1 (Flatten)	(None, 10)	0	['embedding_1[0][0]']
flatten (Flatten)	(None, 10)	0	['embedding[0][0]']
concatenate (Concatenate)	(None, 20)	0	['flatten_1[0][0]', 'flatten[0][0]']
dense (Dense)	(None, 128)	2688	['concatenate[0][0]']
dense_1 (Dense)	(None, 32)	4128	['dense[0][0]']
dense_2 (Dense)	(None, 1)	33	['dense_1[0][0]']

Total params: 641109 (2.45 MB)
Trainable params: 641109 (2.45 MB)
Non-trainable params: 0 (0.00 Byte)

Figure 3. The architecture of the model

The parameters of the model are learned during training, and the total number of trainable parameters is 641,109. In collaborative filtering models like this, the idea is to learn latent representations (embedding vectors) for both users and items (books in this case). These learned representations are then used to predict user preferences or ratings for items they haven't interacted with before.

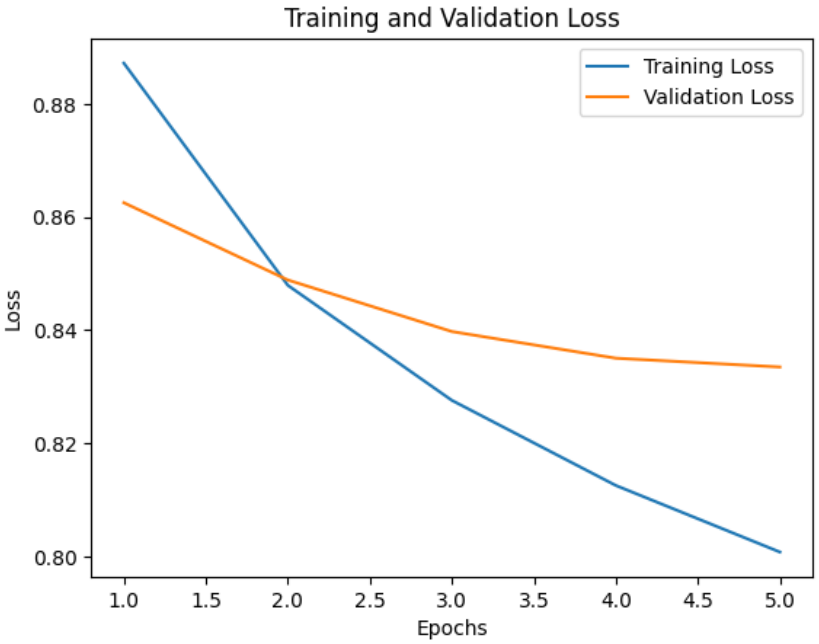

Figure 4.

To predict a new user's rating, we find the three most similar users and take their average ratings predicted by the model.

5.4. Hybrid Recommendation System

In order for the recommendation system to consider both the ratings of similar users and similar books, we combined the two previous models into one hybrid one.

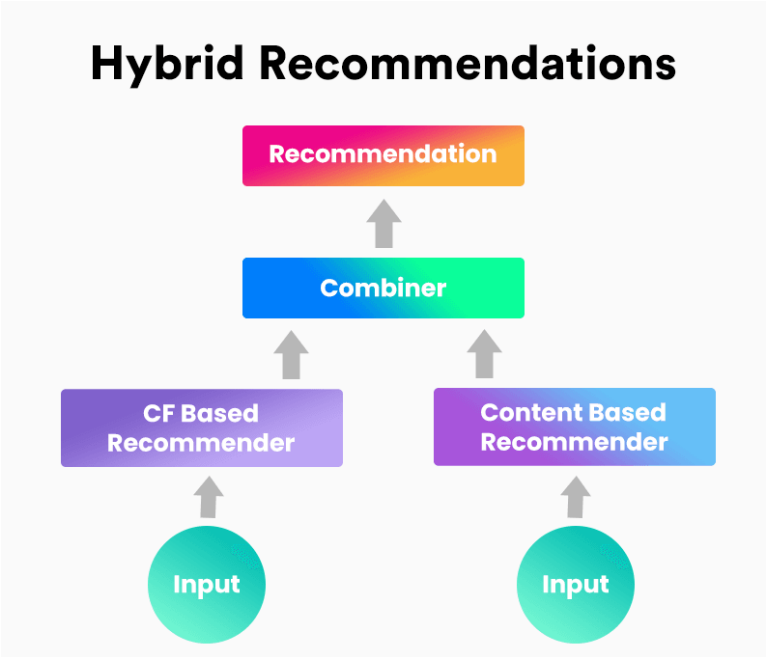

Figure 5.

For predicting the rating of the specific books, the following formula is used,

$$R_n = \alpha \cdot CF_n + (1 - \alpha) \cdot CB_n$$

Where R_n is the rating of book n, CB_n is the rating predicted by content-based recommender for the book n, CF_n is the rating predicted by collaborative filtering model for the book n, and α is hyperparameter to tune.

5.5. Fine-Tuning of the Hybrid System

The hybrid system has two hyperparameters to tune: α (coefficient in the weighted hybrid equation) and β (threshold of the similarities). We have chosen the following values for the tuning: α in [0.3, 0.4, 0.5, 0.6, 0.7, 0.8] and β in [0, 0.1, 0.25, 0.4]. For evaluation the RMSE metric is used.

alpha	beta	RMSE
0,4	0	0,87772
0,4	0,1	0,87325
0,4	0,25	0,89546
0,4	0,4	0,92522
0,5	0	0,88157
0,5	0,1	0,87779
0,5	0,25	0,89595
0,5	0,4	0,92022
0,6	0	0,89213
0,6	0,1	0,88909
0,6	0,25	0,90323
0,6	0,4	0,92208
0,7	0	0,90917
0,7	0,1	0,90689
0,7	0,25	0,91714
0,7	0,4	0,93075

Table 1. Results of the fine-tuning

Table 1 shows all possible configurations and their evaluations. The best obtained results were for $\alpha = 0.4$ and $\beta = 0.1$.

5.6. Telegram-bot for book recommendations

In order for people to use our recommendation system, we decided to create a telegram bot. The bot was written using the telebot library. Using the bot, the user can select from 3 to 15 books, rate them on a scale from 1 to 5 and receive personal recommendations from our system. For the telegram bot, we trained a hybrid model on all data in the dataset. The model is not fully optimized and requires about a minute of waiting. A brief overview of the bot work is shown in the demo in section 6.

6. Main Results

During the project, we studied the topic of recommender systems and their evaluations, found a dataset, analyzed the data, built various recommendation systems, including the hybrid model consisting of both content-based and user-based collaborative filtering models, selected the best parameters and wrote a telegram bot to use the final recommendation system.

	Simple CF	Content-based system	User-based CF using DL	Hybrid model	Fine-tuned hybrid model
RMSE	3.53569	0.95774	0.98924	0.89941	**0.87325**
RMSE for high-rated books	—	0.89935	0.81974	0.76834	**0.74349**

Table 2. Evaluations of the systems

Table 2 contains a comparison of RMSE metric for all created recommender systems. The best performance was showed by the fine-tuned hybrid model.

The telegram bot work is showed in the demo:

https://drive.google.com/file/d/16WNmHHeeOHjvKeEKwjL5OXO3IxJCNzvT/view?usp=sharing

7. Timeline of the project

16-17th September — topic selection, research, dataset selection

9th October — data analysis

20-21st November — simple CF, choosing metrics for evaluations

24-26th November — various systems development: content-based, user-based CF using DL, hybrid model

27-28th November — fine-tuning of the final model, telegram bot development, writing final technical report, testing

8. Individual Contributions of the teammates

Adelina Kildeeva — topic research, data analysis, content-based model, telegram bot, final report

Oksana Konovalova — simple CF, user-based CF using DL, hybrid model, fine-tuning, evaluations

9. References

[1] Goodbooks-10k dataset: https://github.com/zygmuntz/goodbooks-10k

[2] Building a book Recommendation System using Keras, Gilbert Tanner https://gilberttanner.com/blog/building-a-book-recommendation-system-usingkeras/