STA5073Z - Recommender Systems

Nicholas A Limbert

Introduction

Recommender systems have become an important part of modern digital platforms, enhancing user experience by delivering personalized suggestions for items such as books, movies, products, and more. The increasing volume of data and platforms available today has made it essential for companies to provide comprehensive and tailored views to users in order to improve user engagement and satisfaction. We will focus on the development of an ensemble recommender system designed to suggest books to new and current users based on their past ratings and similarities to other users.

The analysis will utilize a dataset from the Book-Crossing (Arashnic 2021) community, which includes over 1.1 million ratings from approximately 279,000 users for more than 271,000 books.

We aim to implement three distinct recommendation methodologies: item-based collaborative filtering, user-based collaborative filtering, and matrix factorization techniques. Subsequently, an ensemble recommender system that incorporates predication's from all three implementations will be constructed to provide a more tailored model. By evaluating these approaches individually and then combining their predictions into an ensemble model, we seek to improve the accuracy of the book recommendations.

At each step we will asses the accuracy of the models. Additional steps such such as k-fold cross-validation will be used to asses matrix factorization. To prevent over fitting, we incorporate regularization into matrix factorization model to help balance the model complexity and generalization.

Prior to model building, basic exploratory data analysis (EDA) is performed to unpack and understand the structure of the data set. This guides the preprocessing steps, for handling missing values, normalizing ratings, and structuring the data. for optimal performance in matrix factorization. Effective preprocessing ensures clean and usable data and ultimately contributing to better model outcomes.

Data Preprocessing and Features

After inspecting the structure of the datasets, a left join merge operation was performed to combine user and book metadata to the ratings records. This merging was crucial for constructing a comprehensive user-item interaction matrix that includes additional information about users and book. Implicit ratings, represented by zero values, were removed to ensure that only explicit ratings were considered. Implciity ratings for recomdeations would hold no value as we can only gauge interest from this measure as no feedback from is received from the users. Additionally, records with missing book titles were excluded from the data set as these are assumed to be invalid ISBN numbers captured in the ratings table.

To reduce the data set size for more efficient processing, two key filters were applied:

- Users with fewer than three ratings were excluded, leaving only the most active users. This leaves only the most active users and provides more substantial information to train the models.
- Books with fewer than ten ratings were excluded, ensuring that only the most popular books, which have enough data to make meaningful predictions are included in the data set

This approach helps eliminate noise from inactive users and unpopular books.

Finally, the data set is transformed into a user-item matrix, where rows represented users, columns represented books, and the matrix cells contained the corresponding book ratings. Missing values (where a user had not rated a specific book) were replaced with zeroes, as they represent no interaction between the user and the book.

Due to performance issues the data set was reduced significantly by further filtering out user rows and book columns with fewer than 10 ratings. This mitigates the impact of sparsity in the data set, ensuring that the recommendations are based on sufficient interaction data. However, it must be noted that we will lose diversity in recommendations, introduce bias from only active users and likely introduce the cold start problem where new users or books that don't meet the threshold won't be included.

Table 1: Sparsity Metrics of the Rating Matrix

Metric	Value
Total Elements	92998338
Non-Zero Elements	114812
Sparsity Proportion	0.00123456
Sparsity Percentage (%)	99.88

The sparsity metrics shown in Table 1.1 of the rating matrix reveal a significant level of emptiness within the data set with only 114,812 of these entries contain non-zero values. With the large number of users and books this type of sparsity is expected in in the data matrix. We attempt to deal with this sparsity by limiting further filtering the data where each row and column should have at least ten results. Table 1.2 shows the results of further filter but does not significantly reduce the sparsity percentage. This sparsity will pose significant challenges when constructing the recommenders.

Table 2: Sparsity Metrics of the Rating Matrix

Metric	Value
Total Elements	9244542
Non-Zero Elements	59004
Sparsity Proportion	0.006382577
Sparsity Percentage (%)	99.36

Figure 1.3 illustrates the distribution of ratings by user from the usable data set with all implicit ratings removed. The distribution is skewed towards higher ratings and suggests that users tend to have a positive view of the books.

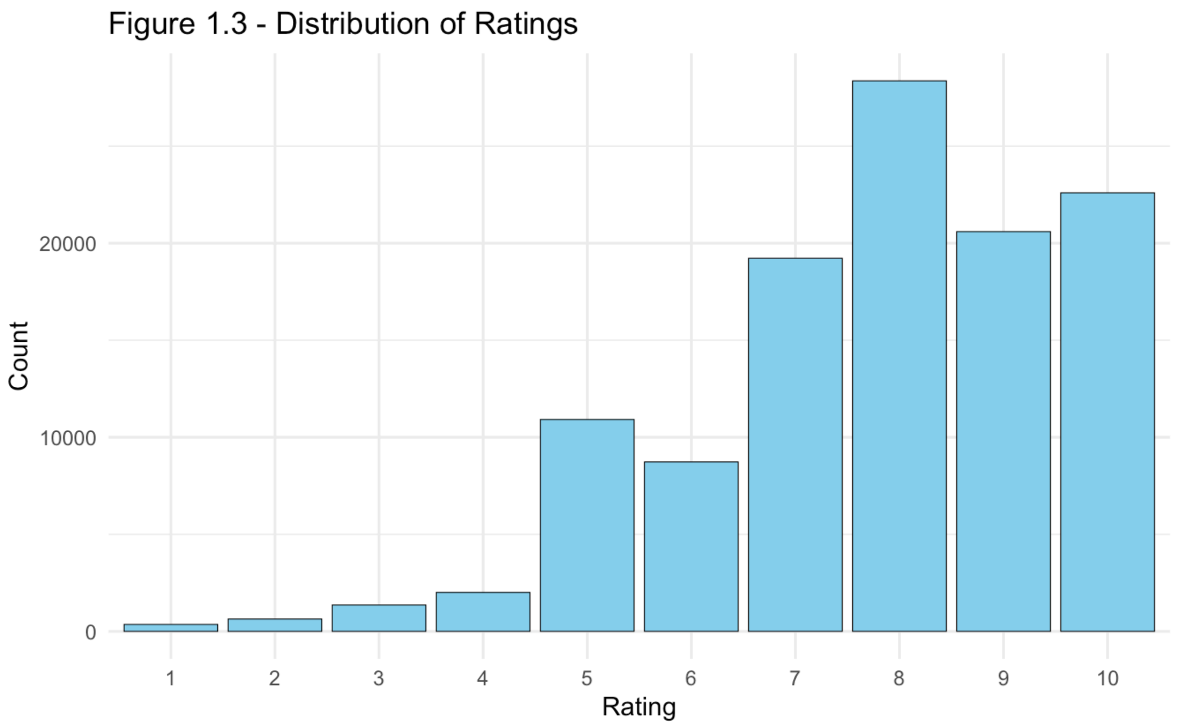

Figure 1: Distribution of Ratings

Methods

Item-based Collaborative Filtering

An item-based collaborative recommendation system is a model that focuses on identifying similarities between items (books) based on the user ratings we received. Item-based methods compare the products themselves, where the core idea is that if two items receive similar ratings from users, they are likely to be similar and could be recommended together. The main advantage of item-based collaborative filtering is that the items (books) generally receive more ratings over time, leading to more consistent similarity measures.

In order to perform item-based CF, we first load and preprocess the ratings data by transposing the user-item rating matrix. Each item's rating vector is normalized using the L2 norm to account for variations in rating as this ensures a consistent scale when computing similarities.

Cosine similarity is utilized as the measure of similarity between items. This is computed through matrix multiplication, where the dot product between item vectors is divided by the product of their norms. The diagonal of the resulting similarity matrix is set to zero since the similarity of an item with itself is not meaningful for recommendation purposes.

We write an *item_based_recommendations* function then generates recommendations for a specific user by calculating the sum of similarities between items the user has rated and other items. Items that have not been rated by the user are sorted by their similarity scores, and the top 3 recommendations are returned. Item based recommendations are illustrated below in Table (**secPlot1?**).

Table 3: Top 3 recommendations for User 243 based on Item CF

book	rating	Book.Title
3596215226	1.343286	Schachnovelle
3423202327	1.343286	MÃ?Â¶rder ohne Gesicht.
3423201509	1.343286	Die Weiss Lowin / Contemporary German Lit

Table 4: Top 3 recommendations for User 1733 based on Item CF

book	rating	Book.Title
0440185327	2.417411	Thurston House
0440214114	2.403642	Mixed Blessings
0440201926	2.378634	Kaleidoscope

Table 5: Top 3 recommendations for User 507 based on Item CF

book	rating	Book.Title
0446527041	1.349675	The Crush
0399149783	1.268641	Monkeewrench
0060542128	1.266967	When the Storm Breaks

User-based Collaborative Filtering

Unlike item-based collaborative filtering, user-based collaborative filtering systems focus on finding similarities between users based on their preferences and ratings. Instead of comparing products, this method identifies users who have rated items in similar ways, under the assumption that users with similar preferences will like similar items. The user-user similarity can be leveraged to make personalized recommendations, suggesting items liked by other users that a given user hasn't yet interacted with.

We follow a similar process to the item-based CF mentioned previously with a few key differences. A function utilized the user similarity matrix to calculate weighted ratings for each book. This function uses a collaborative filtering mechanism, wherein the recommendations were generated based on the ratings provided by users with similar preferences. We prioritized books that the target user had not previously rated and examples are shown in the Table (**secPlot2?**) below.

Table 6: Top 3 recommendations for User 243 based on User CF

book	rating	Book.Title
0316666343	1.2373502	The Lovely Bones: A Novel
0142001740	0.9077407	The Secret Life of Bees
0312195516	0.8410719	The Red Tent (Bestselling Backlist)

Table 7: Top 3 recommendations for User 1733 based on User CF

book	rating	Book.Title
0439064872	1.459780	Harry Potter and the Chamber of Secrets (Book 2)
043935806X	1.245313	Harry Potter and the Order of the Phoenix (Book 5)
0439136369	1.169269	Harry Potter and the Prisoner of Azkaban (Book 3)

Table 8: Top 3 recommendations for User 507 based on User CF

book	rating	Book.Title
0446527041	1.349675	The Crush
0399149783	1.268641	Monkeewrench
0060542128	1.266967	When the Storm Breaks

Matrix Factorization

The matrix factorization approach was implemented using the recosystem package in R. As previously outlined, the data set was preprocessed by filtering users with at least 3 ratings and books with at least 10 ratings in an effort to reduce sparsity The data was split into training (80%) and test (20%) sets.

The recosystem model was tuned using a grid search over dimensions (10, 25, 50) and learning rates (0.1, 0.01), with 20 iterations. The model was then trained both with and without L2 regularization (lambda = 0.1). To assess model performance, Root Mean Square Error (RMSE) was calculated on the test set. Additionally, 5-fold cross-validation was performed to obtain a more robust estimate of model performance. Finally, the model was retrained using the optimal parameters found during tuning.

Table 9: Optimal Parameters for Matrix Factorization

dim	costp_l1	costp_l2	costq_l1	costq_l2	lrate	loss_fun
50	0.1	0.1	0	0.01	0.1	2.168722

Table 10: Model RMSE Result

Model	RMSE	Description
Model without Regularization	1.664640	This model is evaluated without any regularization applied, which may lead to overfitting.
Model with Regularization	1.663445	This model incorporates regularization to prevent overfitting and improve generalization.
Mean Cross-Validated RMSE	1.715000	This RMSE is derived from cross-validation, providing a more robust estimate of model performance.
RMSE using Tuned Paramters	1.745558	This RMSE reflects the best performance obtained through hyperparameter optimization.

The model without regularization achieved an RMSE of 1.66, while the regularized model slightly worse performance with an RMSE of 1.66. This suggests that regularization did not help mitigate over fitting. The cross-validated RMSE of 1.71 provides a more reliable estimate of the model's performance on unseen data. The model with tuned parameters achieved a lower performance, with an RMSE of 1.74.

Ensemble Model

An ensemble approach combines predictions from multiple different models to generate a concensu views. This type of model attempts to draw strgeths from different model. Three recommender system models: item-based collaborative filtering (CF), user-based CF, and matrix factorization (MF) are used for the following ensemble model where with the weightings for each model as follows: 0.2 for item-based CF, 0.2 for user-based CF, and 0.6 for matrix factorization. The weights were chosen to balance the strengths matrix factorization with inputs from more simplistic models such as user and item based CF. The code for this section does unfortunately not run but the workings are provided in the file.

Conclusions

An effective approach to building a recommender system would be to use an ensemble method that combine item-based collaborative filtering (CF), user-based CF, and matrix factorization. Item-based CF analyzes similarities between items based on user ratings, allowing for recommendations of similar books. User-based CF, on the other hand, identifies users with similar preferences to make personalized suggestions. Matrix factorization decomposes the user-item interaction matrix into latent factors, capturing hidden relationships between users and items. By integrating predictions from these three methods through weighted averaging—an ensemble model can enhance overall recommendation accuracy and provide more robust suggestions tailored to individual user preferences.

References

Arashnic. 2021. "Book Recommendation Dataset." https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset/data.