# Enhancing Book Recommendations with a Hybrid Matrix Factorization

Kamel Charaf, Kyuhee Kim, Christina Kopidaki
Recommendos
Kaggle notebook
kamel.charaf@epfl.ch, kyuhee.kim@epfl.ch, christina.kopidaki@epfl.ch

## I. Introduction

Recommender systems are crucial in improving user experiences across digital platforms by providing personalized suggestions. In the context of books, these systems enable readers to discover works aligned with their interests and preferences. This report uses a comprehensive dataset to introduce a book recommendation system to predict user ratings for unseen books. The dataset includes user-book interactions and the ISBNs of the books, which were used to crawl metadata about them.

Our recommendation system utilizes a hybrid approach with a Matrix Factorization combined with a content-based filtering method to predict user preferences. This combination allows us to capture hidden relationships between users and books by decomposing the user-item interaction matrix into latent factors. The system is optimized to minimize the Root Mean Square Error (RMSE), ensuring accurate and personalized predictions. This work highlights the effectiveness of Matrix Factorization in addressing challenges like data sparsity and scalability while delivering high-quality recommendations for readers.

## II. Dataset Overview

**Description.** The book rating dataset includes user ratings identified by book ID and user ID, measured on a 5-star scale with half-star increments ranging from 1.0 to 5.0. The dataset comprises 129,890 entries, divided into two subsets: a training set with 100,523 entries (including ratings) and a test set with 29,367 entries. There are $18,905$ users and $15,712$ items in total. This means the training dataset is $99.966\%$ sparse.

**Additional Data.** As detailed in Table I, most books are associated with their **ISBN** (International Standard Book Number), a unique identifier for books. Using these ISBNs, we retrieve supplementary data, such as subjects, authors, and languages, through the Open Library Search API. Table I summarizes the number of entries enriched with this information. This additional data expand the data set, enabling a more comprehensive feature set for the content-based approach.

| | **ISBN** (%) | **Authors** | **Subjects** | **Languages** | **Combined** (%) |
|---|---|---|---|---|---|
| Train | 99,996 (99.47) | 75,998 | 55,216 | 87,479 | 95,083 (94.58) |
| Test | 29,340 (99.90) | 22,414 | 16,341 | 8,827 | 28,105 (95.70) |

TABLE I
Availability of additional data (Authors, Subjects, Languages) and combined coverage in Train and Test sets.

## III. Preliminary Approaches

**Collaborative Filtering** (CF) is a technique used in recommendation systems to predict user preferences for items based on their historical interactions and the preferences of other users. The key idea behind it is that users with similar preferences in the past will likely have similar preferences in the future. The CF methods we implemented can be found in Appendix A. Due to the sparsity of the data, the performance of these models was poor.

**Graph Neural Network-based** (GNNs) [2] extend collaborative filtering by representing user-item interactions as a graph $G = (V, E)$ and leveraging the graph structure to learn latent representations (embeddings) for users and items. Unlike traditional CF, which computes similarities explicitly, GNNs use message passing to learn embeddings:

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \frac{1}{\sqrt{d_i d_j}} W^{(l)} h_j^{(l)}\right).$$

- $h_i^{(l+1)}$: The updated embedding of node $i$ at layer $l+1$.
- $h_j^{(l)}$: The embedding of neighboring node $j$ at layer $l$.
- $\mathcal{N}(i)$: The set of neighbors of node $i$.
- $d_i, d_j$: The degrees of nodes $i$ and $j$ (number of connected edges).
- $W^{(l)}$: A learnable weight matrix at layer $l$ that transforms the embeddings.
- $\sigma$: A non-linear activation function (e.g., ReLU).

The predicted rating for a user-item pair is given by:

$$\hat{r}_{ui} = h_u^\top h_i,$$

**Content-based Filtering** is a recommendation technique that suggests items based on their features and previous interactions of the user. It creates item profiles using attributes such as genres, authors, or descriptions and matches them with user preferences by calculating similarity scores.

In our approach, we utilized a TF-IDF matrix to represent books using a combination of subjects, authors, and languages. For each user and book in the test dataset, we calculated the cosine similarity between the target book and the books previously rated by the user. Using the ratings of the top N most similar books, we predicted the user's rating for the target book. In cases where similar books could not be found due to missing ISBNs or other metadata, we used the average of the user's past ratings as a fallback.

## IV. Our Method

Matrix factorization is an effective method to predict users' preferences solely based on previous interactions (without additional knowledge about the items). It works by decomposing the original ($R$), user-item rating matrix into two, lower-dimensional

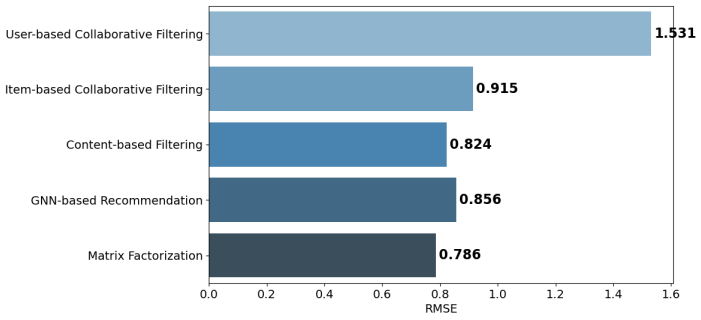


Fig. 1. RMSE values for each created method

(called latent dimension ($d$)) matrices, where one represents the users ($P$), and the other represents the items ($Q$).

$$R \approx P \times Q^T = \hat{R}$$

To predict a user's rating of an item that was not rated before the two matrices are multiplied together, as they approximate the non-zero entries of the original user-item rating matrix. Constructing those two matrices is an optimization problem, defined as:

$$\mathcal{L} = \min_{p,q} \sum_{(i,j) \text{ known}} [(r_{ij} - \hat{r}_{ij})^2] + \lambda_2(\|P\|_F^2 + \|Q\|_F^2),$$

$$\hat{r}_{ij} = \sum_{k=1}^{d} p_{ik} q_{kj}, \qquad e_{ij} = (r_{ij} - \hat{r}_{ij})$$

We applied **Stochastic Gradient Descent** to solve this minimization problem, which has the following update rule for some random $(i, j)$ pair where the rating from user $i$ to item $j$ is known:

$$p_{ik} := p_{ik} + 2\alpha(e_{ij}\ q_{kj} - \lambda_2 p_{ik}),$$
$$q_{kj} := q_{kj} + 2\alpha(e_{ij}\ p_{ik} - \lambda_2 q_{kj})$$

Originally, we initialize $P$ and $Q$ matrices with a random normal distribution with a mean of 0 and a standard deviation of 0.1. To achieve a lower RMSE score, we introduced several enhancements to the original pipeline.

*1) User/Item bias list:* Users may rate different books higher/lower in general and also books may receive higher/lower ratings based on their popularity. To address this issue we introduce the **user bias** ($b_i$) and **item bias** ($b_j$) that will be learnable parameters and the **global average of non-zero ratings** ($\mu$)

$$\hat{R}_{i,j} = \mu + b_i + b_j + P_i \times Q_j^T$$

The objective function will change with the introduced changes:

$$\mathcal{L} = \min_{p,q,b_u,b_i} \sum_{(i,j) \text{ known}} (r_{ij} - \hat{r}_{ij})^2 + \lambda_2 \left(\|P\|_F^2 + \|Q\|_F^2 + b_i^2 + b_j^2\right),$$

$$\hat{r}_{ij} = \mu + b_i + b_j + \sum_{k=1}^{d} p_{ik} q_{kj}$$

As the defined biases are learnable parameters, we can define the update function as:

$$b_i := b_i + 2\alpha(e_{ij} - \lambda_2 \times b_i),$$
$$b_j := b_j + 2\alpha(e_{ij} - \lambda_2 \times b_j)$$

*2) Combining L1 and L2 regularization:* Combining L1 and L2 regularization, known as Elastic Net, is an effective technique since L2 (ridge) helps prevent overfitting, while L1 (lasso) promotes sparsity by shrinking less relevant dimensions to zero. Together, the strength of both methods emerges, balancing model complexity and relevance. The objective function changes slightly (see Appendix B).

*3) Hyperparameter optimizing:* To find the optimal hyperparameters - the $d$ (latent dimension), the $\alpha$ (learning rate), and the $\lambda$ (regularization value) - we applied a 5-fold cross-validation. See Figure 2.

| **Matrix Factorization Methods** | **RMSE Score** |
|---|---|
| Original with l2 regularization | 1.23265 |
| With custom user/item bias | 0.79239 |
| With custom user/item learning rate | 0.79273 |
| P and Q matrices with Xavier initialization | 0.79572 |
| With normalized ratings | 0.93210 |
| With custom user/item regularization | 0.79640 |
| With using papertrick from [1] | 0.79185 |
| **With custom bias and elastic net regularization** | 0.79120 |

TABLE II
PERFORMANCE COMPARISON OF MATRIX FACTORIZATION METHODS

## V. RESULTS

The evaluation of our recommendation models on Kaggle using RMSE as the metric highlights the strengths and limitations of each approach. The item-based collaborative filtering model achieved 0.915, demonstrating its effectiveness in leveraging item similarities for accurate predictions. However, the user-based collaborative filtering model performed worse, 1.531, likely due to the higher variability in user behaviours and sparse interactions. The GNN-based model outperformed both traditional approaches, achieving the lowest RMSE of 0.856. This result underscores the power of GNNs in capturing complex relationships.

Matrix factorization with custom user/item bias lists and elastic-net regularization achieved the best RMSE score with 0.79120 outperforming other methods. However, content-based filtering contributes differently and leverages item-specific information based on the ISBN given, which Matrix factorization can not exploit. By averaging the predictions from these two methods, weighted 1.5 for matrix factorization and 0.5 for content-based filtering, the combined model resulted in a more robust and balanced recommendation system. This approach relies on the accuracy of matrix factorization while integrating the semantical insight provided by content-based filtering. This improved generalization and prediction and we achieved our overall best, 0.786 RMSE score. The results for the different variations of Matrix Factorization are shown in Table II

## VI. CONCLUSION

This work presented a hybrid book recommendation system combining Matrix Factorization and content-based filtering. With enhancements like user/item biases and elastic net regularization, we achieved a 0.786 RMSE, demonstrating improved accuracy and robustness. Future work can explore richer metadata integration to further enhance performance.

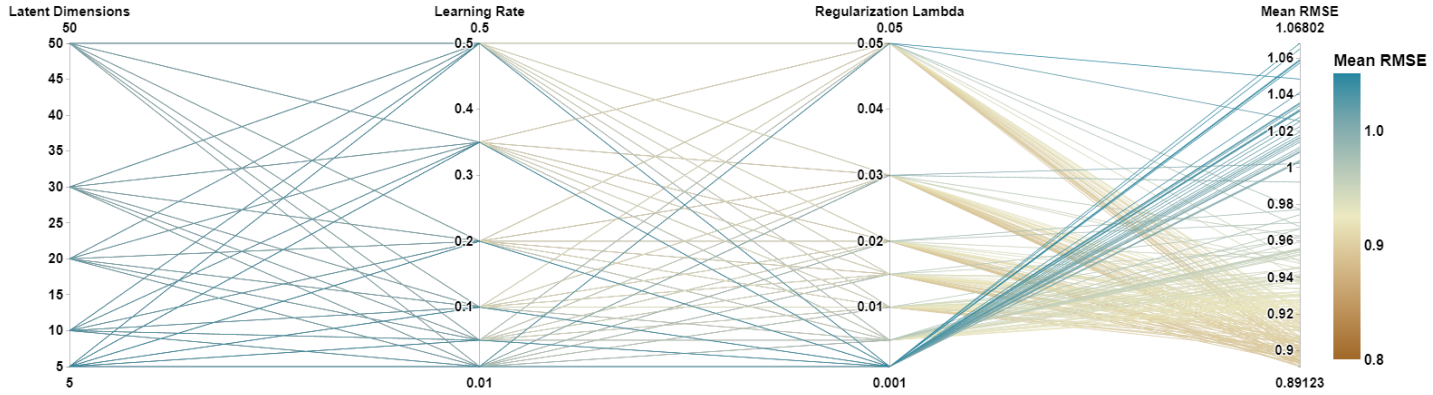


Fig. 2. RMSE scores for various hyperparameter values.

## REFERENCES


[1] Thanh Tran, Kyumin Lee, Yiming Liao, and Dongwon Lee. Regularizing matrix factorization with user and item embeddings for recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18. ACM, October 2018.

[2] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey, 2022.


## APPENDIX

### A. *User and Item-based Collaborative Filtering formulas*

- **User-based**:

$$\hat{r}_{u,j} = \frac{\sum_{v \in N(u)} \text{Similarity}(u,v) \cdot r_{v,j}}{\sum_{v \in N(u)} |\text{Similarity}(u,v)|}$$

- **Item-based**:

$$\hat{r}_{u,j} = \frac{\sum_{i \in N(j)} \text{Similarity}(j,i) \cdot r_{u,i}}{\sum_{i \in N(j)} |\text{Similarity}(j,i)|}$$

- **Where**:
  - $\hat{r}_{u,j}$: The predicted rating for user $u$ on item $j$.
  - $N(u)$: The set of users $v$ who are similar to user $u$.
  - $N(j)$: The set of items $i$ that are similar to item $j$.
  - $\text{Similarity}(x,y)$: The similarity between two entities, computed using cosine similarity.
  - $r_{x,y}$: The rating of entity $x$ for entity $y$ (e.g., $r_{v,j}$ or $r_{u,i}$).

### B. *Objective Function after combining L1 and L2 regularization*

$$\begin{aligned}\mathcal{L} = \min_{p,q,b_u,b_i} & \sum_{(i,j) \text{ known}} (r_{ij} - \hat{r}_{ij})^2 \\ & + \lambda_2 \left( \|P\|_F^2 + \|Q\|_F^2 + b_i^2 + b_j^2 \right) \\ & + \lambda_1 \left( \|P\|_1 + \|Q\|_1 + \|b_i\|_1 + \|b_j\|_1 \right)\end{aligned}$$

Note that $\lambda_1$ and $\lambda_2$ are hyperparameters and cannot be learned. The update functions change to the following:

$$\begin{aligned} p_{ik} &:= p_{ik} + 2\alpha(e_{ij}\ q_{kj} - \lambda_2 p_{ik}) - \frac{\lambda_1}{2}\ sign(p_{ij}), \\ q_{kj} &:= q_{kj} + 2\alpha(e_{ij}\ p_{ik} - \lambda_2 q_{kj}) - \frac{\lambda_1}{2}\ sign(q_{kj}), \\ b_i &:= b_i + 2\alpha(e_{ij} - \lambda_2 b_i + \frac{\lambda_1}{2}\ sign(b_i)), \\ b_j &:= b_j + 2\alpha(e_{ij} - \lambda_2 b_j + \frac{\lambda_1}{2}\ sign(b_j)) \end{aligned}$$