DIS Project 2: Recommender Systems

Team Datanauts - Kaggle Notebook [RMSE: 0.78628]

Stefan Krsteski
stefan.krsteski@epfl.ch

Said Gürbüz
said.gurbuz@epfl.ch

Matea Tashkovska
matea.tashkovska@epfl.ch

1 Introduction

Today, personalized recommendations are an essential part of online platforms. Popular applications such as Spotify and Netflix use recommendation systems to deliver tailored content to users and enhance engagement and satisfaction. These systems analyze user preferences and behaviors to provide content that aligns with individual tastes. Similarly, platforms such as Goodreads allow users to rate books, creating a wealth of data that can be used to develop systems capable of recommending books that users are most likely to enjoy.

In this project, we developed a book recommendation system using matrix factorization augmented with user offsets and linear layers with skip connections. User offsets were included specifically to solve the cold-start problem (Yuan and Hernandez, 2023), which occurs frequently in recommender systems when new users lack interaction data. Our best model achieves a RMSE of 0.78628 on the test set.

2 Methods

2.1 Data Preprocessing

The training dataset consists of 100,523 book ratings provided by 18,905 unique users across 15,712 books. As a first step, we mapped user and book IDs to a dense, zero-indexed range to standardize input. To ensure consistency in our experiments, we split the data into 95% training and 5% validation sets.

After a closer analysis of the rating distributions in the train set, we observed an interesting pattern among cold users, the ones who had less than 5 interactions in the training set. As shown in Figure 1, these users tend to give higher ratings compared to those with more extensive interaction histories. One possible explanation for this is that cold users might only rate books they genuinely enjoyed, thus biasing their personal rating scale upwards.

Ignoring this disparity could lead to biased predictions. To address this, we adapted our models accordingly to ensure that such users are fairly represented in the prediction process, as explained in the following subsection.

2.2 Models

We experimented with collaborative and content-based filtering, with our main focus on collaborative methods due to their superior performance on this dataset. All models minimize the mean squared error (MSE) between the

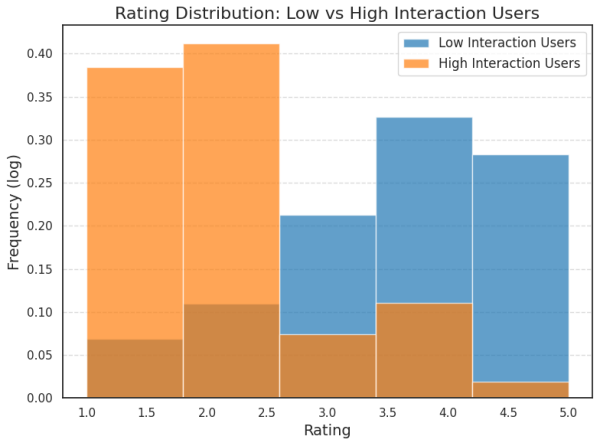

Figure 1: Normalized rating distribution for low interaction (cold) and high interaction users.

predicted and true ratings, with an additional L2 regularization term to prevent overfitting. The loss function is defined as:

$$\mathcal{L} = \text{MSE}(\hat{r}, r) + \lambda \left(\|\mathbf{U}\|_2^2 + \|\mathbf{V}\|_2^2 \right) \quad (1)$$

where $\hat{r}$ and r are the predicted and true ratings, $\mathbf{U}$ and $\mathbf{V}$ are the user and item embeddings, and λ is the regularization strength. We used the Adam optimizer with a ReduceLROnPlateau scheduler (factor 0.5, patience 2) across all models.

Matrix Factorization (MF). As a baseline, we use a simple Matrix Factorization model (Koren et al., 2009), where each user and item is represented by a latent embedding vector. A predicted rating is computed as the dot product of these embeddings, plus user/item biases and global offset. Specifically, for a user u and item i, the predicted rating $\hat{r}_{ui}$ is given by:

$$\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i + b_u + b_i + b_{global} \quad (2)$$

where $\mathbf{p}_u$ is the user embedding, $\mathbf{q}_i$ is the item embedding, b_u and b_i are user and item bias terms, and b_{global} is the global offset.

MF with linear layers. As an enhancement to the previous method, we introduced linear transformations to the user and item embeddings. Specifically, after computing the initial embeddings, we applied two linear layers with skip connections to refine the embeddings, one for the user embedding and one for the item embedding. This approach was inspired by the use of Multi-Layer Perceptrons (MLPs) in Transformers to mix information (Vaswani, 2017), although applied in a different context. The skip connections ensure that the original embedding information is preserved while allowing the linear transformations to add flexibility.

MF with linear layers & user offsets. As mentioned in the previous section, we observed that users with very few ratings tend to have different rating distributions compared to heavy raters. We empirically observed in our training dataset that users with the same number of ratings tend to exhibit similar rating patterns. To handle this, we first precompute a mapping from a rating count c to the average rating of all users who share exactly c number of ratings. For a given user u with c_u ratings, if this mapping provides a direct match, we use that average as the user's initial offset μ_u. For instance, if a user has 10 ratings and we know that average rating of all 10-rating users is 3.7, then we start this user at $\mu_u = 3.7$. This step ensures that users with a known interaction level begin from a realistic baseline that mirrors established behavior patterns for that level of activity.

If there is no exact match, we approximated the offset using a smooth interpolation between two end values that we find empirically: μ_{high} and μ_{low}. The approximation uses a normalized, log-scaled function of c_u:

$$\mu_u = \mu_{high} + (\mu_{low} - \mu_{high}) \times \frac{\log(1 + c_u)}{\log(1 + c_{\max})} \quad (3)$$

Here, $c_{\max}$ is the maximum observed rating count. This interpolation ensures a gradual transition from the high-offset regime to the low-offset regime. This precomputed user offset was used to initialize the user bias weights. Intuitively, this helps the model quickly adapt to user-specific rating scales, especially for those with limited history.

To optimize the hyperparameters of the MF-based models, we used Optuna[1]. The search space included the embedding dimension (ranging from 32 to 128 with steps of 16), the learning rate (log-uniformly sampled between 10^{-5} and 10^{-3}), the regularization strength (log-uniformly sampled between 10^{-5} and 10^{-1}), and the patience parameter for early stopping (ranging from 5 to 20 epochs). The best hyperparameters were determined through iterative trials and include: an embedding dimension of 64, a learning rate of 3.59×10^{-5}, a regularization strength of 0.0016 and a batch size of 64. These parameters were used across all MF-based models.

Additional Methods. Due to space limitations in the main text, we have detailed our experiments with three additional methods: Content-based filtering, Wide & Deep architecture, and LightGCN, in Appendix B. These methods utilize book metadata, shallow and deep learning components, and graph-based embeddings, respectively.

3 Results

We evaluated each approach using the RMSE metric on the Kaggle public test set, and the results are shown in Table 1. While our baseline Matrix Factorization (MF) model provided a solid starting point with an RMSE of 0.80580, some of the more complex architectures, such as the content-based model, LightGCN, and Wide & Deep, did not improve upon this baseline. This underperformance could be due to the dataset's sparsity which makes it challenging for these complex models to fully leverage their complexity without further tuning or additional data.

On the other hand, incrementally enhancing the MF model with simple but effective modifications yielded significant improvements. Adding linear layers with skip connections improved the RMSE to 0.79609. This suggests that linear layers add flexibility to the model without drastically increasing the complexity, can make the latent representations more expressive while skip connections make the model more robust.

We achieved the best results when we added user-specific offsets, which reduced the RMSE to 0.78628. By initializing user bias terms to reflect general tendencies of user with similar interaction levels, the model started from user-tailored point that leads to more accurate predictions for both cold-start and high-interaction users.

Table 1: Performance Comparison of Different Models.

Method	**RMSE** ↓
Matrix Factorization (MF)	0.80580
Content-based - Appendix B.1	0.96318
LightGCN - Appendix B.2	0.81051
Wide & Deep - Appendix B.3	0.81579
MF with linear layers	0.79609
MF with linear layers + offsets	**0.78628**

3.1 Analysis

To further analyze the performance of our model, we split the validation data based on user interaction levels into high and low interaction groups (>=5 and <5 respectively). Using the best-performing model, we observed an RMSE of 0.9697 for low interaction users compared to 0.7581 for high interaction users. These results are expected, as users with fewer interactions pose a greater challenge due to sparse data. For comparison, we performed the same evaluation with the baseline MF model. The low interaction users show a significantly higher RMSE of 1.0143, while the RMSE for high interaction users remained relatively consistent at 0.7611. This shows the advantage of our best approach in better handling cold-start users, although there is still room for improvement in this area.

4 Discussion

This project focused on developing a recommender system for books, with a primary focus on collaborative filtering methods. Our best-performing model, matrix factorization with linear layers and predefined user offsets, achieved an RMSE of 0.78628. This result shows the effectiveness of incorporating user-specific adjustments, particularly for low-interaction users.

The user offsets significantly reduced errors for cold-start users, demonstrating the importance of capturing user-specific rating behaviors. However, a notable performance gap between high and low interaction groups remains, that highlights an area for potential improvement.

Future efforts could focus on "warming up" cold users by linking them to warm users with similar characteristics.

[1] https://optuna.org/

This approach involves leveraging shared demographic attributes, such as age, location, or mutual connections, to align cold users with warm users who have well-defined interaction histories. By inferring preferences from these connections, the system could create richer representations for cold users, and help in reducing the performance gap. While this strategy was not applicable in our case due to the lack of user-specific metadata, it represents a viable option in scenarios where such information is available.

References

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Hongli Yuan and Alexander Hernandez. 2023. User cold start problem in recommendation systems: A systematic review. *IEEE Access*.

A Code

Final submission achieved with MF with linear layers and offsets.
https://www.kaggle.com/code/stefankrsteski/matrix-factorization-with-offsets

GitHub Repository
https://github.com/Stefanstud/Recommender-Systems
Our GitHub repository includes all other approaches and supporting scripts, in addition to the best approach, and will be shared with the responsible TAs. For additional access, please email stefan.krsteski@epfl.ch.

B Appendix

B.1 Content-based

Content-based filtering is often preferred for addressing the cold-start problem, as it relies on item attributes rather than just user interaction history. As the original dataset did not provide additional book details, we enhanced our data by collecting additional information such as the title, description, categories, page count and authors using the Google Books API [2]. These textual components were concatenated and then embedded using a BGE sentence transformer [3] to obtain a fixed-length vector representation. The categories were represented using GloVe embeddings [4], with the mean embedding computed for items associated with multiple categories. Authors were encoded as unique numerical identifiers using a label encoder, ensuring compatibility with embedding layers in the model. Our model processed these features through a neural network, embedding text, categorical, and numerical data into a shared latent space. The final prediction was generated by concatenating these embeddings and passing them through a fully connected layer.

B.2 LightGCN (He et al., 2020)

We further explored LightGCN, which is a recent graph-based collaborative filtering approach. LightGCN constructs a user-item bipartite graph and propagates embeddings through multiple graph convolutional layers without transformation functions. We integrated user and item biases, as well as user-level interaction features into LightGCN to adapt it for rating prediction. We construct a user-item interaction graph, and model the propagation of information through the graph in 3 layers to get higher order relationships. Adam optimizer is used to train this model with a learning rate of 0.001. The results are shown in Table 1.

B.3 Wide & Deep (Cheng et al., 2016)

We additionally explored the Wide & Deep model, a well-known state-of-the-art method designed to combine memorization and generalization in recommendation systems. This architecture integrates two components: a wide linear model and a deep neural network. The wide component captures feature interactions explicitly, leveraging raw numerical and categorical features for memorization. Meanwhile, the deep component learns complex patterns through embeddings and MLPs, enabling the model to generalize to unseen feature combinations.

For our implementation, the wide component consisted of numerical features such as page count, average rating, ratings count, and published year (extracted from Google Books API), processed through a simple linear layer. The deep component incorporated embeddings for user, book, author, category, and publisher features. Additionally, it included pre-trained full-text embeddings (BGE) to provide semantic context. These embeddings were concatenated with the numerical features and passed through multiple fully connected layers with ReLU activations and dropout regularization.

The output of the wide and deep components was combined to generate the final prediction and achieved public test RMSE of 0.81579, as shown in Table 1. While this approach showed promise, its computational complexity and hyperparameter tuning requirements made it less practical for our dataset. Future work could explore optimizing this model for achieving better performance.

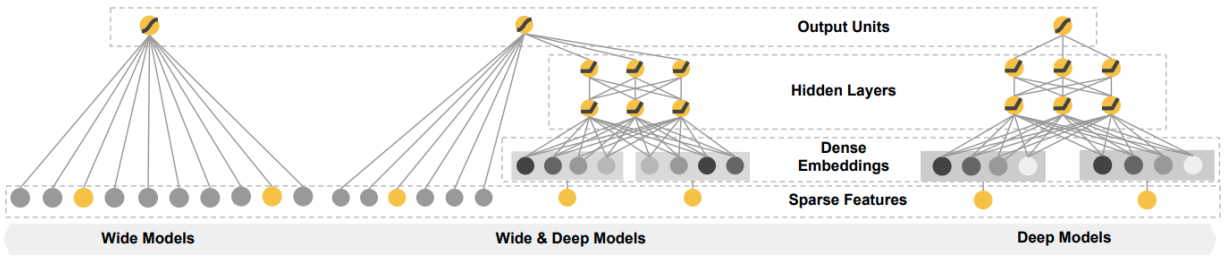

Figure 2: The spectrum of Wide & Deep models

[2] https://developers.google.com/books
[3] https://huggingface.co/BAAI/bge-small-en-v1.5
[4] https://nlp.stanford.edu/projects/glove/