Project 2: Recommender Systems

Luca Engel
luca.engel@epfl.ch

Team Name: Gigubyte
Final notebook: [1]

Damian Kopp
damian.kopp@epfl.ch

1 Introduction

This project builds a recommender system for estimating book ratings, using a sparse user-book ratings dataset. As a baseline, an Alternating Least Squares (ALS) system is implemented, achieving a Root Mean Squared Error (RMSE) of 1.1318. To generate a more sophisticated system, user- and item-based collaborative filtering (CF) models are created that use k-Nearest Neighbors (k-NN) for similarity computation. A hybrid system as well as adding metadata via ISBNs improve the performance of the before-mentioned individual systems, achieving an RMSE score of 0.8242.

The contributions of this project include the effective use of hybrid models with metadata to build a well-performing and scalable book recommender system.

2 Approach

2.1 Alternating Least Squares [2]

As a first approach for predicting book ratings based on existing user-book ratings, ALS was implemented. This method treats missing ratings in a sparse user-item matrix (1) as unknown values that need to be predicted. It factorizes the user-book matrix into two latent factor matrices – U and V. Alternating, both matrices are updated using closed-form solving while the other one is fixed. The key ALS parameters are:

- **Number of latent factors (k)**: Sets the dimensionality of the latent factor matrices.
- **Regularization parameter (λ)**: Penalizes large factor values to improve generalization.
- **Number of iterations (n)**: Fixed at 20 for convergence.

The dot product of the user and book matrix produces the predicted ratings that are then assessed using RMSE. This CF method serves as a baseline for the more sophisticated approaches.

2.2 User-based Collaborative Filtering [3]

User-based CF leverages similarity measurements between users to provide rating predictions. The method relies on the premise that users with similar preferences will likely rate items similarly.

The k-NN algorithm is used to identify similar users. [4] The similarity between users is calculated using the cosine similarity metric based on user ratings. The value for k is is found through hyperparameter tuning with a validation set. [5] Predictions for unseen items are generated by aggregating the weighted ratings of the nearest neighbors to estimate the rating of a given user. Missing ratings are changed to zero values during similarity computation to handle the sparsity challenge. (1)

2.3 Item-based Collaborative Filtering [6]

Item-based CF identifies similarities between items to predict user ratings. Unlike user-based CF, this approach focuses on the relationship between items, working on the assumption that users prefer items that are similar to those they have rated highly in the past.

Similar to the user-based CF, this approach leverages k-NN to identify similar books using the cosine similarity. [4] The value for k is found through hyperparameter tuning with a validation set. Additionally, metadata is extracted for each book, leveraging the books' International Standard Book Number (ISBN), to improve the similarity computation's performance. [7] [8] [9] Here, with the help of a validation set, metadata that improves the model performance is kept and the rest is discarded. [10] The kept metadata includes book subjects, summaries, and language. To properly leverage the subjects and summaries, k-means clustering is applied on sentence transformers' embedding vectors. [11] [12] [13] [14] A fitting k is selected using the elbow method. [15] The clusters are stored as a weighted one-hot encoding. The weights are also found through hyperparameter tuning.

Predictions are generated by aggregating the weighted contributions of the k most similar items for each unrated item

2.4 Hybrid Collaborative Filtering [1]

This approach simply averages the predictions of both the user- and item-based CF approaches.

3 Results

For optimizing ALS, we tuned the number of latent factors, the regularization parameter, and the number of iterations. The following shows the optimal hyperparameters based on the results:

- Number of latent factors $k = 50$
- Regularization parameter $\lambda = 0.8$
- Number of iterations $n = 20$

The corresponding ALS model produced an RMSE score of 1.1318 when applied to the test set.

The RMSE values for the test dataset of different CF approaches are shown in table 1. For both user- and item-based collaborative filtering, smaller k's were found to yield better performances during hyperparameter tuning. However, from k=1 to k=5, there is only a very slight performance drop. For all k's tested except for k=1 where it is equal, hybrid CF outperforms user- and item-based CF. Adding metadata to item-based CF mostly improved its performance, however only slightly.

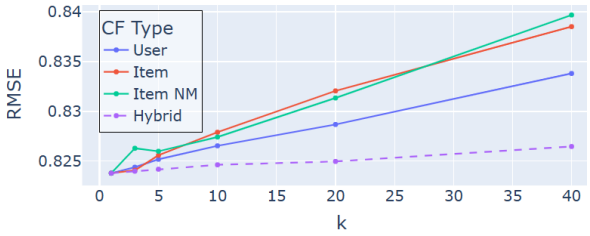

Figure 1: RMSE of CF Approaches by k (k-NN) [16]

3.1 Analysis

Achieving 1.1318 as RMSE score, ALS possesses a mediocre prediction capability. Considering the fact that ALS learns user-item interactions by using latent factors, the system estimates unknown ratings considerably well. Using 50 latent factors optimizes the tradeoff between avoiding overfitting and capturing sufficient complexity in the data. Regarding the regularization parameter λ, setting it to 0.8 prevents getting too large weights in latent factors and still allows for good performance. The most significant learning happens during the first 20 iterations of training which is why n is set to this value.

According to [17], ALS does "a pretty good job at solving scalability and sparseness of the Ratings data, and it's simple and scales well to very large datasets." The outcomes of ALS align with this finding, especially considering the challenging circumstances of working with a very sparse dataset (1) and without the integration of any metadata.

Both user- and item-based CF outperform ALS while user-based CF outperforms item-based CF slightly. In Figure 2, we can see that most users only rated one book. However from Figure 3, we can infer that books were largely rated more than once. In other words, lots of "single-rater" users rated the same books as others, which might result in non-neglectable preference overlaps. Those overlaps allow the algorithm to detect patterns and group users together in a more effective way than for item-based CF where such similarities appear less frequently, which may explain the performance difference in Figure 1.

Our hybrid approach combines the advantages of user-based and item-based CF as it captures both user- and book relationships. For the given data, this approach performs the best. Picking a value k of 5 strikes a good balance between the low RMSE score and staying robust against potential outliers.

Incorporating metadata into the item-based CF system improved the performance, but only minimally and not for all k values as can be seen in the direct comparison of the item-based method in Figure 1.

Also, the test set only contains users and items that appear in the train set at least once. Therefore, each prediction contains at least one known user-item interaction with heightened similarity. This anchoring increases the accuracy of the predictions.

4 Conclusion

This project successfully developed multiple robust CF-based recommender systems for predicting book ratings. After creating a baseline with ALS, this project improved the performance with user- and item-based as well as hybrid CF models. k-NN with the cosine similarity was used for similarity computations. The inclusion of metadata through ISBNs slightly enhanced the prediction accuracy. The hybrid model outperformed the other

approaches by combining the strengths of user- and item-based CF. This work contributes scalable and efficient models that can effectively handle sparse datasets. It also provides useful insights for future recommender system enhancements.

References

[1] L. Engel, "Submission Hybrid Collaborative Filtering." [Online]. Available: https://kaggle.com/code/luca3ngel/submission-hybrid-collaborative-filtering

[2] D. Kopp, "Submission ALS." [Online]. Available: https://www.kaggle.com/thymiantheherb/submission-als

[3] L. Engel, "Submission User-Based Collaborative Filtering." [Online]. Available: https://kaggle.com/code/luca3ngel/submission-user-based-collaborative-filtering

[4] Scikit-learn, "NearestNeighbors." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html

[5] L. Engel, "User-Based Hyperparameter Tuning Luca." [Online]. Available: https://kaggle.com/code/luca3ngel/user-based-hyperparameter-tuning-luca

[6] ——, "Submission Item-Based Collaborative Filtering." [Online]. Available: https://kaggle.com/code/luca3ngel/submission-item-based-collaborative-filtering

[7] I. I. Agency, "What is an ISBN?" [Online]. Available: https://www.isbn-international.org/content/what-isbn/10

[8] L. Engel, "Submission Metadata Extraction." [Online]. Available: https://kaggle.com/code/luca3ngel/submission-metatdata-extraction

[9] ——, "Submission Create Processed Metadata." [Online]. Available: https://kaggle.com/code/luca3ngel/submission-create-processed-metadata

[10] ——, "Item-Based Hyperparameter Tuning." [Online]. Available: https://kaggle.com/code/luca3ngel/item-based-hyperparameter-tuning

[11] Scikit-learn, "KMeans." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[12] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. [Online]. Available: http://arxiv.org/abs/1908.10084

[13] L. Engel, "Submission Subject Clustering." [Online]. Available: https://kaggle.com/code/luca3ngel/submission-subject-clustering

[14] D. Kopp, "Submission Summary Clustering." [Online]. Available: https://www.kaggle.com/code/thymiantheherb/submission-summaries-clustering

[15] GeeksforGeeks, "Elbow Method for optimal value of k in KMeans," Jun. 2019, section: AI-ML-DS. [Online]. Available: https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/

[16] L. Engel, "Submission Performance Plots." [Online]. Available: https://kaggle.com/code/luca3ngel/submission-performance-plots

[17] K. Liao, "Prototyping a recommender system step by step part 2: Alternating least square (als) matrix factorization in collaborative filtering." [Online]. Available: https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1

5 Annex

$$
\begin{aligned}
&\text{User-Item Matrix Sparsity} \\
&= 1 - \frac{\text{Number of non-zero ratings}}{\text{Total number of possible ratings}} \\
&= 1 - \frac{\text{Number of non-zero ratings}}{\text{Nb Users} \times \text{Nb Books}} \\
&= 1 - \frac{100523}{18905 \times 15712} = 0.9997
\end{aligned} \tag{1}
$$

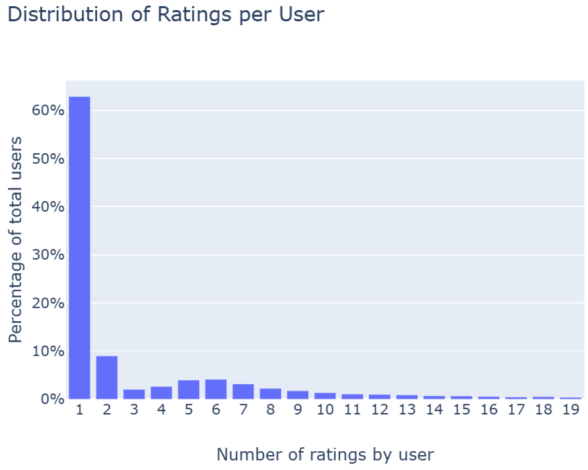

Figure 2: Percentage of Users who Gave a Given Number of Ratings

Figure 3: Percentage of Books with Given Number of Ratings

CF Type	k = 1	k = 3	k = 5	k = 10	k = 20	k = 40
User	0.8238	0.8244	0.8252	0.8266	0.8287	0.8338
Item	0.8238	0.8241	0.8256	0.8279	0.8321	0.8385
Item NM	0.8238	0.8263	0.8260	0.8274	0.8314	0.8397
Hybrid	**0.8238**	**0.8240**	**0.8242**	**0.8246**	**0.8250**	**0.8265**

Table 1: RMSE of CF Approaches by k (k-NN)
NM: no metadata