Project 2: Recommender Systems

Team MAM : Aryan Ahadinia and **Matin Ansaripour** and **Madeleine Hueber**
{aryan.ahadinia, matin.ansaripour, madeleine.hueber}@epfl.ch

Introduction

The rapid growth of online markets has increased the need for accurate recommendation systems to enhance user navigation and enable targeted advertising. The objective of this project is to develop a fast and accurate book recommender system that predicts user ratings. To achieve this, we first enriched the provided dataset with the books' information and descriptions using various book APIs (sec. 1). Next, we explored multiple methodologies (sec. 2), including classical and neural matrix factorization, as well as content-based approaches, to improve recommendations. These methods were then evaluated and compared to baseline models, with the results presented in sec. 3. Finally, our findings are discussed and concluded in sec. 4. Our full notebook is available on Kaggle.

1 Data Enrichment

We enrich the books' metadata by including the title, author, and description. Titles and descriptions can provide rich information about a book's content which can be extracted by using well-established retrieval and embedding techniques. Author information is included, since it is presumed that authors preserve the same tense in their artifacts, making them likely to receive similar ratings.

To collect the book metadata, we implemented a multi-step process to ensure maximal coverage. In these three steps, we have employed *Open Library* (Open Library, n.d.), *Google Books* (Google, n.d.), and *ISBNdb* (ISBNdb, n.d.), respectively, due to their restrictions, prices, and missing values (details in app. A). Once collected, the metadata undergoes a cleaning process to remove hyperlinks, blank lines, and other extraneous elements, followed by translation into English for consistency.

The metadata is embedded into a numerical space using two distinct approaches. In the first approach, descriptions are embedded using the BM-25 method, which focuses on collaborative filtering-based methods. In the second approach, used for neural methods, the descriptions are embedded in a latent feature space using pre-trained `microsoft/deberta-v3-base` (He et al., 2023) model from hugging face (Wolf et al., 2020).

2 Methodologies

2.1 Matrix Factorization (MF)

Matrix factorization is a powerful technique for collaborative filtering in recommendation systems, particularly effective in handling the sparsity of user-item interactions. In our case, the user-book rating matrix R is extremely sparse, as ratings are available only for a small subset of user-book pairs. This method approximates R as the product of two lower-dimensional matrices, P (user embeddings) and Q (book embeddings), which represent users and books in a shared latent space. The matrices P and Q are optimized using gradient descent (details in app. B).

2.2 Neural Matrix Factorization (NeuMF)

NeuMF is a method proposed by (He et al., 2017) to capture non-linear and complex relationships in user-item interactions based on *Generalized Matrix Factorization* and a *Multilayer Perceptron*. Additionally, metadata can be incorporated into the NeuMF model by concatenating external BERT embeddings with the model's internal item representation. Due to the data sparsity, it is necessary to employ regularization techniques to maintain the model's generalization. The hyper-parameters of this model are tuned using the grid-search strategy with *Ray* library (Moritz et al., 2018) for parallel executions (details in app. D).

2.3 Leveraging Book Metadata with BM25

To enhance the model's performance, we develop a hybrid ensemble-like approach that leverages the previously collected book metadata and their BM25

embeddings to find similar books and aggregate their ratings. The underlying intuition is that ratings given by a user to similar items are likely to be similar, so, incorporating ratings from similar books would yield more reliable predictions. Similar books are retrieved using a BM25 retrieval system using the title and description of a book. For a given book b, the top k similar books based on their BM25 scores are retrieved. Additionally, we consider augmenting this set with books written by the same author(s) as b, forming a comprehensive set of related books denoted by $\mathcal{B}$. The predicted rating $R_{b,u}$ of user u for book b is then computed using a weighted averaging as in equation 1.

$$R_{b,u} = \frac{\sum_{b_i \in \mathcal{B}} r_{b_i,u} \cdot score(b, b_i)}{\sum_{b \in \mathcal{B}} score(b, b_i)} \quad (1)$$

In equation 1 $r_{b_i,u}$ represents the rating of book b_i by user u, obtained from the training data if available, or approximated using matrix factorization ($R = PQ$). The term $score(b, b_i)$ corresponds to the similarity score assigned by the BM25 model for the pair of books b and b_i. Note that the BM25 model always returns b as the top similar book, so the final predicted rating $R_{b,u}$ will always incorporate the rating from the factorization model.

Moreover, in another approach, after approximating all the ratings with MF, we aggregate each user's ratings with those of the 300 most similar users, applying a method analogous to the averaging procedure as eq. 1 used for the books' ratings.

3 Experiments

3.1 Data Split and Baselines

Since the provided data did not include a validation set, we created an artificial train-validation split from the training data to evaluate and compare our models. You can find the details in the app. C.

In the first stage the basic models such as SVD (Koren et al., 2009), NMF (Hu et al., 2008), and variants of KNN (Sarwar et al., 2001), are trained, and evaluated on the dataset to have a baseline for comparison using *Surprise* library. All of these methods' hyper-parameters are tuned using the grid-search strategy for lower root mean square deviation (RMSE) using k-fold cross-validation.

3.2 Evaluation

We performed extensive hyperparameter tuning for optimizing P and Q in Matrix Factorization (MF) and NeuMF. Detailed information about this process can be found in the Appendix.

Table 1: Performance of baseline models in comparison to our models.

Model	**RMSE**
SVD	0.890
NMF	0.978
KNN	1.076
KNN (means)	1.021
KNN (z-scores)	1.017
KNN (baseline)	0.939
MF	0.813
MF with BM25	0.787
MF with BM25, Users	0.781
MF with BM25, Authors	0.788
MF with BM25, Authors, Users	0.782
NeuMF	0.856
NeuMF with Embeddings	0.850

Table 1 presents the RMSE results of the various methods explored in our experiments. For MF, we considered the following settings and their combinations: the baseline MF model, MF enhanced with BM25-based content integration, MF integrating content from books by the same authors, and MF incorporating similar users' ratings. When integrating content features (from books or authors), we subsequently incorporated ratings from similar users. For additional implementation details, please refer to the associated Kaggle notebook.

3.3 Results

As we can see, MF and its variants outperformed the other methods. Although the MF model augmented solely with BM25 did not surpass all other integrated approaches, its results remained comparable or even superior to most alternatives. We interpret this to mean that when optimizing the user and item latent matrices, MF allows the flow of information across books and users, and incorporating external content similarity (as provided by BM25) can further enhance this process.

4 Conclusion and Discussion

This study explored developing a book recommendation system for predicting user ratings employing the metadata gathered for the books. Across all the models developed for this project, this report illustrates that a hybrid approach based on matrix factorization and content-based collaborative filtering outperforms the others based on experimental results.

References

Google. n.d. Google books api. Accessed: 2024-12-01.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 173–182, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative non-negative matrix factorization for recommender systems. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*, pages 426–434. ACM.

ISBNdb. n.d. Isbn db api documentation. Accessed: 2024-12-01.

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

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. 2018. Ray: a distributed framework for emerging ai applications. In *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, OSDI'18, page 561–577, USA. USENIX Association.

Open Library. n.d. Open library api. Accessed: 2024-12-01.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web (WWW 2001)*, pages 285–295. ACM.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

A Data Collection

In the first stage, Open Library API has been leveraged since it is freely available without a quota. In the second stage, the missing values have been filled with Google Books API, which is also free but applies a restriction on the number of queries per day. In the last stage, ISBNdb, which requires payment to access has been leveraged to fill the remaining missing values.

B Matrix Factorization

We aim at approximating the user-book rating matrix R, which is a sparse matrix of size $|U||B|$, where $|U|$ is the number of users and $|B|$ is the number of books. To approximate R, we represent it as the product of two low-dimension matrices, P and Q, which encode users and books in a shared latent space of dimension d:

$$R \approx PQ^T, \textit{ with } P \in \mathbb{R}^{|U| \times d}, Q \in \mathbb{R}^{|I| \times d}$$

Our goal is to learn the matrices P and Q by minimizing the reconstruction error, which is defined as:

$$e_{ij}^2 = (r_{ij} - \sum_{k=1}^{d} p_{ik} q_{jk})^2 + \lambda_P ||P||^2 + \lambda_Q ||Q||^2$$

where λ_P and λ_Q are regularization parameters. To minimize this error, we apply gradient descent, iteratively updating the elements of P and Q as follows:

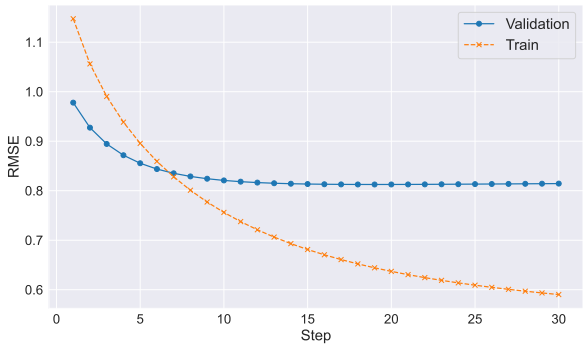
$$p_{ik} \leftarrow p_{ik} + 2\alpha(e_{ij} q_{jk} - \lambda_P p_{ik})$$

$$q_{jk} \leftarrow q_{jk} + 2\alpha(e_{ij} p_{ik} - \lambda_Q q_{jk})$$

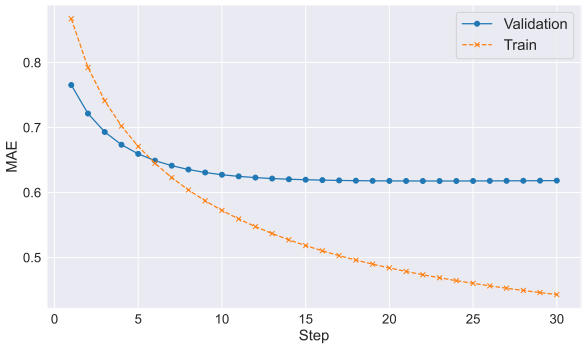
where α is the learning rate, and e_{ij} is the prediction error for a given user-item pair. These updates are applied only for (i, j) pairs where a rating r_{ij} exists, ensuring that the model learns from the available data while efficiently handling the sparsity of R. To be mentioned, we clip the values of $2\alpha(e_{ij} q_{jk} - \lambda_P p_{ik})$ and $2\alpha(e_{ij} p_{ik} - \lambda_Q q_{jk})$ to be between -1000 and 1000 in order to avoid numerical issues.

C Train and Validation Data Split

To avoid cold-start issues, we selected user-book pairs where users had rated at least 5 books and books had at least 5 ratings. Each pair was added to the validation set only if both the user and book



(a) RMSE



(b) MAE

Figure 1: Measurements vs. optimization steps during training the matrix factorization model.

were not already included. Using this method, the original dataset with 100523 ratings was split into a training set with 96704 ratings and a validation set with 3819 ratings.

D NeuMF Hyper-parameter Tuning

The hyper-parameter tuning of NeuMF has been done for its intermediate representation dimension on the set of $\{4, 8, 16\}$, the architecture of its multilayer perceptron on the set of $\{[8, 4], [16, 4], [16, 8], [32, 16, 4], [32, 16, 8]\}$, the dropout rate on set of $\{0.1, 0.15, 0.20, 0.25\}$, the value of learning rate on set of $\{1e-2, 5e-3, 1e-3\}$, the value on weight decay $\{e^{-3}, e^{-4}, e^{-5}\}$ and the batch size on set of $\{512, 1024, 2048, 4096\}$ using grid-search strategy with ray library. The chosen values are in tab. 2.

Hyper-parameter	Plain	With BERT
Repr. Dimention	4	4
Architecture	$4 \rightarrow 8$	$4 \rightarrow 8$
Dropout	0.1	0.1
Learning Rate	0.001	0.01
Weight Decay	0.001	0.001
Batch Size	2048	512

Table 2: Selected hyper-parameter values for the final configuration of MF method.

E MF Hyper-parameter Tuning

We used Weights & Biases (W&B) Sweep to fine-tune the model. The sweep was configured to run for 512 random trials, each aiming to minimize the evaluation loss. The primary hyper-parameters tuned were the learning rate (lr) with the search space of $[0.00001, 0.1]$, the regularization terms (λ_P, λ_Q) with the search space of $[0, 1]$, and the factorization dimension (d) searching between the set of $\{1, 2, 8, 32, 128, 512\}$. Each run executed a fixed number of iterations to systematically explore the hyper-parameter space. By utilizing random search, we ensured coverage across a broad range of values, ultimately identifying configurations that improved the model's performance.

We also performed a grid search for the BM25 parameter k from 1 to 100. Due to the large search space, the number of similar users considered (300) and other hyper-parameters were chosen empirically. The final hyper-parameters are provided in Table 3.

Hyper-parameter	Chosen Value
lr	0.004
λ_P	0.494
λ_Q	0.132
d	8
k_{BM25}	20
# of Optimization Steps	20

Table 3: Selected hyper-parameter values for the final configuration of MF method.

F Experiments Plots

Fig. 1 illustrates the trend of measurements, mean absolute error (MAE), and RMSE, over the training steps of MF.