Book Recommendation Systems

HAY Team : Youssef Belghmi, Amine Belghmi, Hamza Morchid
Distributed Information Systems (CS-423), Department of Computer Science, EPFL, Switzerland

***Abstract*—This project aims to develop a recommendation system capable of predicting book ratings using user preferences from a dataset of over 16,000 books. Leveraging user-book interaction data, we explored and assessed various methods to enhance prediction accuracy. The study offers insights into the design of efficient and scalable recommendation systems, addressing key challenges and trade-offs, while presenting and interpreting the results.**

I. Introduction

Recommendation systems utilize user interests and behaviors to deliver tailored suggestions. In the case of books, they help users find new titles that match their preferences. In this project, we aim to develop a recommender system to predict book ratings based on previous data. We implemented and evaluated several approaches: collaborative filtering, which relies on user-item interaction models; content-based recommendation, which leverages textual embeddings of book metadata to calculate cosine similarities; and matrix factorization, which incorporates latent factors and biases to enhance prediction accuracy. This paper compares these methods to identify the most effective solution while highlighting their strengths and limitations.

II. Collaborative Filtering Methods

Collaborative filtering is one of the most widely used approaches in recommendation systems. Predicts user preferences by analyzing patterns of interactions between users and items. The core idea is that users who have shown similar preferences in the past are likely to share similar tastes in the future.

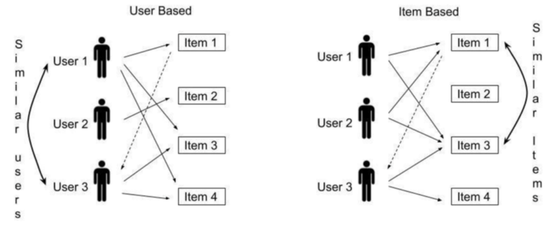


Figure 1. Illustration of collaborative filtering methods [1]

A. User-Based Collaborative Filtering

The user-based collaborative filtering approach predicts a user's rating for an item by analyzing the ratings provided by other users who share similar preferences, called neighbors. The key is to identify these neighbors and aggregate their ratings to estimate the target user's potential rating.

In our implementation, we constructed a user-item matrix representing the interactions between users and books, where each row corresponds to a user and each column corresponds to a book. To measure similarity between users, we compute the cosine similarity between the two vector representations of the two users. For each queried user-book pair, we predicted the rating the user will give to the book by the weighted average of the ratings of similar users for the queried book. The average is weighted by the similarity degree of users to the queried user.

The evaluation of this method yielded a Root Mean Square Error (RMSE) of **1.01336**, reflecting the average deviation of predicted ratings from actual ratings. Although this result demonstrates that user-based collaborative filtering can provide consistent predictions, the RMSE indicates that the method does not achieve high accuracy. The relatively high RMSE highlights the limitations of this approach, particularly its reliance on finding sufficiently similar users in sparse datasets or when users have rated few items.

A major limitation of User-Based Collaborative Filtering approach is its sensitivity to data sparsity. When users have limited interactions with items, finding reliable neighbors becomes challenging, leading to less accurate predictions.

B. Item-Based Collaborative Filtering

Item-based collaborative filtering offers a compelling alternative to the user-based approach, particularly in datasets where user preferences are sparse. Instead of analyzing user-user similarities, this method evaluates the similarity between items themselves. The premise is straightforward: if two books are often rated similarly by various users, they are likely to share attributes that appeal to those users. This approach can often provide more stable recommendations, as item similarities tend to remain consistent over time, unlike user preferences, which can be more volatile.

The key distinction between item-based and user-based collaborative filtering lies in how similarities are computed and used. While user-based filtering focuses on identifying similar users to predict ratings, item-based filtering identifies similar books. This shift often reduces the impact of sparsity in user ratings, as books tend to accumulate more interactions than individual users. Consequently, the item-item similarity matrix is generally denser and more robust compared to the user-user similarity matrix. In our implementation, we leveraged this advantage by predicting a user's rating for a book based on their ratings for similar books, weighted by their respective similarity scores.

Our implementation achieved a Root Mean Square Error (RMSE) of **0.87414**, a marked improvement over the collaborative user-based filtering approach. This score reflects a much closer alignment between predicted and actual ratings. The improved performance can be attributed to the stability and robustness of item-item relationships, which are less influenced by individual user behavior.

However, while item-based collaborative filtering significantly improved accuracy, it is not without limitations. This method assumes that sufficient data exists to construct meaningful item similarities. For rare or niche books with limited user feedback, the predictions may lack reliability.

III. Content-Based Recommendation Approach

Content-based recommendation serves as a valuable alternative to collaborative filtering, particularly in scenarios where interaction data is sparse or unavailable. Instead of relying on user-item interactions, this method focuses on the intrinsic attributes of items, such as textual metadata, to generate recommendations.

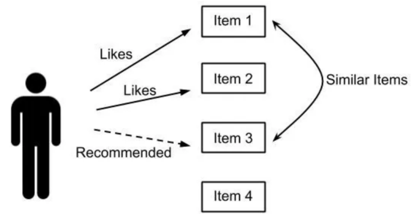


Figure 2. Illustration of content-based recommendation method [1]

To build the metadata for each book, we used two APIs. The Open Library API provided the title, author, and themes of the books, while the Google Books API was used to retrieve detailed descriptions. By combining these metadata elements, we constructed a rich textual representation for each book, which served as the basis for our recommendation system. For books without metadata or for which the APIs could not retrieve complete information, fallback mechanisms such as default values were implemented, but these cases added challenges to the method's effectiveness.

Once we had textual information about each book of the dataset, we used the all-MiniLM-L6-v2 sentence transformer [2] to create embeddings for each book. The sentence transformer maps sentences and paragraphs to a 384 dimensional dense vector space and can be used for tasks like clustering or semantic search. Once we had embeddings for each book of the dataset, we computed a (nbooks, nbooks) item symmetric similarity matrix using cosine similarity that represents similarity between books. Finally the predicted rating of a user for a given book was computed as an average of the ratings the queried user gave to other books, weighted by the similarity between these books and the queried book.

Our implementation achieved a Root Mean Square Error (RMSE) of **0.82904**, marking a significant improvement over both user-based and item-based collaborative filtering methods seen previously. This result demonstrates the potential of content-based systems in leveraging rich metadata to enhance prediction accuracy. The inclusion of detailed descriptions and themes from external APIs significantly contributed to this performance.

However, this method is not without its challenges. A major limitation lies in the process of extracting metadata for each book. Some ISBNs in the dataset were missing or invalid, making it impossible to retrieve metadata for the corresponding books. Additionally, while rare, certain books were unavailable on the APIs or lacked detailed descriptions, further complicating the prediction process and impacting accuracy. These limitations underscore the dependency of content-based approaches on the completeness and quality of metadata, which can constrain their effectiveness in real-world applications. Additionally, content-based systems may lack diversity, as they tend to recommend items similar to those already rated by the user, potentially reinforcing existing preferences without encouraging exploration.

IV. Matrix Factorization (SVD-like Approach)

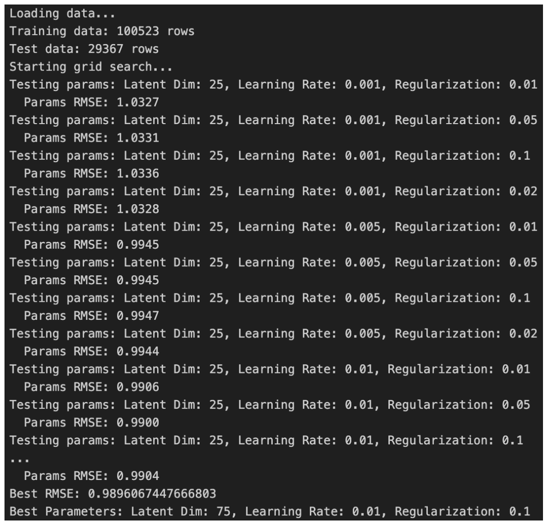
Matrix factorization is a powerful and scalable technique for recommendation systems. It decomposes the user-item interaction matrix into latent factors, capturing hidden features that explain user preferences and item characteristics. By incorporating bias terms for users and items, the SVD-like approach improves accuracy by accounting for systematic tendencies, such as consistently high-rating users or universally popular items. This method effectively balances complexity and predictive power, making it ideal for large, sparse datasets, while also enabling personalized recommendations at scale.

To optimize the performance of our matrix factorization model, we conducted a grid search to tune key hyperparameters. The dataset was first divided into two subsets: a training set for learning the model parameters and a validation set for hyperparameter tuning. Grid search systematically tests different combinations of hyperparameters to identify the configuration that minimizes prediction error. This method ensures a comprehensive exploration of parameter spaces, enabling the selection of optimal settings. The hyperparameters tested included:

- **Latent dimensions**: The size of latent factor vectors, controlling model's capacity to capture patterns.
- **Learning rates**: The step sizes for updating parameters during the training step.
- **Regularization parameters**: Terms added to the loss to penalize large parameter values and prevent overfitting.

For each combination, the model was trained on the training set and evaluated on the validation set using RMSE as the performance metric. After testing various configurations, the best hyperparameters were identified:

- **Latent dimensions**: 75, allowing the model to uncover subtle patterns in the data.
- **Learning rate**: 0.01, ensuring stable convergence without overshooting the optimal solution.
- **Regularization parameter**: 0.1, effectively reducing overfitting, making the model robust to noise.



```
Loading data...
Training data: 100523 rows
Test data: 29367 rows
Starting grid search...
Testing params: Latent Dim: 25, Learning Rate: 0.001, Regularization: 0.01
  Params RMSE: 1.0327
Testing params: Latent Dim: 25, Learning Rate: 0.001, Regularization: 0.05
  Params RMSE: 1.0331
Testing params: Latent Dim: 25, Learning Rate: 0.001, Regularization: 0.1
  Params RMSE: 1.0336
Testing params: Latent Dim: 25, Learning Rate: 0.001, Regularization: 0.02
  Params RMSE: 1.0328
Testing params: Latent Dim: 25, Learning Rate: 0.005, Regularization: 0.01
  Params RMSE: 0.9945
Testing params: Latent Dim: 25, Learning Rate: 0.005, Regularization: 0.05
  Params RMSE: 0.9945
Testing params: Latent Dim: 25, Learning Rate: 0.005, Regularization: 0.1
  Params RMSE: 0.9947
Testing params: Latent Dim: 25, Learning Rate: 0.005, Regularization: 0.02
  Params RMSE: 0.9944
Testing params: Latent Dim: 25, Learning Rate: 0.01, Regularization: 0.01
  Params RMSE: 0.9906
Testing params: Latent Dim: 25, Learning Rate: 0.01, Regularization: 0.05
  Params RMSE: 0.9900
Testing params: Latent Dim: 25, Learning Rate: 0.01, Regularization: 0.1
...
  Params RMSE: 0.9904
Best RMSE: 0.9896067447666803
Best Parameters: Latent Dim: 75, Learning Rate: 0.01, Regularization: 0.1
```

Figure 3. Steps and Results of Grid Search for the Hyperparameterization

Using these optimal hyperparameters, selected because they provided a balance between capturing complex interactions and maintaining generalization, the model was trained on the entire training dataset for 20 epochs. The number of epochs was chosen to ensure sufficient iterations for convergence while avoiding excessive training that might lead to overfitting. The predicted rating $\hat{R}_{ui}$ for a user u on an item i is calculated as:

$$\hat{R}_{ui} = \mu + b_u + b_i + P_u \cdot Q_i^T$$

where μ is the global mean rating, b_u and b_i are the user and item biases, and P_u and Q_i are the latent factor vectors for users and items, respectively. The training process minimizes the following loss function:

$$\mathcal{L} = \sum_{(u,i)} \left(R_{ui} - \hat{R}_{ui}\right)^2 + \lambda \left(\|P_u\|^2 + \|Q_i\|^2 + b_u^2 + b_i^2\right)$$

with R_{ui}, the observed rating, and λ, the regularization term.

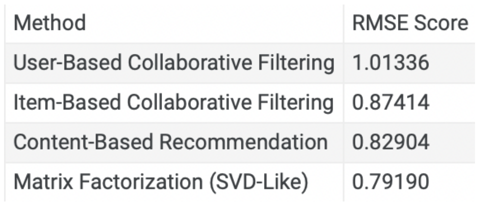
Once trained, the model was used to predict ratings for unseen user-item pairs in the test set. The final Root Mean Square Error (RMSE) achieved was **0.79190**, the best result among all methods tested. This low RMSE demonstrates the model's ability to capture user preferences and item features.

Despite its strengths, the SVD-like approach has limitations. Training can be computationally intensive, particularly for large datasets, as multiple iterations over the data are required for convergence. Additionally, the model struggles with cold-start problems, where new users or items lack interaction data. Nevertheless, the combination of bias terms and latent factors makes matrix factorization a robust and scalable solution for recommendation systems, particularly when sufficient interaction data is available.

V. Summary

A. Project Overview

This project aimed to develop a book recommendation system by evaluating collaborative filtering, content-based recommendation, and matrix factorization methods to predict ratings based on user preferences and metadata, focusing on balancing accuracy and computational efficiency.



Method	RMSE Score
User-Based Collaborative Filtering	1.01336
Item-Based Collaborative Filtering	0.87414
Content-Based Recommendation	0.82904
Matrix Factorization (SVD-Like)	0.79190

Figure 4. RMSE Score Obtained for Each Method

Among the methods tested, we selected the matrix factorization approach (SVD-like) for the final Kaggle submission due to its superior RMSE score, despite the computational demands associated with this method. While matrix factorization requires multiple iterations over the dataset to converge and typically involves a longer running time, limiting the training to 20 epochs allowed us to stay within the 10-minute running time constraint while ensuring the optimality of our results.

B. Conclusion

This project highlights the potential and challenges of building effective recommendation systems. Collaborative filtering, though simple and intuitive, struggled with sparse data. Content-based methods provided competitive results but were limited by their reliance on metadata completeness. Matrix factorization achieved high accuracy while balancing predictive power and computational constraints. Future work could explore hybrid approaches to combine the strengths of these methods and address limitations like the cold-start problem for improved accuracy.

References

[1] Ankita Prasad, "Medium, Level Up Coding: The Mathematics of Recommendation Systems," 09/2020, Available [Online]: https://levelup.gitconnected.com/the-mathematics-of-recommendation-systems-e8922a50bdea.

[2] Reimers Nils and Gurevych Iryna, "Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing," 11/2019, Available [Online]: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.