Project 16 - Book Recommendation Engine

***Abstract* — Recommendation systems are vital for helping users discover new books from an enormous, evergrowing catalogue. In the domain of books, the *GoodReads* platform, a widely popular book cataloging website, has become a hub for readers to discover, discuss, and rate books. Hence, the *GoodReads10k* dataset (10,000 books, 6 million ratings) was used to provide a comprehensive exploration and evaluation of various recommendation techniques. The pipeline encompasses multiple approaches: a *baseline model,* basic and optimized version of *collaborative (user and item-based)*, *content-based filtering,* a *hybrid* model and a deep-learning based *two-tower model*. Necessary features were extracted and pre-processed, then were modeled and evaluated using accuracy metrics like *RMSE* for rating predictions, and *Precision@10, Recall@10,* and *HitRate@10* for top-10 recommendations.**

Keywords—Goodreads10k, baseline, weighted baseline, collaborative-filtering, content-based filtering, hybrid model, two-tower model, RMSE, precision, recall, hitRate

I. Introduction

In an era defined by exponentially increasing products in every sphere, recommendation systems have become an indispensable tool for navigating through these vast repositories of information. Recommendation systems play an essential role in improving a user's experience on an online platform - from e-commerce (Amazon) to media streaming services (Netflix) - being the two main giants in the sphere. Within this landscape, book recommendation provides a unique set of challenges and opportunities as readers' choices are influenced by a multitude of contextual, cultural, and personal factors, and it can be hard to quantify what exactly constitutes a 'good' book recommendation. Hence, the subjective nature of book recommendation is a compelling area for research.

II. Dataset Description

The GoodReads10k[1] dataset is utilized, which is substantial, containing approximately 6 million ratings for 10,000 books from over 50,000 users. The entire dataset consists of multiple CSV files with metadata about books, their corresponding tags, and user ratings. Additionally, another resource was incorporated to enrich the existing book metadata. Each file is described as follows:

1. **Books.csv** contains rich metadata for 10,000 most popular books on GoodReads with the following features: unique identifiers (*id, books_id*) descriptive information (*title, authors, original_publication_year*), and metrics of popularity (*average_rating, ratings_count, image URLs*).
2. **Ratings.csv** captures explicit feedback in the form of ratings on a scale of 1-5 from over 50,000 users on these 10,000 books. It consists of 3 columns - *user_id, book_id* and *rating*. The *book_id* corresponds to *book_id* from the books.csv file.
3. The **Book_tags.csv** file provides a mapping between *goodreads_book_id* and *tag_id*, along with a count representing how frequently a particular tag was applied to a book by users. These tags offer valuable insights into the genres and topics associated with each book from the perspective of the Goodreads community.
4. The **tags.csv** file serves as a translation table, providing the actual tag name for each *tag_id* found in *book_tags.csv*.
5. An additional resource, **books_enriched.csv** file was added[2] which enriches the original booksmeta with book summary and official genre labels, providing richer features for our content-based models.

III. Methods

Due to the substantial size of the dataset, initial cleaning and preparation steps were essential prior to modeling.

1. **Ratings:** To reduce sparsity and improve training stability, users were filtered out with fewer than 50 ratings, retaining only "*active*" users. From this subset, 20% of active users were randomly sampled (keeping all of their ratings), resulting in a working set of ~1.18 million records. Evaluation of a 50% sample was also done, which produced nearly identical modeling results but incurred substantially higher compute cost, so we proceeded with the more efficient 20% dataset.

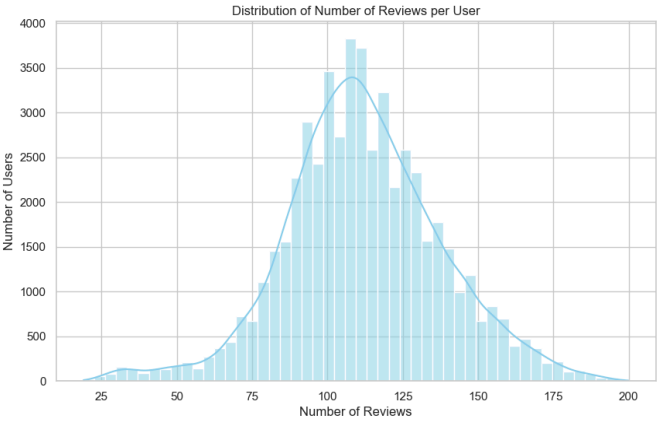

Fig I. Distribution of the Number of Reviews Per User

2. **Book Tags and Tags:** The book and tag datasets were merged based on *tag_id*, and duplicate book-tag pairs were removed, retaining only tags applied to at least *300* distinct books. To address

noisy, user-generated tag variations (e.g., "comic," "comic-books," "comics"), we used TF-IDF vectorization on the remaining tags to cluster them into *70* semantic groups using k-means clustering, and cluster validity were visually validated using t-SNE. Subsequently, approximately 40 normalized genre groups were manually created, collapsing hundreds of raw tags into consistent, interpretable categories (e.g., consolidating all comic-related tags under the unified label "comics") to produce a clean, unified tag feature for each book.

3. **Books:** Normalized tags were merged with the core book metadata and then joined in descriptions and official genres from the enriched dataset, parsed and combined each book's genres+tags to complete the final books dataset.

1. Baseline Model

Establishing a baseline is essential for quantifying the actual improvements gained by implementing more sophisticated algorithms. For this project, two baseline models were considered:

1.1. Non-personalized popularity predictor: Each book's occurrences were counted within the training set (i.e., the number of ratings), and the top-10 most-rated books were recommended to every user.

1.2. Weighted Baseline Model: A weighted popularity predictor (created by IMDb) balances the rating volume (R) and count (v), filtering out unreliable ratings using the following formula:

$$score = \frac{v}{v+m}R + \frac{m}{v+m}C$$

where m is the 90th percentile vote count and C is the global mean. The books are then ranked by this weighted score for the top-10 list.

2. Collaborative Filtering (Memory-based)

Collaborative Filtering bases recommendations on observed user-item interactions, assuming that *"users with similar tastes in the past will have similar tastes in the future"*. Unlike content-based methods, CF does not require item descriptions or metadata - it relies only on the user-item rating matrix. Two CF approaches were explored: *memory-based* (neighbor models) and *model-based* (SVD matrix factorization).

2.1. User-based Collaborative Filtering

This method identifies users who have similar rating histories as the target user and then recommends books that those similar users have liked. *KNNBasic* (with user_based=True) from the *Surprise*[3] library - a scikit-learn based recommendation library in Python - was used to create a user-item matrix from the filtered ratings data, where rows represent users and columns represent books, and the entries are the ratings given. Then, the similarities are compared between users using *cosine-similarity*. Once similar users are identified, the system recommends books that these neighbors have rated highly but the target user has not yet interacted with.

2.2. Item-based Collaborative Filtering

This method focuses on identifying item similarities based on user ratings. The *KNNBasic* algorithm *(user_based=False)* computes an item-item similarity matrix. If a user has positively rated certain books, the system recommends similar books from this matrix. Item-based CF is efficient when users significantly outnumber items (50,000 users vs. 10,000 books), allowing item-item similarities to be precomputed and reused.

3. Optimized Collaborative Filtering (Model-based)

Pure memory-based CF has drawbacks. It struggles with very sparse users or items (cold start), and it can produce unexplainable recommendations. If many users happened to rate two books similarly, even though their content is unrelated, such recommendations can perplex other users. This motivates a more model-based approach which imbues domain knowledge into the recommendations.

Model-based CF seeks to learn latent factors that explain ratings. The most famous approach is Matrix Factorization (MF), popularized by the *Netflix Prize-winning algorithm*, which approximates the large user-item rating matrix with a product of lower-dimensional matrices. Two variants of matrix factorization were implemented: a basic *SVD* model and an *SVD++* model with a bias term and implicit feedback.

3.1. CF using Matrix Factorization (SVD)

Singular Value Decomposition (SVD) is a popular matrix factorization technique that decomposes the user-item matrix into three matrices (U, Σ, V^T) to capture the underlying structure of user preferences and item attributes.

Each user u_i is associated with a vector p_u (here d=50 dimensions) and each book i with a vector q_i. The model predicts a rating by computing the dot product $p_u \cdot q_i$. We then minimize the sum of squared errors between predicted and actual ratings by adding a penalty on large vector magnitudes (L2 regularization) to prevent overfitting.

3.2. CF using Matrix Factorization with Bias (SVD++)

Building on basic SVD, we incorporated implicit feedback by adding an extra term that models the influence of all items a user has interacted with (not just those they've rated). This SVD++ variant further refines predictions in sparse datasets by capturing both explicit ratings and implicit exposure. During training, each user's profile is augmented by a second vector that summarizes all items they have interacted with. By blending explicit and implicit signals, SVD++ improves predictions for users with few ratings. In our experiments, this model consistently outperformed all other models in both rating accuracy and top-K recommendation metrics.

4. Content-Based Filtering Using Leave-One-Out

In this model, a TF-IDF content filter is implemented by concatenating the user-generated tags of a book and its

description into a single field. The raw text obtained is cleaned by converting each character into lowercase, splitting it into tokens, removing non-alphanumeric characters, punctuations and it is then filtered against a list of English stopwords from NLTK. This cleaned text is then passed through a TF-IDF vectorizer which results in the text being converted into high dimensional vectors where tags and distinctive words have greater weights.[16]

A cosine similarity matrix is then computed over the TF-IDF vectors obtained so as to ascertain how similar two books are which assists the model in recommending books which are similar to the genre or any other characteristic of a book that they might like based on the books that they have read before. The model then uses leave-one-out[15] to hide one of a user's books rated greater than or equal to 4.0 (which is generally a book that the user has already read) to mimic a realistic recommendation system. The rest of the high rated books form a user's profile where each profile and each book has their cosine similarity aggregated which helps to generate top K recommended books for a user.

5. Two-Tower Recommendation System

A Two-Tower model is a type of neural network recommendation architecture which has two separate deep neural networks (towers). One tower is for users while the other is for items, and each network learns to embed their inputs into the same low-dimensional vector space that they exist in. Each tower independently learns embeddings in a shared low-dimensional space: the item tower produces embeddings from book features, while the user tower generates embeddings from reading histories.[9]

The two towers are then jointly trained on the user-book interaction data obtained to ensure high similarity between embeddings of users and books they interacted with, and low similarity with unvisited books, thus effectively capturing personalized interactions.

The primary reason why the Two-Tower model was implemented was because of its ability to scale and handle sparse (the data is considered to be sparse due to the fact there are few interactions between the many users and books in the data) and high dimensional data effectively by being able to incorporate diverse features per item or user with the help of the two deep towers used. The model also enables efficient retrieval due to the fact that item and user representations are learned independently.

At inference, the model computes user embeddings and retrieves top-K book recommendations by efficiently matching these embeddings against precomputed item embeddings using approximate nearest neighbor search techniques.

6. Hybrid Recommendation

The hybrid recommendation is implemented by blending collaborative filtering and content-based filtering. First, user-generated tags for each book were vectorized using TF-IDF and then reduced to a lower-dimensional embedding via Truncated SVD[14]. A profile vector for each user was created by averaging the embeddings of all books they rated ≥ 4. Each candidate book was scored according to its cosine similarity to the user profile, the degree of author overlap, its normalized popularity, its average rating, and the user's overall rating bias.

These combined features were fed into a LightGBM[13] regressor trained to predict the rating a user would assign to unrated books. By integrating collaborative signals (through tag co-occurrence and rating behavior) with item content features, this hybrid model delivers more nuanced and personalized recommendations than either pure collaborative filtering or content-based filtering alone.

IV. Results

The performance of the models are summarized below:

TABLE I. Comparative evaluation of all models in terms of RMSE

Model	Metric
	RMSE
Baseline Popularity	0.9562
Weighted Popularity	0.9635
User-User CF	0.9580
Item-Item CF	0.8929
SVD (MF)	0.8492
SVD with bias (MF)	0.8473
Content Based	**0.8364**
Two Tower	0.8649
Hybrid	0.8053

TABLE II. Comparative evaluation of all models in terms of top-10 ranking performance

Model	Metric		
	Precision@10	**Recall@10**	**HitRate@10**
Baseline Popularity	0.0669	0.0302	0.4444
Weighted Popularity	0.0437	0.0195	0.3135
User-User CF	0.7601	0.5312	0.9973
Item-Item CF	0.7179	0.5019	0.9968
SVD	0.7755	0.5443	0.9977
SVD with Bias	**0.7774**	**0.5457**	**0.9980**
Content Based	0.0004	0.0043	0.0043
Two Tower	0.0022	0.0015	0.0211
Hybrid	0.0467	0.0220	0.3000

1. **RMSE:** This metric evaluates the difference between predicted and actual user ratings, indicating how accurately a model predicts user preferences. The Hybrid model achieved the best performance with the lowest RMSE of **0.8053**.

Content-Based Filtering and SVD with Bias also performed strongly, with Content-Based Filtering achieving an RMSE of **0.83**. Additionally, the Two-Tower model (**0.86**) and Item–Item Collaborative Filtering (**0.89**) showed substantial improvements over the popularity-based baselines.

2. Precision@10: This metric indicated the top 10 books that the user genuinely liked (ratings ≥ 4). SVD with bias (MF) model along with User-User CF indicating that the models are able to predict relevant books more frequently. In contrast, content-based filtering had notably low precision, highlighting its inability to capture the nuanced, multi-topic tastes of readers. Even though it excels at matching on explicit text features, it struggles whenever user interests extend beyond straightforward content similarity.

3. Recall@10: This metric measures the fraction of all relevant items in the top 10 recommendations. The SVD with Bias (MF) model achieved the highest recall among the eight models, with a score of **0.5457**, followed closely by the standard SVD (MF) model, which recorded a recall of **0.5443**.

4. HitRate@10: This metric measures whether at least one relevant item appears in the top 10 recommendations for each user. In this evaluation, the top-performing models are SVD with Bias (Matrix Factorization) and standard SVD (MF), with SVD with Bias achieving the highest Hit Rate of **0.9980**. These results indicate that the models are effective in consistently providing users with at least one relevant recommendation.

V. Discussion

It is observed that even though some models seemed to be performing well based on the fact that their RMSE is lower than the Baseline Model Although *Content-Based filtering* attains the lowest RMSE, it fails to recommend books users actually care about - in both precision and hit-rate it is near zero. Similarly, the two-tower model, trained purely on MSE, predicts ratings reasonably well (RMSE **0.8649**) but does not translate into effective top-N recommendations. This was the main reason that multiple metrics were used so that each model's effectiveness could be properly ascertained.

On the other hand, the *SVD with bias (MF)* model is the most effective model for recommending books. By correcting for individual rating habits and overall book popularity, it achieves RMSE of **0.8473**, highest precision of **0.7774**, recall of **0.5457**, and a hit rate of (**0.9980**). Simple neighborhood methods: user-user and item-item CF aren't far behind, with precision around **75%** and hit-rates above **99%,** proving that classic collaborative filtering still rules when it comes to top-N recommendations.

VI. Conclusion

Overall, the *bias-aware SVD* model emerges as the clear leader, combining low prediction error (RMSE **0.8473**) with outstanding ranking performance (Precision@10 **0.7774**, Recall@10 **0.5457**, HitRate@10 **0.9980**). *Basic SVD* follows closely, offering nearly identical top-N quality alongside a slightly higher RMSE (**0.8492**). Both models shine when accuracy and personalized ranking matter most.

Classic neighborhood methods: item-item and user-user CF remain excellent alternatives, delivering precision around **75%** and hit-rates above **99%** with minimal computational overhead and straightforward interpretability. In practice, *bias-aware SVD* is the go-to choice for applications that demand the very best in both rating prediction and recommendation ranking, while *item-based CF* offers a fast, transparent solution when ease of deployment and explainability are priorities.

References

[1] Z. Zając, goodbooks-10k dataset, GitHub, 2025 [Online]. Available: https://github.com/zygmuntz/goodbooks-10k

[2] O. Simard-Hanley, Goodbooks-10k-extended. GitHub, 2025, [Online].Available: .https://github.com/malcolmosh/goodbooks-10k-extended

[3] Surprise Library, A Python scikit for building and analyzing recommender systems, 2025, [Online]. Available: https://surpriselib.com/

[4] Evidently AI, Evaluating Recommender Systems, 2025 [Online]. Available:https://www.evidentlyai.com/ranking-metrics/evaluating-recommender-systems#predictive-quality-metrics.

[5] G. Torbati, A. Tigunova, and G. Weikum, Unveiling Challenging Cases in Text-based Recommender Systems, CEUR-WS, vol. 3476, 2024, [Online]. Available: https://ceur-ws.org/Vol-3476/paper5.pdf

[6] S. Prabha, Introduction to Recommender Systems, Medium, 2020. [Online]. Available: https://medium.com/@swetaprabha/introduction-cbc02aedc03d

[7] S. Chirravuri and K. Immidi, Major Challenges of Recommender System and Related Solutions, International Journal of Innovative Research in Computer Science & Technology , 2022, [Online]. Available:https://www.ijircst.org/DOC/3-major-challenges-of-recommender-system-and-related-solutions.pdf

[8] D. Bakde, S. Girase and D. Mukhopadhyay, Matrix Factorization Model in Collaborative Filtering AlgorithmsL A Survey, Procedia Computer Science, vol. 49, pg 136-146, 2015, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050915007462

[9] J. Totten, J. Wortz and L. Sethu, Implement Two-Tower Retrieval For large-scale Candidate Generation, Google Cloud, 2025, [Online]. Available:https://cloud.google.com/architecture/implement-two-tower-retrieval-large-scale-candidate-generation

[10] A. Jaiswal, Towards a Theoretical Understanding of Two-Stage Recommender Systems, arXiv:2403.00802v1, 2024, [Online]. Available: https://arxiv.org/html/2403.00802v1

[11] J. Tae, Recommendation Algorithm with SVD, GitHub, 2025, [Online], Available: https://jaketae.github.io/study/svd/

[12] R. Salakhutdinov and A. Mnih, Probabilistic Matrix Factorization,, 2007, [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2007/file/d7322ed717dedf1eb4e6e52a37ea7bcd-Paper.pdf

[13] G. Ke et. al, LightGBM: A Highly Efficient Gradient Boosting Decision Tree, Advances in Neural Information Processing Systems 30, 2017 , [Online]. Available: **https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html**

[14] S. Deerwester, S. Dumais, G. Furnas, T. Laundauer and R. Harshman, Indexing by latent semantic analysis, Journal of the American Society

for Information Science, vol. 41, pg 391-407, 1990, [Online]. Available:**https://asistdl.onlinelibrary.wiley.com/doi/10.1002/(SICI)1097-4571(199009)41:6%3C391::AID-ASI1%3E3.0.CO;2-9**

[15] T. Pathak, Understanding and Implementing Leave-One-Out Cross Validation for Measuring Accuracy of Recommendation Systems, Medium, 2024, [Online]. Available: https://medium.com/@tejupathak/understanding-and-implementing-leave-one-out-cross-validation-for-measuring-accuracy-of-425b62b01d38

[16] T. Phalle and S. Bhushan, Content Based Filtering and Collaborative Filtering: A Comparative Study, Journal of Advanced Zoology, 2024, [Online]. Available: https://www.researchgate.net/publication/378841543_Content_Based_Filtering_And_Collaborative_Filtering_A_Comparative_Study