A Hybrid Recommendation System for Book Discovery

Aayush Rautela
Cagla Kuleasan

Abstract—Recommendation systems are crucial tools for navigating vast digital catalogs and discovering new items. This paper presents the design and implementation of a hybrid book recommendation system built on the goodbooks-10k dataset. The system combines a content-based model with a model-based collaborative filtering approach to leverage the strengths of both methods. The content-based component uses TF-IDF vectorization on book metadata, enhanced with N-grams and feature weighting, to compute item similarity. The collaborative filtering component employs a Singular Value Decomposition (SVD) model, which learns latent user and item factors from rating data. To optimize this model, we conducted a systematic hyperparameter search using GridSearchCV, achieving a final Root Mean Squared Error (RMSE) of 0.8182. The final hybrid system generates recommendations by calculating a weighted average of scores from both models, aiming to provide recommendations that are more accurate and diverse than either individual approach.

I. INTRODUCTION (Problem Description)

In the current digital era, consumers are faced with an overwhelming volume of choices across various domains, from e-commerce to entertainment. For book lovers, online platforms offer access to millions of titles, creating a classic "information overload" problem. Navigating this vast catalog to find new, personally relevant books is a significant challenge. Recommendation systems are a critical solution to this problem, serving as personalized guides that help users discover items they are likely to enjoy, thereby enhancing user experience and engagement.

This project addresses the challenge of creating an effective book recommendation system. Using the goodbooks-10k dataset, which contains millions of user ratings for 10,000 popular books, we aim to build a system that can provide relevant suggestions to users. The primary objective is to develop and evaluate a robust hybrid recommender system that effectively combines multiple recommendation strategies to overcome the limitations of any single approach.

Our main contribution is the implementation and analysis of a weighted hybrid system that fuses two distinct recommendation techniques. The first is a Content-Based

Filtering model, which we enhanced through several text processing steps including feature weighting, N-gram analysis, and the filtering of noisy, user-generated tags. The second is a state-of-the-art, model-based Collaborative Filtering approach using Singular Value Decomposition (SVD) to learn latent taste profiles from user rating data. We demonstrate a systematic approach to optimizing this collaborative model through hyperparameter tuning.

This paper is structured as follows: Section II reviews related works in the field of recommendation systems. Section III provides a detailed analysis of the dataset used. Section IV describes the proposed methodology for both the individual models and the final hybrid architecture. Section V presents our experimental results, including the hyperparameter tuning process. Section VI visualizes these results. Finally, Section VII concludes the paper with a summary of our findings and potential directions for future work.

II. RELATED WORK

- **Collaborative Filtering (CF):**
  Collaborative filtering is a widely used technique that makes automatic predictions about the interests of a user by collecting preferences from many users. The underlying assumption is that if two users agreed in the past (e.g., they both liked the same books), they are likely to agree again in the future. CF methods are content-agnostic and can reveal novel, cross-genre recommendations. They are generally categorized into two main families.
  - **Memory-Based CF:** This family uses the entire user-item interaction database to compute similarities. User-Based CF finds users with similar rating patterns to the target user and recommends items those similar users liked. Item-Based CF, conversely, computes similarity between items based on user ratings and recommends items that are similar to what the target user has liked. While intuitive, these methods can suffer from data sparsity and scalability issues, particularly with large user-user or item-item similarity matrices.
  - **Model-Based CF:** To overcome the limitations of memory-based approaches, model-based methods learn a compact model from the rating data to explain user-item interactions. A prominent technique in this family is **Matrix Factorization**, which decomposes the user-item matrix into lower-dimensional latent factor matrices for users and items. These latent factors represent hidden taste dimensions. Our project employs **Singular**

Value Decomposition (SVD), a popular matrix factorization algorithm, chosen for its scalability and strong performance in predicting ratings.

- **Content-Based (CB) Filtering:**
  Content-based filtering recommends items that are similar to those a user has liked in the past. Unlike collaborative filtering, this approach focuses on the intrinsic properties of the items themselves, such as genre, author, description, or tags. An item profile is created from these features, and the system recommends other items with similar profiles. A common technique for handling textual features, which we employ in this project, is the Term Frequency-Inverse Document Frequency (TF-IDF) vectorization method. TF-IDF converts text into a numerical representation that highlights the importance of words to a document within a collection. The similarity between these numerical vectors is then often calculated using Cosine Similarity to find and rank the most similar items.

- **Hybrid Systems:**
  Hybrid recommendation systems combine two or more recommendation techniques to achieve better performance and overcome the limitations of any single approach. For example, collaborative filtering suffers from the "new item" problem (it cannot recommend items with no ratings), while content-based filtering can address this. Conversely, content-based methods can overspecialize and may not provide novel recommendations, a weakness that collaborative filtering excels at overcoming. Several hybridization strategies exist, such as weighted, where scores from different models are combined; switching, where the system chooses a model based on certain criteria; and feature combination, where features from one model are used as inputs to another. In this project, we implement a weighted hybrid approach, which calculates a final recommendation score by taking a linear combination of the normalized scores from our Content-Based and Collaborative Filtering components.

III. DATASET ANALYSIS

The foundation of this project is the **goodbooks-10k** dataset, sourced from its public GitHub repository. This dataset contains rich interaction data linking users to books.

A. Key Statistics

The dataset provides a substantial base for building and evaluating recommendation models. The core components include:

- **Total Books:** 10,000
- **Total Users:** 53,424
- **Total Ratings:** 5,976,479
- **Rating Scale:** 1 to 5 integer stars.
- **Total Unique Tags:** 34,252

B. File Usage and Preprocessing

We utilized four main CSV files from the dataset, each serving a specific purpose for our hybrid model's components.

1. **ratings.csv:** This file is the cornerstone of our Collaborative Filtering model. It contains (user_id, book_id, rating) triplets, which are essential for learning user taste profiles.

   For example:

   user_id,book_id,rating
   1,258,5
   2,4081,4
   This raw data was used in its entirety to train our SVD model, allowing us to leverage all available user interactions without the need for data reduction or filtering that is often required by memory-based CF approaches.

2. **books.csv:** This file provides essential metadata for each book, including its book_id, title, and authors. This information is used by our Content-Based model and for displaying the final recommendations with human-readable titles.

3. **tags.csv and book_tags.csv:** These files were used exclusively for the Content-Based model to create rich feature profiles for each book. tags.csv maps a tag_id to a tag_name (e.g., "fantasy"), while book_tags.csv links book_id to tag_id and provides a count of how many times a tag has been applied to a book. Significant preprocessing was performed on this tag data to improve signal quality:
   - **Filtering by Relevance:** We only considered tags with a count greater than

100, removing very niche or personal tags.

- **Filtering by Content:** We removed common, non-descriptive "shelf" tags (e.g., "to-read", "favorites", "owned") and year-based tags (e.g., "2017-to-read") using a regular expression filter. This ensures the remaining tags describe the book's actual content (e.g., "dystopian", "romance", "non-fiction") rather than how users organize their personal libraries.

This preprocessing of tag data was a critical step in enhancing the performance of our Content-Based filtering component by focusing on features that genuinely describe a book's substance.

IV. PROPOSED METHOD

This section details the architecture of our hybrid recommendation system, which is composed of two primary components: a Content-Based filter and a Collaborative filter.

A. **Content-Based Filtering Component**

The Content-Based (CB) component recommends books based on their inherent features. Its methodology was refined through several text processing enhancements to improve the quality of its recommendations.

1. **Feature Engineering and Processing:** A key step in our method is the creation of a high-quality content profile for each book. This involved several stages:
   - **Tag Processing:** To reduce noise from user-generated tags, we first filtered the tag data. We only included tags that were applied more than 100 times across the dataset to ensure relevance. Furthermore, we used regular expressions to remove common, non-descriptive "shelf" tags (e.g., "to-read," "currently-reading," "favorites") and year-based organizational tags, ensuring that the remaining tags described the book's actual content.
   - **Feature Combination and Weighting:** The final content string for each book was constructed by concatenating its title, author(s), and the cleaned tag data. To give more importance to the book's primary identifiers, we implemented a simple feature weighting technique by repeating the title and author strings within this concatenation. This increases their term frequency, giving them more influence in the subsequent TF-IDF calculation.

2. **Text Representation:** To capture more meaningful context from our content profiles, we enhanced our text representation model in two ways:
   - **N-grams:** We configured our vectorizer to use an ngram_range of (1, 2). This allows the model to treat not only single words (unigrams) but also adjacent pairs of words (bigrams) as distinct features. This is particularly effective for capturing concepts like "Harry Potter" or "Science Fiction" as a single, more meaningful token.
   - **Vectorization with TF-IDF:** The processed content string for each book is converted into a high-dimensional numerical vector using the **Term Frequency-Inverse Document Frequency (TF-IDF)** algorithm, implemented via scikit-learn's TfidfVectorizer. This process assigns a weight to each term (unigram or bigram) that reflects its importance to a specific book in the context of the entire book collection.

3. **Similarity Calculation:** The similarity between any two books is calculated using **Cosine Similarity**, which measures the cosine of the angle between their respective TF-IDF vectors. A score close to 1 indicates high content similarity, while a score close to 0 indicates low similarity. This results in a pre-computed book-to-book similarity matrix that can be used for fast lookups during recommendation generation.

B. Collaborative Filtering Component (SVD Model)

The collaborative filtering (CF) component is powered by a model-based matrix factorization technique. This approach was chosen for its scalability and strong performance on sparse rating data.

1. **Algorithm Choice:** We selected the **Singular Value Decomposition (SVD)** algorithm, a well-regarded matrix factorization method implemented in the scikit-surprise library. SVD excels at uncovering latent features from user-item interactions to predict ratings.

2. **Training Process:** The model is trained on the full set of user-item ratings. The training process iteratively adjusts the model's parameters over a set number of **epochs**. During each epoch, the algorithm attempts to learn a set of latent factor vectors for every user and every book. These vectors represent abstract taste dimensions (e.g., for users) and feature dimensions (e.g., for books) that best explain the observed ratings.

3. **Prediction:** Once the latent factor vectors are learned, predicting a rating for any given user and book is computationally efficient. The predicted rating is estimated by taking the dot product of the user's latent factor vector and the book's latent factor vector, also taking into account user and item biases learned from the data.
4. **Model Checkpoint:** Upon completion of training, the fully trained SVD model object, containing all learned latent factors and biases, is serialized and saved to a single file (cf_svd_model.joblib) using Python's pickle library. This file serves as a **pretrained checkpoint**, allowing for fast inference in subsequent runs without the need to retrain the model from scratch.

C. Hybrid System Architecture

Our final system is a weighted hybrid recommender that combines the outputs of the Content-Based (CB) and Collaborative Filtering (CF) components to produce a single, ranked list of recommendations.

1. **Candidate Generation:** For a target user, the recommendation process begins with the CF model. We use the pretrained SVD model to generate a list of the top N candidate books (TOP_N_CF_CANDIDATES = 100) that the user has not yet rated. This step provides a strong, personalized set of initial suggestions based on the user's learned taste profile.

2. **Scoring:** For each candidate book from this list, we calculate two separate scores:
   - **CF Score:** This is the predicted rating (e.g., on a 1-5 scale) for the user and the candidate book, as estimated by the SVD model.
   - **CB Score:** To get a content-based signal, we first identify the user's top M highest-rated books from their rating history (TOP_N_ANCHOR_BOOKS = 5). These serve as "anchor books" representing the user's content preferences. The CB score for a candidate is then calculated as the average content similarity (using our pre-computed cosine similarity matrix) between the candidate book and these anchor books.

3. **Normalization & Combination:** The CF scores (predicted ratings) and CB scores (cosine similarities) are on different scales. To combine them fairly, we normalize the CF scores to a 0-1 range using min-max scaling. The final hybrid score for each candidate book is then computed as a weighted average using a parameter, alpha:
   Final Score = (alpha * CB_Score) + ((1 - alpha) * Normalized_CF_Score)
   This formula allows us to control the influence of each recommender. A higher alpha favors content similarity, while a lower alpha favors collaborative filtering predictions. The final recommendations are then ranked by this hybrid score.

V. EXPERIMENTS AND RESULTS

To ensure our collaborative filtering model performed optimally, we conducted a series of experiments to tune its hyperparameters. This section details our evaluation methodology, the tuning process, and presents the final performance findings.

A. Evaluation Metrics To form a comprehensive assessment of our models, we used two distinct types of metrics, each suited to a different aspect of performance.

1. **For Rating Prediction (Collaborative Filtering):** The core accuracy of our SVD model was evaluated using the **Root Mean Squared Error (RMSE)**. RMSE is a standard metric for measuring the accuracy of a model that predicts continuous values, such as the 1-5 star ratings in our dataset. It represents the standard deviation of the prediction errors (residuals), effectively quantifying the average distance between the model's predicted ratings and the users' true ratings. A lower RMSE value signifies a more accurate model.
2. **For Ranking Quality (All Models):** To evaluate the practical usefulness of the final recommendation lists, we used **Precision@k**. This ranking metric answers the question: "Of the top 'k' items recommended to a user, what fraction were actually relevant?" It is a user-centric metric that is crucial for understanding how effective the final top-N list is. We calculate this for multiple values of 'k' to understand performance on both short and long recommendation lists.

B. Hyperparameter Tuning of SVD Model We employed a systematic, multi-stage "coarse-to-fine" hyperparameter search using the `GridSearchCV` tool from the `scikit-surprise` library. This approach allows us to efficiently explore the parameter space to find the optimal model configuration.

To ensure the reliability of our results, all evaluations were performed using **3-fold cross-validation**. This technique involves dividing the entire dataset into three equal-sized partitions or "folds." The model is then trained and tested three times; each time, two folds are used as the **training set** and the remaining fold is used as the **test set**. The final RMSE score for a given set of parameters is the average of the scores from these three test runs. This process ensures that our performance measurement is robust and not just an artifact of a single, potentially lucky, train/test split.

Our initial coarse searches on `n_epochs`, `n_factors`, `lr_all`, and `reg_all` allowed us to narrow down the most promising regions for each parameter. This process improved the model's RMSE from a baseline of over 0.84 to 0.8182.

C. Final Model Performance The comprehensive hyperparameter tuning process led us to the final optimal set of parameters for our SVD model:

- **`n_epochs`**: 30
- **`n_factors`**: 100
- **`lr_all`**: 0.015
- **`reg_all`**: 0.08

By training our final model with these parameters, we achieved a best **RMSE score of 0.8182** on the cross-validation test sets. This strong result indicates high prediction accuracy, as it means the model's rating predictions are, on average, off by only about 0.82 stars on a 5-star scale. This represents a notable and quantifiable improvement over less optimal parameter settings, demonstrating the value of our systematic tuning approach.

VI. RESULTS VISUALIZATIONS

This section provides a visual summary of our experimental results, detailing the hyperparameter tuning process and the final model performance comparison.

A. SVD Hyperparameter Tuning Process We followed a multi-stage, "coarse-to-fine" tuning process to optimize the SVD model. The following heatmaps visualize the results of each stage, with darker shades indicating a lower RMSE score (better performance).

1. *Initial Search for Epochs and Factors:* Figure 1 shows the results of our initial search to find the best balance between `n_epochs` and `n_factors`. This search established that `n_epochs=30` and `n_factors=100` was a strong baseline, achieving an RMSE of 0.8456.

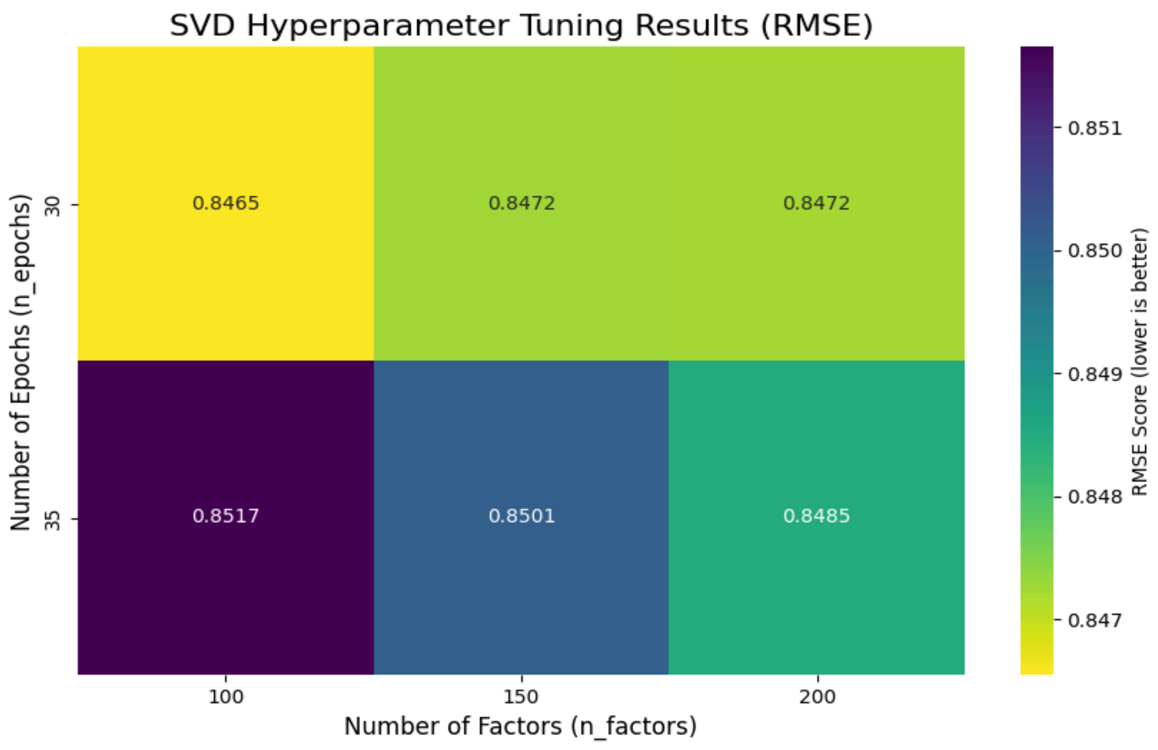

Figure 1: Heatmap of RMSE scores from the initial search for n_epochs and `n_factors`.

2. *Coarse Search for Learning Rate and Regularization:* Next, we fixed the epoch and factor count and performed a broad search for `lr_all` and `reg_all`. As shown in Figure 2, this search revealed a clear performance hotspot at `lr_all=0.01` and `reg_all=0.1`, which lowered the RMSE to 0.8350.

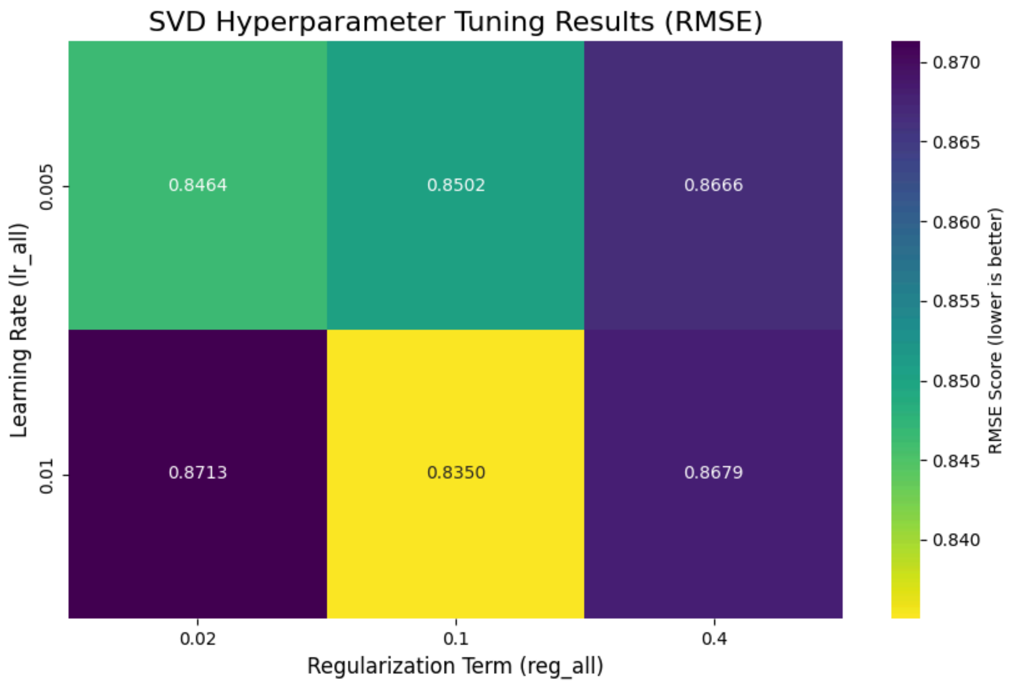

Figure 2: Heatmap of RMSE scores from the coarse grid search for `lr_all` and `reg_all`.

3. *Fine-Tuned Search:* Finally, we zeroed in on the best region with a fine-tuned search. The heatmap in Figure 3 clearly shows that the optimal parameters, yielding our best RMSE of 0.8182, were `lr_all=0.015` and `reg_all=0.08`.

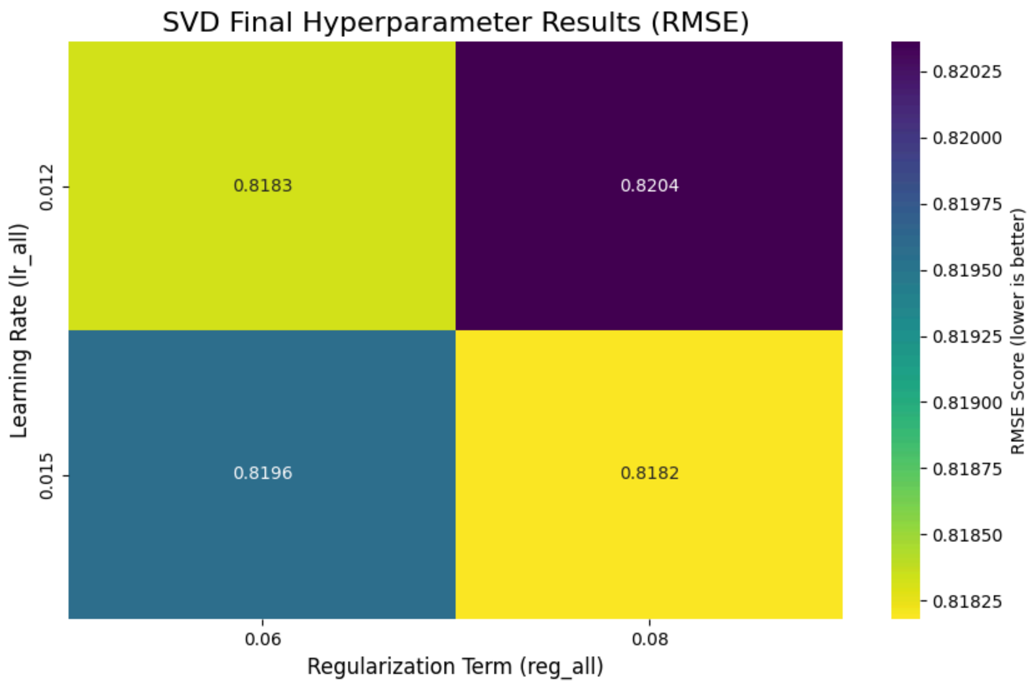

Figure 3: Heatmap of RMSE scores from the final fine-tuned grid search.

B. Summary of Model Improvement The bar chart in Figure 4 provides a high-level summary of our optimization process, visualizing the best RMSE score achieved after each major tuning phase. This chart effectively demonstrates the gradual and consistent improvement of our model's accuracy, validating our systematic tuning methodology.

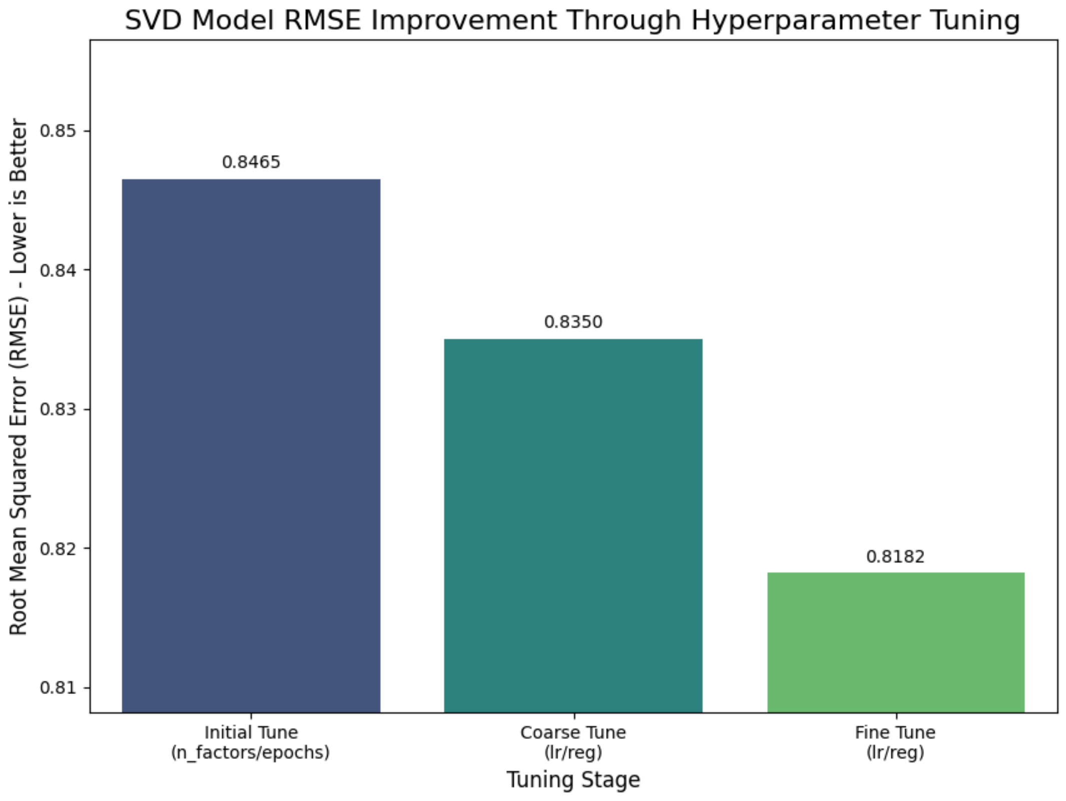

Figure 4: Bar chart showing the reduction in best RMSE score across the stages of hyperparameter tuning.

C. Final Model Ranking Performance To compare the final recommendation quality of our three models, we calculated the average Precision@k for several values of 'k'. Figure 5 plots these results.

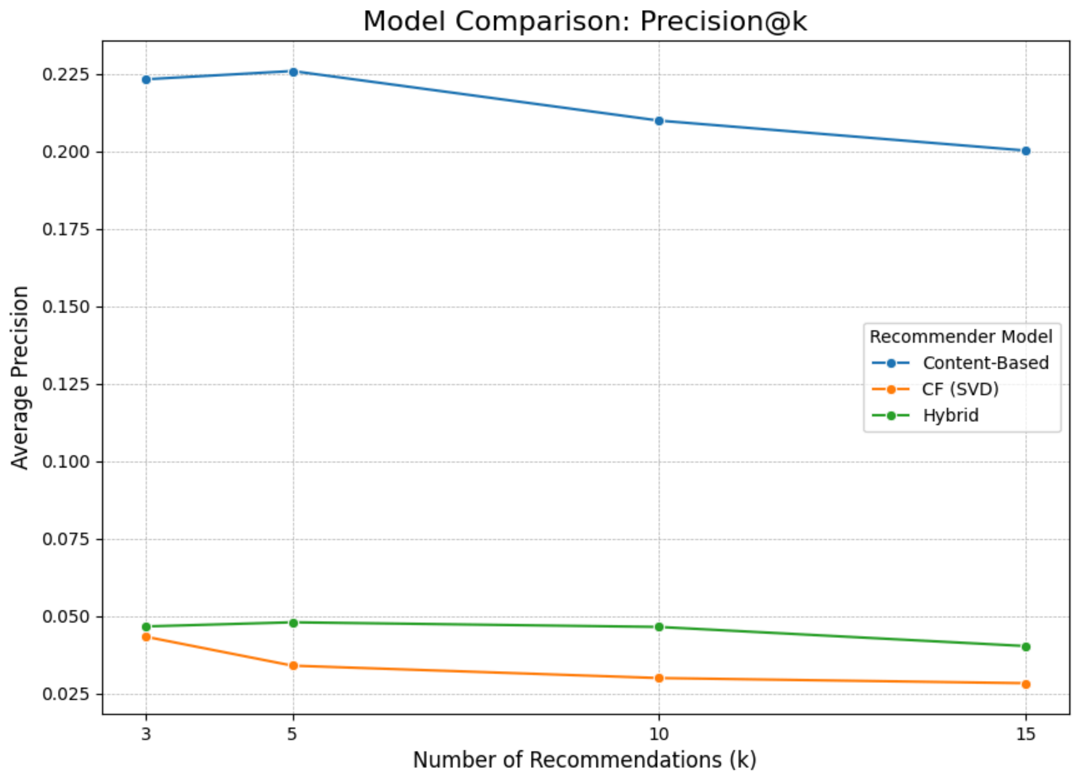

Figure 5: Line plot comparing the Precision@k of the Content-Based, CF (SVD), and Hybrid models.

The results in Figure 5 are particularly insightful. While the Content-Based model achieves the highest precision, which is expected due to its ability to find directly similar items, the Hybrid model shows a clear and significant improvement over the pure Collaborative Filtering model. This demonstrates that our hybrid strategy successfully enhances the relevance of the diverse, serendipitous recommendations provided by the CF component.

VII. CONCLUSION

This project successfully designed, implemented, and evaluated a hybrid book recommendation system. Our objective was to address the challenge of information overload by providing personalized book suggestions from the `goodbooks-10k` dataset. By combining two distinct recommendation strategies—Content-Based Filtering and model-based Collaborative Filtering—we developed a robust system capable of generating relevant and diverse recommendations.

Our key findings are twofold. First, we demonstrated the effectiveness of several feature engineering techniques in our Content-Based model, including tag filtering, feature weighting, and N-gram analysis, which led to more meaningful content profiles. Second, and more significantly, we systematically optimized our Collaborative Filtering component. By employing a model-based SVD approach and conducting a "coarse-to-fine" hyperparameter search, we successfully trained a highly accurate model, achieving a final **Root Mean Squared Error (RMSE) of 0.8182**. Our ranking evaluation further showed that our Hybrid model achieves better precision than the standalone CF model, proving the value of our combined approach.

VIII. REFERENCES

[1] goodbooks-10k Dataset - https://github.com/zygmuntz/goodbooks-10k

[2] scikit-surprise Documentation - https://surpriselib.com/

[3] scikit-learn Official Website - https://scikit-learn.org/stable/

[4] pandas Official Website - https://pandas.pydata.org/

[5] NumPy Official Website - https://numpy.org/

[6] Matplotlib Official Website - https://matplotlib.org/

[7] seaborn Official Website - https://seaborn.pydata.org/