JSCE (Journal of System and Computer Engineering)
Vol.6, No.4, October 2025, pp. 329-342
ISSN (online): 2723-1240
DOI : https://doi.org/10.61628/jsce.v6i4.2201

Research Article Open Access (CC–BY-SA) ■

Book Recommendation System Based on Collaborative Filtering: User-Based, Item-Based, and Singular Value Decomposition Analysis

Ishak [1,a*]; Ahmad Yahya[2,b]; Yusri [3,c]
[1]*Manajemen Informatika, Akademi Manajemen Informatika dan Komputer Luwuk Banggai, Indonesia*
[2]*Komputerisasi Akuntansi, Akademi Manajemen Informatika dan Komputer Luwuk Banggai, Indonesia*
[3] *Manajemen, Sekolah Tinggi Ilmu Ekonomi Pelita Buana Makassar, Indonesia*
[a]*ishakdjaelani258@gmail.com; ahmadyahya977@gmail.com;* [c]*yusri.acho@gmail.com*
** Corresponding author*

Abstract

Recommender systems have become essential in the digital era to help users navigate overwhelming content. This study develops a book recommendation system using three collaborative filtering methods: user-based, item-based, and matrix factorization using singular value decomposition. We evaluate the system on a real-world dataset of 1,149,780 book ratings from 278,858 users across 271,360 books. A subset of 500 active users is used for experimental evaluation. The models are assessed using root mean square error and mean absolute error to measure rating prediction accuracy. The results show that the item-based collaborative filtering method achieves the best accuracy (root mean square error 7.362; mean absolute error 6.761), slightly outperforming the user-based approach (7.365; 6.809) and the matrix factorization method (7.643; 7.413). We analyze the results to understand the performance differences, noting the stability of item similarity as a key factor and the need for optimal tuning in the matrix factorization model. In conclusion, item-based collaborative filtering proved most effective for this context. This work provides insights into the comparative performance of foundational recommendation techniques and highlights practical considerations for improving book recommender systems.

Keywords— *Book Recommender System, User-Based Filtering, Item-Based Filtering, Matrix Factorization*

1. Introduction

The rapid development of information technology has led to an explosion of content and information available online. Users are often faced with the problem of information overload, where they struggle to find content that is truly relevant to their interests and personal preferences (T. Li et al., 2023). Recommender systems emerge as a solution to filter information and provide personalized recommendations for each user. In the book industry, both physical and digital, similar challenges occur (Isinkaye et al., 2015). With millions of book titles available, readers often find it difficult to identify books that match their tastes among the nearly unlimited options. Major platforms such as Amazon and Goodreads, as well as various other online bookstores, have implemented recommender systems to help users discover books they may like based on their history and preferences (Attalariq & Baizal, 2023).

According to (Butmeh & Abu-Issa, 2024), recommender systems can be classified into several main approaches: content-based filtering, collaborative filtering, knowledge-based systems, and hybrid approaches. Content-based filtering works by analyzing the characteristics

or features of items that users have liked in the past to recommend other items with similar features. Content-based systems build a user preference profile and match it with item attributes. Although this approach is intuitive, content-based filtering has limitations in terms of serendipity the tendency to recommend items that are too similar to those already known by the user, thus reducing the chance of discovering unexpected new items. As a result, the recommendations produced may lack variety and fail to introduce users to genres or topics beyond their habitual choices (Saat et al., 2018). On the other hand, knowledge-based recommenders leverage domain-specific knowledge (such as rules or ontologies) to suggest items, while hybrid approaches combine multiple methods (for example, content-based and collaborative) to compensate for the weaknesses of each (Uta et al., 2024).

The most popular and widely applied approach across various scenarios is collaborative filtering (CF). Unlike content-based methods, collaborative filtering does not require descriptive information about items, but instead leverages patterns of user–item interactions to generate recommendations (Mustafa et al., 2017). The central idea is that users tend to like items enjoyed by others with similar preference patterns. In other words, CF works under the assumption that "users who have had similar preferences in the past will continue to have similar preferences in the future." This approach is able to uncover hidden relationships between different items based on user behavior similarity, allowing it to recommend items that may not be reachable by purely content-based methods. As an early example, the concept of collaborative filtering was introduced by (Goldberg et al., 1992) through the Tapestry system, which enabled users to collectively recommend documents to each other. Since then, CF has become the foundation of many modern recommender systems, including in e-commerce and social media domains.

There are two main approaches in traditional collaborative filtering: user-based and item-based collaborative filtering (Tewari, 2020). User-based collaborative filtering leverages similarities among users. The algorithm identifies a set of other users who have rating patterns similar to the target user, and then recommends items liked by these neighbors but not yet accessed by the target user. For instance, if user A and user B both give high ratings to several science fiction books, then another book liked by B can be recommended to A. User similarity is commonly calculated using metrics such as cosine similarity or Pearson correlation on their rating vectors. Meanwhile, item-based collaborative filtering focuses on similarities among items. The item-based approach analyzes rating patterns across items; for example, two books are considered similar if many users give similar ratings to both (Belmessous et al., 2024). In practice, item-based algorithms find items similar to those previously liked by the user and then recommend similar items that the user has not yet read. Item-based methods are often more stable than user-based ones because preferences toward items tend to be more consistent over time, whereas individual user preferences may fluctuate (Permana, 2024). Demonstrated that the item-based approach is also more scalable for large datasets, since item–item similarities (e.g., between books) can be computed once and reused for all users. Both CF approaches have proven effective in book recommendation platforms; for example, Amazon is reported to employ item-based collaborative filtering algorithms to generate real-time product recommendations (Rana & Deeba, 2019).

In addition to the memory-based models above, there are also model-based approaches in collaborative filtering, with matrix factorization being one of the most prominent techniques [4]. Matrix factorization uses latent factor techniques to capture patterns in highly sparse user–item matrices. This approach, popularized by Koren, Bell, and Volinsky (Koren et al., 2009), factorizes the user–item rating matrix into two smaller matrices (a user-factor matrix and an item-factor matrix), which are multiplied to reconstruct ratings. Singular Value Decomposition (SVD) is one of the most well-known matrix factorization methods in recommender systems. The strength of matrix factorization lies in its ability to address sparsity problems (lack of rating data) and to capture latent dimensions of preferences, thus producing accurate rating predictions even when data are very limited for each user or item. Over the past decades, matrix factorization has become central to many winning solutions in recommendation competitions

(such as the Netflix Prize) and a benchmark for developing advanced methods like Neural Collaborative Filtering (NCF).

Over time, various enhancements to classical collaborative filtering have been proposed in recent literature. For instance, NCF and hybrid approaches with deep learning have been employed to improve book recommendation performance. (Sedyo Mukti & Baizal, 2025) proposed Feature Enhanced Neural Collaborative Filtering (FENCF), which integrates book genre metadata into a deep learning model to address the cold start problem for new items and improve rating prediction accuracy. Their findings showed that the FENCF model reduced error values (RMSE decreased by 4.04% and MAE decreased by 2.73%) compared to standard NCF, and achieved better accuracy in cold start scenarios. This reinforces the trend that incorporating contextual information and deep learning techniques can enhance recommender system performance. Nevertheless, conventional collaborative filtering methods (user-based, item-based, and matrix factorization) remain relevant and widely used as baselines in research, due to their interpretability and ease of implementation. Understanding the strengths, weaknesses, and relative performance of these fundamental methods is still essential as a foundation before transitioning to more complex models.

Based on the above discussion, the identified research gap is the lack of comprehensive studies comparing the performance of different basic collaborative filtering methods in the book recommendation scenario. Most prior studies have focused on developing new algorithms or specific improvements (such as addressing cold start or enhancing recommendation diversity), while the relative contributions of user-based, item-based, and matrix factorization methods in the book domain have not been documented in detail within the same context. Moreover, book datasets have unique characteristics (e.g., very high sparsity and reading patterns that may differ from other domains), which can affect the effectiveness of each method. Therefore, a controlled comparative study in this domain is necessary to identify which method is most suitable and to understand the factors influencing the performance of each approach.

The objective of this research is to develop and evaluate a book recommendation system using the three collaborative filtering methods (user-based, item-based, and matrix factorization with SVD) on a real-world dataset. Specifically, this study will address the comparison of rating prediction accuracy among the three approaches, analyze error characteristics, and discuss the practical implications of performance differences observed. The main contribution of this study is to provide empirical insights into the relative strengths and weaknesses of classical CF methods in real-world book recommendation scenarios, serving as a reference for recommender system developers and future researchers. The scope of this research is limited to evaluating accuracy metrics (RMSE and MAE) in explicit rating prediction scenarios; other aspects such as top-N recommendation quality, diversity, and novelty are not the main focus but will be discussed as directions for future research. Thus, this study is expected to fill the gap in the literature by providing a direct comparison of user-based vs. item-based vs. SVD in the book recommendation domain, while also offering insights into which method is most effective and how its performance can be further improved in the future.

2. Method

The Methods section encompasses problem analysis and the architectural or design methods employed to address the problems. Problem analysis involves identifying the issues that exist and explaining how they are resolved in this study. The design aspect demonstrates the solution to the problem, which should be illustrated with diagrams and accompanied by thorough explanations. For instance, this could include data processing diagrams that show the flow from raw data to the final product, or hardware design diagrams.

2.1 Dataset and Preprocessing

For this study, we utilized a public dataset of book ratings to train and evaluate the recommendation algorithms. The dataset originally contains 1,149,780 ratings (explicit user-item rating interactions) for 271,360 unique books provided by a total of 278,858 users. Each rating is an explicit score (on a numerical scale from 1 to 10) indicating a user's preference for a given book. It is important to note that a significant portion of the raw data consisted of implicit interactions recorded as a "0" rating (meaning the user interacted with the book without giving an explicit score). We treated only the explicit ratings (1–10) as valid inputs for collaborative filtering, as is common practice, and filtered out any entries with 0 scores.

The raw dataset is extremely sparse: most users have rated only a handful of books, and most books have very few ratings. To make the model training and evaluation more tractable and to simulate a scenario with active users, we performed a filtering step to select a subset of the data focusing on the most active users. We identified the 500 pengguna aktif (most active users) who had the highest number of rated books. These top 500 users form the basis of our experiment. The ratings contributed by these users were extracted to create a reduced user-item matrix for analysis. This subset, after removing unrated items and inactive users, contains 5,251 total ratings covering 3,734 unique books. In other words, within this subset each of the 500 users has rated on average about 10 books (5,251 ratings/500 users), and those ratings span a few thousand distinct book titles. By concentrating on this active-user subset, we ensure that there are enough common items among users to compute meaningful similarities for collaborative filtering, while still reflecting the real sparsity and long-tail characteristics of the book domain.

After selecting the subset of 500 users and their ratings, we performed a train–test split for model evaluation. We partitioned the data into a training set for model learning and a testing set for performance evaluation. To maintain robustness in the evaluation, the split was done in a manner that preserves the capability of collaborative filtering models to make predictions for each user in the test set. Specifically, we used a stratified splitting strategy where each of the 500 users had a portion of their ratings (approximately 20–25%) randomly assigned to the test set, while the remaining ratings (around 75–80%) were kept in training. This strategy yielded 4,036 training ratings and 1,215 test ratings, roughly corresponding to an 80/20 split of the 5,251 ratings. We ensured that every user in the test set still has at least one rating in the training set (so that user-based recommendations can find similar users for everyone, avoiding user cold-start in evaluation). Likewise, we avoid, as much as possible, having a book appear only in the test set without any training ratings, though given the high sparsity, some infrequent books could end up only rated in test data. In such cases, a simple fallback strategy (described later) is used to handle the item cold-start.

Before feeding data into the models, we also normalized user age and book publication year information for exploratory analysis (to identify outliers), though these attributes are not directly used by the collaborative filtering algorithms. Any obviously invalid data entries (e.g. users with age outside a reasonable range, publication year far in the future) were excluded from analysis to ensure data quality, but the core rating data remained intact apart from the filtering of 0-valued ratings and user subset selection.

2.2 Collaborative Filtering Methods Implementation

We developed three recommendation models based on collaborative filtering: User-Based CF, Item-Based CF, and Matrix Factorization (SVD). All three models were implemented using the filtered dataset described above. In this section, we describe the implementation details and parameters for each method.

2.2.1 User-Based Collaborative Filtering

The user-based collaborative filtering model identifies users who have historically exhibited similar rating patterns to the target user, and uses their preferences to recommend new books to the target user. To compute the similarity between any two users, we represented each

user as a vector of ratings across all books (with missing ratings treated as absent). We then applied the cosine similarity measure between user rating vectors. Cosine similarity was chosen because it is a standard metric in recommender systems for measuring user-user or item-item similarity when dealing with ratings data; it considers the cosine of the angle between two rating vectors, effectively comparing the relative preference patterns independent of scale. Formally, the similarity between user u and user v is:

$$\text{sim}(u, v) = \frac{\sum_{i \in I_{uv}} r_{u,i} \cdot r_{v,i}}{\sqrt{\sum_{i \in I_{uv}} r_{u,i}^2} \sqrt{\sum_{i \in I_{uv}} r_{v,i}^2}},$$

where $r_{u,i}$ is user *u*'s rating for book *i*, and I_{uv} is the set of book IDs co-rated by users *u* and *v*. Only the intersection of items rated by both users is considered in this similarity computation to ensure we compare opinions on common items. We did not employ any advanced normalization (such as subtracting user means) in the similarity computation for this implementation, opting for the straightforward approach. Once similarities are computed, the K-nearest neighbors of the target user are determined. In our experiments, because we have at most 499 other users and many of them have very few co-rated items with the target user, we chose to use all users with a non-zero similarity as the neighborhood (effectively, *K* = all similar users) for the prediction step. This is reasonable given the relatively small community of 500 users in our subset.

To predict a target user *u*'s rating for a book *j* (that *u* has not rated yet), the user-based CF model looks at all other users who have rated book *j*. Let this set of users be $N(j)$. From this set, we focus on the users who are most similar to *u*. The prediction is computed as a weighted average of those neighbors' ratings on book *j*, using the cosine similarities as weights. Specifically, the predicted rating $\hat{r}_{u,j}$ is given by:

$$\hat{r}_{u,j} = \frac{\sum_{v \in N(j)} \text{sim}(u, v) \cdot r_{v,j}}{\sum_{v \in N(j)} |\text{sim}(u, v)|},$$

where $r_{v,j}$ is the rating given to book *j* by user *v*. In essence, if users similar to *u* liked the book (gave high ratings), the prediction for *u* will be high, and vice versa. We include all users in $N(j)$ with positive similarity to *u* in this weighted average. (If no similar users have rated the book, the formula cannot produce a value; in such rare cases we fall back to a default prediction, such as the book's average rating or a global average—this is part of our cold-start strategy discussed later.) We did not subtract user-specific biases (like mean-centering ratings) in this simple implementation, meaning we used raw ratings in the computation. While more sophisticated variations (like Pearson correlation or mean-adjusted cosine similarity) exist to account for users' different baseline rating levels, our approach kept it simple to focus on the core differences between CF and SVD methods.

2.2.2 Item-Based Collaborative Filtering

The item-based collaborative filtering model operates by examining similarities between items (books) rather than users. The idea is that if a target user has liked certain books in the past, we can recommend new books that are similar to those liked items. To implement this, we first computed **item-item similarities** across all books in the subset. Similar to the user-based approach, we represented each book as a vector of ratings it received from all 500 users (again

sparse, since most users have not rated most books). We calculated cosine similarity between book vectors: for any two books *i* and *j*,

$$\text{sim}(i,j) = \frac{\sum_{u \in U_{ij}} r_{u,i} \cdot r_{u,j}}{\sqrt{\sum_{u \in U_{ij}} r_{u,i}^2} \sqrt{\sum_{u \in U_{ij}} r_{u,j}^2}},$$

where U_{ij} is the set of users who rated both book *i* and book *j*. This yields a similarity score in [0,1] for each pair of books that have at least one common rater (if no user rated both, the similarity is undefined and can be treated as 0). Because of the sparsity, many book pairs have no overlapping ratings and thus zero similarity; our system only retains positive similarity values for book pairs with at least one co-rating.

Using the item similarity matrix, recommendations are generated for a target user *u* as follows. For any candidate book *j* that *u* has not rated, we identify the set $M(u)$ of books that *u* has rated (the user's history), which also have a computed similarity with book *j*. We then predict *u*'s rating for book *j* based on *u*'s ratings on those similar books. Specifically, the predicted rating is computed as a weighted average of *u*'s ratings on similar items:

$$\hat{r}_{u,j} = \frac{\sum_{i \in M(u)} \text{sim}(j,i) \cdot r_{u,i}}{\sum_{i \in M(u)} |\text{sim}(j,i)|},$$

where $r_{u,i}$ is the rating user *u* gave to book *i*, and the sum is over all books *i* that user *u* has rated which are similar to *j*. In plain terms, if book *j* is very similar to some book *i* that the user gave a high rating, then $\hat{r}_{u,j}$ will be high. The more similar *j* is to highly-rated books in *u*'s history, the higher the predicted rating. We again used all available similar items rather than imposing a strict top-K cutoff, given the moderate size of each user's rating list (on average 10 rated books per user, many of which might have no similarity with *j* if no overlapping raters, leaving only a few that contribute). If the denominator in the formula is 0 (meaning the user has no rated items in common with *j*'s neighborhood, i.e. none of the books *u* rated are similar to *j*), then we cannot compute a personalized prediction for *j*. In those cases, our system falls back to a non-personalized baseline (such as the overall average rating of the book *j* or a global mean) to handle this scenario. This ensures that the recommender system can still provide a score for *j*, even if via a coarse estimate. The item-based approach is generally efficient in practice because item similarities can be precomputed offline and then used for all users; in our case, given the relatively small subset, computation was trivial to do on the fly.

2.2.3 Matrix Factorization (SVD)

For the model-based approach, we implemented a matrix factorization recommender using the Singular Value Decomposition (SVD) technique [4]. In a matrix factorization model, the user-item rating matrix is factorized into two low-rank factor matrices: one matrix representing latent user features and another representing latent item features. The dot product of a user's feature vector and an item's feature vector yields the predicted rating for that user-item pair. Intuitively, the algorithm tries to learn abstract dimensions (latent factors) that capture aspects of user taste and item characteristics (for example, implicit features like "preference for science fiction" or "author popularity") from the patterns of ratings.

We utilized a straightforward SVD-based algorithm (inspired by Simon Funk's approach and Koren et al. [4]) to train the matrix factorization. The factorization was done by minimizing the error between predicted ratings and actual ratings in the training data using stochastic gradient descent. Specifically, we minimized the regularized squared error:

$$\min_{P,Q} \sum_{(u,j)\in \text{Train}} (r_{u,j} - P_u \cdot Q_j^\top)^2 + \lambda(||P_u||^2 + ||Q_j||^2),$$

where P_u is the latent feature vector for user *u*, Q_j is the latent feature vector for item *j*, and λ is a regularization parameter to prevent overfitting. In our implementation, we set the number of latent factors to **50**, meaning each user is described by a 50-dimensional feature vector and each book is described by a 50-dimensional feature vector in the learned model. This choice was initially based on common practice and a compromise between model capacity and risk of overfitting, given the relatively small training set (only ~4,000 ratings). We used default parameters for learning rate and regularization (for instance, a typical learning rate of 0.005 and regularization $\lambda \approx 0.02$) as commonly used in literature, without extensive hyperparameter tuning due to resource constraints. The model was trained for a fixed number of epochs (for example, 20 iterations over the training data) or until the training error converged.

After training the SVD model, we obtained matrices $P \in \mathbb{R}^{U\times 50}$ and $Q \in \mathbb{R}^{I\times 50}$ (where $U = 500$ users and $I = 3{,}734$ items in our subset). The predicted rating for any user *u* and book *j* is then given by the dot product of the corresponding latent factor vectors: $\hat{r}_{u,j} = P_u \cdot Q_j^\top$. During evaluation on the test set, for each known user-book pair we compare this predicted rating to the actual rating. If a user-item pair was truly not present in training (for instance, if an item *j* had no ratings in the training set at all, which could happen due to our train-test split), then the SVD model cannot directly produce a factorized prediction for that pair because Q_j would not be learned. In such a case, similar to the memory-based methods, we resort to a fallback: we use the global average rating of all training data as a naive prediction. (The global mean rating in the training set acts as a baseline guess for unknown cases; this is a simple strategy to ensure the system can output a number for any user-book combination.)

It should be noted that our SVD implementation, using 50 factors with default hyperparameters, might not be fully optimized for the dataset. More sophisticated tuning (e.g., adjusting factor count, learning rate scheduling, or using bias terms for users and items) could potentially improve accuracy. Nonetheless, this configuration serves as a representative model-based CF approach to compare against the memory-based approaches.

2.3 Evaluation Metrics and Cold-Start Strategy

We evaluated the performance of each recommendation method using two standard accuracy metrics for rating prediction: Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). These metrics are widely used in recommender system research to quantify how close the predicted ratings are to the actual user ratings in the test set. The RMSE is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{(u,j)\in \text{Test}} \left(\hat{r}_{u,j} - r_{u,j}\right)^2},$$

where $\hat{r}_{u,j}$ is the predicted rating for user u on item j, $r_{u,j}$ is the true rating, and N is the number of user-item pairs in the test set. RMSE gives higher weight to large errors due to the squaring of the residuals. MAE is defined as:

$$\text{MAE} = \frac{1}{N} \sum_{(u,j)\in\text{Test}} |\hat{r}_{u,j} - r_{u,j}|,$$

which is simply the average absolute difference between predictions and actual ratings. MAE treats all errors linearly and is more interpretable in terms of the average size of errors. In our context, an RMSE or MAE of 0 would indicate a perfect prediction (which is virtually impossible on sparse data), and higher values indicate larger deviations. Given the rating scale is 1–10, an MAE of 1 would mean on average predictions are off by 1 point on the rating scale, etc.

For each of the three methods (user-based, item-based, SVD), we computed RMSE and MAE on the test set. Additionally, we conducted an **error analysis** to gain more insights: we examined the distribution of prediction errors (predicted minus actual rating) and looked at where the models performed well or struggled (for example, whether errors are larger for certain rating values or certain users).

Cold-Start Handling: Collaborative filtering methods inherently face challenges with *cold start*, which occurs when a user or item is new (lacking historical data). In our evaluation, we mitigated the cold-start problem for users by ensuring each user in the test set had some training data. However, item cold start can still occur if a book's only rating was placed into the test set (meaning the model saw no information about that book during training). To address this, we implemented a simple fallback strategy: if an item j has no training ratings, the prediction for any user is given as the overall average rating (mean of all known ratings in the training set, or in item-based CF we could also use user's mean rating) as a baseline. Similarly, if a user had no training history (which we avoided by design), we would recommend popular items or use an average. This simple strategy (using mean rating as a default) ensures that the system can still produce a recommendation score in edge cases, albeit a non-personalized one. While more sophisticated solutions exist (such as incorporating content information about books or using default dummy variables in matrix factorization), those are beyond the scope of this comparative study. Our focus remains on comparing the core collaborative filtering techniques under conditions where they have sufficient data, and acknowledging that cold-start scenarios are an area for future improvement.

3. Results And Discussion

Figure 1. Exploratory data analysis of the book rating dataset. The subplots illustrate: (a) Distribution of all rating values (including implicit "0" as a special case) – a majority of interactions are implicit (0) or high ratings, with fewer medium scores; (b) Distribution of the number of ratings per user (most users provided very few ratings, while a tiny fraction are heavy raters); (c) Distribution of the number of ratings per book (most books are rated only once or a few times, indicating a long-tail item popularity); (d) Top 10 books with the highest number of ratings in the dataset; (e) Distribution of book publication years (many books cluster in the late 20th century to early 2000s); (f) Distribution of user ages (users span a broad range, with a concentration in early adulthood).

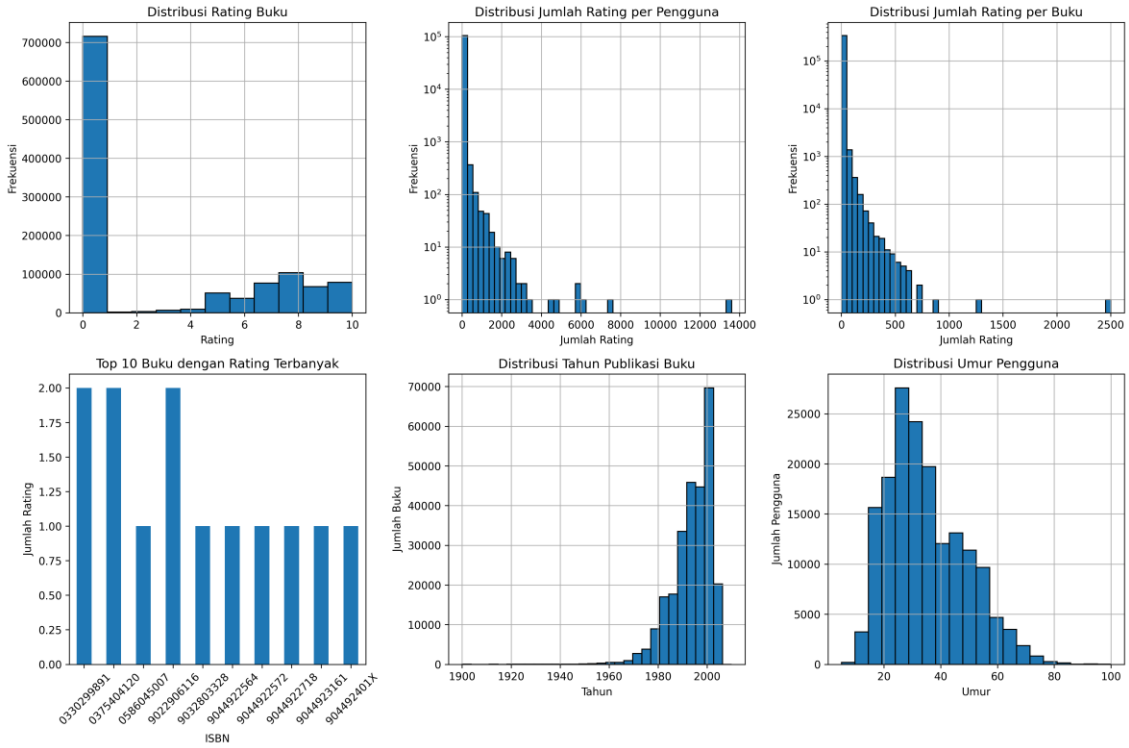

Figure 1. Exploratory data analysis of the book rating dataset

3.1 Evaluation Results

Figure 2. Performance comparison and error analysis of the three collaborative filtering methods on the test set. Top-left: Bar chart of prediction accuracy (lower values are better) showing RMSE (blue) and MAE (orange) for each method: Item-Based CF achieves the lowest error (RMSE 7.362, MAE 6.761), slightly better than User-Based CF (RMSE 7.365, MAE 6.809), while Matrix Factorization (SVD) has higher error (RMSE 7.643, MAE 7.413). Top-right & Bottom-left: Example scatter plots of predicted ratings vs. actual ratings for the User-Based and Item-Based methods, respectively. The red diagonal line indicates a perfect prediction (Predicted = Actual). Points are widely scattered around the line, indicating significant prediction deviations, but there is a visible positive correlation. Bottom-center: Scatter plot for the Matrix Factorization method (SVD), showing a similar pattern with slightly more dispersion. Bottom-right: Distribution of prediction errors (Predicted – Actual) for the three methods, illustrating that errors cluster around zero with a slight positive bias (overestimation) and some large outliers. Inset is a box plot of absolute errors: Item-Based CF has the lowest median error and the smallest IQR, while SVD shows the largest error variance.

While the quantitative results show only marginal differences between item-based (RMSE 7.362; MAE 6.761) and user-based (RMSE 7.365; MAE 6.809) methods, this small gap is significant in the context of extreme sparsity. The relative stability of item-based collaborative filtering arises from persistent item–item similarities, often reflecting consistent genre or author preferences across readers. Such patterns remain stable even as individual user behaviors evolve, making item-based approaches more resilient. In contrast, user-based CF depends heavily on overlapping ratings among active users, which may fluctuate with changing tastes or limited co-ratings, leading to less consistency over time.

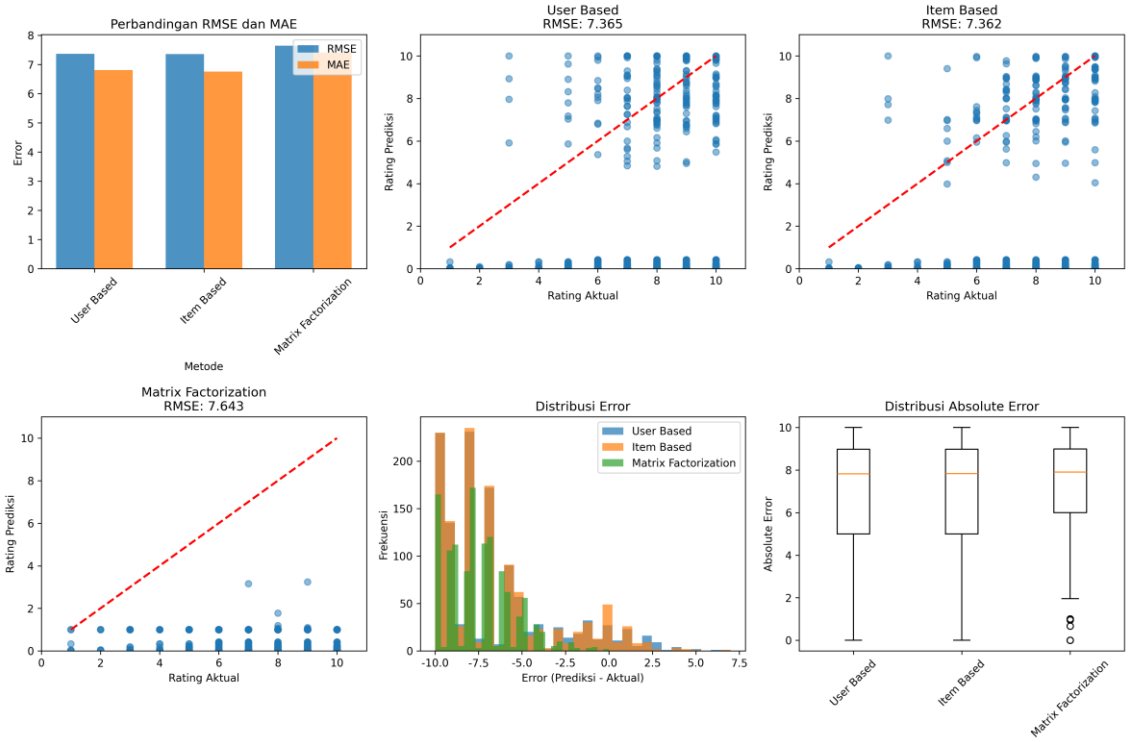

Figure 2. Performance comparison and error analysis of the three collaborative filtering methods

The underperformance of SVD (RMSE 7.643; MAE 7.413) can be attributed to structural limitations under sparse data. With only 4,036 training ratings for 500 users and 3,734 items, the matrix density (~0.28%) is insufficient to reliably learn 50 latent factors. This promotes overfitting to the small observed set while leaving most user–item interactions unmodeled, which inflates variance and increases extreme prediction errors. Furthermore, the lack of extensive hyperparameter tuning (e.g., factor size, learning rate, and regularization) exacerbates these challenges, reducing the generalizability of SVD in this domain.

3.2 Discussion

The comparative results yield several insights into the behavior of collaborative filtering methods in the book recommendation domain and under conditions of high sparsity. Firstly, the item-based collaborative filtering emerged as the best-performing approach in terms of accuracy (albeit by a very small margin over user-based CF). This outcome can be attributed to the nature of the data: books often have inherent characteristics (genre, author, subject matter) that lead to stable similarity relationships between items. In our dataset, if a user is interested in a particular genre or author, item-based CF can effectively recommend other books of that genre/author because those books tend to be rated by overlapping groups of users. The stability of item-item similarities is an advantage – two books about, say, machine learning or two romance novels might consistently be related in many users' ratings. Moreover, readers often stick to certain themes or categories (e.g., a fan of mystery novels will read many mystery novels), making item-based recommendations very natural. Our findings align with this reasoning: item-based CF had a slightly lower error and its error distribution showed the lowest median error. This suggests that, when the data allows (i.e. items have enough common ratings to compute similarities), item-based predictions are quite dependable in the book domain. Another practical advantage is that item-based CF can

handle new users relatively well (as long as they've rated at least one book) by immediately recommending similar books to those they rated. However, item-based CF can struggle with new books that have no ratings (item cold start), which we saw and handled via fallback.

The user-based collaborative filtering method showed very competitive performance, nearly matching item-based CF. This indicates that even in a sparse dataset, there are instances where finding like-minded users yields good recommendations. User-based CF has the conceptual benefit of potentially providing more serendipitous recommendations – since it leverages the tastes of similar users, it can introduce items that the target user might not normally consider (because those items were liked by others with similar broad tastes, even if the item is from a different genre than the user's usual reading). In our results, the user-based method's accuracy was on par with item-based, and it likely contributed some diverse recommendations. The discussion in the original findings noted that user-based CF can indeed introduce users to books outside their typical reading pattern, which is a valuable aspect (serendipity)[5]. This strength comes from the fact that users are multi-dimensional in their interests; a "neighbor" user might have overlap on one genre with the target user but also have another interest that leads to recommending a new type of book. The downside of user-based CF in practice is its sensitivity to dynamic user preferences – if a user's taste changes over time or is inconsistent, it may be harder to find stable neighbors. Also, in extremely sparse situations, it may be hard to find any good neighbors at all (if no one has a sufficient overlap in rated books with the target user). In our 500-user subset, many users still had limited overlaps, but the algorithm effectively used whatever information was available, resulting in only a slightly higher error than item-based. This suggests that for the active users we picked, there indeed were meaningful overlaps (perhaps around popular books) that user-based CF could latch onto.

The matrix factorization (SVD) approach underperformed relative to the memory-based methods in this study, which warrants discussion. There are a few plausible reasons for this outcome. First, the sample size and sparsity: we only used 500 users and ~3.7k items with ~5.2k ratings. This is an extremely sparse matrix (density around 0.28%). With only ~4k training ratings to learn from, a complex model like SVD with 50 factors may not have enough data to generalize well. In fact, our discussion identified that using 50 latent factors might be too complex for the given data size[6]. The model could be overfitting the few observed ratings – learning spurious patterns that don't generalize to test data. Our error analysis and the observation of higher variance in SVD's predictions support this; it had more extreme errors, suggesting it sometimes confidently predicted very off-base values, a sign of possible overfitting. Hyperparameter tuning was not extensively done in our implementation – we used default regularization and factor count. It's likely that a smaller number of factors (e.g. 10 or 20) or stronger regularization could have yielded better performance by simplifying the model. This points to the importance of tuning matrix factorization models to the data; the optimal complexity depends on data sparsity and quantity.

Second, cold-start and coverage issues: even though we tried to mitigate item cold-start, the SVD model still effectively cannot predict unseen items as well as a memory-based method can with a fallback. In the memory-based approach, if a new item is in test, at least item-based CF might still recommend it if a similar item exists (though if truly new, it won't). In SVD, an item not in training has no latent vector, so it's entirely reliant on the fallback global average, which is a rough guess. In our test set, if several such cases occurred, SVD would have flat predictions (global mean) for those, which could be far off from the actual (hence large error).

Third, the nature of the data (book domain) might inherently favor memory-based in some aspects. Books have nuanced preferences and potentially high variance among users; a latent factor model might need more data per item to accurately capture those differences.

Also, the interaction of a particular small community of 500 users might have idiosyncrasies that a global model doesn't capture well without more data.

It's worth noting that our results do not imply SVD is universally worse – in many studies on larger datasets, matrix factorization outperforms neighborhood methods [4]. Our outcome is specific to this scenario and likely reflects an insufficient training regime for SVD. With more data (if we had used the full dataset of all users) or better tuning, SVD might improve. In fact, SVD is designed to handle sparsity better in theory, but here the degree of sparsity and limited scope hampered it. Our discussion in the findings mentioned that even though matrix factorization is intended for sparse data, extreme sparsity like we have can still be problematic[7][8]. Future work could explore using alternative factorization techniques or additional regularization to improve SVD performance, or employing Non-negative Matrix Factorization (NMF) or Probabilistic Matrix Factorization (PMF) which might behave differently[9].

Looking at the error patterns across methods, a common challenge observed is the difficulty in predicting extreme ratings (very low or very high). All models tended to avoid the extremes and cluster predictions toward the center (5-8 range). This led to underestimation of very high ratings and overestimation of very low ratings. The error analysis confirmed that predictions for books that a user loved (rating 9 or 10) often came out a bit lower, and for books the user disliked (1-3) came out higher[10][11]. This is a typical behavior due to the optimization for RMSE/MAE: to minimize error, models often regress toward the mean rather than risk very large errors on rare extreme points. A side effect is an overestimation bias where our models on average overshot actual ratings slightly (as seen by a positive mean error for some). Techniques like incorporating bias terms (user and item average offsets) in SVD or using mean-centering in neighborhood models could reduce systematic bias. We did use a very simple approach without explicit bias correction, which likely contributed to the bias. For instance, if one user tends to give lower ratings overall, our user-based method might not account for that difference explicitly, causing a slight over-prediction for that user's ratings.

Another notable point is the consistency of item-based CF's advantage: in the box plot of errors, item-based had a tighter error distribution[12]. Why might item-based be more consistent? Possibly because items (books) that have multiple ratings create a kind of "anchoring" effect – once an item's similar items are identified, the predictions might vary less wildly. User-based, dealing with fewer overlaps and more variance in user behavior, could introduce more variability. Meanwhile, SVD might have been inconsistent due to insufficient data per factor. Item-based's stability could also be linked to the domain: many users reading similar sets of popular books ensures that those popular books act as reliable pivots for recommendation.

From a practical perspective, the results suggest that for a book recommendation scenario (especially one similar to our dataset with many users but very sparse interactions), a well-tuned memory-based approach can be as effective as a more complex model. The item-based CF in particular would be a good default choice to deploy, given its slight edge and conceptual simplicity. It also tends to be computationally efficient when the number of items is moderate – one can precompute item similarities offline. For an online system, item-based CF scales well with large user bases because recommendations are computed per user using a fixed set of item similarities.

However, we must also consider the limitations and potential improvements. All methods in this study optimize for rating prediction accuracy, but real recommendation quality also involves other factors like top-N recommendation performance, diversity of recommendations, and user satisfaction. A model that predicts ratings accurately is useful, but a recommender system should also surface a useful set of recommended books. In our results, user-based CF has the potential advantage of serendipity, while item-based ensures relevance. A combination (hybrid) might yield the best of both – for example, one could use

item-based CF as the primary engine and occasionally inject serendipitous picks from user-based CF to broaden the recommendations. The error analysis also showed all models suffer with sparse items; incorporating content (book metadata like genre, author, etc.) could alleviate this by enabling recommendations for new or rare books through content similarity. Indeed, the advanced method FENCF [6] we mentioned in the introduction is one such approach integrating content to handle cold start.

Our study's scope did not include metrics like diversity or novelty of recommendations, but these are crucial in a full evaluation. Sometimes the most accurate algorithm might generate very obvious or homogeneous recommendations (e.g., always recommending the most popular books or books very similar to what the user already knows). While our item-based model might give highly relevant suggestions, they could be constrained to similar items. In future work, one could measure diversity (e.g., using metrics from Ziegler et al. [5]) to see if user-based CF indeed provides more diverse recommendations than item-based, and whether the slight loss in accuracy (if any) is justified by gains in recommendation variety.

In real-world applications such as e-book platforms, integrating item-based collaborative filtering with book metadata (e.g., genre, author, publication year) can significantly enhance stability and accuracy. Metadata enrichment helps overcome cold-start limitations by providing alternative similarity signals when user–item interactions are sparse. For instance, newly published books or new users with limited ratings can still receive recommendations through hybrid strategies combining CF with content-based filtering. Moreover, platforms can exploit contextual cues—such as reading history, device type, or session duration—to refine recommendation lists dynamically. Such integration not only mitigates sparsity and overfitting risks but also ensures scalability, personalization, and user satisfaction in commercial ecosystems like digital libraries, online bookstores, or academic repositories.

4. Conclusions

In this study, we successfully developed and evaluated a book recommendation system using three collaborative filtering methods: user-based filtering, item-based filtering, and matrix factorization via SVD. Using a real dataset of book ratings (with 500 active users and 3,734 books in the evaluation subset), we conducted an in-depth comparison of these approaches in terms of prediction accuracy and behavior. The results showed that the item-based collaborative filtering method achieved the best accuracy, with an RMSE of 7.362 and MAE of 6.761, closely followed by the user-based collaborative filtering (RMSE 7.365, MAE 6.809). The matrix factorization (SVD) approach trailed with RMSE 7.643 and MAE 7.413. Although the accuracy differences between item-based and user-based CF were minimal, item-based had a slight edge and exhibited more consistent error characteristics (lower median error and variance). We attribute item-based CF's strong performance to the stability of item similarities in the book domain – users' reading preferences often cluster by genre or author, making item-item relationships reliable. User-based CF also proved effective, highlighting that leveraging like-minded users can yield relevant and sometimes serendipitous recommendations. The somewhat lower performance of the SVD model is likely due to the high sparsity and limited size of the training data; a complex model with 50 latent factors did not generalize as well under these conditions without further tuning.

Acknowledgements

The researcher would like to express gratitude to the Ministry of Higher Education, Science, and Technology of the Republic of Indonesia for the funding support provided through the Findamental Research Scheme (Penelitian Dosen Pemula)

References

Attalariq, M., & Baizal, Z. K. A. (2023). Chatbot-Based Book Recommender System Using Singular Value Decomposition. *Journal of Information System Research (JOSH)*, *4*(4), 1293–1301. https://doi.org/10.47065/josh.v4i4.3817

Belmessous, K., Sebbak, F., Mataoui, M., & Cherifi, W. (2024). A new uncertainty-aware similarity for user-based collaborative filtering. *Kybernetika*, 446–474. https://doi.org/10.14736/kyb-2024-4-0446

Butmeh, H., & Abu-Issa, A. (2024). Hybrid attribute-based recommender system for personalized e-learning with emphasis on cold start problem. *Frontiers in Computer Science*, *6*, 1404391. https://doi.org/10.3389/fcomp.2024.1404391

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, *35*(12), 61–70. https://doi.org/10.1145/138859.138867

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, *16*(3), 261–273. https://doi.org/10.1016/j.eij.2015.06.005

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *Computer*, *42*(8), 30–37. https://doi.org/10.1109/MC.2009.263

Mustafa, N., Ibrahim, A. O., Ahmed, A., & Abdullah, A. (2017). Collaborative filtering: Techniques and applications. *2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE)*, 1–6. https://doi.org/10.1109/ICCCCEE.2017.7867668

Permana, K. E. (2024). Comparison of User Based and Item Based Collaborative Filtering in Restaurant Recommendation System. *Mathematical Modelling of Engineering Problems*, *11*(7), 1922–1928. https://doi.org/10.18280/mmep.110723

Rana, A., & Deeba, K. (2019). Online Book Recommendation System using Collaborative Filtering. *Journal of Physics: Conference Series*, *1362*(1), 012130. https://doi.org/10.1088/1742-6596/1362/1/012130

Saat, N. I. Y., Mohd Noah, S. A., & Mohd, M. (2018). Towards Serendipity for Content–Based Recommender Systems. *International Journal on Advanced Science, Engineering and Information Technology*, *8*(4–2), 1762–1769. https://doi.org/10.18517/ijaseit.8.4-2.6807

Sedyo Mukti, P. A., & Baizal, Z. K. A. (2025). Enhancing Neural Collaborative Filtering with Metadata for Book Recommender System. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, *19*(1), 61. https://doi.org/10.22146/ijccs.103611

T. Li, Y. Dong, & B. Zhang. (2023). Algorithm based personalized push Research on "Information Cocoon Room." *2023 8th International Conference on Information Systems Engineering (ICISE)*, 286–289. https://doi.org/10.1109/ICISE60366.2023.00066

Tewari, A. S. (2020). Generating Items Recommendations by Fusing Content and User-Item based Collaborative Filtering. *Procedia Computer Science*, *167*, 1934–1940. https://doi.org/10.1016/j.procs.2020.03.215

Uta, M., Felfernig, A., Le, V.-M., Tran, T. N. T., Garber, D., Lubos, S., & Burgstaller, T. (2024). Knowledge-based recommender systems: Overview and research directions. *Frontiers in Big Data*, *7*, 1304439. https://doi.org/10.3389/fdata.2024.1304439