Penerbit ITDel

Journal of Intelligent Computing, Systems, and Applications
e-ISSN: 3123-1705

Vol. 2 No. 1 (2026) 1-18
https://journal.del.ac.id/jicsa

JICSA

Performance Comparison of Recommendation Models: IBCF and ALS Based on Implicit Feedback

Bronson Siallagan[1*], Wilson Sihombing[1], Winfrey Nainggolan[1], Samuel Sitio[1], Reinhard Batubara[1], Wiwin Sry Adinda Banjarnahor[2]

[1]*Information System, Faculty of Informatics and Electrical Engineering, Institut Teknologi Del, Toba, North Sumatera, 22381, Indonesia*
[2]*Computer Engineering, Department of Computer Engineering, Politeknik Negeri Medan, Medan, North Sumatera, 20155, Indonesia*

*Corresponding Author: iss22025@students.del.ac.id

Article Info

Received: 19 March 2026
Revised: 18 September 2026
Accepted: 19 September 2026
Available online: 30 September 2026

Keywords

Recommendation System, Collaborative Filtering, Implicit Feedback, Data Sparsity, Alternating Least Squares (ALS), Item-Based Collaborative Filtering (IBCF)

Abstract

The rapid growth of digital platforms has led to an overwhelming amount of information, making it difficult for users to discover relevant books among the vast selection available. Recommendation systems, particularly collaborative filtering techniques, help address this issue by personalizing recommendations based on user interactions. This study compares the performance of three recommendation models: Item-Based Collaborative Filtering (IBCF) as a baseline, Alternating Least Squares (ALS), and ALS with hyperparameter tuning (ALS-Tuning) in the context of implicit feedback data. Using two evaluation metrics, Mean Average Precision at 10 (MAP@10) and Root Mean Squared Error (RMSE), the results demonstrate that ALS-Tuning outperforms both IBCF and the standard ALS model in term of MAP@10 where hyperparameter tuning playing a significant role in enhancing the model's ability to uncover latent user preferences. While IBCF struggles with sparse data, ALS, especially with tuning, proves to be more effective in capturing latent patterns in implicit feedback data. This research provides empirical evidence supporting the use of ALS-Tuning as a robust approach for recommendation systems with sparse and implicit data, offering valuable insights for future development in personalized recommendation systems.

This is an open access article under the CC BY-NC-SA 4.0 license.

1. 1. Introduction

The rapid growth of digital platforms that provide a vast collection of books, such as online bookstores and digital book catalog services, has led to an exponential increase in the number of cataloged items and logged user interactions [1], [2]. This situation inevitably creates an information overload phenomenon, where users struggle to discover books that align with their personal interests and reading preferences among thousands or even millions of available titles [3]. In this context, recommendation systems play a crucial role as an automated personalization mechanism that assists users in discovering relevant books by filtering, scoring, and prioritizing items based on historical user interaction patterns, such as searches, clicks, page views, and purchases [4], [5].

One of the most widely adopted paradigms in recommendation systems is Collaborative Filtering, specifically Item-Based Collaborative Filtering (IBCF), which operates by calculating statistical similarities between items based on historical user interaction overlaps [6]. While IBCF is relatively intuitive, computationally transparent, and stable across expanding user bases, its performance degrades significantly in scenarios where user–item interactions are extremely sparse [7]. Nevertheless, IBCF remains an essential, widely utilized empirical baseline across diverse recommender system studies owing to its algorithmic simplicity and strong capability in capturing direct item-to-item associations [8].

On the other hand, matrix factorization techniques such as Alternating Least Squares (ALS) offer a more flexible mathematical framework by projecting users and items into a shared low-dimensional latent factor space [9]. ALS has proven particularly effective in uncovering subtle, unobserved interaction patterns from implicit feedback data [10]. However, ALS models exhibit high sensitivity to hyperparameter configurations, including the number of latent factors, regularization penalties, and implicit confidence weighting factors [11]. Consequently, an arbitrary or default parameter selection frequently leads to severe underfitting or overfitting. To systematically address this challenge, this study incorporates automated hyperparameter tuning powered by Optuna, an efficient define-by-run hyperparameter optimization framework utilizing Bayesian Tree-structured Parzen Estimators (TPE) to discover optimal model parameter combinations [12].

This study aims to conduct a rigorous comparative performance analysis between IBCF and ALS, incorporating an in-depth evaluation of ALS performance gains attained through Optuna-guided hyperparameter optimization. The primary focus is to assess the relative ranking and prediction accuracy of ALS against IBCF under real-world implicit feedback conditions. By adopting standardized evaluation metrics—specifically Mean Average Precision at 10 (MAP@10) to quantify ranking relevance and Root Mean Squared Error (RMSE) to evaluate error divergence [13]—this work provides empirical insights into how latent representations compare against explicit similarity heuristics in sparse environments.

Although IBCF and ALS are individually well-established techniques, the specific contribution of this study lies in the joint evaluation of both paradigms on an authentic, extremely sparse implicit-feedback book-interaction dataset (sparsity = 99.984203%, density = 0.015797%), combined with a transparent, documented hyperparameter search trajectory. Unlike prior comparative works that typically present only a single best configuration without optimization context [14], this work details the hyperparameter search space and links the resulting performance gap directly to the mathematical interaction between latent-factor dimensionality, regularization strength, and confidence weighting under severe data sparsity, providing actionable empirical guidance for real-world implicit recommendation deployment [15].

2. Literature Review

This chapter will discuss the theories underlying the development of recommendation systems, including the basic concepts of recommendation systems, the Item-Based Collaborative Filtering (IBCF) algorithm, Alternating Least Squares (ALS), and the evaluation techniques used in this research. A strong understanding of these theories is crucial for gaining insight into how recommendation systems work and why these two approaches were chosen for this study.

2.1 Recommendation System

Recommendation systems are algorithmic architectures engineered to suggest relevant items to users by evaluating available data sources, including user preference profiles, item metadata, and historical interaction logs [4], [5]. Today, recommendation engines are broadly deployed across diverse online domains such as e-commerce, digital streaming, social platforms, and digital libraries, serving as essential mechanisms to guide users toward items tailored to their personal tastes [9], [10], [16].

Recommendation systems can be categorized into several foundational paradigms. Content-Based Filtering recommends items based on the descriptive features of items and their similarity to items previously favored by a user [5]. Collaborative Filtering generates recommendations by discovering patterns and correlations across user–item interactions without requiring explicit item attributes [3], [10]. Collaborative filtering is commonly bifurcated into User-Based Collaborative Filtering, which computes neighborhood similarities among users, and Item-Based Collaborative Filtering, which computes similarity structures among items based on shared co-consumption [3], [6]. Another distinct category is Knowledge-Based Recommendation Systems, which leverage structured domain rules and explicit user constraints to generate recommendations, particularly useful in infrequent purchasing scenarios [1]. Finally, Hybrid Recommendation Systems combine multiple techniques to overcome individual algorithmic limitations such as cold-start and sparsity [5], [17].

Content-based recommendation approaches have also been thoroughly investigated in specialized application domains. In [18], the authors conducted a comparative analysis between Content-Based Filtering and TF-IDF term weighting for sports recommendation systems, highlighting the critical role of textual representations in item profiling. In contrast to attribute-reliant content methods, the present investigation focuses purely on collaborative filtering architectures that infer user preferences directly from implicit behavioral signals.

2.2 Item-Based Collaborative Filtering (IBCF)

Item-Based Collaborative Filtering (IBCF) is a neighborhood recommendation approach that determines item similarities by analyzing historical user interaction vectors [6], [7]. Unlike user-based approaches whose neighborhood structures fluctuate rapidly as new users join, item similarities in IBCF remain substantially more stable over time because items accumulate interaction histories gradually. This stability makes IBCF computationally efficient and scalable for production environments.

Collaborative filtering algorithms have also been explored to advance recommendation goals beyond traditional accuracy. Pardede et al. [8] investigated recommendation diversity in media content using collaborative filtering, emphasizing the necessity of delivering a varied assortment of items alongside preference alignment. While their work focuses on recommendation diversity, the present study investigates the comparative robustness of item similarity versus latent factor modeling under extreme data sparsity.

In IBCF, user interactions—such as clicks, views, or purchase events—are formalized in an m x n user–item interaction matrix. Each item i is represented by an interaction vector over all users. The similarity between

two items i and j is commonly calculated using Cosine Similarity [6], [19], which evaluates the geometric angle between the two vectors as formulated in Eq. (1).

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \, \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \, \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{1}$$

Eq. (1) shows the dot product of the interaction vectors of items i and j as the numerator and the product of the magnitudes of their vectors as a denominator. Higher similarity scores indicate that items are more similar in the pattern of user interaction and can be used to generate recommendations.

IBCF has the advantage of being conceptually simple and interpretable, but it relies on enough shared user interactions. In extremely sparse datasets, item co-occurrence is absent which may affect the trustworthiness of similarity estimates, and hence the quality of recommendations. This limitation motivates the comparison with latent factor-based approaches, such as ALS, which learn representations of users and items from the interaction matrix.

2.3 Alternating Least Squares (ALS)

Alternating Least Squares (ALS) is a prominent matrix factorization approach designed to factorize the sparse user–item interaction matrix into low-rank latent user factors and latent item factors [9], [10]. Rather than relying strictly on direct item co-occurrence, matrix factorization decomposes the interaction space into a k-dimensional latent representation, enabling the model to predict preference scores for unobserved user–item pairs across the entire catalog.

Matrix factorization has been extended to various complex recommendation scenarios. TopC-CAMF is a context-aware matrix factorization model proposed by Lumbantoruan et al. [20] that integrates contextual variables directly into the factorization process. While their work explores context-aware factorization, the present study investigates ALS specifically under implicit feedback and analyzes how automated Bayesian hyperparameter tuning impacts latent factor convergence and recommendation accuracy in sparse settings.

In implicit feedback settings, user preferences are derived from indirect behavioral signals (clicks, views, purchases) rather than explicit numerical ratings [10]. Under implicit feedback, observed interactions represent positive preference with varying degrees of confidence, whereas unobserved entries do not necessarily indicate dislike, but rather a lack of exposure. Confidence weights are thus employed during ALS training to formalize this asymmetric distinction.

ALS optimizes the objective loss function by alternately fixing one factor matrix while solving for the other via quadratic least squares. In each iteration, item factors are held constant to optimize user factors, and subsequently user factors are held constant to optimize item factors. This alternating sequence is executed iteratively until convergence or until a predefined iteration count is reached [9], [10].

Unlike neighborhood techniques such as IBCF, ALS uncovers latent semantic relationships that are not explicitly captured by surface-level item co-occurrence, making it a highly effective framework for sparse interaction data [21]. Nevertheless, model convergence and predictive quality remain heavily dependent on hyperparameter tuning, especially latent dimensionality, regularization penalties, and confidence coefficients [11], [12], [22]. Therefore, comparing standard ALS against hyperparameter-tuned ALS is essential to quantify the empirical benefits of systematic optimization.

2.4 Challenges in Recommendation Systems

The development of recommendation systems, particularly those relying on Collaborative Filtering techniques and operating with implicit feedback, faces several fundamental challenges that can affect the accuracy and scalability of the model. These main challenges include data sparsity, the system's ability to handle new users and items, and computational efficiency.

2.4.1 Data Sparsity

Data sparsity is a pervasive challenge where user–item interactions constitute only a minute fraction of the possible interaction space, resulting in a largely vacant interaction matrix [7], [23]. This condition severely restricts the ability of similarity-based algorithms like IBCF to calculate reliable co-occurrence statistics. Consequently, latent factor models like ALS that map interactions into compact embeddings and leverage implicit confidence weighting are vital to achieving consistent recommendation quality under extreme sparsity [10], [15].

2.4.2 Cold Start Problem

The Cold Start Problem emerges whenever a new system is deployed, a new user registers (new user problem), or a new item is added to the catalog (new item problem) [24]. Under these conditions, the lack of sufficient historical interactions impedes both similarity calculations and latent factor initialization. While matrix factorization mitigates this challenge more effectively than pure neighborhood approaches once minimal interactions are recorded, cold-start handling remains an active area of recommender system research.

2.5 Evaluation Metrics

Evaluating a recommendation system is a crucial procedure to assess the ranking quality, accuracy, and practical utility of developed models [4], [5], [13]. In implicit feedback environments where explicit ratings are absent, evaluating recommendation effectiveness requires a dual perspective: ranking relevance and error divergence. This study utilizes Mean Average Precision at 10 (MAP@10) to evaluate ranking relevance [25] and Root Mean Squared Error (RMSE) to assess numerical prediction error [26].

2.5.1 Mean Average Precision (MAP@10)

Mean Average Precision at 10 (MAP@10) is an evaluation metric used to assess the performance of a recommendation system, specifically in evaluating the quality of the ranking of recommendations generated. MAP@10 specifically measures the system's ability to place relevant items within the top ten positions of the recommendation list provided to each user.

In general, the MAP value is obtained by computing the mean of the Average Precision (AP) scores across all evaluated users in the test set. The mathematical formulation for calculating MAP is expressed in Eq. (2), where AP(u) is summed across all users and divided by the total number of users |U|.

$$MAP = \frac{1}{|U|} \sum_{u \in U} AP(u) \tag{2}$$

The Average Precision (AP) for a given user u is calculated based on precision scores at each rank position k up to cutoff K = 10. The formula for AP is given in Eq. (3), where P(k) represents the precision at cutoff k, and rel(k) is a binary indicator denoting whether the item at position k is relevant.

$$AP(u) = \frac{1}{K} \sum_{k=1}^{K} P(k) \cdot rel(k) \tag{3}$$

The use of MAP@10 as an evaluation metric provides a comprehensive overview of the performance of a recommendation system, as it not only considers the number of relevant items recommended but also takes into account the order in which these items appear. The higher the MAP@10 value, the better the quality of the system's recommendations in consistently presenting relevant items to users.

2.5.2 Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) is a commonly used evaluation metric to measure the prediction error in recommendation systems, particularly in systems that generate numerical predictions, such as ratings. RMSE measures the difference between the actual value and the predicted value generated by the system, providing an indication of how accurate the model is in predicting user preferences.

The mathematical formulation for calculating RMSE is defined in Eq. (4). In this equation, r_ui represents the actual interaction signal, r̂_ui denotes the reconstructed preference score estimated by the model, and N denotes the total number of evaluated user–item interaction pairs.

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (r_i - \hat{r}_i)^2} \tag{4}$$

A smaller RMSE value indicates that the difference between the predicted and actual values is smaller, suggesting that the recommendation system's performance is better. Conversely, a larger RMSE value indicates that the system generates less accurate predictions. Therefore, RMSE is often used as a primary evaluation metric when comparing the performance of different recommendation methods or models, especially in approaches based on rating predictions.

3. Research Methodology

3.1 Research Workflow

Research Workflow illustrates the steps taken in this research to develop and evaluate a recommendation system based on user-item interaction data. The flowchart presented outlines the overall process, starting from problem formulation to the final conclusion based on the model evaluation results.

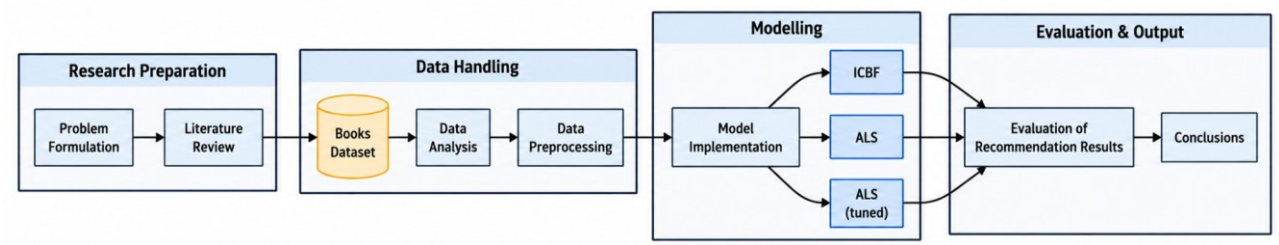

Fig. 1 Research phases workflow

This research begins with the problem formulation stage, which focuses on comparing the performance of several recommendation system models in handling sparse data dominated by implicit feedback. The main

issue addressed is the limited user preference information due to the absence of explicit ratings, which makes it difficult for recommendation systems to generate relevant recommendations. Based on this issue, the study aims to compare the effectiveness of Item-Based Collaborative Filtering (IBCF), ALS (Baseline), and ALS (Tuned) in generating recommendations in the book domain.

The next stage is the literature review, conducted to examine previous research related to recommendation systems, particularly Collaborative Filtering approaches. This stage discusses the basic concepts of IBCF and ALS, the characteristics of their application on implicit feedback data, and common challenges that arise, such as data sparsity and the limitations of similarity-based methods. The findings from this literature review serve as the theoretical foundation for model selection and experimental design in this study.

Once the theoretical foundation is established, the research proceeds with the use of the Books Dataset as the primary data source. This dataset contains interactions between users and items (books), represented by implicit feedback such as clicks, views, or purchases. The dataset is used as a representation of user interest in items without explicit ratings, aligning with the focus of the research problem.

In the data analysis stage, an initial exploration of the dataset is conducted to understand its characteristics, including the distribution of user-item interactions and the existing sparsity level. This analysis aims to identify challenges that might affect model performance and ensure that the approach used is relevant to the data conditions. This stage also includes a data quality check to ensure the data is ready for the modeling phase.

During the data preprocessing stage, data filtering is performed to reduce noise and improve the quality of the user–item interaction matrix before the modeling phase. Preprocessing focuses on removing users and items with very low interaction counts, as they provide limited information and may exacerbate data sparsity. This process is carried out iteratively by applying minimum interaction thresholds until the dataset size becomes stable. The iterative approach ensures consistency in the filtering process, since removing users can affect item interaction counts and vice versa. After preprocessing, the resulting dataset is reanalyzed to determine the number of users, items, total interactions, and the sparsity level in order to evaluate the impact of preprocessing on data density.

The next step is the model implementation stage, where three recommendation system approaches are applied in parallel: IBCF, ALS (Baseline), and ALS (Tuned). IBCF is implemented by calculating the similarity between items based on user interaction patterns, generating recommendations from items that have a high similarity. Meanwhile, ALS (Baseline) uses a matrix factorization approach to map users and items into a latent space to capture hidden preference patterns in sparse data. Additionally, ALS (Tuned) is applied as a variant of ALS by tuning hyperparameters, such as the number of latent factors and regularization parameters, to achieve an optimal model configuration.

After the models are implemented, the next step is the evaluation of the recommendation results. Evaluation is conducted to assess and compare the performance of the three models using the MAP@10 and RMSE metrics. MAP@10 is used to measure the quality of rankings by assessing the relevance of items that appear in the top positions, while RMSE is used to measure the prediction error of the models. The use of these two metrics allows for a more comprehensive evaluation of the quality of recommendations.

Based on the evaluation results, the final stage of the research is the conclusion, where the findings are used to determine the most effective model for generating recommendations in sparse data conditions and with implicit feedback. Additionally, this research provides recommendations on model selection and highlights the important role of hyperparameter tuning in improving the performance of recommendation systems.

3.2 Dataset and Exploratory Analysis

The dataset used in this research consists of user-item interaction data that is based on implicit feedback. This data represents user activity with items without explicit ratings. The recorded interactions reflect the user's interest in a specific book, such as clicks, views, or purchases, which are then used as the foundation for building the recommendation system. The dataset is stored in a file named data-books, with each row formatted as a pair of user_id and item_id. Each row represents one interaction between a user and a book. In this dataset, user_id represents a unique identifier for the user, while item_id represents a unique identifier for the book. The absence of explicit rating information means that all interactions are treated as positive signals (positive-only feedback).

	user_id	item_id
0	8	0002005018
1	8	074322678X
2	8	0887841740
3	8	1552041778
4	8	1567407781
...	...	...
269759	278854	0316184152
269760	278854	0375703053
269761	278854	042516098X
269762	278854	0425163393
269763	278854	0553275739

Fig. 2 Example of books dataset

3.3 Data Distribution Analysis

An initial analysis of the dataset was conducted to understand the distribution of user interactions, the sparsity level of the data, and relevant usage patterns for selecting the recommendation system model. This stage aims to identify key characteristics of the dataset as well as potential challenges that may impact the performance of the recommendation models to be applied.

Based on the distribution of the number of interactions per user, a significant imbalance in activity levels is observed. This is shown in Figure 3, which displays the top 10 most active users in terms of interaction counts. One user with User ID 11676 has a substantially higher number of interactions compared to other users, with more than 6,450 interactions, followed by User ID 98391 with over 4,642 interactions. Meanwhile, other users in the top 10 group show a sharp decline in interaction numbers, ranging from 700 to 1,500 interactions.

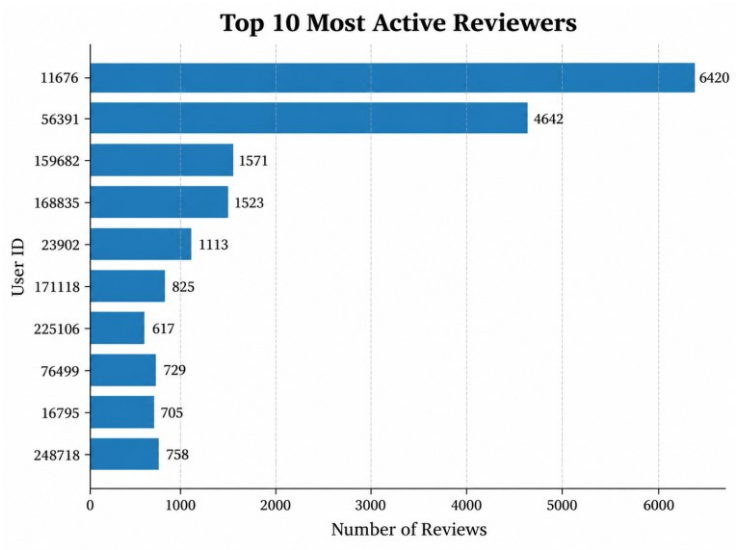

Fig. 3 Top 10 users with the highest number of interactions

This distribution of interactions is also reflected in the descriptive statistics presented in Table X. The dataset consists of 13,876 users, with an average of 19,441 interactions per user, a median of 8, and a third quartile (75%) value of 16 interactions. The high standard deviation of 79.274 indicates significant variation among users. This condition reflects a common characteristic in book-based recommendation systems, where most users are passive, and only a few users have intensive interaction activity. This pattern directly indicates the challenge of data sparsity in recommendation modeling.

Table 1 Descriptive statistics of user interactions in the book dataset

Statistic	Value
Count	13876
Mean	19.4410
Std. Deviation	79.274
Minimum	4
Q1 25%	5
50%	8
Q3 75%	16
Maximum	64

To quantitatively measure the data sparsity, the calculation of total potential interactions was performed, defined as the product of total users and total items (|U| x |I|). With a total of 13,876 users and 123,069 items, there are 1,707,705,444 possible interaction pairs. However, the recorded interactions amount to only 269,764, resulting in a matrix density of just 0.015797%. Formally, sparsity S is defined as S = (1 - |R| / (|U| x |I|)) x 100%, demonstrating that over 99.98% of potential entries are unobserved.

This very low density results in a sparsity value of 99.984203%, which emphasizes that the dataset is in a highly sparse condition. This condition is a key factor in choosing a recommendation system approach based on implicit feedback. Therefore, Item-Based Collaborative Filtering (IBCF) and Alternating Least Squares

(ALS) were selected, as both methods are designed to work effectively with highly sparse data, either by leveraging item similarities or by learning latent representations of users and items.

$$\begin{aligned}\text{Possible Interactions} &= \text{Total Users} \times \text{Total Items} \\ &= 13{,}876 \times 123{,}069 \\ &= 1{,}707{,}705{,}444\end{aligned}$$

$$\begin{aligned}\text{Density} &= \frac{\text{Total Interactions}}{\text{Possible Interactions}} \times 100\% \\ &= \frac{269{,}764}{1{,}707{,}705{,}444} \times 100\% \\ &= 0.0001579687 \times 100\% \\ &= 0.015797\%\end{aligned}$$

$$\begin{aligned}\text{Sparsity} &= (1 - \text{Density}) \times 100\% \\ &= (1 - 0.0001579687) \times 100\% \\ &= 99.984203\%\end{aligned}$$

Fig. 4 Sparsity distribution of the books interaction dataset

3.4 Data Preprocessing

The data preprocessing stage is designed to address the primary operational challenges of the dataset, namely severe matrix sparsity and the presence of low-activity users and infrequent items. The complete preprocessing workflow is illustrated in Fig. 5.

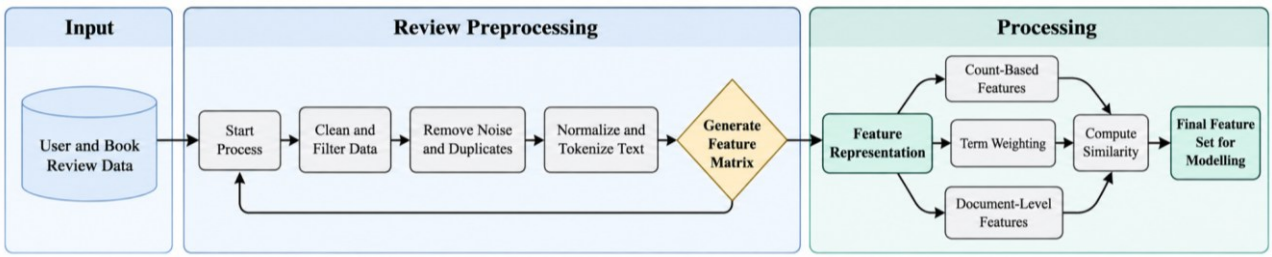

Fig. 5 Flow of the data preprocessing stages

Data filtering is performed iteratively to ensure consistency in the user–item interaction structure. This approach is based on the assumption that removing users with low interaction counts may cause some items to fall below the minimum interaction threshold, and vice versa. Therefore, the filtering process is repeated until the dataset size no longer changes, indicating that a stable condition has been reached.

In each iteration, the number of interactions per user is calculated, and users with fewer than five interactions are removed from the dataset. Subsequently, the number of interactions per item is computed, and items with fewer than fifteen interactions are also removed. Analytically, applying different thresholds for users and items aims to reduce the effect of the extreme long tail in the interaction distribution while improving the relative density of the user–item matrix to better support the modeling process.

After the preprocessing stage is completed, the dataset is reanalyzed by computing the number of unique users, the number of unique items, and the total remaining interactions. Based on these values, a new sparsity level is calculated to evaluate the effectiveness of preprocessing in reducing data sparsity. This step ensures that the resulting dataset is more balanced, more informative, and better suited for the application of implicit-

feedback-based recommendation models, particularly Item-Based Collaborative Filtering and Alternating Least Squares.

3.5 Recommendation Models Implementation

3.5.1 Item-Based Collaborative Filtering (IBCF)

Item-Based Collaborative Filtering (IBCF) is a recommendation system model that generates recommendations based on the similarity between items, which is calculated from the user interaction patterns with those items. This approach focuses on the relationships between items, rather than between users, so recommendations are made by leveraging items that the user has previously interacted with.

Based on the flow shown in Fig. 6, the IBCF process begins with the input stage, which ingests user–item interaction data in the form of implicit feedback (searches, clicks, views, purchases). This data is structured into a sparse user–item matrix reflecting historical interaction patterns.

Next, the system performs the similarity computation stage, where the similarity between items is calculated based on the user interaction vectors. In this stage, each item is represented as a user interaction vector, and the similarity between items is computed using a specific metric, such as Cosine Similarity. The result of this process is an item-item similarity matrix, which serves as the basis for the recommendation process.

The following stage is the recommendation process. In this stage, the system first retrieves the items that the user has interacted with (retrieve user's interacted items). Then, the system selects the items that have the highest similarity to those items (select similar items). The similarity values between items are then aggregated (aggregate similarity) to generate recommendation scores for each candidate item. Based on these scores, the system performs the ranking (rank candidate) to determine the order of the most relevant items.

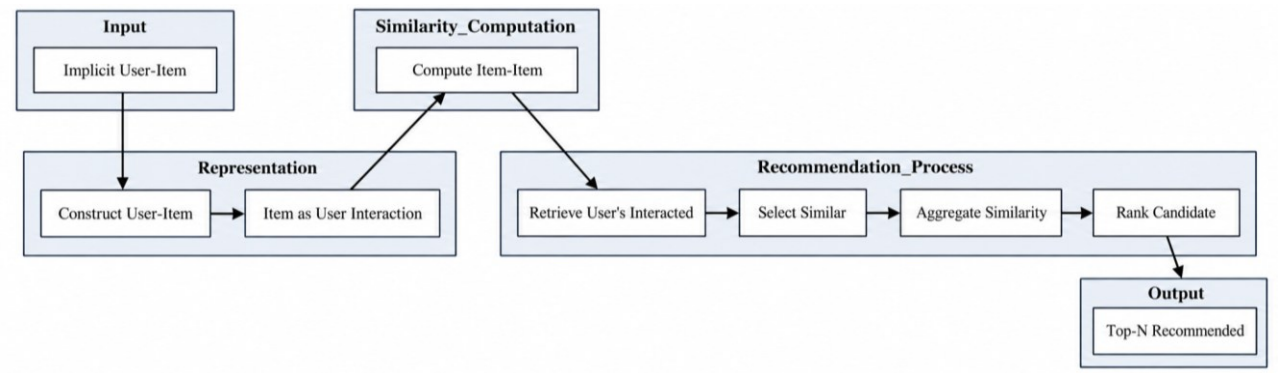

Fig. 6 IBCF model recommendation process flow

The final output of the IBCF pipeline is a Top-N list of recommended items, as depicted in the output stage of Fig. 6. The IBCF model offers high stability and interpretability, though its ranking fidelity drops in extremely sparse regions.

3.5.2 Alternating Least Squares (ALS)

Alternating Least Squares (ALS) is a matrix factorization-based recommendation method widely used to handle implicit user-item interaction data. This method works by factorizing the user-item interaction matrix

into two latent matrices: the user factor matrix and the item factor matrix, which are used to model user preferences towards items.

Based on the workflow depicted in Fig. 7, the ALS process begins with the input stage containing implicit user–item interactions. Confidence weights are computed to assign proportional trust to observed interactions during latent decomposition.

Next, in the factorization stage, the system initializes the latent factors for users and items (initialize latent factors). Optimization is then carried out alternately (alternating optimization), where the item factor matrix is updated and the user factor matrix is optimized (optimize user), followed by the user factor matrix being updated and the item factor matrix optimized (optimize item). This process is done iteratively until convergence is achieved, indicated by a reduction in error values or by meeting the stopping criteria (converged).

Once factorization converges, the system computes preference prediction scores across user–item pairs. These scores are ranked to produce a Top-N recommendation list for each active user, as shown in the output stage of Fig. 7.

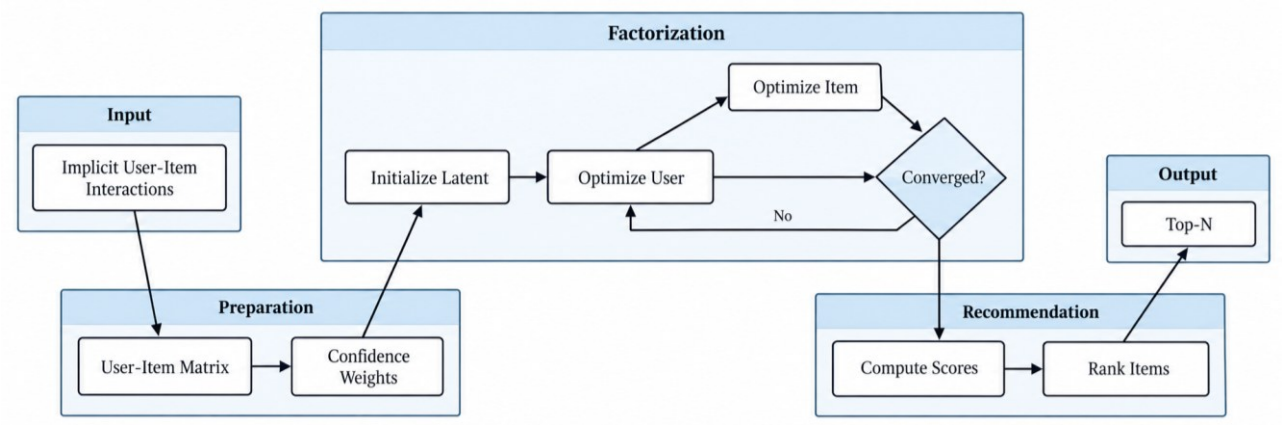

Fig. 7 Process flow of the ALS recommendation model

The ALS model has the advantage of handling the sparsity problem in interaction data because it leverages latent factors to capture hidden patterns in the data. Additionally, ALS is efficient and easily parallelizable, making it suitable for use with large-scale datasets. However, the selection of parameters such as the number of latent factors and confidence weights significantly impacts the model's performance and must be optimized carefully.

3.5.3 ALS Hyperparameter Tuning with Optuna

The tuning process in the Alternating Least Squares (ALS) method is carried out to achieve the best configuration of parameters for optimal recommendation system performance. The goal of parameter tuning is to adjust the model so that it can effectively represent user-item interaction patterns, especially in implicit feedback data, which is often sparse. By fine-tuning parameters such as the number of latent factors, regularization terms, and confidence weights, the model is better able to capture hidden patterns in the data, thereby improving the accuracy and relevance of the recommendations generated.

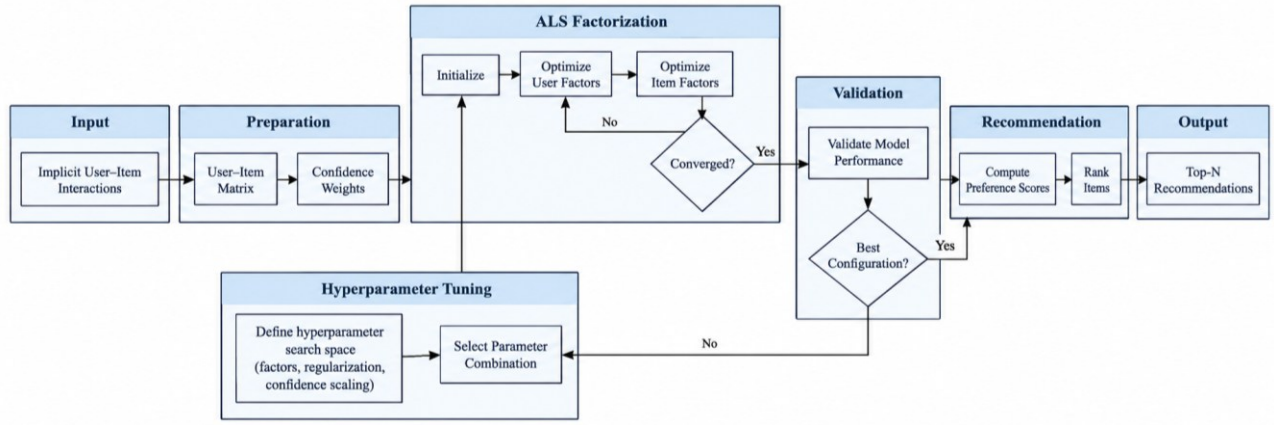

Fig. 8 ALS model recommendation process flow with hyperparameter tuning

Based on the flow in Fig. 8, the ALS tuning pipeline ingests implicit interaction records, constructs the sparse matrix, and sets up confidence weighting before initiating automated optimization.

In the hyperparameter optimization stage, search ranges for latent factors (factors), regularization penalty (regularization), and confidence scaling factor (alpha) are systematically configured within the Optuna framework [12]. Optuna explores parameter configurations using Tree-structured Parzen Estimators to iteratively maximize recommendation relevance.

Once the parameters are defined, the model moves to the ALS factorization stage. At this stage, initial latent factors are initialized, followed by alternating optimization of the user factors and item factors. This optimization process is performed iteratively until the model reaches convergence, indicated by a decrease in the error value or the fulfillment of stopping criteria (converged).

After the model reaches convergence, the validation stage is carried out to evaluate the model's performance based on the predetermined evaluation metrics. If the parameter configuration does not produce the best performance (best configuration), the tuning process will return to the parameter combination selection stage and proceed with retraining the model. This process is repeated until the optimal parameter configuration is obtained.

Once the best configuration is found, the model moves to the recommendation stage to calculate user preference scores for items (compute preference scores). Based on these scores, the system ranks the items (rank items) and generates a list of the Top-N most relevant items for each user.

Through this systematic ALS tuning process, the model is expected to generate more accurate and stable recommendations. This process also plays a key role in improving recommendation quality by balancing model complexity and the ability to generalize to user-item interaction data.

4. Results and Discussion

This section presents the experimental results comparing the performance of three recommendation models: Item-Based Collaborative Filtering (IBCF) as the baseline, Implicit Feedback with ALS, and ALS with hyperparameter tuning (ALS-Tuning). The evaluation is conducted using two primary metrics: Mean Average Precision at 10 (MAP@10) to measure ranking relevance, and Root Mean Squared Error (RMSE) to assess prediction accuracy.

Table 2 Model evaluation performance results comparison

Method	MAP@10	RMSE
IBCF	0.108897	1.822437
ALS	0.159405	0.642507
ALS (Tuning)	**0.180693**	**0.661034**

The quantitative evaluation presented in Table 2 benchmarks the three evaluated models: IBCF, default ALS, and Optuna-tuned ALS. Specifically, the ALS-Tuning model achieves superior performance in terms of MAP@10 followed by standard ALS and baseline IBCF. To provide a clear visual comparison of performance dynamics, Fig. 9 presents the comparative MAP@10 and RMSE metrics across all models.

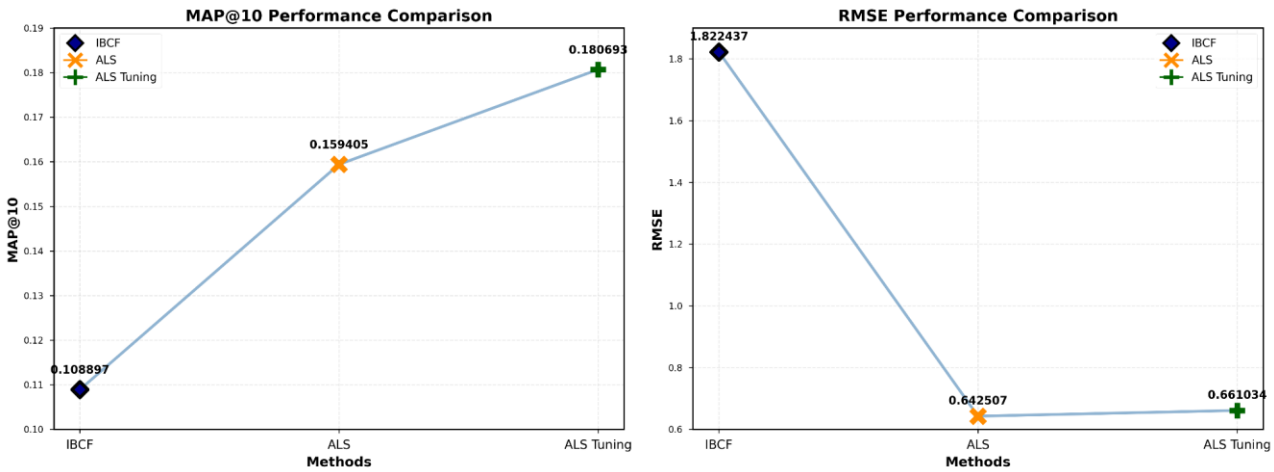

Fig. 9 Comparison of MAP@10 and RMSE performance across models

The empirical performance profiles depicted in Fig. 9 show that ALS-Tuning delivers the highest MAP@10 score while maintaining a competitive RMSE compared to IBCF. In the context of top-N recommendation where top rankings directly determine user CTR and satisfaction, the MAP@10 improvement demonstrates clear practical advantage.

4.1 Relevance Performance Analysis (MAP@10)

Based on Table 2, the ALS-Tuning model achieves the best relevance performance with a MAP@10 value of 0.180693, followed by the standard ALS model (0.159405), and IBCF (0.108897). Substantially, the increase in MAP@10 for ALS-Tuning indicates that hyperparameter tuning has improved the model's ability to rank relevant items higher in the recommendation list. Although the difference in MAP@10 values between the models is relatively small, this difference still has a significant impact in top-N recommendation scenarios, as small changes in MAP@10 can reflect changes in the ranking of items in important positions, potentially increasing user exposure and the number of clicks.

Furthermore, the improvement gap between ALS and ALS-Tuning specifically originates from the interaction between the number of latent factors and the regularization strength. A higher number of latent factors allows the model to represent more nuanced user-item preference structures, but without sufficient regularization this capacity leads to overfitting on the extremely sparse interaction matrix (density = 0.015797%). The tuning process, by jointly searching over latent factors and regularization via Optuna, converges toward a configuration that balances model expressiveness against the risk of overfitting, which explains why the MAP@10 gain, although numerically modest, is methodologically consistent rather than incidental.

The methodological explanation for the relevance improvement in ALS-Tuning can be attributed to three tuning components commonly used in ALS for implicit feedback: (1) the number of latent factors, which controls the model's capacity to represent user preferences and item characteristics; (2) regularization, which balances bias and variance to prevent overfitting on limited interactions; and (3) the alpha value (confidence adjustment), which determines how much the model "trusts" implicit signals (e.g., clicks, views, transactions) compared to unobserved data. The more precise combination of parameters allows ALS-Tuning to form more stable latent representations, resulting in relevant items being ranked higher in the recommendations more often.

4.2 Prediction Accuracy Analysis (RMSE)

The most noticeable difference is observed in the RMSE metric. The IBCF model produces an RMSE of 1.822437, which is much higher compared to ALS (0.642507) and ALS-Tuning (0.661034). The high RMSE value in IBCF indicates that the model's predictions often deviate significantly from the actual/estimated preferences in the evaluation. This can be understood because the item-based approach relies on item similarity calculated from co-occurrence interactions. In recommendation datasets that are generally very sparse (with most user-item interactions missing), sufficient co-occurrence data to form reliable similarity measures is often unavailable, making the similarity estimation fragile and predictions unstable.

In contrast, ALS, based on matrix factorization, projects users and items into a low-dimensional latent factor space. This approach enables the model to capture latent patterns (e.g., implicit preferences that are not captured by direct item similarity) and make better generalizations, even with sparse interaction data. In other words, when two items lack sufficient co-occurrence, ALS can still link them through their proximity in the latent space based on global interaction patterns. This explains why the RMSE for ALS drops significantly compared to IBCF.

Furthermore, in terms of prediction error, baseline IBCF exhibits an elevated RMSE of 1.822437 due to its inability to generalize over unobserved interaction pairs. In contrast, standard ALS dramatically reduces RMSE to 0.642507, and ALS-Tuning maintains a similarly low error rate of 0.661034 while prioritizing ranking relevance (MAP@10). Regularization prevents overfitting to sparse vectors, and calibrated confidence weighting (alpha) ensures balanced learning from implicit clicks without over-trusting occasional noise.

4.3 Implications and Synthesis of Findings

Overall, the results from both metrics affirm that ALS-Tuning is the most robust approach for implicit feedback scenarios and highly sparse datasets. The improvement in MAP@10 indicates higher relevance in top-N recommendations, while the decrease in RMSE shows more accurate preference predictions. These findings reinforce the argument that latent factor-based models tend to outperform item-based similarity methods on sparse implicit data. It should be noted, however, that the MAP@10 and RMSE values reported in Table 2 are point estimates obtained from a single train–test split, and no significance test (e.g., a paired bootstrap or a Wilcoxon signed-rank test over per-user metric scores) was performed to establish whether the gap between ALS and ALS-Tuning is statistically significant. This is acknowledged as a limitation of the current evaluation, and a bootstrap-based significance analysis is planned as part of future revisions to more rigorously support the claim of ALS-Tuning's superiority.

However, the interpretation of these results should also take into account the characteristics of the metrics used. MAP@10 is more focused on the quality of recommendations at the top of the list, making it a more direct reflection of the user experience in a recommendation system. RMSE, on the other hand, measures the numerical prediction error and may be less representative for pure ranking purposes, especially when the evaluation target is not explicit ratings. Nevertheless, within the context of this experiment, the consistent

improvement in both metrics (MAP@10 and RMSE) suggests that ALS-Tuning enhances the overall model quality, both in terms of ranking and preference estimation accuracy. Therefore, ALS-Tuning is highly recommended as the primary model for implementing recommendation systems based on implicit feedback.

5. Conclusion

This study aims to compare the performance of three recommendation models: Item-Based Collaborative Filtering (IBCF) as a baseline, Alternating Least Squares (ALS), and ALS with hyperparameter tuning (ALS-Tuning) in the context of implicit feedback data. Based on the evaluation results using the Mean Average Precision at 10 (MAP@10) and Root Mean Squared Error (RMSE) metrics, it can be concluded that the ALS-Tuning model outperforms the other two models. This model successfully improves recommendation relevance by ranking more relevant items at the top, even though the difference with standard ALS and IBCF is not large. However, this difference is still significant in the context of top-N recommendations, which impact user exposure and clicks. Additionally, ALS-Tuning also shows a significant reduction in RMSE, indicating an improvement in the accuracy of user preference predictions. Compared to IBCF, which produces a much higher RMSE, ALS yields better results, demonstrating that matrix factorization methods are more effective in handling sparse data. The use of Optuna for hyperparameter tuning in ALS-Tuning has proven to have a positive impact on model performance, with adjustments to the number of latent factors, regularization, and alpha values improving recommendation accuracy and relevance. Therefore, the ALS-Tuning model is highly recommended for use in implicit feedback-based recommendation systems, particularly in highly sparse datasets such as those found in e-commerce platforms or digital book applications. This study has limitations, particularly regarding the type of interaction data used, and future research could explore applying this model to broader datasets or other techniques such as deep learning to further enhance performance. Overall, this study provides empirical evidence supporting the use of ALS-Tuning as a superior approach in implicit feedback-based recommendation systems.

Acknowledgement

The authors would like to thank the Faculty of Informatics and Electrical Engineering, Institut Teknologi Del, and the Department of Computer Engineering, Politeknik Negeri Medan, for the facilities and academic support provided throughout this research.

Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of this article. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Author Contribution

Bronson Siallagan: conceptualization, methodology, data preprocessing, ALS model implementation, and writing – original draft. Wilson Sihombing: IBCF model implementation, data analysis, and visualization. Winfrey Nainggolan: hyperparameter tuning experiments and validation. Samuel Sitio: manuscript preparation, revision coordination, and writing – review and editing. Reinhard Batubara: literature review and formal analysis. Wiwin Sry Adinda Banjarnahor: supervision, review, and editing. All authors have read and approved the final version of the manuscript.

AI Usage Declaration

We declare that we use AI Tools Grammarly in this manuscript to check for grammar and paraphrasing. The result from AI is not used directly and is under human supervision for error checking.

References

[1] S. Bouraga, I. Jureta, S. Faulkner, and C. Herssens, "Knowledge-based recommendation systems: A survey," International Journal of Intelligent Information Technologies, vol. 10, no. 2, pp. 1–19, 2014, doi: 10.4018/IJIIT.2014040101.

[2] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," Egyptian Informatics Journal, vol. 16, no. 3, pp. 261–273, 2015, doi: 10.1016/j.eij.2015.06.005.

[3] J. Ben Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in The Adaptive Web, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds., Lecture Notes in Computer Science, vol. 4321. Berlin, Heidelberg: Springer, 2007, pp. 291–324, doi: 10.1007/978-3-540-72079-9_9.

[4] J. A. Konstan, "Introduction to recommender systems: Algorithms and evaluation," ACM Transactions on Information Systems, vol. 22, no. 1, pp. 1–4, 2004, doi: 10.1145/963770.963771.

[5] F. Cacheda, V. Carneiro, D. Fernández, and V. Formoso, "Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems," ACM Transactions on the Web, vol. 5, no. 1, Art. no. 2, pp. 1–33, 2011, doi: 10.1145/1921591.1921593.

[6] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," IEEE Internet Computing, vol. 7, no. 1, pp. 76–80, 2003, doi: 10.1109/MIC.2003.1167344.

[7] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in Proceedings of the 10th International Conference on World Wide Web (WWW), Hong Kong, 2001, pp. 285–295, doi: 10.1145/371920.372071.

[8] C. Pardede, P. R. Togatorop, and P. G. Panjaitan, "Advancing diversity in recommendation systems through collaborative filtering: A focus on media content," Journal of Information Systems and Informatics, vol. 7, no. 1, pp. 731–742, 2025, doi: 10.51519/journalisi.v7i1.1045.

[9] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, vol. 42, no. 8, pp. 30–37, 2009, doi: 10.1109/MC.2009.263.

[10] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM), Pisa, Italy, 2008, pp. 263–272, doi: 10.1109/ICDM.2008.22.

[11] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 639–648, doi: 10.1145/3397271.3401063.

[12] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 2019, pp. 2623–2631, doi: 10.1145/3292500.3330701.

[13] G. Schröder, M. Thiele, and W. Lehner, "Setting goals and choosing metrics for recommender system evaluations," in Proceedings of the UCERSTI Workshop at the 5th ACM Conference on Recommender Systems (RecSys), Chicago, IL, USA, 2011, pp. 78–85.

[14] M. Ferrari Dacrema, P. Cremonesi, and D. Jannach, "Are we really making much progress? A worrying analysis of recent neural recommendation approaches," ACM Transactions on Information Systems, vol. 39, no. 2, Art. no. 12, pp. 1–49, 2021, doi: 10.1145/3417516.

[15] Z. Sun, Q. Guo, J. Yang, C. Fang, G. Guo, and J. Zhang, "Research commentary on recommendations with implicit feedback," IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 1, pp. 1–20, 2023, doi: 10.1109/TKDE.2021.3090040.

[16] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," ACM Computing Surveys, vol. 52, no. 1, Art. no. 5, pp. 1–38, 2019, doi: 10.1145/3285029.

[17] E. Çano and M. Morisio, "Hybrid recommender systems: A systematic literature review," Intelligent Data Analysis, vol. 21, no. 6, pp. 1487–1524, 2017, doi: 10.3233/IDA-163209.

[18] Herimanto, K. Samosir, and F. Ginting, "A comparative analysis of content-based filtering and TF-IDF approaches for enhancing sports recommendation systems," Innovation in Research of Informatics (INNOVATICS), vol. 6, no. 2, pp. 90–97, 2024, doi: 10.37058/innovatics.v6i2.12404.

[19] M. S. P. Subhadra, B. Vikas, and S. B. Nair, "Evaluation of similarity measures for collaborative filtering recommender systems in sparse environments," IEEE Access, vol. 10, pp. 42318–42330, 2022, doi: 10.1109/ACCESS.2022.3168231.

[20] R. Lumbantoruan, P. Simanjuntak, I. Aritonang, and E. Simaremare, "TopC-CAMF: A top context aware matrix factorization recommender system," Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI), vol. 11, no. 4, pp. 278–285, 2022, doi: 10.22146/jnteti.v11i4.4623.

[21] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in Proceedings of the 26th International Conference on World Wide Web (WWW), Perth, Australia, 2017, pp. 173–182, doi: 10.1145/3038912.3052569.

[22] S. Rendle, W. Krichene, L. Zhang, and J. Anderson, "Neural collaborative filtering vs. matrix factorization revisited," in Proceedings of the 14th ACM Conference on Recommender Systems (RecSys), Virtual Event, Brazil, 2020, pp. 240–248, doi: 10.1145/3383313.3412488.

[23] Y. Deldjoo, M. Schedl, P. Cremonesi, and G. Pasi, "Recommender systems driven by multimedia content: A survey and future directions," ACM Computing Surveys, vol. 53, no. 5, Art. no. 95, pp. 1–36, 2020, doi: 10.1145/3405527.

[24] B. L. William, R. S. Oetama, and H. L. H. S. Warnars, "A systematic literature review on cold-start problem in collaborative filtering recommender systems," Journal of Theoretical and Applied Information Technology, vol. 99, no. 18, pp. 4390–4403, 2021.

[25] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge: Cambridge University Press, 2008, doi: 10.1017/CBO9780511809071.

[26] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," ACM Transactions on Information Systems, vol. 22, no. 1, pp. 5–53, 2004, doi: 10.1145/963770.963772.