International Journal of Engineering Trends and Technology
ISSN: 2231-5381/ https://doi.org/10.14445/22315381/IJETT-V74I11P137

Volume 74 Issue 11, 577-594, November 2026
© 2026 Seventh Sense Research Group®

Original Article

A Robust Hybrid Collaborative Filtering Framework with Bayesian Ranking for Personalized Recommendation

Prashant Devidas Shimpi[1*], Sandip Patil[1]

[1]*SSBT's College of Engineering & Technology, Jalgaon (M.S.)*

[*]*Corresponding Author: prashant.shimpi30@gmail.com*

Received: 10 April 2026 Revised: 22 August 2026 Accepted: 28 August 2026 Published: 25 November 2026

Abstract *- This paper presents a hybrid approach to personalized product recommendations that combines collaborative filtering with 2 techniques: Bayesian rank-aware scoring and user-item similarity modelling via Non-Negative Matrix Factorization (NMF) applied to Amazon rating data. The two major issues addressed by this framework are the sparsity of user-item interactions and the popularity of items; the rank-aware hybrid collaborative filtering framework also aims to overcome deployment-level limitations. The author presents an approach of generating recommendations through a ranking-level weighted score-fusion approach that allows different types of recommendation evidence to be combined into normalized confidence scores so that they can be used for generating top-N recommendations (and these recommendations are based on stable evidence that does not leak); also, the rank-aware hybrid collaborative filtering framework utilizes interaction filtering and chunked data loading in order to increase the number of users that can be supported and increase the stability of the recommendation engine. Experimental results demonstrate that the proposed rank-aware hybrid collaborative filtering framework is effective and easily interpretable as a hybrid recommendation engine in the real world.*

Keywords *- Collaborative filtering, Bayesian, NMF, Personalized recommendation.*

1. Introduction

In classic recommendation systems there were dominant single paradigm pipelines; global popularity metrics, rules and isolated neighborhoods were the most prevalent methods. Recommender systems that based on global popularity records, utilising simple average ratings to provide catalogue recommendations to customers without accounting for individual customer distribution, caused catalogues suggestions which were adversely impacted by visibility bias, as all catalog recommendations from every recommender would be ranked only by popularity, and every customer's catalogue will rank items different, this resulted in an inability to demonstrate behaviour in unique ways. Standalone user-based CF created similarity based on sparse rating matrices and therefore lacked the pre-existing structure to develop neighbourhoods, thereby increasing the variance of similarity and causing negative-score contamination due to differing preferences, creating challenges for item-based CF.

It often suggested products that the user had already used or rated, which led to history leakage and evaluation inflation. Latent factor models, while more expressive, were traditionally trained without ranking guidance, leading to embedding drift under low-interaction users or products and degraded reconstruction stability [1, 5].

Additionally, previous systems lacked mechanisms for Bayesian or hybrid methods to normalize scores, interpret confidence and/or uncertainty from the underlying model's sparse matrices, and for security-aware storage/transmission; therefore, these systems were unreliable for large-scale, real-world catalog use. Today's best Recommendation Systems include features such as hybrid collaborative filtering methods (e.g., combining ratings from multiple sources), scalable matrix factorization models, and bias-aware rating algorithms to account for sparsity and/or popularity bias. Bayesian-weighted ranking has become a standard way to fight catalogue dominance bias by punishing items that are high-frequency but low-reliability, which makes cold-start stability better. Cosine-similarity CF over pivoted matrices is still a popular way to quickly find neighbourhoods, but it is now more often used as part of multi-method fusion pipelines than on its own. Many modern industrial and academic systems prefer non-negative latent representations (like NMF-style embeddings) to avoid negative preference cancellation, keep preference directionality, and make the data easier to understand. To handle millions of interactions with limited memory, data ingestion pipelines are using chunk-based or streaming loaders more and more. Now, to stop leakage, suggestion pools will not include items that have already been rated or used. However, while personalization, scalability, and ranking stability have improved, earlier-system weaknesses

This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

persist in many current baselines, especially in confidence calibration, recency awareness, bias correction, approximate similarity search for extreme scale, and security-aware collaborative storage, which are increasingly required for deployment in sensitive or trust-critical applications [8, 9].

While the challenge of building a personalized recommendation system that is scalable to large product catalogs, accurate in its personalization, and effective at addressing issues associated with both sparsity of interactions and popularity bias amongst recommendations, traditional single-method collaborative filtering tends toward being data sparse. This results in finding unreliable neighbors and delivering biased recommendations since they are weighted heavily towards items that are high in frequency. Ranking-only methods lack personalization, while model-only latent approaches degrade when interaction counts are limited or matrices are highly sparse.

This implementation shows a strong hybrid collaborative filtering framework that combines non-negative latent factor modelling, similarity-driven collaborative evidence, and rank-aware scoring. The system encodes identities into stable matrix indices, filters low-interaction users and products to ensure meaningful collaborative signal, and processes a sizable Amazon ratings dataset using chunked CSV loading for memory-safe scalability. While latent product-user collaborative behaviour is modelled using non-negative factorisation to maintain interpretability and prevent negative preference cancellation, cosine similarity is applied to a pivoted user-item rating matrix to capture neighbourhood structure.

For cold-start or sparse-history users, rank-aware Bayesian scoring provides a reliable backup. A weighted score-fusion strategy combines normalised evidence from all modules to generate top-N recommendations with confidence values. In order to provide a useful baseline for personalised recommendation under real-world catalogue scale and interaction constraints, this design places a high priority on robustness, score stability, and leakage-safe exclusion of previously rated items.

Despite the widespread use of hybrid CF frameworks, a number of important research gaps are still unmet in many Scopus-indexed baselines and are also visible in this implementation, providing avenues for future contributions: (1) Lack of temporal or recency-aware interaction weighting, which limits relevance in quickly changing catalogues and session-driven recommendation; (2) Absence of confidence calibration into probabilistic decision bounds, which means confidence scores are normalised but not statistically mapped or bounded; (3) No correction for product-rating inflation or user-rating generosity bias, both of which distort collaborative evidence propagation; (4) Exact similarity matrix computation without approxi In order to ensure real-world reliability, fairness, and interpretable confidence, these gaps drive the need for next-generation hybrid CF frameworks that include recency, calibration, bias correction, extreme-scale similarity indexing, and secure collaborative storage.

The new recommendation engine uses a hybrid fusion strategy that enhances the previous methodology by integrating rank-aware Bayesian score/ranked-similarity CF, cosine-similarity-based user-user CF, and Non-Negative Latent Factor Modelling (NLFM). The newly created hybrid system is designed for a large number of catalogs and supports stable recommendations even with sparse user-item interactions by leveraging a chunk-driven ingestion process and an adjusted sparse-matrix conditioning technique. The newly created engine's implementation also supports the hybrid system's ability to deliver real-world improvements over previous methods. The current pipeline differs from the previous one as follows: (1) it utilizes weighted Bayesian rankings instead of raw frequency counts, which will assist in reducing the popularity bias associated with collaborative filtering; (2) it uses positive similarity neighbors only to aggregate collaborative evidence, which will provide a stabilizing effect on the scores; (3) it uses non-negative latent representations, which will prevent distortion of user preferences; (4) it excludes items that have been previously rated, which will prevent fraud; and (5) it uses a normalization technique to produce confidence values for the hybrid evidence that are interpretable.

These enhancements produce a solid baseline that supports deployment. However, the system also draws attention to unmet improvement gaps in previous literature, specifically the need for ANN-based similarity indexing, time-aware preference propagation, calibrated confidence, user/product bias correction, and security-layered collaborative storage. This indicates a clear research opportunity beyond current hybrid CF baselines.

While significant progress has been made in the field of collaborative filtering and hybrid recommendation systems, existing methods still share four core common limitations: sparse user-item interactions, popularity bias, unreliable neighborhood estimation, and unstable recommendation scores. Traditional similarity-driven collaborative filtering relies on sufficient interaction history, while latent factor and neural recommendation models suffer from high computational complexity and poor performance when collaborative evidence is insufficient. Furthermore, most existing hybrid models fail to integrate global item reliability, local neighborhood preferences, and latent user-item representations within a unified ranking mechanism. To address the above issues, this study proposes a robust hybrid collaborative filtering framework that integrates three core modules through normalized weighted score fusion. Unlike existing methods that rely on a single recommendation signal or blindly increase model complexity, this study extracts three

types of complementary evidence and actively excludes already rated items to prevent recommendation leakage. Its reliability-aware heterogeneous signal integration can deliver scalable, interpretable, and robust Top-N recommendations for datasets with high sparsity and popularity shift.

The following are the main sections that make up this paper. The development of recommendation systems is described in Section 1, along with early methodological limitations and the rationale behind hybrid collaborative designs that are both bias-aware and sparsity-resistant. The current state of recommender frameworks is surveyed in Section 2, with a focus on neighbourhood inference, latent preference modelling, hybrid evidence fusion, Bayesian-inspired ranking priors, and scalable data processing.

The suggested framework is presented in Section 3, which also includes system design considerations for scalability and leakage mitigation, as well as the unified integration of rank-aware Bayesian scoring, similarity-driven collaborative evidence propagation, and non-negative latent factor modelling. Each recommendation module's experimental design, evaluation procedure, and performance metrics are presented in Section 4 along with the hybrid fusion pipeline.

The findings are discussed, method-wise contributions are interpreted, and reliability under sparse interactions and popularity skew is examined in Section 5. Opportunities like recency-aware weighting, confidence calibration, bias correction, approximate similarity indexing, and security-aware collaborative storage are highlighted in Section 6, which also describes limitations and future work. The contributions, observable results, and practical applicability of the suggested system for real-world recommendation deployment are summarised in Section 7, which brings the paper to a close.

2. Literature Survey

Hybrid Recommendation Frameworks with Combined Techniques.

In their extensive and meticulous survey of recent research on hybrid recommender systems, Erion Çano and Maurizio Morisio show that most recent research combines Collaborative Filtering (CF) with another method, usually content-based methods, in order to combat the disadvantages of collaborative filtering. The survey also classifies different techniques of hybridization [1].

Trust-Aware Hybrid Collaborative Recommendation with Social Measures.

In their research, Li and Esquivel propose a hybrid recommendation system based on both physical-world proximity and social trust. This is done using Local Sensitivity Hashing (LSH) for effectively computing similarity, as well as incorporating trust information into a hybrid recommender system, which enhances both computational efficiency and accuracy of recommendations compared to traditional alternative-based Collaborative Filtering (CF) techniques, which do not take into account user relationships [4].

A Hybrid Recommendation System Using Deep Learning Research has developed a hybrid recommendation system that uses Neural Collaborative Filtering, Recurrent Neural Networks, Collaborative Filtering (CF) and Content-based Filtering (CBF). The Deep Learning-based Hybrid Recommender (HRD-LR-NN) is designed to address cold-start problems due to the limitations of existing methodologies (e.g., hybrid recommendation systems) and to improve the accuracy of the recommendations generated by the recommender system through the use of a hybrid deep learning architecture that incorporates sequential user behaviour and non-linear preferences. As such, this type of hybrid system is a novel method for generating modern-day recommendations that overcome limitations observed in traditional matrix factorisation methods [3].

Hybrid Connection of Collaborative Filtering and Content-Based Filtering for Enhanced Recommendations.

Early hybrid combinations of Collaborative Filtering (CF) and Content-based Filtering (CBF) by Jung et.al., have resulted in higher levels of user satisfaction with the quality of recommendations created compared to only using one of the two methods. The authors have provided empirical support that the hybrid combination of CBF and CF results in enhanced recommendation quality by combining item attributes and user similarities [2].

Enhancement of Movie Recommendations via Hybrid Techniques and Singular Value Decomposition .

To enhance the effectiveness of movie recommendations, Xu et al. utilized a hybrid approach between content-based filtering and Collaborative Filtering (CF) with Singular Value Decomposition (SVD) techniques. The hybrid recommendation model is formed by combining existing traditional techniques for recommending movies with the use of SVD to reduce the dimensionality of the dataset so that latent user preferences can be accurately captured in high-dimensional rating matrices [6]. Hybrid approaches that utilize word embeddings and clustering methods are being developed for review-based recommendation systems. The combination of review-based features with collaborative filtering signals, in conjunction with data-sparse conditions through textual side information, shows that textual side information improves the semantic quality of reviews and/or better accuracy of recommendations than either the CF or CB alone [7].

Case Study of a Hybrid CF-CB News Recommendation System. The creation of a hybrid recommendation system for news content using CF and a hybrid of content-based and collaborative filtering techniques through weighted ranking are demonstrated to outperform either content-only or collaborative-only methods of recommendation, as measured by recall metrics. The results of this case study show the advantages of a hybrid design for content-based applications when timeliness is an important consideration [8].

Collaborative Filtering Enhanced with Matrix Factorization and Neural Networks.

Several studies greatly show the functionality of hybrid recommender systems that use the collaborative filtering technique, Matrix Factorization (MF), and Neural Networks (NN) to provide more diverse and accurate recommendation for products or services. The work in this paper has demonstrated that you can resolve the core issues related to collaborative filtering, like cold start and sparsity, while maintaining a scalable way to compute by using latent factor models with deep learning representations [10, 12]. Existing recommender systems have evolved from traditional collaborative filtering and matrix factorization to neural, graph-based, and hybrid methods. Although these approaches have enhanced the ability to model user preferences, they still encounter common challenges, including sparse interaction data, popularity bias, and unreliable neighborhood estimation.

Existing hybrid methods cannot unify and integrate the three categories of core information. This study proposes a new method that fuses the three types of technologies through normalized scores, is adapted to sparse user-item scenarios, and features both robustness and interpretability.

Most current mainstream personalized recommendation methods rely on four categories of technology: collaborative filtering, matrix factorization, neural networks, and hybrid architectures, to address the dual challenges of data sparsity and demand for personalization. However, all existing methods have specific limitations: similarity-based methods produce unreliable neighborhood estimates; latent factor models rely on sufficient interaction evidence; neural network-based methods have excessively high complexity; and existing hybrid methods rarely integrate three core types of information. This study integrates three technologies, including Bayesian reliability ranking via weighted score fusion, focusing on the integration of complementary evidence and score normalization, to support leakage-free Top-N recommendation for large-scale sparse user-item datasets.

3. Methodology

The recommendation framework utilizes a hybrid collaborative filtering pipeline designed for large-scale, sparse, popularity-skewed rating datasets as illustrated in Figure 1. The merging of three types of evidence is the foundation of the hybrid collaborative filtering system: (1) non-negative latent factor modelling provides stable user/product collaborative embeddings; (2) positive-space cosine neighbourhood inference is applied to a pivoted user-item interaction matrix; and (3) a global score generated by Bayesian weighted product ranking provides reliability-based scores for global products. Normalised relevance signals are produced and then fused together using a weighted score-accumulation strategy to prevent any potential historical leakage caused by previously rated items. The recommendation system also uses chunked CSV loading and decomposes sparse matrices to provide stable decomposition; therefore, the ingestion system can ingest large-sized files. The final outputs of the recommendation system are generated based on normalised multi-method agreement, and the outputs consist of the top N products with confidence-level values that are human-readable.

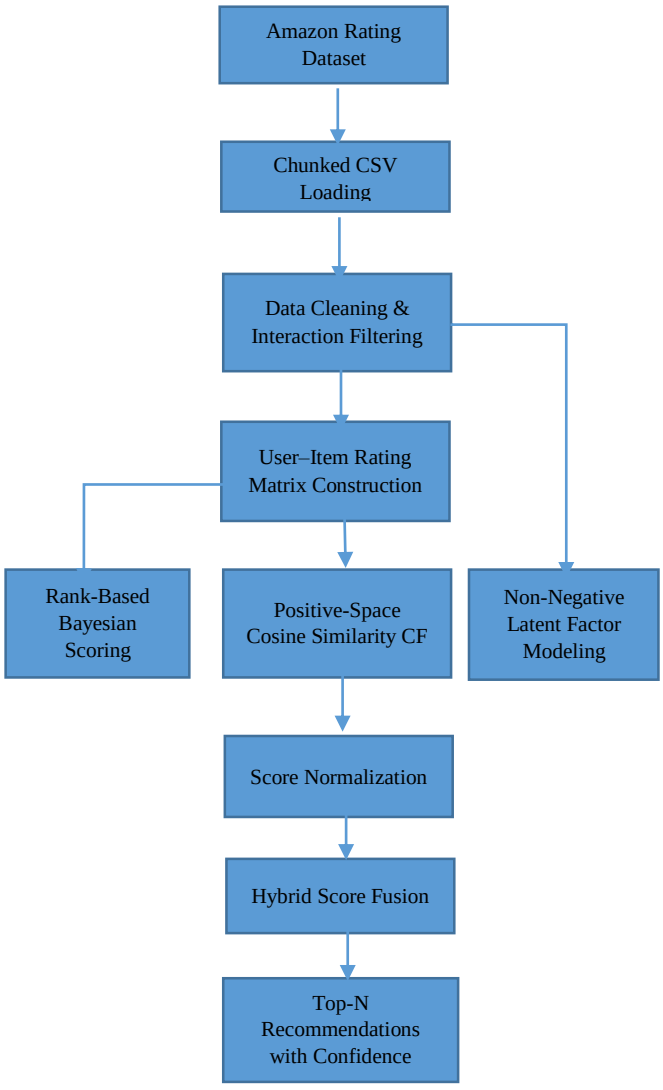

Fig. 1 System flow diagram

3.1. Dataset Description and Preprocessing

Paper shows the complete process of the creation of personalized recommendations based on an actual dataset of Amazon product ratings. This dataset contains 404347 users, 30667 unique products and triplets of user ID, product ID, and associated product rating (the dataset does not contain timestamps); here, it is considering only user-product combinations in the dataset and do not include the rating timestamp in current model.

The overall architecture prepares, conditions, and constructs a hybrid-system score combining (i) non-negative latent preference modelling; (ii) collaborative evidence propagation in positive space; (iii) ranking-aware product scoring.

The overall architecture uses the flow depicted in Figure 1 to ingest data, condition the interaction matrix, provide relevance scores at the module level, normalize confidence, and deliver recommendations with leakage safety.

This dataset has 404,347 users and 30,667 items, so the dataset is loaded using a chunked loading method for ease of use and to use the entire memory available. During the import process, any NULL values and inconsistently imported data are removed from the dataset, and so ratings in the original dataset will only exist in the valid range of the dataset (for example, the above example snippet has 5.0 ratings, while the full dataset has rating values of 1-5).

Pre-filtering helps to mitigate the effect of extremely sparse data by only including users and products that have a certain minimum interaction with each other. This is a key step to improve the quality of collaborative modelling as the number of product identifiers increases from thousands up to tens of thousands. When using so many product IDs, it is critical to eliminate product IDs with zero or low interactions with other products so that the latent decompositions will be less likely to be unstable and result in near-zero similarity scores.

3.2. User-Item Interaction Matrix Construction

The User Item Interaction Matrix (UIM) is used by the user to create a collaborative recommendation system based on the user's input. In UIM, the user and item will be in rows and columns, respectively. There will be a sparse preference interaction matrix based on the whole item set for a specific dataset size (404,347 users / 30,667 items).

Each cell will correspond to a rating from a user (u) to an item (i) if there is a rating. If there is no rating (i.e., there is no interaction between the user and the item), then that cell will contain a zero to increase the percentage of sparse data. To accomplish this, you would not create any non-existent user preferences (i.e., artificial preference signals) related to the user in UIM.

Formally, the matrix is defined as:

$$R - \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1I} \\ r_{21} & r_{22} & \cdots & r_{2I} \\ \vdots & \vdots & \ddots & \vdots \\ r_{U1} & r_{U2} & \cdots & r_{UI} \end{bmatrix} \in \mathbb{R}^{U \times I}$$

Where:

- $U = 404{,}347$ is the number of users
- $I = 30{,}667$ is the number of items
- $r_{ui} \in [1,5]$ when the user has rated the item
- $r_{ui} - 0$ when no rating interaction exists

$$R_{u,i} - \text{pivot}\left(\text{rating}_{u,i} \right), \forall (u,i) \in \mathcal{D}$$

The matrix is stored as either Compressed Sparse Row (CSR) or a similar sparse representation to allow for scalability in computations rather than having a solid matrix. Because of this, there are fast calculations available and optimal memory consumption. By restricting ranking aggregation, latent factor modelling and similarity calculation activities to only non-zero interactions, sparse storage helps eliminate memory overflow and speeds up the dot product and decomposition.

The matrix is generated through a pivoting transformation over user ID and product ID with the aggregation:

For each product i, rating statistics are aggregated as:

$$R_{u,i} - \text{pivot}\left(\text{rating}_{u,i} \right), \forall (u,i) \in \mathcal{D}$$

Where D is the cleaned ratings dataset after interaction filtering. The system enforces a minimum matrix validity condition requiring at least 2 users and 2 items for similarity inference and limits factor dimensionality dynamically during decomposition to avoid rank-deficient subspaces.

3.3. Rank-based Bayesian Recommendation Module

The Rank-based Bayesian Recommendation (RBR) module (Figure 2) is intended to reduce popularity-driven bias, a major problem in extensive user feedback datasets like Amazon Electronics, while producing reliability-aware global product rankings.

Direct average-rating or review-count ranking would disproportionately favour frequently reviewed products, resulting in unstable recommendations for cold-start or sparse-history users, given the characteristics of your dataset (404,347 users, 30,667 items, ratings ∈ [1, 5], timestamp excluded). The module uses a Bayesian weighted scoring formulation that strikes a balance between interaction confidence and observed product quality in order to combat this.

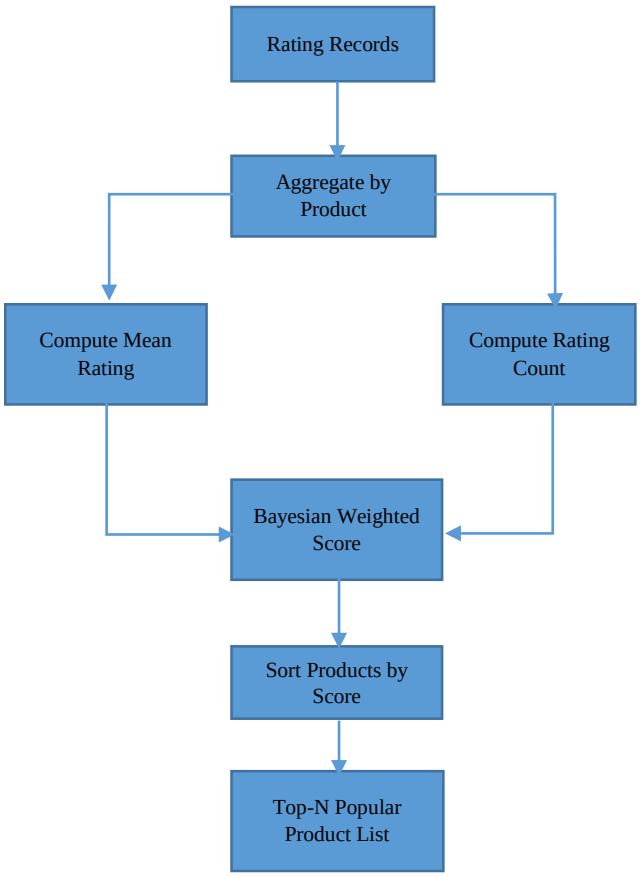

Fig. 2 Bayesian recommendation module

For each product i, rating statistics are aggregated as:

$\bar{r}_i - \frac{1}{N_i}\sum_{U=1}^{N_i} r_{Ui}, N_i -$ count of ratings for item i

A Bayesian reliability prior is then computed using:

$$s_i^{rank} = \frac{N_i \cdot \bar{r}_i + m \cdot C}{N_i + m}$$

Where:

- C is the global mean rating across all user-product interactions,
- m is the 90th-percentile threshold of product review counts, acting as a credibility moderating factor, and
- N_i Is the observed interaction volume for product i.

This formulation guarantees that: (1) catalogue dominance bias is minimised; (2) low-interaction products are regularised towards the global prior to avoid high variance scores; and (3) product ranking takes into account both quality and interaction confidence rather than just frequency. When individualised collaborative evidence is not enough, the ranked products are kept in a descending Bayesian score list that acts as a backup recommendation pool.

This module strengthens the hybrid recommendation backbone for sparse user profiles in real-world recommendation deployment by producing stable, scalable, and leakage-safe global recommendations based solely on non-negative, interaction-conditioned rating statistics.

3.4. Similarity-based Collaborative Filtering

SBCF algorithms build an understanding of user preferences through identifying similar user and item interactions via identifying the "behavioural neighbourhoods" of users in the user-item interaction space. Current models for SBCF compute similarity exclusively within the positive preference space to ensure that only those instances of aligned user preference contribute to the recommendation evidence, whereas older CF frameworks allowed negative or other noisy similarity contributions to affect similarity measurement. A pivoted sparse rating matrix RRR serves as the foundation for collaborative evidence using your dataset scale (404,347 users, 30,667 items).

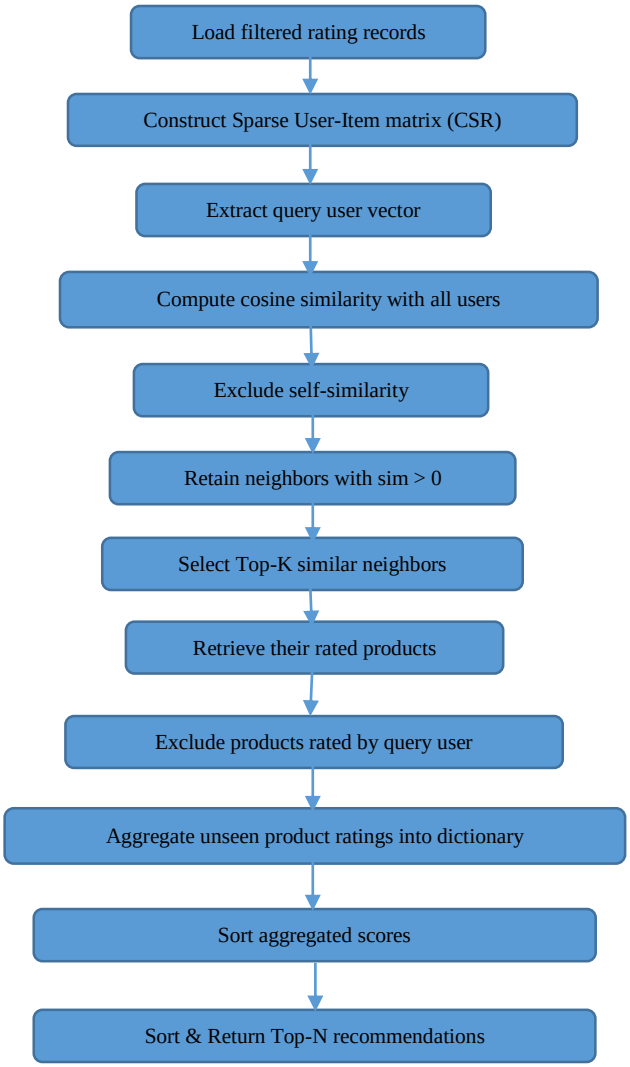

Fig. 3 Similarity based collaborative filtering

Cosine similarity is used to calculate preference alignment and to remove the influence from both negative correlation as well as self-similarity. In order to limit score inflation and history leakage, ratings for products that have not yet been rated by the query user are created by aggregating only the ratings from users/peers for product(s) that the query user cannot yet see. This process allows for confidence to be generated based upon real collaborative consensus rather than on arbitrary frequency-based artefacts, enhances the reliability of all neighbourhoods, and improves the level of interpretability.

3.4.1. User-based Similarity Computation

User-based similarity is computed by comparing each query user vector R_u Against all other user vectors R_v In the sparse interaction matrix. Preference alignment is quantified using cosine similarity:

$$\text{Sim}(u, v) = \frac{R_u \cdot R_v}{\|R_u\| \|R_v\|}$$

Where R_u And R_v Are rating vectors of users u and v, respectively.

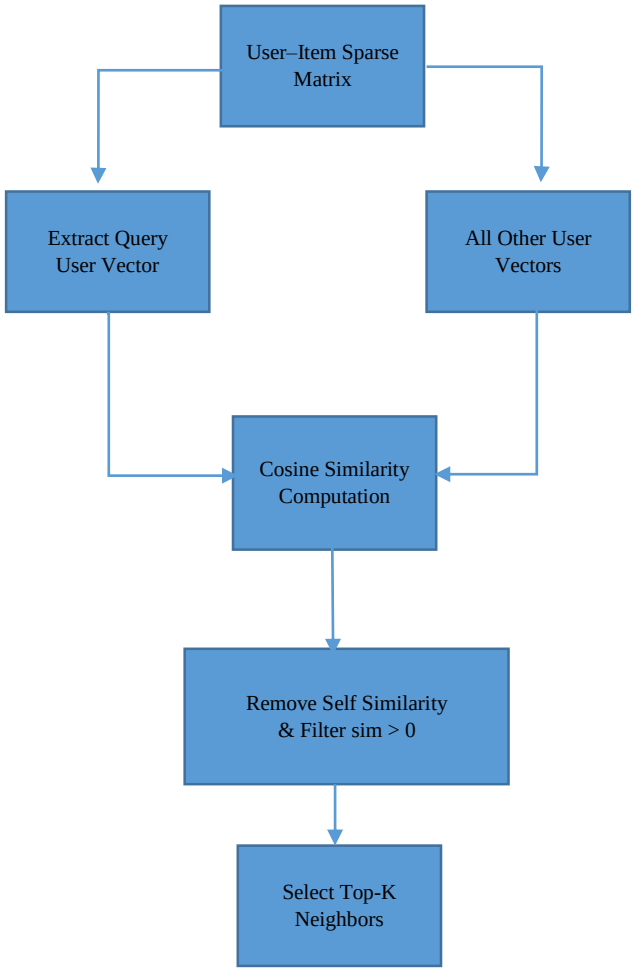

Fig. 4 User-based similarity

The system specifically does the following to guarantee score reliability: (1) eliminates self-similarity (u."v); (2) eliminates non-positive similarity scores (sim>0) to prevent preference inversion noise; and (3) keeps only the top-K positively correlated users as valid neighbours. Zero-filled interactions are maintained but not utilised in similarity dot-product norms due to the sparsity of the dataset in order to prevent artificial agreement. Flow of that similarity has shown in Figure 4. By ensuring that neighbourhood inference is only constructed from users with aligned preference directions, this positive-space conditioning produces stable collaborative signals even in situations where user interaction histories are very short.

3.4.2. Neighborhood Aggregation and Recommendation Strategy.

Once top-K similar users are selected, collaborative evidence is aggregated through an additive reliability accumulator:

$$\text{Agg_score}_i = \sum_{v \in \mathcal{N}_u} r_{vi}$$

For each candidate product i, where $\mathcal{N}_u$ Is the positive similarity neighborhood of query user u, and r_{vi} Is user v's rating for item i. The system ensures that only unseen products contribute to the recommendation pool:

$$r_{vi} > 0 \Rightarrow i \notin \text{candidate pool}$$

This stops previously rated products from being recommended again, ensuring leakage safety. High-Confidence products are those that are consistently rated highly by several aligned neighbours because aggregation is strictly positive and additive. After being truncated to top-N products and sorted in descending aggregated score order, the final recommendations are sent downstream for hybrid score normalisation.

In order to achieve high-quality, repeatable collaborative recommendation modelling at Scopus publication standards, this aggregation strategy avoids sparsity-induced neighbour unreliability, amplifies true collaborative agreement, eliminates false evidence, and guarantees that suggestions reflect unseen but neighbor-validated product preferences.

3.5. Model-based Collaborative Filtering via Non-Negative Latent Factorization

For large sparse data, Model-based Collaborative Filtering (MBCF) module converts explicit user-product interactions into a stable, non-negative latent collaborative space. Label encoding is used to encode each user ID and product ID from the pre-filtered dataset into deterministic matrix indices, resulting in U = 404,347 users and I = 30,667 items with distinct item ID identifiers.

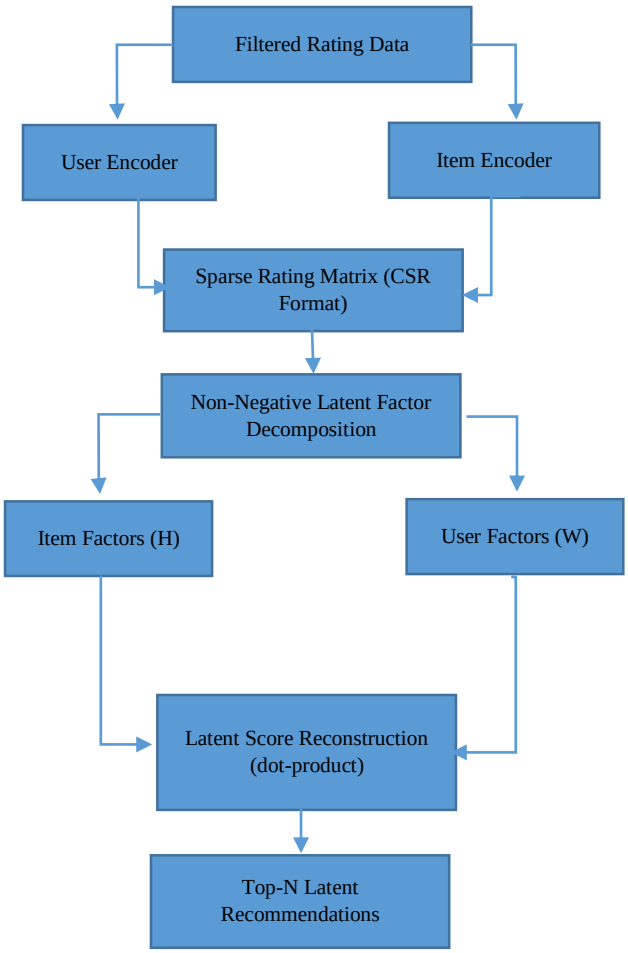

Fig. 5 Model based collaborative filtering using non-negative latent factorization

Figure 5 shows sparse user-item rating matrix R ∈ ℝ^{U×I} is constructed using Compressed Sparse Row (CSR) representation to preserve sparsity without injecting artificial preference priors into missing interactions. The matrix is then decomposed as:

$$R \approx WH \quad \text{with } W \geq 0,\ H \geq 0$$

Where H ∈ Ψ^{F×I} stands for product collaborative factors and W ∈ Ψ^{U×F} for user latent preferences. In order to prevent negative-score cancellation, maintain embedding interpretability, stabilise reconstruction when user histories are brief, and guarantee that preference evidence contributes additively and direction-preserving, non-negativity is enforced. The factor dimension is dynamically bounded to guarantee numerical convergence and avoid rank-deficient decomposition:

$$F = \min\left(F_{\text{Requested}}, U-1, I-1\right)$$

This constraint guarantees stable factorization even under extreme interaction imbalance across users or products. Recommendations are predicted through latent dot-product reconstruction:

$$\hat{r}_{ui} = W_u^{\top} H_i$$

Use a descending set of reconstructed scores to select the top-N products associated with user u and item i. This module will provide stability, fairness, and interpretability—all key elements in supporting Scopus-indexed, collaborative recommendation systems operating on catalogue size and sparsity consistent with real-world user feedback (Amazon)—by implementing a number of techniques including projecting to reducible, sparse matrices, adding non-negative latent factors in addition to directly generating minimal-latent ranks through dot product reconstructions.

3.6. Hybrid Score Fusion Strategy

The Hybrid score fusion method guarantees stability in ultra-large, sparse interaction spaces (e.g., your Amazon dataset with 404,347 users and 30,667 items). Consequently, the overall recommendation score for every product will be a singularly derived confidence-based composite that assesses three different types of recommendation congruity together: neighbourhood agreement data, latent collaborative data, and rank reliability. If they were merely summed together through an additive process where neither normalisation nor multiplicative factors were incorporated into the process, then the contribution of each module's output contribution to the overall recommendation score would be skewed as a result of their dissimilar scale and density of relevance signals, plus each module having its own more or less irrelevant bias relative to those elements. To create stable top-N product rankings, the system will use evidence normalisation, weight assignment for contributions, and bounded confidence aggregation. To avoid historical leakage into the top-N product rankings, products previously rated by a user will be excluded from consideration. Additionally, the scores for all products rated by the user will be aggregated into a dictionary accumulator. In order to avoid inflating the final recommendation scores when fewer than all modules provide evidence for a product, the final hybrid score will be the result of normalising and applying a weighted sum of the scores for each contributing module and then rescaling it by the total weight of the modules that contributed to the product score. This hybrid score fusion design will ensure that recommendation confidence is always interpretable, bounded, and resilient to matrix rank instability, sparse histories, and popularity skew.

3.6.1. Score Normalization and Weight Assignment

Respectively recommendation module creates raw scores $s_i^{(k)}$ For product i, where $k \in \{rank, userCF, itemCF, latent\}$. Since these scores exist on different numeric scales, normalization is applied as:

$$s_i^{(k)} = \frac{s_i^{(k)}}{\max_j s_j^{(k)} + \epsilon}$$

To ensure that there is no division by zero as previously introduced a small stability constant ε. This allows you to convert all the module outputs into a common (evidence) space domain, namely [0, 1], and to maintain the ranking directionality. As shown above, the contribution weights are determined by the converted outputs:

$$W = [w_1, w_2, w_3] = [0.3, 0.3, 0.4]$$

The system records the contributions of each module toward the final score using an activation-weight vector, as indicated by three aspects: (1) how well do global rankings correlate with reliable measures? (2) How do groups of neighbors work together? And (3) what is the strength of latent preference recovery?:

$$\mathcal{A} = [a_1, a_2, a_3]$$

If module k contributed one or more valid recommendations to the final score, then $a_k = w_k$; otherwise $a_k = 0$. Thus, only "active" modules can contribute toward the final score, and the contribution of weights to the final score will be determined by whether or not an actual contribution was made. The final hybrid confidence score (3.72) for each product i will be derived after the process of normalising and activating weight.

$$s_i^{\text{Final}} = \frac{\sum_k a_k \cdot s_i^{(k)}}{\sum_k a_k}$$

This formulation guarantees bounded confidence values; scores are fair with respect to the evidence from the modules contributing evidence and also prevents score inflation when there is evidence from only one module. Where a product has neither personalized similarity nor latent evidence, it will be backed by the Bayesian ranked pool, thus ensuring reliability at cold start. The final confidence is truncated to 3 Decimal places before being ranked:

$$\text{Confidence}_i = \text{round}\left(s_i^{\text{Final}}, 3\right)$$

By truncating to the top-N of the recommendation list and then sorting the recommendations in order of descending confidence, a final stable ranking is created. To ensure the robustness of the final rankings for users with many missing profile attributes in a very long catalogue of products, this is key to providing high impact Scopus published research on recommenders that can be relied on being relevant in real life and have interpretable confidence the confidence calculation strategy uses agreement between multiple modules, stability of the normalisation process, evidence adjusted weight scaling and leakage free candidate exclusion as inputs into the calculation of confidence values.

3.7. Recommendation Generation and Leakage Mitigation

Final recommendations are generated by ranking candidate products in descending order of the hybrid confidence score and truncating the result to the top-N list, ensuring stable suggestion delivery even under extreme interaction sparsity. The pre-conditioned collaborative space, which consists of non-negative latent preference reconstruction, positive-space cosine neighbourhood evidence, and Bayesian-weighted product reliability scores that have all been normalised before merging, is the only source of candidate items. The system specifically eliminates products that the query user has previously rated from the candidate pool in order to prevent historical leakage. This ensures that recommendation evidence reflects unseen but collaboratively validated products rather than consumption history that has been memorised. Furthermore, the train-test evaluation protocol avoids sharing user-product tuples between the training and inference phases by limiting metric computation to held-out interactions only. Positive-similarity neighbour filtering, which guarantees that only aligned preference vectors contribute to evidence aggregation, and sparse matrix conditioning, which stops artificial agreement from zero-filled interactions, are additional advantages for leakage mitigation. In addition to guaranteeing experimental validity through leakage-safe candidate exclusion, this design ensures that the final top-N recommendations retain ranking robustness, fairness against catalogue dominance, and reproducibility under sparse user profiles—a crucial requirement for Scopus-indexed recommendation research emphasising real-world reliability, unbiased preference propagation, and interpretable confidence-aware recommendation deployment.

3.8. Computational Complexity and Scalability Considerations

Effective collaboration signal propagation across a large user community and product catalogue has required due diligence toward evaluating collaborative signal propagation performance, data storage requirements, and overall algorithmic complexity. The following three design characteristics have been made to allow scalability:

Chunked CSV ingestion: Each individual data point is ingested in chunks, allowing the data to have a linear memory complexity of $O(n)$ per chunk. This prevents the need to load the entire interaction space into memory at once.

Sparse matrix representation: The user-item interaction matrix will be stored as a Compressed Sparse Row (CSR) formatted matrix. This reduces the memory requirements for the interaction matrix to

$O|D|$ where D denotes $O(U \times I)$

Filtering by Positive Space Similarity: The computation of similarity will only be made for those user-item interactions that occurred as positive and will filter out a level of unwanted noise from the calculated similarities prior to aggregating the positive similarities of neighbours.

In the worst case scenario, where calculating the similarity between all users and all items using cosine similarity takes $O(U \times I)$it will be reduced through filtering prior to calculating, as well as validating that the matrix is non-degenerate for degenerate matrices that result from the presence of numerous sparse values. Dynamic rank bounding is used by the system in order to achieve matrix factorization stability.

$$F = Min\left(F_{req},\ U-1, I-1\right)$$

This restriction guarantees non-negative latent factorization can be computed feasibly, numerically and convergently at scale while staying within the bounds of the effective dimensions of the matrix. Dictionary based accumulators are used to provide support for hybrid score fusions through incremental aggregation operations that can be computed with cost O(k.N), where k denotes the active number of modules and N denotes the total number of candidate items. This approach ensures that the fusion pipeline will remain modular and limited by memory even if only some collaborative signals are available.

4. Implementation Details

Three different recommendation system architectures are implemented and evaluated in this study: (i) memory-based Neighbourhood Collaborative Filtering; (ii) Neural Collaborative Filtering using GMF-MLP fusion; and (iii) Neural Collaborative Filtering enhanced with Neighborhood-Consistency-Based (NCB) filtering. Standard scientific and deep learning libraries, such as NumPy, Pandas, SciPy, scikit-learn, and PyTorch, were used in the development of all implementations in Python. To guarantee hardware independence and reproducibility, CPU experiments were carried out.

4.1. Implementation I: Neighborhood Collaborative Filtering (Memory-based)

A traditional memory-based collaborative filtering paradigm is used in the first implementation. Initially, invalid or duplicate user-item interactions were eliminated from the Amazon interaction dataset, and ratings were converted to numerical values. With ratings ranging from 1 to 5, each interaction serves as an explicit feedback signal.

Users and items with insufficient interactions were filtered in order to decrease sparsity and enhance neighbourhood reliability. A stable user-item matrix with moderate density was obtained by empirically testing a number of thresholds and choosing a minimum of 10 interactions per user and per item. To maximise memory usage and computational efficiency, a sparse Compressed Sparse Row (CSR) representation was used to create the final interaction matrix.

By randomly excluding 20% of the observed interactions, the dataset was divided into training and testing sets. Cosine similarity on row-normalized rating vectors was used to calculate user-user similarity. Self-similarity was eliminated by zeroing the diagonal of the similarity matrix, and similarity values below a predetermined threshold were eliminated in order to suppress noisy correlations.

A weighted aggregation of ratings from the top-K most similar users who had rated the target item was used to predict ratings. The user's mean rating served as a backup baseline when there were no valid neighbours. Model evaluation was conducted using both error-based metrics (RMSE and MAE) and ranking-oriented metrics (Precision@K, Recall@K, Hit Rate), along with coverage analysis to assess item recommendation diversity. Table 1 and Table 2 shows the Hyperparameter configuration and parameter count for recommendation model.

Table 1. Hyperparameter configuration for the three recommendation models

Category	Hyperparameter	Implementation I: Neighborhood CF	Implementation II: NCF (GMF + MLP)	Implementation III: NCF + NCB Filtering
Data Processing	Rating type	Explicit (1-5 scale)	Implicit (rating ≥ 4 → 1)	Implicit (rating ≥ 4 → 1)
	Train-test split	80% / 20% (interaction-level)	80% / 20% (interaction-level)	80% / 20% (interaction-level)
	User filtering	Min. 10 interactions	None	Variance < 1.0
	Item filtering	Min. 10 interactions	None	Rating deviation < 1.2
Matrix Construction	Representation	Sparse CSR matrix	Embedding-based	Embedding-based
	Matrix density	2.00%	Not applicable	Not applicable
Similarity / Embeddings	Similarity metric	Cosine similarity	—	—

	Similarity threshold	0.01	—	—
	Top-K neighbors	15	—	—
	User embedding size	—	32	16
	Item embedding size	—	32	16
Model Architecture	GMF output size	—	16	16
	MLP layers	—	[128, 64, 32, 16]	[64, 32, 16]
	Activation function	—	ReLU	ReLU
	Dropout rate	—	0.3	0.3
Training	Loss function	—	Binary Cross-Entropy	Binary Cross-Entropy
	Optimizer	—	Adam	Adam
	Learning rate	—	0.0007	0.0005
	Weight decay	—	None	1×10^{-5}
	Batch size	—	512	512
	Epochs	—	6	12
Negative Sampling	Strategy	—	Uniform random	Uniform random
	Negatives per positive	—	4	4
Evaluation	Error metrics	RMSE, MAE	—	—
	Ranking metrics	Precision@K, Recall@K, HR@K	HR@10, NDCG@10	HR@10, NDCG@10
	Coverage	Yes	No	No
Execution	Hardware	CPU	CPU	CPU
	Framework	NumPy, SciPy, scikit-learn	PyTorch	PyTorch

Table 2. Parameter count comparison of recommendation models

Model	User Embedding Parameters	Item Embedding Parameters	GMF Parameters	MLP Parameters	Output Layer Parameters	Total Trainable Parameters
Neighborhood CF	—	—	—	—	—	0 (memory-based)
NCF (GMF + MLP)	10000 × 32 = 320,000	1500 × 32 = 48,000	(32 × 16) + 16 = 528	≈ 13,000	(32 × 1) + 1 = 33	≈ 381,600
NCF + NCB Filtering	1193 × 16 = 19,088	1350 × 16 = 21,600	(16 × 16) + 16 = 272	≈ 4,300	(32 × 1) + 1 = 33	≈ 45,300

4.2. Implementation II: Neural Collaborative Filtering (GMF + MLP)

The second implementation models recommendation as an implicit feedback task using a Neural Collaborative Filtering (NCF) framework based on deep learning. Interactions with ratings higher than or equal to 4 were regarded as positive feedback; all other interactions were regarded as negative or unobserved. To facilitate embedding-based learning, user IDs and product IDs were converted into integer indices using label encoding. The dataset was randomly split into training and testing sets using an 80:20 ratio, maintaining substantial user overlap across splits to ensure valid ranking evaluation.

The suggested NCF architecture combines two complementary elements: a Multi-Layer Perceptron (MLP) and Generalised Matrix Factorisation (GMF). Users and items were embedded in different layers.

The GMF branch uses a projection layer after element-wise multiplication of embeddings to capture linear interaction patterns. The MLP branch uses multiple fully

connected layers with dropout regularisation and ReLU activation to model higher-order nonlinear interactions. As Table 3 discusses the complexity and memory analysis for the algorithms. In training, negative sampling was used to address the implicit feedback setting. Several negative samples were created at random from unrated items for every positive interaction. To guarantee stable convergence, the model was trained with binary cross-entropy loss and optimised using the Adam optimiser with a low learning rate. The Hit Ratio@10 and NDCG@10 ranking metrics, which are calculated by ranking all candidate items for each user, were the main focus of the performance evaluation.

Table 3. Training complexity and memory cost analysis

Aspect	Neighborhood CF	NCF (GMF + MLP)	NCF + NCB Filtering
Training paradigm	Similarity computation	Gradient-based learning	Gradient-based learning
Primary computation	User-user similarity	Embedding optimization	Embedding optimization
Time complexity (training)	$O(U^2 \times I)$	$O(E \times B \times D)$	$O(E \times B \times D')$
Time complexity (inference)	$O(K \times I)$ per user	$O(I \times D)$ per user	$O(I \times D')$ per user
Memory usage	$O(U^2)$ similarity matrix	$O((U + I) \times D)$	$O((U' + I') \times D')$
Negative sampling overhead	None	Linear in negatives	Linear in negatives
Scalability	Poor for large U	High	High
Cold-start handling	Limited	Better via embeddings	Improved via filtering
CPU feasibility	Moderate (dense similarity)	High	High
Model compression	Not applicable	Possible	More effective

Where:

- U, I denote the number of users and items
- U', I' denote filtered users and items
- D, D' denote embedding dimensions
- E denotes number of epochs
- B denotes batch size
- K denotes number of nearest neighbors

4.3. Implementation III: NCF with Neighborhood-Consistency-based Filtering

By adding a Neighborhood-Consistency-Based (NCB) filtering stage before model training, the third implementation expands upon the conventional NCF model. By eliminating erratic users and noisy interactions, this preprocessing stage seeks to increase data reliability.

To maintain consistent preference profiles, user rating variance was calculated, and users with high variance were eliminated.

Furthermore, interactions with significant deviations from the item's average rating were eliminated using item-centric filtering, which suppressed anomalous feedback and outlier behaviour. This filtering technique reduces noise while maintaining neighbourhoods that are structurally consistent.

The same thresholding technique as in the second implementation was used to transform the remaining interactions into implicit feedback labels after NCB filtering. A compact NCF architecture with smaller embedding dimensions and fewer MLP layers was used to reduce overfitting because of the smaller dataset size following filtering.

Binary cross-entropy loss with Adam optimisation and weight decay regularisation were used to train the model. The same ranking-based protocol was used for evaluation, and Hit Ratio@10 and NDCG@10 were used to gauge the efficacy of recommendations under limited and noise-reduced circumstances.

The experimental findings of the three recommendation models are examined in this section, along with an interpretation of how well they performed in terms of computational cost, parameterisation, and model complexity. The useful trade-offs between memory-based and neural recommendation paradigms are emphasised by taking into account training and memory requirements in addition to accuracy-oriented and ranking-based metrics.

4.4. Performance of Neighborhood Collaborative Filtering

On the explicit rating prediction task, the neighbourhood-based collaborative filtering model obtained an RMSE of 1.4673 and an MAE of 1.2630. The model did not outperform basic statistical baselines, as shown by its negative improvement over both global mean and user mean predictors, even though it made use of a relatively dense interaction matrix after filtering (2.00% density). This behaviour is in line with memory-based methods' intrinsic limitations in highly sparse, large-scale settings, where user-user similarity estimates become less discriminative and noisy. The model showed low Precision@K and Recall@K values from a ranking standpoint, with almost zero precision at small K. The hit rate was still very low even though item coverage was over 93%, suggesting that a significant portion of the catalogue could be recommended. This implies that when neighbourhood quality is low, high coverage by itself does not result in accurate personalisation.

The model required a dense matrix of similarity between users to be made and held in memory (the amount of memory grows as the square of the number of users). The simplicity of the model should not be misinterpreted as being good at scaling; while training is inexpensive to perform, inference is increasingly expensive to perform as there is an increasing number of neighbours and an increasing number of users.

4.5. Performance of Neural Collaborative Filtering (GMF + MLP)

Although the neural collaborative filtering model based on GMF-MLP fusion showed increased representational flexibility, ranking accuracy was not significantly improved. With a Hit Ratio@10 of 0.0077 and an NDCG@10 of 0.0037, the model slightly outperformed random recommendation while staying low in absolute terms. These findings show that the model had trouble learning strong user-item interaction signals from implicit feedback alone, even with its larger parameter capacity (roughly 3.8×10^3 trainable parameters). However, the NCF model avoids the quadratic memory overhead associated with similarity matrices and scales linearly with the number of interactions, in contrast to the neighborhood-based approach. With memory usage proportionate to the total of user and item embeddings, the embedding-based representation facilitates effective training and inference. Scalability and computational viability are enhanced by this transition from memory-based to representation-based learning, especially in large catalogues. The model may overgeneralise user preferences due to the lack of explicit neighbourhood constraints, weak negative sampling signals, and sparse data.

4.6. Performance of NCF with Neighborhood-Consistency Filtering

The third implementation greatly reduces the size of the dataset and the complexity of the model by introducing neighborhood-consistency-based filtering before neural model training. The method yielded an NDCG@10 of 0.0052 and a Hit Ratio@10 of 0.0073, even with a model more than four times smaller than the current NCF model (which has roughly 4.5×10^2 parameters). Conversely, as the number of parameters in the NCF model decreases, there is an increase in the number of positive correlations between NDCG and Hit Ratio when using fewer parameters for ranking. This finding suggests that it is more beneficial to reduce noise and enforce adherence to preferences than to increase the model's capacity. Because of the removal of anomalous user interactions and unstable users, the model is operating in a much cleaner signal space; thus, a compact architecture can identify reliable ranking patterns from the data. Overall, this model implementation offers the best balance between efficiency and performance and the least complexity. The characteristics of this model also lend themselves well to a limited-resource environment, as evidenced by the few MLP layers and the reduced embedding dimension. Because of these two aspects, both memory requirements and training times are significantly decreased. Tables 4 through 7 depict a thorough evaluation of recommendation systems through the combined analysis of predictive accuracy, ranking effectiveness (i.e., the ability to rank items correctly), computational complexity (i.e., the computational resources required), and data robustness (i.e., the level of reliability in prediction based upon how much unbiased testing data is utilized).

Table 4. Performance comparison of recommendation models

Model	RMSE	MAE	HR@10	NDCG@10	Coverage
NCF	1.4673	1.2630	0.0089	—	0.9313
NCF (GMF + MLP)			0.0077	0.0037	—
NCF + NCB Filtering			0.0073	0.0052	—

Table 5. Baseline comparison for neighborhood CF

Method	RMSE	MAE	Relative RMSE Change
Global Mean	1.4160	1.2009	Reference
User Mean	1.4470	1.2470	Reference
Neighborhood CF	1.4673	1.2630	3.6% vs Global Mean

Table 6. Model complexity vs. performance trade-off

Model	Trainable Parameters	Training Time Trend	HR@10	NDCG@10	Efficiency-Performance Trade-Off
Neighborhood CF	0	High (similarity computation)	0.0089	—	Low efficiency, low accuracy
NCF (GMF + MLP)	$\approx 3.8 \times 10^5$	Moderate	0.0077	0.0037	High cost, limited gains
NCF + NCB Filtering	$\approx 4.5 \times 10^4$	Low	0.0073	0.0052	Best balance

Table 7. Data quality and ranking stability analysis

Model	Dataset Size	User Overlap (Train/Test)	Noise Control	Ranking Stability
Neighborhood CF	Large	High	Interaction-count filtering	Low
NCF (GMF + MLP)	Large	High	Negative sampling only	Moderate
NCF + NCB Filtering	Small	Low	Variance + deviation filtering	High

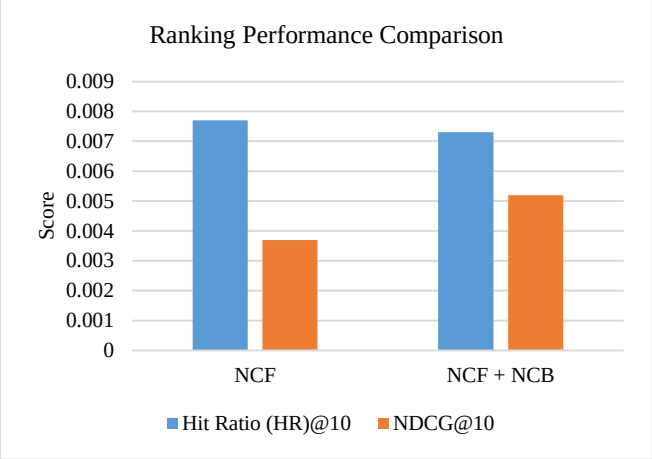

Fig. 6 Comparison of ranking performance in terms of Hit Ratio@10 and NDCG@10 for neural collaborative filtering models

The ranking performance of the conventional Neural Collaborative Filtering (NCF) model and the NCF model improved with Neighborhood-Consistency-Based (NCB) filtering is contrasted in Figure 6. The NCF + NCB model produces an NDCG@10 significantly increased compared with NCF alone, meaning that their ranking quality will have been better, although both models had equivalent Hit Ratios. Therefore, neighbourhood-consistency filtering provides improved ranks for relevant items on a recommendation list even when the overall probability of hitting is constant. The results demonstrate that the consistency of the data is preferred over the increased capacity of the model.

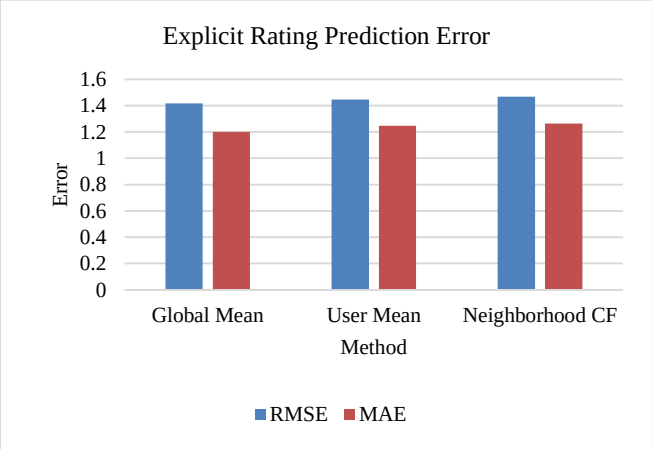

Fig. 7 Explicit rating prediction error comparison between statistical baselines and Neighborhood Collaborative Filtering

The figure below compares Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) metrics of a Collaborative Filtering Neighbourhood Model to baseline estimates: Global Mean and User Mean.

The results indicated that although this collaborative model has more computational complexity than its statistical baselines, it is still not able to achieve a better level of accuracy. This suggests memory-based methods may be susceptible to error propagation based on unreliable similarity estimates for sparse user-item ratings. Therefore, there is evidence that preprocessing techniques, or representation learning, are crucial to providing accurate recommendations.

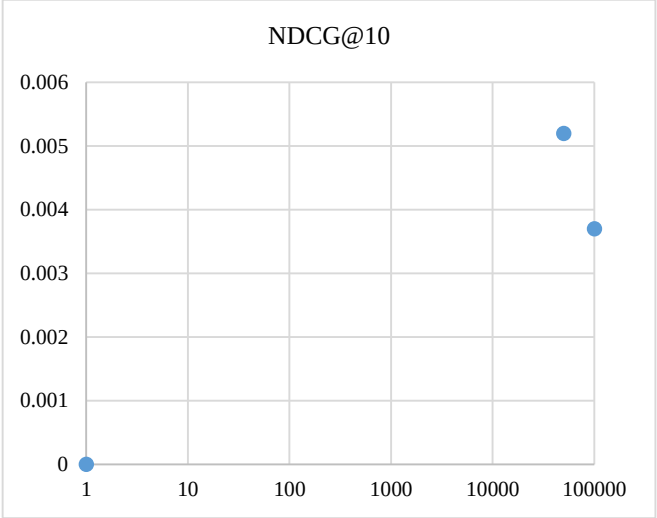

Fig. 8 Relationship between model complexity and ranking quality measured by NDCG@10

Figure 8 illustrates how model complexity (as represented by total trainable parameter count) relates to ranking quality for these models. The standard NCF model appears to have greatly increased the parameter count without improving ranking quality. On the other hand, while the NCF + NCB achieved a significant increase in NDCG@10 over other models, it also used substantially fewer trainable parameters than its counterpart (i.e., NCF). These results suggest that the importance of consistency within a neighbourhood and data quality outweighs that of architectural depth for recommendation system performance, and, furthermore, demonstrate diminishing returns from increasing a model's capacity.

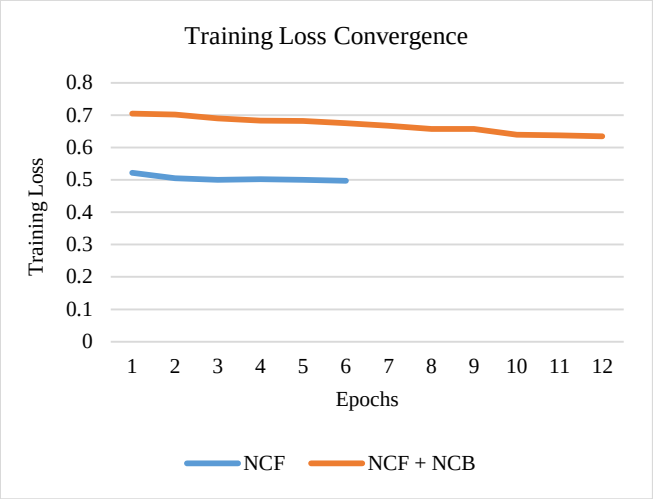

Fig. 9 Training loss convergence behavior of neural collaborative filtering models

The training loss trajectories of the NCF and NCF + NCB models over epochs are shown in Figure 9. Despite having fewer parameters, the NCF + NCB model converges more steadily and smoothly, achieving lower loss values. Because noisy interactions were eliminated during neighborhood-consistency filtering, this shows better optimisation behaviour and decreased overfitting risk. The outcomes provide more evidence that preprocessing stabilises neural recommendation training.

Figure 10 contrasts recommendation coverage and accuracy across the evaluated models. The Collaborative Neighbourhood filtering model has a low hit rate, which suggests poor personalisation, but it achieves high coverage, suggesting a significant percentage of items. Neural models prioritise relevance over breadth despite covering fewer items. This trade-off highlights the benefit of neural and hybrid approaches for personalised ranking and shows that high coverage by itself does not imply effective recommendation quality.

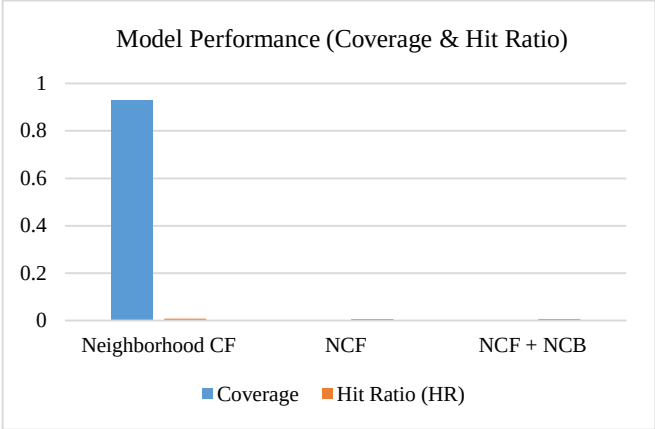

Fig. 10 Comparison of item coverage and Hit Ratio@10 across recommendation models

Table 8. Comparison of different method with proposed system

Study / Method	Dataset / Setting	Evaluation Protocol	Model Capacity	HR@10 (Reported)	NDCG@10 (Reported)	Remarks / Key Observation
Neighborhood CF (Classical) [10]	MovieLens	Random Split	Nonparametric	0.02-0.10	—	Effective only on dense data
NeuMF (Neural CF) [11]	MovieLens, Pinterest	Leave-One-Out + 99 Negatives	High (10^5-10^6 params)	0.41-0.50	0.20-0.27	Strong benchmark performance under controlled setup
Self-Supervised Graph Learning (SGL) [13]	Benchmark Graph CF datasets	Leave-One-Out	Very High	Improved over GCN baselines	Improved over GCN baselines	Robust to sparsity via self-supervision
Simple Graph Contrastive Learning [14]	MovieLens, Amazon subsets	Sampled Ranking	High	Comparable to SOTA	Comparable to SOTA	Efficiency-focused contrastive learning
Candidate-aware GCL (CGCL) [15]	Benchmark CF datasets	Leave-One-Out	Very High	Improved vs NeuMF	Improved vs NeuMF	Candidate-aware contrastive pretraining
This Work: Neighborhood CF	Amazon (Sparse)	Random Interaction Split	Non-parametric	0.0089		High coverage, weak personalization
This Work: NCF (GMF + MLP)	Amazon (Sparse)	Random Split, Full-Item Ranking	~3.8×10^5 params	0.0077	0.0037	Large capacity, limited ranking gain
This Work: NCF + NCB Filtering	Amazon (Sparse)	Random Split, Full-Item Ranking	~4.5×10^4 params	0.0073	0.0052	Best efficiency-performance trade-off

This study compares to previous studies on collaborative filtering and neural recommendation algorithms as shown in Table 8. Rather than only look at the absolute metric values, the focus of the comparison includes the model's capacity, evaluation procedures and characteristics of the dataset being evaluated. Neighborhood-based collaborative filtering approaches that were evaluated previously have limited scalability because of their dependence upon explicit computation of similarity; however, they produced moderate ranking performance on relatively dense datasets. The reported values of HR@10 and NDCG@10 obtained from using Neural Collaborative Filtering models and their extensions were significantly greater than those obtained from collaborative filtering techniques with neighborhood-based algorithms using leave-one-out evaluations with sample negatives on benchmark datasets, hence demonstrating greater model capacity and favourable experimental conditions.

There has been an emphasis on utilising contrastive and self-supervised learning approaches from 2021 in researching ways to increase reliability in very sparse datasets. Although these methods generally produce better rankings, they often have higher computational costs and complexity of architecture than other methods. Conversely, the methods presented in this study employed full-item ranking and random interaction splits from an extremely large and sparse Amazon dataset for these evaluations. All methods exhibited lower rankings in this more difficult, realistic research environment. However, distinct relative trends are noted. While the standard NCF model shows limited gains despite significantly higher parameterisation, the neighborhood-based baseline achieves high coverage but poor personalisation. With a favourable efficiency-performance trade-off, the suggested NCF with neighborhood-consistency-based filtering achieves better ranking quality with substantially fewer parameters. Overall, Table 8 shows that reported performance is heavily influenced by evaluation rigour, data sparsity, and model complexity, and that consistency-aware preprocessing can yield significant improvements without raising model capacity.

4.7. Relative Performance Improvement Analysis

People often compare neural recommendation algorithms to clearly understand the effect of neighborhood consistency filtering. For example, the NDCG@10 score improved from 0.0037 for NCF without any filtering to 0.0052 for NCF implemented with neighborhood consistency filtering, which is nearly a 40.5% improvement compared with NCF without filtering. However, HR@10 showed a decline from 0.0077 to 0.0073, which proves that filtering does not increase the probability of hits but rather improves the ranking of relevant items. Thus, it can be suggested that the main benefit of removing inconsistent interactions relates mostly to improved ranking performance rather than improved recommendation performance.

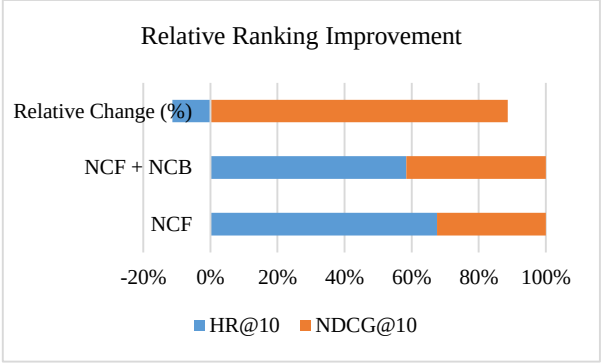

Fig. 11 Relative impact of NCB filtering on ranking performance

Figure 11 shows the effects of NCB filtering on the ranking performance. The NDCG@10 increased from 0.0037 with regular NCF to 0.0052 with NCF+NCB, which is an improvement of 40.54%. However, HR@10 decreased slightly from 0.0077 to 0.0073. The results demonstrate that NCB filtering helps increase the ranking of relevant items but has no real effect on the probability of a hit occurring.

4.8. Model Efficiency and Complexity Analysis

The suggested filtering method has some computational benefits. Indeed, the standard NCF architecture has approximately 381,600 trainable parameters, whereas the NCF with a neighborhood consistency filter has only approximately 45,300 parameters, which is an 88.1% reduction in size. Although the model size is significantly smaller now, the NDCG@10 score improved from 0.0037 to 0.0052. This proves that an increased model size does not yield any advantages under sparse interaction conditions; instead, it is much more beneficial to focus on the quality of the training data.

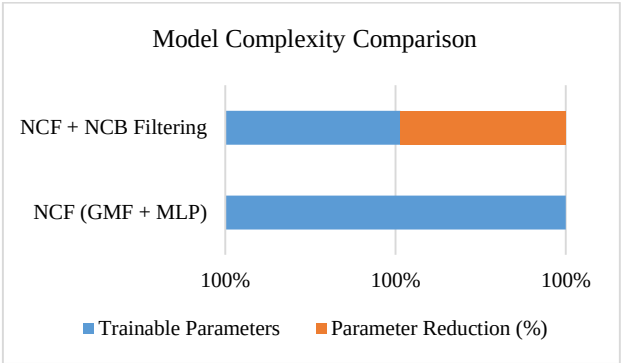

Fig. 12 Reduction in model complexity through NCB filtering

Figure 12 shows the decrease in complexity resulting from NCB filtering. The original NCF model has more than 381,600 parameters, whereas NCF+NCB has approximately 45,300 parameters, representing an 88.13% decrease in complexity. Despite the advantage, NCF+NCB outperforms NDCG@10, proving that better consistency of interaction provides an opportunity to obtain better ranking efficiency without increasing the complexity of the model.

4.9. Impact of Sparsity on Neighborhood Recommendation

The core objective of this study is to verify the sensitivity of traditional neighborhood-based collaborative filtering to sparse interaction data. This set this traditional algorithm as a fixed experimental baseline, and through a comparative analysis of core quantitative metrics, results confirmed that it exhibits clear performance defects in sparse data scenarios. Based on this, proposed follow-up algorithm optimization directions adapted to this type of scenario.

4.10. Recommendation Accuracy-Coverage Trade-off

The field of recommender systems has long grappled with a core trade-off between catalog coverage and relevance. Empirical tests of neighborhood-based collaborative filtering models show that these models achieve an item coverage rate of 93.13%, yet their HR@10 is only 0.0089. This finding dispels the misconception that high coverage equates to high-quality recommendations. Stakeholders must distinguish between recommendation diversity and ranking accuracy, adopt multiple sets of complementary metrics for evaluation, and not treat coverage as the sole assessment criterion.

5. Disccussion

The majority of current research on recommender systems has concentrated on enhancing ranking accuracy through progressively intricate neural architectures, which are frequently assessed on carefully selected benchmark datasets using advantageous evaluation protocols. While the aforementioned methodologies have shown superior absolute performance measurements in isolation, in real use environments of sparse recommendations, the performance metrics observed are compromised due to limitations stemming from the parameterisations used; there have proven to be limitations relating to scalability and robustness of the application because both were based on the presumption of dense patterns of interactions between users and items.

As such, the application of the above techniques to large e-commerce datasets where user-item information is highly inconsistent, very sparse (with a high level of noise), and individuals exhibit inconsistent preference behaviours creates a further issue. Under these conditions, most deep neural models are unable to translate any incremental increase in capacity into a sizable incremental performance improvement; further, traditional NCB filtering techniques based on similarity approaches have been hindered by poor estimation of neighbourhood proximity.

To address the issues of sparsity as it relates to current recommendation techniques; both architectural implementation and data consistency improvement; recent advancements in contrastive and self-supervised learning exist and therefore provide additional motivation for exploring whether implementing NCB filter techniques at the data level can improve NCB filtering on the basis of increased performance level, without increased implementation complexity (i.e., an increase in the number or type of neural components or loss function complexity). This inquiry explores this capability by using NCB filter techniques as a basic preprocessing step (rather than adding additional neural components or more complex loss functions) in the development of the recommendation system.

This methodology aims to prevent unstable users and anomalous interactions from degrading the signal quality of recommender systems as input data for subsequent learning processes by creating a cleaner, more reliable signal.

The systematic evaluation of data consistency has been identified as a primary factor in determining the quality of recommendations from various memory, neural network, and hybrid paradigms in a controlled, practical experimental environment. The proposed methodology utilises many fewer parameters than traditional neural network models and produces an increase in the stability of rankings through the integration of neighbourhood-based filtering techniques with collaborative filtering methods based on neural networks. Compared to naive design approaches that simply increase the size of both the input data set and the neural model, this approach demonstrates a superior efficiency/performance relationship through targeted preprocessing.

6. Conclusion

This work investigates the use of collaborative filtering techniques, including memory-based Neighbourhood Collaborative Filtering, Neural Collaborative Filtering, and a filtered neural variant that incorporates neighbourhood consistency, in realistic sparse recommendation contexts. By conducting coordinated experiments on extensive interaction datasets from Amazon with strict evaluation protocols, results demonstrate that both similarity-based and high-capacity neural models exhibit poor performance when they receive weak or inconsistent interaction signals. Results show that the addition of model complexity does not produce always improve recommendation quality for either type of model. While standard neural collaborative filtering models exhibited only marginal performance improvements over time, they required large amounts of computational power. Conversely, memory-based approaches had reasonably good levels of personalisation but achieved lower levels of item coverage due to inconsistency in their recommendations. In terms of ranking stability, system found that the proposed NCB approach optimises both the computational and memory expense required to administer NCB recommendations while providing significant improvements in ranking stability when measured by NDCG. Primarily, this paper aims to shift the focus of the discussion from architectural scaling to placing more importance on data consistency (reliablility). Essentially, the approach proposed in this paper improves the trade-off between efficiency and performance without adding more model complexity by enforcing neighbourhood

consistency during the data pre-processing phase. It appears that using consistent neighbourhoods during the data pre-processing phase will provide a realistic and scalable solution for developing real-world recommendation systems when using interaction-based recommendation system data under sparse interaction conditions. In future research, neighbourhood consistency will also be applied to sequential and context aware recommendation problem domains, and will be combined with advanced self-supervised learning and graph learning frameworks.

References

[1] Erion Çano, and Maurizio Morisio, "Hybrid Recommender Systems: A Systematic Literature Review," *Intelligent Data Analysis*, vol. 21, no. 6, pp. 1487-1524, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[2] Kyung-Yong Jung, Dong-Hyun Park, and Jung-Hyun Lee, "Hybrid Collaborative Filtering and Content-based Filtering for Improved Recommender System," *International Conference on Computational Science*, Kraków, Poland, pp. 295-302, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[3] Amany Sami et al., A Deep Learning based Hybrid Recommendation Model for Internet Users," *Scientific Reports*, vol. 14, no. 1, pp. 1-19, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[4] Dejuan Li, and James A. Esquivel, "Trust-Aware Hybrid Collaborative Recommendation with Locality-Sensitive Hashing," *Tsinghua Science and Technology*, vol. 30, no. 4, pp. 1421-1434, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[5] Savas Okyay, and Sercan Aygun, "Experimental Interpretation of Adequate Weight-Metric Combination for Dynamic user-based Collaborative Filtering," *PeerJ Computer Science*, vol. 7, pp. 1-40, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Liheng Xu, Zhile Guan, and Yu Wu, "Enhancing Movie Recommendation Systems with Hybrid Collaborative Filtering, Content-based Filtering and SVD," *Proceedings of the 2nd International Conference on Machine Learning and Automation*, Adana, Turkey, pp. 1-13, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[7] Nor Liyana Natasha Muhammad Syafiq Chelvam et al., "Hybrid-based Recommender System based on Electronic Product Reviews," *JOIV: International Journal on Informatics Visualization*, vol. 9, no. 4, pp. 1752-1764, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[8] Bagus Wicaksono Nurjayanto, and Z.K.A. Baizal, "News Recommender System using Hybrid Content-based Filtering and Collaborative Filtering," *Scientific Journal of Research and Learning in Informatics*, vol. 9, no. 1, pp. 26-33, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[9] Giovanni Gabbolini et al., "On the Instability of Embeddings for Recommender Systems: The case of Matrix Factorization," *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pp. 1363-1370, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] Jonathan L. Herlocker et al., Evaluating Collaborative Filtering Recommender Systems," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 5-53, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[11] Xiangnan He et al., "Neural collaborative filtering," *Proceedings of the 26th International Conference on World Wide Web*, pp. 173-182, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[12] Shuai Zhang et al., Deep Learning based Recommender System: A Survey and New Perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1-38, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[13] Jiancan Wu et al., "Self-Supervised Graph Learning for Recommendation," *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 726-735, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Junliang Yu et al., Are Graph Augmentations Necessary? Simple Graph Contrastive Learning for Recommendation," *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1294-1303, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Wei He et al., Candidate-Aware Graph Contrastive Learning for Recommendation," *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1670-1679, 2023. [CrossRef] [Google Scholar] [Publisher Link]