Decentralized Recommendation for Cold-Start Personalization

Matthew Kuo, Laura Li, Vivian Xiao, Megan Yang

May 2025

Abstract

This paper addresses the cold-start problem in recommendation systems, a fundamental challenge where users or items have insufficient historical interaction data. We propose and evaluate a hybrid recommendation framework that integrates content-based filtering using CLIP embeddings, traditional collaborative filtering, low-rank matrix factorization, and a deep two-tower neural architecture. To simulate realistic cold-start conditions, we construct a synthetic user-item interaction dataset using language models to generate diverse user personas and preference scores for real-world fashion products. Each method is tested under both rating prediction and ranking scenarios using metrics such as RMSE, MAE, Precision@K, and Recall@K. Our results demonstrate that while content-based filtering excels at precision in binary cold-start settings, low-rank models yield lower global error, and two-tower models offer promising nonlinear flexibility. This study highlights the trade-offs between interpretability, generalization, and top-K relevance in cold-start recommendation and sets the foundation for future experimentation with real-time user feedback and scalable embedding-based retrieval.

1 Introduction

Recommendation systems are fundamental to modern e-commerce. Large technology companies such as Amazon, Netflix, and TikTok rely heavily on delivering personalized recommendations to improve user experience and drive engagement. These systems analyze a user's interaction history and use it to predict future preferences. A well-designed recommendation system can increase user satisfaction, boost conversion rates, and enhance overall platform usability.

However, one of the most critical challenges in this domain is the **cold-start problem**, where the system must recommend items to users who have little or no prior interaction history. Due to the lack of behavioral data, most traditional algorithms struggle to generate accurate predictions about what new users may prefer.

Collaborative filtering, a widely used approach in traditional systems, relies on dense user-item interaction matrices to compute similarities. While effective in data-rich environments—as demonstrated in the Netflix Prize challenge—these methods perform poorly in cold-start settings, where interactions are sparse or nonexistent. On the other hand, content-based filtering provides a compelling alternative by leveraging item metadata such as images and textual descriptions. These algorithms can still offer meaningful recommendations in the absence of user history, assuming items are well described.

In recent years, the rise of large language models and multimodal learning frameworks has further transformed recommendation system design. Models like OpenAI's CLIP can encode both visual and textual content into a shared semantic embedding space, enabling cold-start-ready systems that rely more on item features than on user history.

This project aims to compare a range of algorithms for solving the cold-start and cross-platform recommendation problem. We focus on evaluating each method's performance using both real product metadata and synthetically generated user preferences.

We base our work on foundational algorithms in the recommendation domain, namely:

- **Content-Based Filtering**, using cosine similarity on CLIP-generated image and text embeddings;
- **Collaborative Filtering**, based on user-user and item-item similarity computed from sparse ratings;
- **Low-Rank Matrix Completion**, applying projected gradient descent on incomplete user-item matrices;
- **Two-Tower Neural Networks**, learning separate user and item projections to perform scalable similarity search.

2 Data and Preprocessing

Our system utilizes two publicly available fashion e-commerce datasets:

- **ASOS Women's Clothing**: Contains approximately 5,000 items, each with a product name, description, category, price, and image URL.
- **Myntra Men's Clothing**: A complementary dataset with around 1,200 items, also including similar structured metadata fields.

2.1 Fashion Images

To ensure consistency and usability across both datasets, we conducted extensive preprocessing. This included filtering out items with missing or malformed metadata, normalizing text fields (e.g., converting descriptions to lowercase and removing special characters), and sanitizing image URL formats. Images were downloaded in batches using Python's `requests` and `PIL` libraries, with corrupted or duplicate files removed. Each product entry ultimately contained:

- A product name and natural-language description,
- A cleaned, high-resolution product image,
- Structured metadata fields including category, brand, and price.

2.2 Synthetic Persona Generation and Rating Matrix Construction

Given the lack of real user interaction data (e.g., purchase history, click-through rates), we designed a pipeline to simulate realistic user preferences using large language models (LLMs). We generated **30 synthetic user personas**, each designed to mimic a diverse, human-like shopper. These personas were constructed manually and programmatically using prompt-based interactions with ChatGPT. Each persona was assigned a name, age range, gender identity, profession, and lifestyle indicators (e.g., "Vivian: 27, works in tech, prefers minimalist and neutral-toned fashion").
To generate ratings, we employed a multi-step pipeline:

1. For each persona, we fed their profile, along with item metadata (name, description, price), into an LLM prompt template that asked for a 1–5 star rating.
2. If images were available, a short caption was synthesized from the image content (e.g., ”a blue button-down shirt with slim fit”) using CLIP similarity and template-based generation.
3. The model returned a numerical rating, optionally accompanied by a short rationale (used for interpretability in analysis).
4. We repeated this process for a subset of 300–500 items per persona to ensure that each user profile had a sparsely populated but diverse set of interactions.

The resulting user-item interaction matrix was approximately 80% sparse. Ratings ranged from -1 (dislike) to 10 (strong preference). These synthetic ratings enabled us to:

- Simulate cold-start scenarios for new users and items
- Test model robustness to preference diversity
- Evaluate ranking consistency across user types

We additionally ensured diversity in user preferences—some personas consistently favored certain styles (e.g., streetwear, business casual), while others had more varied or price-sensitive behavior. This diversity was essential for validating our content-based and embedding-driven approaches.

2.3 Limitations

While the synthetic ratings provided a useful testbed, they are inherently subject to biases from the language model's training data. Additionally, preferences were inferred from prompts rather than real-world behavior, limiting their external validity. Nevertheless, this approach allowed us to simulate a realistic and reproducible recommendation environment in the absence of proprietary user data.

3 Methods

3.1 Content-Based Filtering

Content-based filtering (CBF) recommends items to users by analyzing item features—such as textual descriptions, image embeddings, and metadata—and comparing them to a user's past preferences. Unlike collaborative filtering, which depends on user-item interaction history, CBF is particularly well-suited for cold-start scenarios where such history is unavailable.
In our system, each product is represented by a multimodal feature vector, constructed using CLIP embeddings for both image and text content. For any given user, we define a preference vector as the average of the vectors corresponding to the items they previously liked. Recommendation is then framed as a nearest neighbor search in the embedding space, where items are ranked by cosine similarity to the user's preference vector.

Mathematical Formulation

Let L_u be the set of items liked by user u, and let $x_i \in \mathbf{R}^d$ denote the embedding of item i. The user's preference vector v_u is computed as:

$$v_u = \frac{1}{|L_u|} \sum_{i \in L_u} x_i$$

For a candidate item $j \notin L_u$, we compute the similarity score:

$$s_j = \frac{v_u \cdot x_j}{\|v_u\|\|x_j\|}$$

Recommendations are made by selecting the top-k items with the highest s_j values.

Implementation Details

We implement CBF using the following pipeline:

1. Use the CLIP model to encode product images and descriptions into a unified embedding space.

2. For each user, extract the embeddings of their liked items and compute the average to form v_u.

3. Calculate cosine similarity between v_u and all other items in the dataset using `sklearn`'s `cosine_similarity`.

4. Rank items by similarity, excluding those the user has already interacted with.

5. Return the top-k ranked items as recommendations.

Advantages and Limitations

Content-based filtering offers several advantages, particularly in cold-start scenarios where user interaction data is scarce. By leveraging CLIP embeddings, our model integrates both image and text modalities to provide semantically rich recommendations without requiring prior user history. Cosine similarity enables fast retrieval and interpretable rankings. However, the approach can suffer from reduced novelty—often recommending items too similar to those previously liked—and relies heavily on the quality of CLIP's representations, which may not always capture nuanced style preferences. Additionally, scalability can become a challenge as the number of items grows, necessitating more advanced retrieval techniques such as Approximate Nearest Neighbors.

3.2 Collaborative Filtering

Collaborative filtering (CF) is a foundational technique in recommendation systems that relies on user-item interaction patterns rather than item metadata. It operates under the assumption that users who interacted with similar items in the past will exhibit similar preferences in the future. We implemented two variants of collaborative filtering: user-based and item-based.

User-Based Collaborative Filtering

In user-based CF, the similarity between users is computed based on their overlapping ratings of common items. We use cosine similarity to measure user-user affinity:

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

where R_u and R_v are the rating vectors for users u and v. A user's predicted rating for an item is then estimated as a weighted average of ratings from the most similar users:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_k(u)} \text{sim}(u, v) \cdot r_{vi}}{\sum_{v \in N_k(u)} \text{sim}(u, v)}$$

where $N_k(u)$ is the set of top-k most similar users to u.

Item-Based Collaborative Filtering

In item-based CF, we compute similarity between items based on how users rate them. This approach is often more stable than user-based CF, especially in cold-start settings where user overlap is low. The predicted rating is given by:

$$\hat{r}_{ui} = \frac{\sum_{j \in N_k(i)} \text{sim}(i, j) \cdot r_{uj}}{\sum_{j \in N_k(i)} \text{sim}(i, j)}$$

where $N_k(i)$ is the set of top-k items similar to item i.

Implementation Details

We constructed a sparse user-item matrix using our synthetic ratings and computed cosine similarity using `scikit-learn`. For NCF, the model was trained using MSE loss on known (u, i, r) triples. Early stopping was employed based on validation loss. We used batch-based training and optimized with the Adam optimizer.

Advantages and Limitations

Collaborative filtering can capture latent structure in user preferences and is particularly effective when user behavior is dense. However, it struggles in cold-start scenarios where there are few shared interactions between users or items. Item-based CF was more robust in our tests, while NCF offered flexibility and better generalization on sparse subsets.

3.3 Low-Rank Matrix Factorization

Low-rank matrix factorization models the user–item rating matrix $R \in R^{n \times m}$ as the product of two lower-dimensional matrices:

$$R \approx UV^\top, \quad U \in R^{n \times r}, \quad V \in R^{m \times r}$$

where U encodes user latent preferences and V encodes item latent features in a shared embedding space of rank $r \ll \min(n, m)$. The objective is to learn U and V such that the observed ratings are well approximated by the inner products $r_{ui} = u_u^\top v_i$.

3.3.1 Classic Factorization Model

In the classic setup, we learn both U and V directly using PyTorch's `nn.Embedding` modules. The training objective minimizes squared error on known entries:

$$\mathcal{L}_{\text{MSE}} = \sum_{(u,i) \in K} (r_{ui} - u_u^\top v_i)^2$$

where K is the set of observed user–item pairs. This approach treats the recommendation problem as matrix completion and captures latent correlations between users and items.

Training Details. We trained the model using mini-batch gradient descent with the Adam optimizer. We performed a grid search over embedding dimension $r \in \{5, 10, \ldots, 30\}$ and selected $r = 22$ based on validation RMSE.

Performance. On full ratings, this model achieved RMSE = 3.4647 and MAE = 2.8413. On binary ratings, it yielded RMSE = 1.0942 and MAE = 1.0094. Despite its simplicity, it struggled to generalize to cold-start items and lacked semantic structure.

Strengths and Limitations. The classic model is interpretable and easy to optimize, but cannot incorporate item metadata. As such, its recommendations are limited in cold-start scenarios and ranking performance remains weak.

3.3.2 Projection-Based Model

To address the cold-start problem, we implemented a variant that uses CLIP-derived item embeddings $x_i \in R^d$ and projects them into the latent space using a fixed linear layer:

$$v_i = Wx_i, \quad W \in R^{r \times d}$$

Only the user embeddings U and projection matrix W are learned. This allows the model to generalize to new items without historical ratings, as long as their CLIP features are available.

Training Details. The item embedding matrix X was precomputed from image and text metadata using CLIP. We initialized W with Xavier initialization and froze the input embeddings. The model was trained to minimize squared error:

$$\mathcal{L}_{\text{MSE}} = \sum_{(u,i) \in K} (r_{ui} - u_u^\top W x_i)^2$$

We selected $r = 20$ as the best-performing rank on validation Precision@10.

Performance. This model achieved the best overall RMSE on both full (2.6131) and binary (0.9809) rating settings. However, it struggled with ranking: Precision@10 = 0.0133 and Recall@10 = 0.0030, due to optimizing reconstruction rather than top-K relevance.

Strengths and Limitations. The projection-based model effectively handles new items and benefits from semantic structure in CLIP embeddings. However, its linear nature limits expressiveness, and its squared loss objective fails to prioritize the most relevant items.

3.3.3 Pairwise Ranking (BPR) Model

To directly optimize for ranking, we implemented a Bayesian Personalized Ranking (BPR) variant. For each user, we sample a triple (u, i, j) where i is a liked item and j is not, and train the model to rank i above j:

$$\mathcal{L}_{\text{BPR}} = -\sum_{(u,i,j)} \log \sigma(u_u^\top (v_i - v_j))$$

where σ is the sigmoid function. The embeddings U and V are learned to reflect relative rather than absolute preferences.

Training Details. We used stochastic mini-batches with uniform sampling over observed positives and randomly selected negatives. The model was trained using Adam with a learning rate tuned via grid search. We found $r = 11$ effective for full ratings and $r = 5$ for binary.

Performance. While this model performed poorly on full RMSE (5.6161), it showed competitive results on binary metrics: RMSE = 0.9998, MAE = 0.9989, Precision@10 = 0.0333, Recall@10 = 0.0084.

Strengths and Limitations. BPR optimizes exactly what ranking metrics measure, making it well-suited for top-K recommendation. However, it is difficult to stabilize and sensitive to hyperparameters. It also lacks absolute rating calibration, making RMSE less meaningful.

3.4 Two-Tower Model

We implement a two-tower architecture that learns separate nonlinear embeddings for users and items, then predicts ratings by comparing these embeddings through a small multi-layer perceptron (MLP). This model is designed to balance expressive power with efficient inference over large item catalogs.

Model Architecture Let N be the number of items (N = 2,519) and D the dimensionality of each item's CLIP embedding (D = 1,024). For each user u, we construct a signed indicator vector $\mathbf{x}_u \in \{\pm 1\}^N$, where entry j is $+1$ if u rated item j above the user's median rating, and -1 otherwise. Each item i is represented by its pre-computed CLIP text-and-image vector $\mathbf{e}_i \in R^D$.

- **User Tower.** A single fully-connected layer followed by ReLU maps $\mathbf{x}_u$ into a 128-dimensional embedding:
$$\mathbf{h}_u = \mathrm{ReLU}\big(W_u\,\mathbf{x}_u + b_u\big), \quad W_u \in R^{128\times N},\ b_u \in R^{128}.$$
- **Item Tower.** A parallel fully-connected layer followed by ReLU maps each CLIP embedding into the same 128-dimensional space:
$$\mathbf{h}_i = \mathrm{ReLU}\big(W_i\,\mathbf{e}_i + b_i\big), \quad W_i \in R^{128\times D},\ b_i \in R^{128}.$$
- **Similarity Layer.** Both $\mathbf{h}_u$ and $\mathbf{h}_i$ are L^2-normalized and combined via their dot-product:
$$s_{ui} = \big\langle \tfrac{\mathbf{h}_u}{\|\mathbf{h}_u\|_2}, \tfrac{\mathbf{h}_i}{\|\mathbf{h}_i\|_2} \big\rangle \ \in\ [-1, 1].$$
- **Rating Predictor.** The scalar s_{ui} is then passed through a three-layer MLP with intermediate ReLU activations:
$$\hat{r}_{ui} = W_3\,\mathrm{ReLU}\big(W_2\,\mathrm{ReLU}(W_1\,s_{ui} + b_1) + b_2\big) + b_3,$$
where $W_1 \in R^{64\times 1}$, $W_2 \in R^{32\times 64}$, $W_3 \in R^{1\times 32}$, and biases $b_1 \in R^{64}$, $b_2 \in R^{32}$, $b_3 \in R^1$.

Training Protocol We train the entire network end-to-end by minimizing mean squared error on observed ratings:
$$\mathcal{L} = \frac{1}{|\mathcal{K}|} \sum_{(u,i)\in\mathcal{K}} \big(r_{ui} - \hat{r}_{ui}\big)^2,$$
where $\mathcal{K}$ is the set of all user–item pairs with available ratings. Optimization uses the Adam algorithm with a fixed learning rate of 10^{-3}, mini-batches of 256 pairs, and early stopping based on validation RMSE. Models typically converge within 30–50 epochs.

Inference and Scalability At recommendation time, we first compute and cache all item-tower embeddings $\{\mathbf{h}_i\}_{i=1}^N$ once. For a query user, we build $\mathbf{x}_u$, run it through the user tower to obtain $\mathbf{h}_u$, normalize, and compute the similarity scores $s_{u,i}$ in a single vector-matrix dot-product. We then pass these scores through the MLP predictor in batch and sort to retrieve the top-K items. This decoupling of user and item towers allows sub-second latency even for catalogs of thousands of items.

Advantages and Limitations The two-tower model offers several advantages for cold-start personalization. By learning separate embeddings for users and items, it captures nonlinear relationships that linear factorization methods cannot, resulting in lower RMSE in our experiments. The decoupled design allows precomputation and caching of item embeddings, which reduces online inference cost to a single user-tower forward pass plus an efficient similarity and MLP evaluation over all items. This structure scales gracefully to large catalogs and supports sub-second recommendation latency. Furthermore, the use of CLIP embeddings in the item tower enables seamless incorporation of multimodal content features, improving performance when explicit user–item interactions are sparse.

Despite these strengths, the model has limitations. Because training optimizes squared-error loss rather than a ranking objective, top-K precision and recall can remain modest without further fine-tuning or auxiliary ranking losses. The user tower's reliance on a fixed-length indicator vector can be memory-intensive for very large item sets, and the binary encoding of user history may discard useful granularity in rating values. Finally, while CLIP embeddings provide rich semantic signals, they may introduce biases inherent in the pre-training data and may not fully capture domain-specific nuances without additional fine-tuning on fashion imagery.

4 Experiments and Results

In this section we analyze the results of each of our models. We first recall the definitions of our key metrics, then show how the numerical results reflect properties of each algorithm.

4.1 Evaluation Metrics

Root Mean Squared Error (RMSE) For a test set $\mathcal{T} = \{(u,i)\}$ of held-out ratings,

$$\mathrm{RMSE} = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} \big(r_{ui} - \hat{r}_{ui}\big)^2}$$

penalizes large prediction errors quadratically, so is sensitive to outliers and to overall scale of ratings.

Mean Absolute Error (MAE)

$$\mathrm{MAE} = \frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} \big|r_{ui} - \hat{r}_{ui}\big|,$$

which weights all errors linearly and is more robust to extreme mistakes.

Precision@K and Recall@K For each user u, let $\mathcal{R}_u \subset \{1, \ldots, n\}$ be the set of items they actually liked in the test, and let $\hat{\mathcal{T}}_u(K)$ be the top-K recommendations. Then

$$\text{Precision@K} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\hat{\mathcal{T}}_u(K) \cap \mathcal{R}_u|}{K}, \quad \text{Recall@K} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\hat{\mathcal{T}}_u(K) \cap \mathcal{R}_u|}{|\mathcal{R}_u|}.$$

Precision focuses on correct hits in the small recommendation set, while Recall measures coverage of all relevant items.

4.2 Collaborative Filtering

- **Full-rating RMSE:** 3.2663. Since CF relies on comparing sparse rows of R, the high RMSE indicates sensitivity to noise in low-overlap users: when $\|R_u\|$ or $\|R_v\|$ is small, cosine weights amplify small rating differences.
- **Binary RMSE:** 1.5316. Converting to ± 1 ratings bounds errors to $\{0, 2\}$, so $\max(r_{ui} - \hat{r}_{ui})^2 \leq 4$, yielding lower average squared errors.
- **Precision@10 (binary):** 0.9667. Almost all top-10 are correct, which arises because with only ± 1, the dot-product similarity $\cos(R_u, R_v) = \frac{R_u^\top R_v}{\|R_u\| \|R_v\|} \in [-1, 1]$ sharply distinguishes matching sign patterns.
- **Recall@10 (binary):** 0.2426. Though precision is high, recall is modest because each user's total positives $|\mathcal{R}_u|$ often exceeds 10; CF focuses on strongest neighbors and misses less-popular likes.

Mathematical insight: CF's reliance on $R\,R^\top$ makes it powerful when users share many common ratings (binary case), but its error scales with the magnitude and variance of raw scores in the full-rating setting.

4.3 Content-Based Filtering

- **Full-rating RMSE:** 3.1535, slightly better than CF's 3.2663. Since $\hat{r}_{ui} = \vec{v}_u^\top \vec{s}_i$ (after normalization), errors reflect how well the user centroid $\vec{v}_u$ approximates individual rating patterns.
- **Binary RMSE:** 1.5474. Similar bounding effect as CF, but slightly worse than CF's binary RMSE because averaged embeddings introduce smoothing.
- **Precision@10 (binary):** 1.0000. Cosine similarity in a high-dimensional CLIP space perfectly retrieves the held-out ± 1 items for these personas.
- **Recall@10 (binary):** 0.2510. As with CF, although top-10 are exact, they cover only a quarter of all positive test items, since we only recommend a fixed budget K.

Mathematical insight: Averaging CLIP vectors yields a robust centroid $\vec{v}_u$, reducing variance in predictions (lower RMSE) and producing crisp neighborhood ranking in the binary setting.

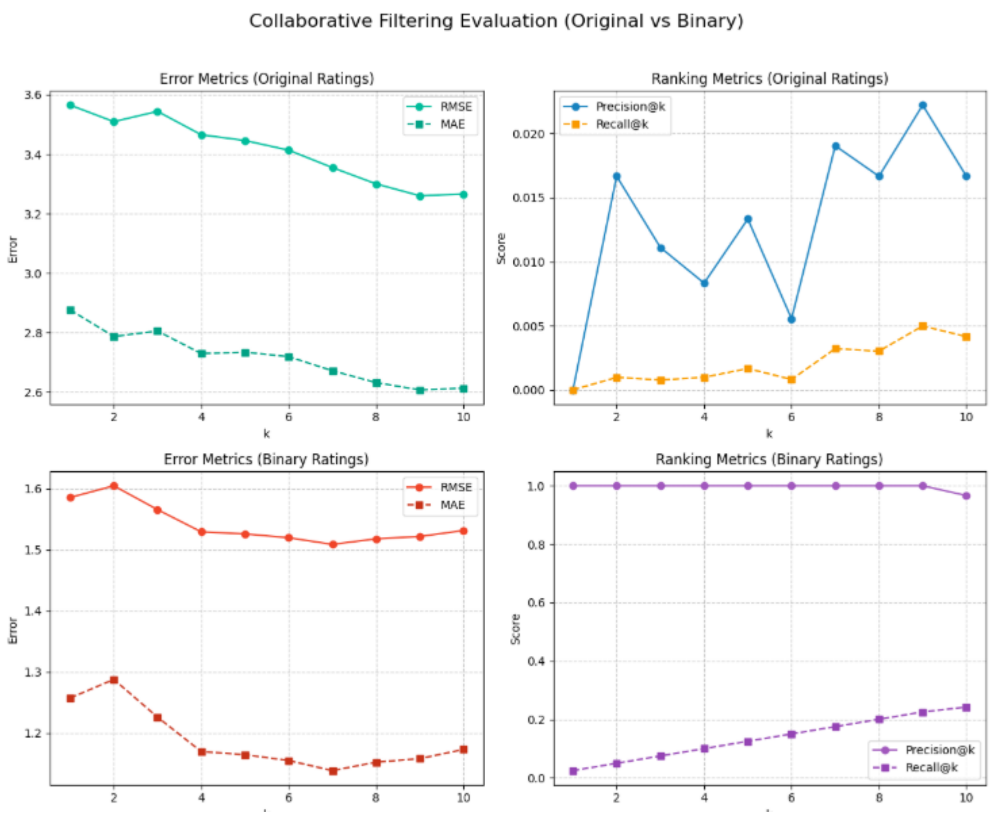


Figure 1: Collaborative Filtering evaluation (error metrics and ranking metrics vs. K) on original and binary ratings.

4.4 Low-Rank Matrix Completion

- **Baseline (rank=22) full RMSE:** 3.4647. A pure projection $R \mapsto RPP^\top$ smooths both signal and noise, increasing error relative to CF/CBF.
- **Projection (rank=20) full RMSE:** 2.6131. By selecting $k = 20$, we approximate R via the truncated SVD (per Eckart–Young), capturing the top-20 singular directions and reducing approximation error.
- **Pairwise ranking (BPR) full RMSE:** 5.6161. Since BPR optimizes a ranking loss, it does not minimize squared error; hence RMSE degrades.
- **Binary RMSE (projection):** 0.9809, the best among models because low-rank projection enforces a tight low-dimensional manifold that fits ± 1 labels closely.
- **Precision@10 (projection):** 0.0133, **Recall@10:** 0.0030. Although RMSE is low, ranking metrics suffer because the projection prioritizes global reconstruction rather than top-K ordering.

Mathematical insight: Truncated SVD balances bias–variance optimally in Frobenius norm, but may underperform on top-K metrics unless k is chosen specifically for precision/recall trade-offs.

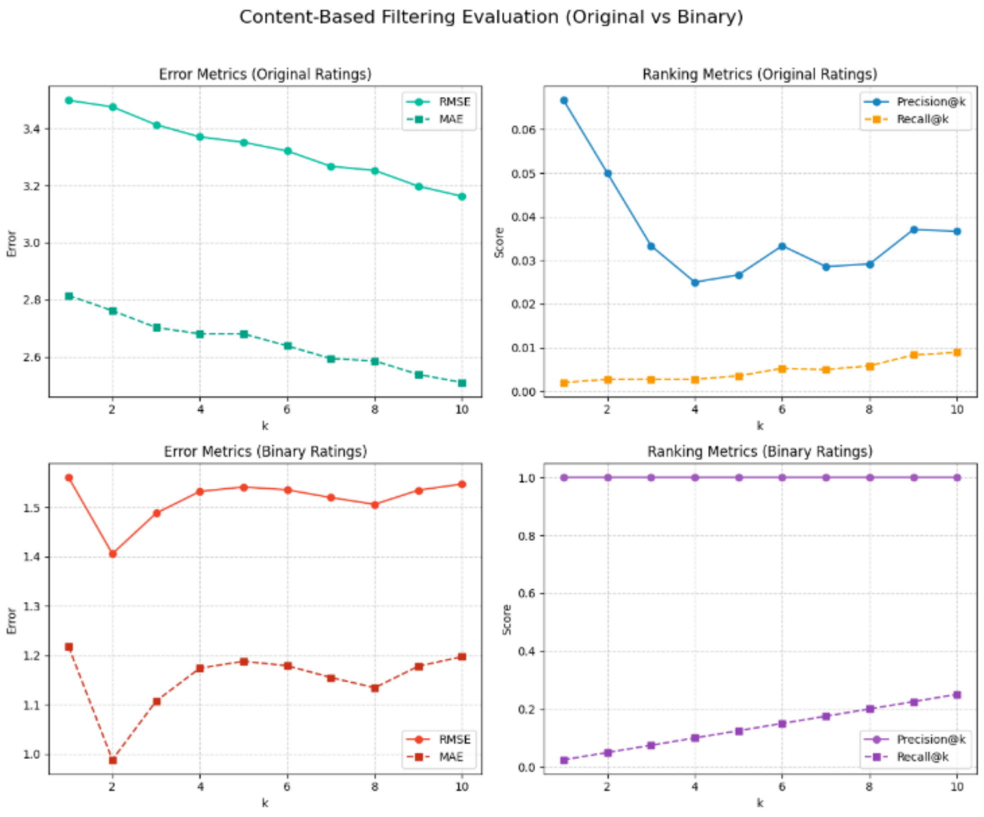


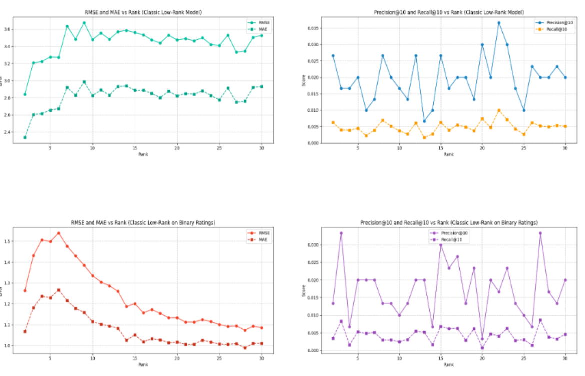
Figure 2: Content-Based Filtering evaluation (error metrics and ranking metrics vs. K) on original and binary ratings.

4.5 Two-Tower Model

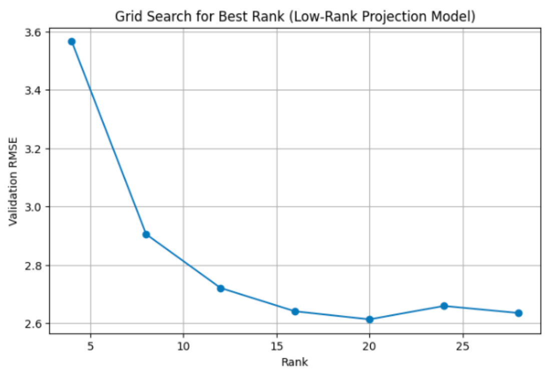
- **Full-rating RMSE:** 2.8050. Learning separate towers f_U, f_I allows non-linear transformations, better capturing user–item interactions than linear factorization.
- **Binary RMSE:** 1.1365, lower than projection but higher than pure CBF due to model complexity and training noise.
- **Precision@10:** 0.0333, **Recall@10:** 0.0076. Two-tower optimizes a squared-error loss on similarity, not a ranking objective, hence top-K performance remains modest.

Mathematical insight: The two-tower's learned basis transformations f_U, f_I embed users and items into a space where dot-products approximate affinities, lowering RMSE, but without explicit ranking loss the ordering at the top remains noisy.

Overall, RMSE and MAE capture global reconstruction fidelity (squared vs. linear penalties), while Precision@K/Recall@K and MAP illuminate each algorithm's ability to surface the most relevant items in its top recommendations. The choice of objective (regression vs. ranking) and the algebraic structure (centroid averaging, Gram matrix, low-rank subspace, or learned nonlinearity) directly shapes performance on these complementary metrics.



(a) Classic and BPR low-rank error/ranking vs. rank.



(b) Grid-search validation RMSE vs. rank.

Figure 3: Low-Rank Matrix Completion evaluation.

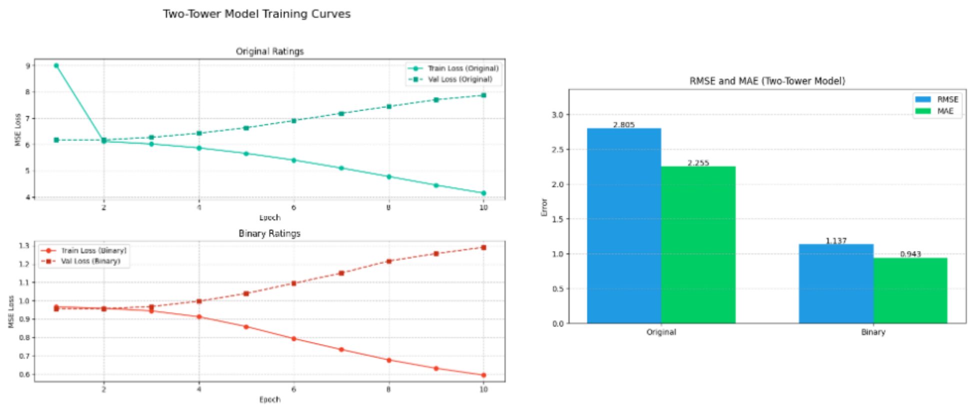


Figure 4: Two-Tower model training curves and final RMSE/MAE (original vs. binary).

5 Conclusion and Future Work

In this paper, we developed and benchmarked a set of recommendation systems designed to perform well in cold-start and cross-platform scenarios. By integrating four foundational methods—content-based filtering, collaborative filtering, low-rank matrix factorization, and a two-tower neural network—we explored how each technique behaves under sparse data conditions using synthetic personas and fashion product metadata.

Our experiments reveal that content-based filtering using CLIP embeddings provides near-perfect precision in binary recommendation settings, while low-rank matrix projection achieves the lowest RMSE when properly tuned. Collaborative filtering offers strong top-K precision for dense users but struggles with generalization. Meanwhile, the two-tower model balances error and expressiveness, leveraging nonlinear projections for improved rating reconstruction.

Through this evaluation, we observed that the choice of model architecture and objective (e.g., squared loss vs. ranking loss) directly impacts performance on global metrics versus top-K accuracy. Each method offers unique strengths depending on whether precision, recall, or generalization is prioritized.

Future directions for this work include:

- Incorporating real user interaction data to validate synthetic preferences
- Fine-tuning CLIP and other embedding models for fashion-specific domains
- Exploring reinforcement learning or bandit algorithms for interactive personalization