Report: Recommender Systems — Part 1

1. Dataset & Evaluation Setup

We work with **MovieLens 1M**: 1,000,209 ratings from 6,040 users on 3,706 movies. Ratings are on a 1–5 integer scale. Every rating carries a timestamp.

The user-item matrix is **95.5% sparse** — the average user rated ~166 out of 3,706 movies. Both user activity and item popularity follow a power-law distribution: a small number of "power users" contributed thousands of ratings, and a handful of blockbusters received the majority of all ratings. About 19% of movies have fewer than 20 ratings — the cold-start zone.

Evaluation Protocol

All models share a single evaluation protocol:

- **Per-user temporal split (80/10/10):** For each user, we sort their ratings by timestamp. The earliest 80% go to training, the next 10% to validation, and the final 10% to the test set. This prevents data leakage — we always train on past behavior and evaluate on future behavior.
- **Relevance threshold:** A rating ≥ 4.0 counts as relevant.
- **Primary metric:** NDCG@K — measures how well relevant items are ranked near the top of the recommendation list.
- **Secondary metrics:** Recall@K, Precision@K, Coverage (fraction of catalog appearing across all recommendations), and Popularity Bias (average popularity of recommended items).
- **K values:** 5, 10, 20. We report K=10 as the primary comparison point.
- All metrics are **averaged per user**, so power users do not dominate.

2. Models Implemented

We implemented three families of recommenders plus a Popularity baseline.

2.1 Content-Based Filtering

Each movie is represented as a feature vector. The user profile is built by aggregating feature vectors of movies the user rated, weighted by `(rating - mean_rating)`.

Basic CB: Binary genre vectors (18 genres). We tested Jaccard, Cosine, and Pearson similarity between user profiles and item vectors. This performed poorly because genre is a coarse feature — many movies share the

same genre vector.

Enhanced CB: TF-IDF weighted genre features + decade feature + popularity blending. The TF-IDF weighting helps: rare genres like "Film-Noir" get higher weight than common genres like "Drama." We also added a `popularity_weight = 0.5` term that blends the content score with a popularity score. This was necessary because on this dataset, knowing what's popular is almost as informative as knowing genre preferences.

2.2 Item-Item Collaborative Filtering

Computes item-item similarity from user co-interaction patterns. We compared two similarity functions:

- **Jaccard similarity** on binarized interactions (rating ≥ 4.0):
  `|users who liked both| / |users who liked either|`
- **Adjusted Cosine similarity** on raw ratings

Jaccard was better. Binarizing at threshold 4.0 strips out noise from mediocre ratings and focuses on strong positive signals. Adjusted Cosine tries to use the actual rating values, but the extra precision hurts more than it helps — users are inconsistent raters.

Hyperparameters: K=50 nearest neighbors per item, min_cooccurrence=2. The top-50 neighbors and similarities are precomputed offline. At inference, for each user, we look up the neighbors of their liked items, aggregate weighted scores, and return the top-K.

2.3 Matrix Factorization

Both models decompose the rating matrix into user factors P and item factors Q, predict `r̂ = μ + b_u + b_i + P_u · Q_i`, and minimize MSE.

FunkSVD: SGD-based. Hyperparameters: n_factors=30, lr=0.005, reg=0.02, 20 epochs. Initialization: N(0, 0.1). Global bias μ = mean training rating. Compiled via Numba (`@njit(parallel=True, fastmath=True)`) — each epoch iterates over all interactions, shuffled, updating user/item factors and biases with gradient descent.

ALS (Alternating Least Squares): Closed-form alternating optimization. Hyperparameters: n_factors=60, reg=0.1, 15 iterations. Each iteration: fix item factors → solve `(Q^T Q + λI) P_u = Q^T r_u` for each user; fix user factors → solve the analogous system for each item. More stable convergence than SGD, but higher per-iteration cost.

Both models optimize **RMSE** (rating prediction error), while we evaluate on **NDCG** (ranking quality). This is a fundamental mismatch: a model can predict ratings accurately (3.8 vs true 4.0) but still fail at ranking (placing irrelevant items above relevant ones in the top-10). This explains their weak performance below.

2.4 Popularity Baseline

Simply counts the number of ratings per movie (`groupby('item_id').size()`) and recommends the most-rated ones. All ratings count regardless of value — a movie with 500 one-star ratings and one with 500 five-star

ratings get the same score.

3. Part 1 Results

Model	NDCG@5	NDCG@10	NDCG@20	Precision@10	Recall@10	Coverage	Pop. Bias
Item-Item CF (Jaccard)	**0.0562**	**0.0625**	**0.0770**	**0.0457**	**0.0593**	19.7%	1681
Popularity	0.0437	0.0490	0.0615	0.0393	0.0433	5.1%	2086
Enhanced CB	0.0391	0.0460	0.0577	0.0302	0.0495	40.4%	978
FunkSVD	0.0245	0.0265	0.0313	0.0218	0.0223	16.8%	799
ALS	0.0196	0.0219	0.0271	0.0173	0.0185	84.5%	470
Basic CB	0.0073	0.0087	0.0116	0.0064	0.0090	100.2%	237

Key observations

Item-Item CF is the clear Part 1 winner. NDCG@10 = 0.0625 — roughly 28% better than the Popularity baseline. The Jaccard similarity on binarized interactions was the right choice for this dataset: it captures the "co-liking" signal cleanly.

The Popularity baseline is surprisingly strong. NDCG@10 = 0.049 just from recommending the most popular movies. Enhanced CB (0.046) barely beats it, and half of the Enhanced CB's signal actually comes from its popularity_weight term. This tells us that on MovieLens, "what's popular" is a very informative feature.

Matrix Factorization underperforms. FunkSVD (0.0265) and ALS (0.0219) rank below even Popularity. The reason is **objective mismatch**: they optimize MSE on rating prediction, but we measure NDCG on ranking. A model that predicts "you'd rate this 3.8" for every movie produces low RMSE but useless rankings.

Basic CB is the worst model. NDCG@10 = 0.0087. Binary genre vectors are too coarse — many movies share identical genre vectors, making personalization near-random. However, it achieves 100.2% coverage — slightly above 100% — because CB scores items using only genre metadata, not interaction history. This means it can recommend movies that appear in validation or test data but have zero training ratings. Those items are not counted in the training catalog denominator, so the ratio of unique recommended items to catalog size can marginally exceed 1.0.

The accuracy–coverage trade-off

This is the most important structural finding:

Model	NDCG@10	Coverage
Item-Item CF	0.0625	19.7%
ALS	0.0219	84.5%
Basic CB	0.0087	100.2%

High accuracy models (Item-Item CF) collapse into a narrow set of popular items — only 730 out of 3,706 movies ever get recommended. High coverage models (ALS, Basic CB) spread recommendations across the catalog but sacrifice ranking quality. No single model wins both dimensions.

4. Failure Analysis

Model	Primary Failure	Root Cause
Item-Item CF	Cold-start items	Cannot compute similarity for items with <20 ratings (19% of catalog). Needs co-interaction data.
Content-Based	Filter bubble	Only recommends movies similar to what the user already watched. A Sci-Fi fan will never see a Drama recommendation, even a great one.
FunkSVD	Users with mixed tastes	A user who likes both arthouse cinema and action blockbusters doesn't fit neatly into 30 latent dimensions. The model averages their preferences, losing the signal for both.
ALS	Misleading offline results	Recommends many lesser-known items that users may have enjoyed but didn't rate in the test set. Offline metrics count this as error, but in a live system these could be good recommendations.
Popularity	Zero personalization	Completely fails for niche users. Everyone gets the same list regardless of their taste.

5. Bias Analysis

Popularity Bias

The Popularity baseline (NDCG@10 = 0.049) is a shockingly strong competitor. This highlights the severity of popularity bias in the dataset: the power-law distribution means popular movies dominate test sets too.

Item-Item CF has the highest popularity bias among Part 1 models (1,681) because popular movies have more co-interactions, inflating their similarity scores. This is why it achieves high accuracy — it recommends movies that are both relevant and popular — but at the cost of 19.7% coverage.

Enhanced CB required a popularity_weight of 0.5 to be competitive. Without it, pure content features underperform. This means half of the Enhanced CB signal is just popularity in disguise.

Activity Bias

Per-user averaging in our evaluation prevents power users from dominating metric computation. However, during training, Matrix Factorization (FunkSVD, ALS) inherently over-represents power users: they contribute more terms to the loss function, so the learned item factors Q primarily satisfy power user preferences.

6. Design Choices & Struggles

Jaccard vs. Adjusted Cosine: We initially expected Adjusted Cosine to win because it uses rating magnitudes. In practice, Jaccard with binary threshold 4.0 was better. Users are inconsistent raters — a 3 from one user means the same as a 4 from another. Binarizing removes this noise.

MF initialization: Both FunkSVD and ALS initialize factors from N(0, 0.1). We tested N(0, 0.01) but it converged too slowly. Biases start at 0; the global bias is set to the mean training rating (≈3.58).

Why TF-IDF helped CB: Raw binary genre vectors produce near-identical feature vectors for many movies. TF-IDF re-weights: rare genres (Film-Noir, Documentary) become more discriminative, common genres (Drama, Comedy) get downweighted. Adding a decade feature further helped differentiate movies within the same genre.

Numba acceleration: FunkSVD and ALS without Numba were impractically slow on 1M interactions. JIT compilation brought FunkSVD from minutes to seconds per epoch. ALS's closed-form solve is fast per-user but iterates over 6,040 users and 3,706 items each iteration — still manageable but not instant.

7. Deployment Strategy (Part 1 Only)

Based on Part 1 results alone, we would deploy a **Hybrid Switching Strategy**:

1. **Primary: Item-Item CF.** Best accuracy. Precompute top-50 neighbors per item offline. At inference time: look up neighbors of the user's liked items, aggregate scores, return top-K. Sub-millisecond.
2. **Cold-start fallback: Enhanced CB.** For new movies or items with <50 ratings, where CF similarity is unreliable. CB can score any item with metadata immediately.
3. **Diversity slot: ALS.** Insert 1–2 ALS recommendations into the final list to promote discovery of lesser-known movies. Offline metrics punish this, but in a live system it drives valuable catalog exploration.

This strategy is superseded by the Part 2 deployment recommendation (Two-Tower + Hybrid Reranking) once BPR and neural models are available.

8. Next Steps (Addressed in Part 2)

- **Objective alignment:** Replace MSE with a pairwise ranking loss (BPR) to directly optimize what we measure.
- **Hybrid models:** Combine collaborative and content signals to address cold-start without sacrificing accuracy.
- **Neural architectures:** Explore whether deep learning can improve on linear factorization.
- **Online evaluation:** Move beyond static offline metrics to simulate live traffic routing with bandits.