Report: Recommender Systems — Part 2

Building on Part 1, Part 2 addresses the core limitations identified there: objective mismatch (MF models optimize MSE, not ranking quality), cold-start (CF models need interaction history), and narrow coverage (accurate models collapse into a small subset of popular items). We implement ranking heuristics, pairwise BPR, hybrid architectures, neural models, and bandit-based online evaluation.

All models share the same evaluation protocol from Part 1: per-user temporal split 80/10/10, relevance threshold ≥ 4.0, NDCG@K as primary metric, K ∈ {5, 10, 20}.

1. Ranking Heuristics

Three non-learned baselines that require no training.

1.1 Popularity Ranker

Counts all interactions per item (`train_df.groupby('item_id').size()`). Every rating counts — a 1-star and a 5-star rating contribute equally. The item's score is simply "how many people rated this movie." Minimum threshold: `min_ratings=5` .

1.2 Recency Ranker

Each interaction gets a time-decay weight: `exp(−ln2 · days_ago / half_life)` with `half_life=90` days. A rating from yesterday weighs ≈1.0; a rating from 180 days ago weighs ≈0.25. The actual rating value (1–5) is ignored — only the timestamp matters. The item's score is the sum of all its time-decayed weights. On MovieLens with only a 3-year temporal span, recency added little over raw popularity.

1.3 Personalized PageRank (PPR)

Builds a bipartite graph: nodes = all users + all items, edges = "user u liked item i" (rating ≥ 4.0). Row-normalizes the adjacency matrix to get a transition matrix (each node distributes 1/degree to each neighbor). Then runs power iteration for 20 steps with teleport probability α=0.15:

```
scores = (1 − α) · M^T · scores + α · teleport
```

where `teleport` is 1.0 at the target user and 0.0 elsewhere. The result: every item reachable through any path from the user gets a score — even items the user never interacted with, reached through multi-hop chains (user→movie→user→movie).

Heuristic Results

Model	NDCG@5	NDCG@10	NDCG@20	Precision@10	Recall@10	Coverage	Pop. Bias
PPR	**0.0475**	**0.0544**	**0.0667**	**0.0428**	**0.0497**	5.8%	2053
Popularity	0.0437	0.0490	0.0615	0.0393	0.0433	5.1%	2086
Recency	0.0380	0.0438	0.0547	0.0353	0.0410	4.4%	1797

PPR is the best heuristic — it captures indirect user-item connections through the graph (e.g., "users who liked the same movies as you also liked this"). But all heuristics have extremely low coverage (<6%) because they collapse into the popular, well-connected core.

2. BPR-MF: Pairwise Learning-to-Rank

The core idea

BPR uses the same latent factor architecture as FunkSVD (user vectors P, item vectors Q, biases) but replaces the MSE loss with a pairwise ranking loss: for each training step, sample a user, one of their liked items (positive), and a random item they haven't interacted with (negative), then push `score(user, positive) > score(user, negative)`.

The loss per sample: `−log σ(x_ui − x_uj)` where σ is the sigmoid function.

Implementation details

- **Architecture:** n_factors=64, user_bias + item_bias
- **Initialization:** user_factors and item_factors from N(0, 0.01), biases at 0
- **Training:** lr=0.01, reg=0.001, 40 epochs
- **Negative sampling:** n_samples = len(positive_interactions) × 5 per epoch. Each sample: pick random user → pick one of their positives → sample one random negative.
- **Acceleration:** Full training loop compiled via Numba `@njit(fastmath=True)`
- **Scoring:** `score_all_items(u) = item_bias + item_factors @ user_factors[u]` — a single matrix-vector multiply producing 3,706 scores

BPR vs. Baselines

Model	NDCG@5	NDCG@10	NDCG@20	Precision@10	Recall@10	Coverage	Pop. Bias
BPR-MF	**0.0567**	**0.0664**	**0.0829**	**0.0466**	**0.0704**	**45.9%**	1095
PPR	0.0475	0.0544	0.0667	0.0428	0.0497	5.8%	2053
Popularity	0.0437	0.0490	0.0615	0.0393	0.0433	5.1%	2086

BPR outperforms all heuristic baselines and all Part 1 models (best Part 1: Item-Item CF at NDCG@10 = 0.0625) while dramatically improving coverage. Compared to FunkSVD (same architecture, MSE loss): NDCG@10 jumps from 0.0265 → 0.0664 — a **+150% improvement** from switching the loss function alone. This is the single most impactful design choice in the project.

Negative Sampling Sensitivity

Multiplier	Samples/epoch	NDCG@10	Recall@10	Coverage
1×	469,223	0.0502	0.0449	2.9%
5×	**2,346,115**	**0.0661**	**0.0700**	**34.2%**
10×	4,692,230	0.0651	0.0698	42.4%
20×	9,384,460	0.0646	0.0697	46.9%

At 1× the model sees too little contrast — each positive gets roughly one gradient update per epoch. At 5× each positive is sampled ~5 times (paired with different random negatives), giving rich contrast. Beyond 5× accuracy slightly degrades while coverage keeps increasing — the model learns to distribute scores across more items but loses some precision at the top.

Head / Mid / Tail Analysis

Items split by popularity quantiles: head = top 10% (≥ q90), mid = 50th–90th percentile, tail = bottom 50%.

Model	Head NDCG@10	Mid NDCG@10	Tail NDCG@10
BPR-MF	**0.0884**	**0.0238**	**0.0012**
PPR	0.0737	0.0	0.0
Popularity	0.0662	0.0	0.0

Popularity and PPR recommend exclusively head items — literally 0.0 on mid and tail. BPR is the only model that can surface mid-tier and tail items at all. The tail number (0.0012) is tiny but it is nonzero: BPR occasionally recommends a niche movie that happens to be in the user's test set. This happens because negative sampling forces the model to learn embeddings for tail items — their positions in latent space are shaped by the gradient updates even though few users interacted with them directly.

3. Hybrid Recommender

Motivation

BPR struggles with cold-start (users/items with few interactions). Content-Based can score any item with metadata but misses collaborative signal. The hybrid combines both.

Two strategies implemented

Weighted Blending: Score all 3,706 items with both models. Min-max normalize each score array to [0, 1] (necessary because BPR scores are unbounded while CB scores are bounded). Compute `final = α · BPR_norm + (1−α) · CB_norm`. Return top-K.

Candidate Generation + Reranking: BPR retrieves top-100 candidates. For each candidate: `combined = 0.6 · bpr_score + 0.4 · cb_score` (raw scores, not normalized). Return top-K from the reranked list. This is more computationally efficient — CB only scores 100 items instead of 3,706.

Alpha Tuning (Weighted Blending, cold users only)

α (BPR weight)	NDCG@10	Recall@10
0.3	0.0642	0.0750
0.5	0.0720	0.0840
0.7	0.0785	0.0914
0.8	**0.0811**	**0.0938**
0.9	0.0805	0.0922

α=0.8 is optimal — 80% BPR, 20% CB. Below 0.8 the content signal dilutes the collaborative signal that BPR already provides. Above 0.8 there's not enough content fallback for cold items.

Hybrid Results (Full Test Set)

Model	NDCG@10	Precision@10	Recall@10	Coverage	Pop. Bias
Blend (α=0.8)	**0.0657**	**0.0452**	**0.0708**	**48.1%**	1058

Model	NDCG@10	Precision@10	Recall@10	Coverage	Pop. Bias
CandGen+Rerank	0.0655	0.0455	0.0701	47.2%	1067
BPR-MF (pure)	0.0649	0.0455	0.0689	45.8%	1094
Content-Based	0.0359	0.0224	0.0382	55.0%	734

Blend and CandGen+Rerank are nearly identical (0.002 difference = noise). The CandGen+Rerank architecture's value is not empirical on this dataset — it's architectural: scalability (CB scores 100 items instead of 3,706) and modularity (swappable retrieval and reranking stages).

Cold vs. Warm User Segmentation

Model	Cold NDCG@10 (≤30 ratings)	Warm NDCG@10 (≥100 ratings)
Blend (α=0.8)	**0.0806**	0.0668
CandGen+Rerank	0.0769	0.0693
BPR-MF	0.0730	**0.0707**
Content-Based	0.0478	0.0312

For cold users the Hybrid boosts NDCG by +10% over pure BPR. Content features fill in when interaction history is sparse. For warm users pure BPR is actually better — when the collaborative signal is strong, injecting generic content features adds noise.

4. Deep Learning Models

4.1 NeuMF (Neural Collaborative Filtering)

Combines a GMF path (element-wise product of user/item embeddings) and an MLP path (concat, then feed-forward). Four embedding tables: gmf_user, gmf_item, mlp_user, mlp_item — all emb_dim=32, initialized from N(0, 0.01). MLP dims: [64, 32] with ReLU and Dropout(0.1). Linear layers Xavier-initialized. Output: `concat(GMF 32-dim, MLP 32-dim) → Linear(64, 1)`.

Training: BPR pairwise loss (same as BPR-MF for fair comparison), Adam optimizer, lr=0.001, weight_decay=1e-5, batch_size=2048, 30 epochs.

Inference limitation: NeuMF mixes user and item data inside the network — to score one user against all items requires 3,706 separate forward passes. This makes it impractical for production.

4.2 Two-Tower

Separate user and item encoder towers:

- **User tower:** Embedding(32) → Linear(32→64) → ReLU → Dropout(0.1) → Linear(64→64) → 64-dim output
- **Item tower:** [Embedding(32) ; TF-IDF features(~30-dim)] → concat → Linear(~62→64) → ReLU → Dropout(0.1) → Linear(64→64) → 64-dim output
- **Score:** dot product of the two 64-dim vectors

Same BPR pairwise loss, same optimizer settings as NeuMF.

Key advantage: The item tower takes content features (TF-IDF genre + decade) alongside the learned embedding. At inference, all 3,706 item representations are precomputed through the item tower once and cached. Per-user inference reduces to one user tower forward pass + a matrix-vector dot product — compatible with ANN search (FAISS) for fast serving.

Deep Learning Results

Model	NDCG@5	NDCG@10	NDCG@20	Precision@10	Recall@10	Coverage	Pop. Bias
BPR-MF (baseline)	0.0571	0.0657	0.0828	0.0457	0.0692	45.4%	1099
Two-Tower	0.0553	0.0648	0.0800	0.0477	0.0666	54.1%	1199
NeuMF	0.0529	0.0619	0.0765	0.0447	0.0622	56.8%	1227

Both neural models slightly underperform the linear BPR-MF. Neural networks need large amounts of data to learn well: on MovieLens 1M with purely structural (ID-based) interactions, the extra MLP parameters learn noise rather than useful patterns. A well-regularized linear dot product generalizes better on a dataset of this size.

Two-Tower beats NeuMF because its item tower incorporates TF-IDF content features — a richer item representation. Both neural models achieve higher coverage (54–57%) than BPR-MF (45.4%), suggesting the MLP spreads scores across a wider range of items.

5. Online Evaluation with Multi-Armed Bandits

Setup

Offline metrics evaluate models on frozen historical data — they cannot capture exploration value or feedback loops. We simulated online evaluation using multi-armed bandits.

Arms: 4 static recommendation policies — Popularity, BPR-MF, Hybrid, Random. Each arm is a function: `user_id → top-K recommendations`.

Reward: Precision@K — `hits / K` where a hit is a recommended item that appears in the user's test set with rating ≥ 4.0. This is a continuous value in {0, 0.1, 0.2, ..., 1.0}, not strictly binary.

Bandit strategies: ε-greedy (ε=0.1, ε=0.3), UCB1, Thompson Sampling. For Thompson Sampling, the internal update is binarized (`reward > 0 → success`), but the reported metrics use the raw precision values.

Simulation: 6,040 users arrive sequentially in random order. At each step, the bandit picks an arm, the arm generates top-10 recommendations, the reward is computed from the test set, and the bandit updates its beliefs.

Bandit Results

Strategy	Avg Reward	Total Reward
Static: Hybrid	0.0487	293.9
Static: BPR-MF	0.0470	283.7
Static: Popularity	0.0393	237.2
Static: Random	0.0022	13.3
Thompson Sampling	**0.0485**	**292.7**
ε-greedy (0.1)	0.0474	286.1
ε-greedy (0.3)	0.0427	258.0
UCB1	0.0406	245.2

Thompson Sampling nearly matches the best static policy (Hybrid: 293.9 vs Thompson: 292.7) without knowing in advance which arm is best. It maintains a Beta(α, β) distribution for each arm — samples a random value from each distribution, then pulls the arm with the highest sample. After a success α += 1, after failure β += 1. Over time, the distributions narrow around the true reward, naturally reducing exploration.

ε-greedy (0.1) permanently wastes 10% of traffic on random exploration — acceptable but suboptimal. At ε=0.3, 30% random exploration wastes too much traffic and drops total reward by 10%.

UCB1 failed. Rewards are small (≈0.04) while UCB1's uncertainty bonus starts at ≈1.0 (`sqrt(2·ln(t)/n)`). The bonus dominates arm selection for thousands of rounds, causing severe over-exploration before the average rewards become distinct enough to pick a winner.

6. Part 2 Summary Table

Model	NDCG@10	Precision@10	Recall@10	Coverage	Pop. Bias	Best For
Hybrid Blend (α=0.8)	**0.0657**	**0.0452**	**0.0708**	48.1%	1058	Cold-start users
BPR-MF	**0.0664**	**0.0466**	**0.0704**	45.9%	1095	Overall ranking
CandGen+Rerank	0.0655	0.0455	0.0701	47.2%	1067	Production architecture
Two-Tower	0.0648	0.0477	0.0666	54.1%	1199	Production retrieval
NeuMF	0.0619	0.0447	0.0622	56.8%	1227	Flexibility
PPR	0.0544	0.0428	0.0497	5.8%	2053	Graph structure
Popularity	0.0490	0.0393	0.0433	5.1%	2086	Baseline
Recency	0.0438	0.0353	0.0410	4.4%	1797	Freshness

7. Unified Results: All Models

Rank	Model	NDCG@10	Precision@10	Recall@10	Coverage	Pop. Bias
1	BPR-MF	0.0664	0.0466	0.0704	45.9%	1095
2	Hybrid Blend (α=0.8)	0.0657	0.0452	0.0708	48.1%	1058
3	CandGen+Rerank	0.0655	0.0455	0.0701	47.2%	1067
4	Two-Tower	0.0648	0.0477	0.0666	54.1%	1199
5	Item-Item CF	0.0625	0.0457	0.0593	19.7%	1681

Rank	Model	NDCG@10	Precision@10	Recall@10	Coverage	Pop. Bias
6	NeuMF	0.0619	0.0447	0.0622	56.8%	1227
7	PPR	0.0544	0.0428	0.0497	5.8%	2053
8	Popularity	0.0490	0.0393	0.0433	5.1%	2086
9	Enhanced CB	0.0460	0.0302	0.0495	40.4%	978
10	Recency	0.0438	0.0353	0.0410	4.4%	1797
11	FunkSVD	0.0265	0.0218	0.0223	16.8%	799
12	ALS	0.0219	0.0173	0.0185	84.5%	470
13	Basic CB	0.0087	0.0064	0.0090	100.2%	237

8. Key Insights

1. **Objective alignment > model complexity.** BPR-MF and FunkSVD are the same architecture. Switching from MSE to pairwise ranking loss: +150% NDCG. This is the single most impactful finding.
2. **The accuracy–coverage trade-off is real.** Item-Item CF: 0.0625 NDCG, 19.7% coverage. ALS: 0.0219 NDCG, 84.5% coverage. No model wins both.
3. **Hybrid helps cold-start, hurts warm users.** Blend (α=0.8) boosts cold-user NDCG by +10% over BPR. But for warm users, BPR alone is better — content features add noise when collaborative signal is strong.
4. **Deep learning ≠ automatic improvement.** On MovieLens 1M, linear BPR-MF beats both NeuMF and Two-Tower. Neural networks need more data (or richer features) to justify their extra parameters.
5. **Offline metrics are insufficient.** Thompson Sampling matched the best static policy without prior knowledge. The exploration it performs creates value invisible to offline evaluation.

9. Problems & Struggles

Objective mismatch (FunkSVD/ALS): Took real debugging time to understand why MF models had excellent RMSE but terrible NDCG. Not a bug — a fundamental misalignment between training objective and evaluation metric.

BPR loss sign: The BPR log-likelihood is negative and increases toward 0 as the model improves. Our training plots initially looked like divergence (loss going "up"). In the PyTorch models (NeuMF, Two-Tower) we negated the loss to get a standard decreasing curve.

Score normalization in hybrid: BPR scores are unbounded; CB scores are in [0, 1]. Blending without normalization produces nonsensical results. We used min-max normalization, which works but is sensitive to outliers. The CandGen+Rerank approach sidesteps this by using raw BPR scores with a lower weight (0.6) — it works because within the top-100 BPR candidates, the score range is narrower.

UCB1 failure: Specific to our reward scale. Rewards ≈ 0.04, but the UCB1 bonus `sqrt(2·ln(t)/n)` starts at ≈1.0, causing thousands of rounds of over-exploration before the average rewards become distinct enough to pick a winner. A reward-scaled variant or different confidence parameter would fix this.

Computational cost: BPR at 20× sampling = 9.4M samples/epoch. Without Numba JIT, each epoch would take minutes instead of seconds. Neural models required GPU. Item-Item CF similarity is O(n_items^2) — several minutes on 3,706 items.

10. Deployment Recommendation

Two-stage architecture:

1. **Retrieval: Two-Tower.** Precompute all 3,706 item representations through the item tower. Index them in FAISS. Per-user inference: one forward pass through the user tower + ANN search → top-100 candidates. Content features in the item tower help with cold-start items.
2. **Reranking: Lightweight Hybrid.** Rerank the 100 candidates using `α · retrieval_score + (1−α) · business_signals`. Business signals include: CB genre match, recency boost, diversity constraint, already-seen genre demotion.

Post-deployment — two stages:

- **Stage 1 — A/B test:** Validate the new pipeline against the Item-Item CF baseline. Primary metric: CTR on top-5 recommendations. Guardrail: Watch Time (to avoid optimizing for clickbait). Requires statistical significance before rollout.
- **Stage 2 — Thompson Sampling:** Once validated, use a bandit to dynamically route traffic between model versions as they evolve. No need to rerun A/B tests for every model update — the bandit converges to the best option automatically.