title: From Neighborhoods to Neural Rankers—Designing Recommenders for Discovery
theme: technology
subtopic: recommender-systems
keywords: [ai, retrieval, embeddings, vector-stores, mmr, cosine]
approx_word_count: 880
suggested_sources:

* Wikipedia: Recommender system
* News/Report: Netflix Tech Blog — “The Netflix Recommender System: Algorithms, Business Value, and Innovation”

From Neighborhoods to Neural Rankers—Designing Recommenders for Discovery

Overview

Recommender systems power feeds, playlists, and product carousels. Their job is not simply to predict clicks but to **shape what users discover**, balancing accuracy with diversity, novelty, and fairness. This briefing traces the path from classic collaborative filtering to **deep ranking models**, and explains practical levers—like **MMR re-ranking**—that navigate the diversity/serendipity trade-off.

The Classic Foundations

Memory-Based Collaborative Filtering

Early systems relied on **user–user** or **item–item** similarity. Given a sparse matrix of interactions, you compute neighbors using **cosine** similarity and recommend items favored by close neighbors. Strengths: simplicity and explainability (“people like you watched…”). Weaknesses: cold start and limited generalization.

Matrix Factorization

Factorization maps users and items into a **latent space** where inner products approximate preferences. It compresses the matrix and reveals hidden structure (“action + sci-fi” as a dimension). With implicit feedback (views, dwell), weighted variants perform well at scale. Still, they struggle with context (time, device, season) and content cold start.

Modern Pipelines

Large platforms usually split the problem into:

1. **Candidate Generation** (fast, recall-oriented): retrieve hundreds to thousands of candidates using vector **embeddings** (e.g., two-tower models placing users and items in the same space), approximate nearest neighbor over **vector-stores**, and classical text search for keywords.
2. **Ranking** (slow, precision-oriented): a learned model (GBDT or deep) scores candidates with features like recency, popularity, personalization signals, and **retrieval features** (e.g., similarity scores).
3. **Re-ranking** (layout-aware): enforce constraints—diversity, freshness, business rules—and smooth the sequence (e.g., avoid five near-duplicates in a row).

Two-tower candidate generators deliver **sub-10 ms** retrieval using ANN indexes and shard-friendly architectures. Rankers add richer features but must keep **p95 inference** to tens of milliseconds to stay interactive.

Neural Ranking and Representation

Text, images, and audio features are now embedded using multimodal encoders. Examples:

* **Text**: transformer encoders produce semantic vectors that handle synonyms (“couch/sofa”).
* **Vision**: CNN/ViT encoders capture style and color for fashion similarity.
* **Audio**: spectral embeddings capture tempo and mood for music.

Training schemes mix **contrastive learning** (push clicked item close to user embedding) with **hard negative mining** (items viewed but skipped). Rankers often use **listwise** losses to model the whole slate, not just pairs.

Diversity vs. Serendipity vs. Relevance

Why “More of the Same” Fails

Pure accuracy drives the system toward filter bubbles and boredom. Users value **novelty** (new artists), **coverage** (long-tail items), and **serendipity** (pleasant surprises). Concrete metrics:

* **Intra-list diversity (ILD)**: average pairwise distance within a slate.
* **Coverage**: fraction of catalog recommended over a window.
* **Novelty@k**: penalize overexposure of popular items.
* **Calibrated relevance**: match topical proportions to the user’s historical mix.

Practical Levers

* **MMR (Maximal Marginal Relevance)**: greedily build the list by combining relevance with dissimilarity to already chosen items. A single λ parameter trades off precision vs. diversity.
* **Category Quotas**: e.g., at most two items per brand.
* **Fairness Constraints**: ensure exposure for underrepresented creators or sellers.
* **Exploration**: contextual bandits or ε-greedy inject low-risk trials to learn new tastes.

A common pattern: rank for relevance, then apply **MMR** or submodular maximization to increase ILD by **10–30%** with minimal CTR loss, and recover the small loss by improved **long-term retention**.

Handling Cold Start

* **Item cold start**: content embeddings (text/images) and **zero-shot** similarity to existing items.
* **User cold start**: short onboarding quizzes, inferred cohorts (“new parents”), and **popularity-boosted** defaults.
* **Marketplaces**: seller quality features (shipping speed, return rate) provide robust priors.

Retrieval and Vector Infrastructure

ANN libraries (HNSW, IVF-PQ, ScaNN) power candidate recall. Key knobs:

* **Dimensionality**: 128–768 is common; higher dims increase recall but cost memory.
* **Quantization**: product quantization cuts RAM 4–16× with small accuracy loss.
* **Hybrid retrieval**: union of lexical (BM25) and vector results improves robustness to

typos and rare terms.

Evaluation Beyond CTR

Short-term metrics can mislead. Mature programs include:

* **User-centric**: time to first satisfying view, dwell on new categories, skip rates.
* **Catalog-centric**: creator exposure fairness, tail coverage.
* **Causal**: interleaving, bandit-off-policy estimators, and long-horizon retention experiments.

Guardrails and Ethics

* Limit runaway reinforcement of sensitive attributes; monitor for disparate impact.
* Provide explanations (“recommended because you liked…”).
* Allow explicit controls (mute, “less like this”), feeding signals back into embeddings.

Key Takeaways

* Candidate generation retrieves; ranking personalizes; re-ranking shapes the final slate.
* Embeddings and ANN over vector stores underpin sub-10 ms retrieval at scale.
* MMR and simple constraints can raise diversity without tanking relevance.
* Balance short-term CTR with long-term engagement, fairness, and catalog health.
* Cold-start is manageable with multimodal content features and light exploration.