Report: Recommender Systems Analysis

1. Performance Comparison & Methodology Analysis

We implemented and evaluated three distinct families of recommender systems: Content-Based Filtering, Memory-Based Collaborative Filtering, and Model-Based Matrix Factorization.

Why Item-Item CF Outperformed Others

The **Item-Item Collaborative Filtering (Jaccard)** emerged as the clear winner in terms of ranking accuracy (NDCG@10 = 0.0625). This dominance can be attributed to several factors specific to the MovieLens dataset:

1. **Nature of Interaction:** The Jaccard similarity, when applied to a binary threshold (Rating ≥ 4.0), effectively captured the "co-liking" signal. It turns out that knowing *whether* two items are liked by the same people is a more robust predictor for ranking than estimating *how much* they are liked (which Adjusted Cosine and FunkSVD attempt to do).
2. **Stability of Item Correlations:** In movie domains, item correlations (e.g., people who like "Star Wars" also like "Empire Strikes Back") are stable and strong. Item-Item CF directly exploits these explicit graphs.
3. **Objective Mismatch in MF:** Matrix Factorization models (FunkSVD) optimized for RMSE (rating prediction error). A model can have a low RMSE (predicting a 3.8 when the truth is 4.0) but still fail at ranking (ordering items correctly in a top-10 list). Item-Item CF was tuned specifically for top-K retrieval.

The Trade-off: Accuracy vs. Coverage

A distinct trade-off emerged between accuracy and catalog coverage:

- **Item-Item CF:** High Accuracy, Low Coverage (12%). It tends to reinforce satisfying, safe recommendations.
- **ALS (Matrix Estimation):** Low Accuracy, High Coverage (85%). It effectively spreads probability density across the "long tail," but often recommends obscure items that users in the test set didn't interact with, leading to "false negatives" in offline evaluation (the user might have liked it, but we don't know).

2. Failure Analysis

Model Family	Primary Failure Scenario	Reason
Collaborative Filtering (Item-Item)	**Cold-Start Items**	Cannot recommend an item until it has been rated by other users. It is blind to 19% of the catalog (items with <20 ratings).
Content-Based Filtering	**The "Filter Bubble"**	Fails to provide serendipity. If a user only watched interaction-heavy Sci-Fi, it will never recommend a great Drama, even if the user would love it. It lacks the "wisdom of the crowd."
Matrix Factorization (FunkSVD)	**The "Gray Sheep"**	Struggles with users whose tastes are eclectic or inconsistent with the dominant latent factors. It forces users into a fixed-dimensional space which may not capture unique idiosyncrasies.
Popularity Baseline	**Niche Users**	Completely fails for users with specific, non-mainstream tastes (e.g., horror cult classics), offering zero personalization.

3. Bias Analysis

Popularity Bias

The dataset follows a strict Power Law distribution.

- **The Findings:** The popularity baseline alone achieved an NDCG@10 of 0.022. This is a very strong baseline. Any Personalized model must fight the urge to simply mimic this baseline.
- **Model Behavior:** The Enhanced Content-Based model required a massive popularity weight (0.5) to be competitive. This indicates that for this dataset, "quality" (popularity) is often a better feature than "genre". Item-Item CF naturally biases towards popular items because popular items have more co-occurrences, leading to higher similarity scores. This explains its high accuracy but low coverage.

Activity Bias

- **Power Users:** The evaluation metric (average across users) prevents power users (who have rated 1000+ movies) from dominating the scores. However, the *training* process of Matrix Factorization is inherently biased towards power users, as they contribute more terms to the loss function. This means the latent factors Q_i (items) are learned primarily to satisfy power users, potentially marginalizing casual users.

4. Deployment Strategy

If we were to deploy this system into a production environment today, we would not choose a single model. Instead, we would implement a **Hybrid Switching Strategy**:

The Architecture

1. **Primary Candidate Generator: Item-Item CF**
   - **Why:** Best accuracy. Users trust systems that give "right" answers.
   - **Implementation:** Pre-compute the Top-50 neighbors for every item offline. At inference time, this becomes a fast O(1) lookup.
2. **Fallback / Cold-Start Handler: Content-Based Filtering**
   - **Scenario:** When a new movie is added or for items with <50 ratings.
   - **Why:** CF cannot handle these. CB uses metadata (genres, year, director) to place the item near similar items immediately.
   - **Threshold Rationale:**
     - **< 20 ratings (19% of catalog):** This is the statistical noise floor identified in EDA. With fewer than 20 ratings, calculating reliable item-item similarity is mathematically unstable (high noise/variance).
     - **20–50 ratings:** Our hyperparameter tuning found $K = 50$ to be the optimal neighbor count. Items in this range are in a "warm start" phase—they exist but lack enough history to support a full 50-neighbor calculation, making the Content-Based fallback a safer choice.
3. **Experimental Slot: ALS**
   - **Scenario:** Insert 1 or 2 items from the ALS model into the Top-10 list.
   - **Why:** To inject diversity and "long tail" discovery. offline metrics punish this, but in an online setting, this drives catalog exploration.

Why not pure Matrix Factorization?

While FunkSVD is good in theory, its "Rating Prediction" nature is less suited for a "Top-N Recommendation" user interface (like Netflix/Spotify). Users care about "What should I watch?", not "What will be my rating?". Item-Item CF aligns better with this mental model.

5. Thoughts & Considerations for Next Steps

1. Implicit vs. Explicit Feedback

We treated 4.0+ ratings as "positive" and others as negative/noise. In a real-world scenario, we would also incorporate **Implicit Feedback** (clicks, watch time) into the model. As we assume - a user clicking "play" is a stronger signal than a user *not* rating a movie.

2. Context-Awareness

Our current models are static. They ignore:

- **Time:** A user watching cartoons on Saturday morning might be doing so for their kids, while watching crime fiction on Friday night is for themselves.
- **Sequence:** The user just watched "The Fellowship of the Ring". Recommending "The Two Towers" is obvious, but standard MF might not catch this sequential dependency as strongly as a Sequence-Aware Recommender.

3. Beyond Genres

The Content-Based model was weak because "Genre" is a weak feature. Movies are defined by Directors, Actors, Plot keywords, and Visual style.

- **Next Step:** Use NLP (BERT/LLMs) on movie plot summaries to generate rich item embeddings, rather than just using TF-IDF on genres.

4. Online Evaluation

Offline metrics (NDCG) are proxies. High accuracy on historic data does not guarantee user engagement. The only true test is an **A/B Test** to see if users actually click and watch the recommendations provided by the diverse ALS model versus the safe Item-Item model.