**LLM-Based Movie Recommendation Systems**

Practical Work Report

Author: Tamas Scheidl Date: 09.06.2025

# Introduction

This practical work investigates the application of large language models (LLMs) for movie recommendation tasks using the MovieLens 1M dataset. The goal is to explore a range of prompting strategies and ranking methods that leverage semantic reasoning, surprise, diversity, and genre alignment, while balancing relevance and novelty.

The system builds on and adapts techniques from Delbar et al. (2024) [1] and extends the GitHub implementation Benchmark\_RecLLM\_Fairness. The project is implemented using Python and evaluated on a local Apple Silicon environment (MPS backend), using a Hugging Face implementation of the Phi-3.5-mini-instruct model.

# Model Configuration

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| **LLM** | microsoft/Phi-3.5-mini-instruct |
| **Architecture** | Decoder-based (small LLM, ~1.8B params) |
| **Platform** | Local inference with Hugging Face |
| **Device** | Apple Mac (MPS backend) |

# Dataset

* Dataset: MovieLens 1M
* User-Movie Interactions: 1M ratings (userId, itemId, rating, timestamp)
* Metadata:
  + df\_users\_ml: demographics o df\_items\_ml: movie titles, genres
* Evaluation:
  + 10 users × 120 candidate movies per user
  + 5 movies as input history, remainder as ground truth

# Methodology

Each model (S1–S7) is implemented as a separate Python script located in the scripts/ directory. All models take as input a short history of previously watched movies and output a top-10 ranked list of recommendations. Depending on the strategy, candidates are filtered by genre, popularity, or model scores.

The recommendations are then evaluated using both standard accuracy metrics and diversity indicators. Additional visualizations are produced via Plot.ipynb to support interpretation.

# Model Architectures

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Strategy Description** |
| S1 | Simple | Input history → predict top 10 |
| S2 | Genre-Focused | Match user genres |
| S3 | Genre-Contrast Diversity | Select unseen or unusual genres |
| S4 | Quality + Novelty | Recommend lesser-known high-quality films |
| S5 | Surprise | High deviation from history to induce surprise |
| S6 | Motivate Reasoning | Thematically motivated recommendations |
| S7 | Chain-of-Thought  Reasoning(COT) | Step-by-step reasoning + thematic alignment |

LLMs are used in S6 and S7 via prompt-based generation using Hugging Face Transformers with Phi-3.5-mini-instruct.

S1 Simple model

**Overview**

The **S1 model** is a zero-shot movie recommender that uses the microsoft/Phi-3.5-mini-instruct large language model (LLM) to generate recommendations directly from a user’s previously watched movie titles. No fine-tuning is performed; instead, recommendations are generated via carefully designed prompts. The architecture emphasizes simplicity and leverages the LLM's language understanding capabilities to infer preferences.

**Data Handling**

* **User and item metadata** (df\_users\_ml, df\_items\_ml) and interaction logs (test\_data) are loaded from preprocessed CSV files.
* The system evaluates the model on 10 users with at least 6 known movie interactions.
* Each user's movie history is split:
  + The first 3 titles are used as the input context.
  + The remaining are used as ground truth for evaluation.

**Model Initialization**

* The Phi-3.5 model and its tokenizer are loaded using the transformers library.
* The model runs on mps (Apple Silicon) or falls back to cuda/cpu based on availability.
* The model is forced to re-download (force\_download=True) to avoid caching issues.

**Recommendation Generation Logic**

The function get\_recommendations(user\_movies\_string, candidate\_movies) creates a structured prompt:

text

CopyEdit

Based on these movies: [user history], recommend 10 movies...

Choose only from the following list: [candidate pool]...

Return only the movie titles, comma-separated, with no extra text.

* This prompt is tokenized and passed to the Phi-3.5 model.
* The output is decoded and parsed to extract titles that match the candidate pool.
* Only exact matches are retained to maintain evaluation accuracy.

**Candidate Pool Construction**

* 120 titles are sampled from the full movie list excluding the user’s known items.
* These are concatenated with the 3 input movies to create a complete set the model can choose from.

**Evaluation Metrics**

The model is evaluated on a variety of standard recommendation metrics:

* **Hit Rate @3**: Measures if at least one ground truth item appears in top 3 recommendations.
* **Average Rank**: Computes the mean rank of ground truth items in the output list.
* **Recall@5 / NDCG@5**: Focused relevance-based metrics at cut-off 5.
* **HHI / Entropy / Gini**: Diversity-focused metrics analyzing distribution of recommendations across users.

Each user’s metrics are stored in a DataFrame and saved to results/metrics\_s1.csv.

**Summary of Evaluation**

The S1 model serves as a **baseline architecture** that leverages a general-purpose LLM to perform recommendation via textual inference. Its key advantages include:

* No training required.
* Flexible and adaptable to different prompt formulations.
* Easy integration with other LLM pipelines.

However, it also has limitations:

* Recommendations can be imprecise due to lack of structure in output parsing.
* No personalization beyond short-term history.
* May repeat popular or generic movies.

S2 Genre-Focused

**Overview**

The **S2 model** is a rule-based recommender that prioritizes strict genre matching between the user’s viewing history and candidate movies. Unlike S1, which uses a language model to infer preferences, S2 uses interpretable logic based on movie metadata (genre and popularity). It ranks candidate movies based on genre overlap and a mild popularity bias.

**Data Handling**

* The model loads interaction logs and movie metadata from CSV files.
* The test set includes 10 users with at least 6 watched movies.
* Each user’s history is split:
  + The first 5 titles form the input set.
  + The remaining are treated as ground truth for evaluation.

**Recommendation Generation Logic**

The core recommendation function is get\_recommendations\_genre\_strict, which follows this logic:

1. **Extract Top Genres**:  
   Identify the most common genres in the user’s 5 input movies.
2. **Filter Candidates**:  
   Select movies from the full item list that share at least one of the top 3 genres and exclude previously seen movies.
3. **Popularity Scoring**:
   * Candidate popularity is calculated from the frequency of interactions in the test dataset.
   * Popularity is normalized and lightly perturbed with Gaussian noise to avoid ties.
4. **Scoring and Ranking**:  
   Each candidate receives a score based on:  
   python  
   CopyEdit  
     
     
   score = genre\_match\_count + 0.1 \* normalized\_popularity
5. where genre\_match\_count is the number of overlapping genres with the user's top genres.
6. **Top-N Selection**:  
   The 10 highest-scoring titles are returned as recommendations.

**Prompt Generation (Optional)**

Although this model is rule-based, a descriptive prompt is also generated internally for each user. This could be used for logging or for passing to a language model in future hybrid models.

Example:

text

CopyEdit

Provide 10 movie recommendations that strictly match the user's favorite genres.

User history includes: [movie titles].

Focus on genre overlap with top genres: [genres].

**Evaluation Metrics**

Per-user metrics include:

* **Hit Rate**: Whether at least one ground truth movie appears in the recommendations.
* **Average Rank**: Average index position of ground truth items in the recommended list.
* **HHI**: Herfindahl-Hirschman Index to measure recommendation concentration.
* **Entropy**: Diversity in the recommendation distribution.
* **Recall@5 & NDCG@5**: Calculated across all users for relevance assessment.
* **Gini Index**: Measures inequality in how often movies appear in recommendations.

All metrics are saved to results/s2\_metrics.csv.

**Summary of Evaluation**

The S2 model serves as an **interpretable, metadata-driven baseline**. It benefits from:

* **High transparency** – every recommendation is explainable via genre overlap.
* **Efficient computation** – no deep model inference is needed.
* **Increased diversity** – boosted by genre filtering and random sampling.

**Limitations**:

* The model cannot capture subtle user preferences (e.g., themes, moods).
* Its performance is bounded by the quality and granularity of genre metadata.
* Over-dependence on popularity can reduce novelty.

S3 Genre-Contrast Diversity

**Overview**

The **S3 model** introduces a diversity-focused recommendation strategy by balancing genre-based relevance with controlled randomness and a light popularity signal. This approach encourages **exploration of new genres** while maintaining a connection to user preferences. Unlike S2's strict genre matching, S3 rewards novelty, using genre preference weights, noise injection, and biasing to break symmetry and promote varied recommendations.

**Data Handling**

* The model loads MovieLens user interaction data and item metadata.
* Each user must have at least 6 rated movies:
  + The first 5 titles form the user’s input history.
  + The remaining titles are used as the ground truth for evaluation.

**Recommendation Generation Logic**

The core recommendation logic is encapsulated in get\_recommendations\_s3\_diversify, which operates in the following steps:

**1. Genre Preference Weighting**

* Each movie in the user history is mapped to its genres.
* For each genre, a cumulative score is calculated based on the user’s ratings.
* This creates a weighted genre profile representing the user's preferences.

**2. Candidate Scoring**

Each candidate movie receives a composite score based on:

python

CopyEdit

score = genre\_score + 0.7 \* popularity + noise + bias

* genre\_score: Sum of the genre weights associated with the movie.
* popularity: Normalized frequency of interactions with the movie across all users.
* noise: Gaussian noise (N(0, 0.5)) to promote diversity and avoid deterministic top results.
* bias: A tiny index-based value to enforce deterministic ordering in case of ties.

**3. Top-K Selection**

* The top 10 candidates based on the composite score are selected.
* If score variance is too low (i.e., all candidates are equally good), additional noise is injected.

**4. Prompt Generation (for future use or logging)**

A natural language prompt is generated to document the user context and recommendation intention:

text

CopyEdit

Provide 10 movie recommendations that explore new genres or themes while remaining interesting to the user.

User's previously watched movies: [input titles].

Aim to balance diversity and relevance.

**Evaluation Metrics**

Each user’s recommendations are evaluated using:

* **Hit Rate**: Measures if at least one ground truth item appears in recommendations.
* **Average Rank**: Average position of ground truth titles in the ranked recommendation list.
* **Recall@5** and **NDCG@5**: Relevance-based top-K metrics.
* **HHI** and **Entropy**: Assess recommendation diversity.
* **Gini Index**: Measures inequality in recommendation scores (interpreted as concentration).

Metrics are aggregated across 10 users to determine the model’s overall performance.

**Summary of Evaluation**

The S3 model demonstrates a **semi-randomized, controlled diversification strategy** that still leverages user preferences but encourages **genre exploration**:

**Strengths:**

* Encourages novelty while maintaining personal relevance.
* Incorporates user preference via genre weights.
* Score noise prevents repetitive recommendations.

**Limitations:**

* Noise and bias can introduce instability in evaluation.
* Genre-based scoring may overlook semantic themes (e.g., tone, mood).
* Ground truth movies may not always reflect user interest in diversity.

S3 is suitable as a **counterpoint to over-personalized or popularity-driven recommenders**, providing more creative, exploration-friendly suggestions.

S4 Quality + Novelty

**Overview**

The **S4 model** is a hybrid recommender designed to promote **diversity and novelty** in suggestions by incorporating **genre preferences**, **popularity penalties**, and **random noise**. It deviates from recommending only mainstream or highly popular items, and instead, it encourages exploration of **lesser-known films** that still align with a user’s taste.

This model is deterministic in structure but introduces stochasticity through a Gaussian noise component, helping it avoid recommendation homogenization.

**Data Setup**

* The model processes MovieLens 1M data, focusing on 10 test users.
* Each user’s historical interaction is split:
  + **First 5 movies** → User history
  + **Remaining** → Ground truth
* Candidate movies are filtered to include only those that **share genres** with the user's top 3 most watched genres and have not been seen already.

**Recommendation Generation Logic**

The recommendation engine is implemented in the function get\_recommendations\_s4\_diversify, which executes the following key steps:

**1. Genre Scoring Based on Ratings**

* From the user’s past interactions, genres are extracted and scored based on the **sum of ratings** the user gave to movies in each genre.
* This results in a **user-specific genre affinity profile**.

**2. Scoring Candidate Movies**

Each candidate movie is scored using:

ini

CopyEdit

score = genre\_score + 0.8 \* popularity\_penalty + random\_noise

Where:

* **genre\_score** is the sum of the user’s affinity scores for the genres in the movie.
* **popularity\_penalty** is -log(1 + normalized\_popularity), which pushes down mainstream items.
* **random\_noise** adds Gaussian noise (mean=0, std=0.2) to support diversity.

A small value idx \* 1e-4 is added to break score ties consistently.

**3. Top-K Selection**

* After scoring, the top 10 highest-scoring items are selected as recommendations.

**Evaluation Metrics**

The model is evaluated using standard quality and diversity metrics:

* **Hit Rate**: Measures whether any ground truth item is in the top recommendations.
* **Average Rank**: Average position of ground truth items in the ranked list.
* **Recall@5** and **NDCG@5**: Assess top-5 effectiveness.
* **HHI (Herfindahl-Hirschman Index)** and **Entropy**: Measure the diversity of recommendations.
* **Gini Index**: Calculated using the score distribution of recommended items to assess inequality.

**Key Characteristics**

* ✅ **Promotes Exploration**: By penalizing popularity and injecting randomness, S4 avoids over-recommending blockbusters.
* ✅ **Still Anchored in User Taste**: Genre scores ensure alignment with user preferences.
* ❌ **Less Precise than Genre-Matching Models**: The noise and penalization can reduce hit rate and precision.
* ✅ **Higher Diversity**: Often yields higher entropy and lower HHI than stricter models like S2.

**Summary**

The S4 model exemplifies a trade-off between **accuracy and novelty**. It is especially useful in settings where recommender fatigue, echo chambers, or lack of discovery are problems. By balancing genre relevance with anti-popularity and stochastic influence, it produces recommendations that are both relevant and **refreshingly diverse**.

S5 Surprise

**Overview**

The **S5 Surprise Model** is a recommender system designed to deliberately **prioritize obscure, unconventional, and unexpected movie recommendations**. It intentionally deviates from traditional top-popular or highly-rated suggestions to encourage **novelty and surprise**.

This model leverages genre familiarity but **heavily penalizes popularity**, ensuring that recommended titles are **less predictable** and more likely to introduce new experiences to the user.

**Design Motivation**

The S5 model is inspired by the need to counter **recommendation fatigue** and avoid **echo chambers** where users are repeatedly shown variations of what they've already seen. By downweighting genre conformity and introducing significant noise, the system aims to **delight through unpredictability**, while still maintaining a loose alignment with the user’s historical preferences.

**Recommendation Strategy**

Recommendations are generated using the get\_recommendations\_s5\_surprise() function, following these key steps:

**1. Genre Scoring**

* The user’s watched and rated movies are used to compute a **genre affinity profile**, where each genre’s score is the sum of the user’s ratings for movies containing that genre.

**2. Candidate Pool Preparation**

* Candidate movies exclude any already seen by the user.
* There is **no genre restriction**, allowing for exploration across all available movies.
* Popularity is calculated and normalized, with a small noise term added to break ties and enhance randomness.

**3. Scoring with Surprise Factors**

Each candidate movie is scored using the following formula:

ini

CopyEdit

score = 0.5 \* genre\_score + (-2.0 \* log(1 + normalized\_popularity)) + strong\_noise

* **Genre score** is present but downweighted.
* **Popularity penalty** is multiplied by -2.0 to strongly suppress mainstream choices.
* **Gaussian noise** (std=0.6) is added to simulate surprise.
* The index-based term idx \* 1e-4 is again used to ensure stable sorting.

**4. Top-K Selection**

* The top 10 highest-scoring movies are selected.

**Prompt Generation (Optional)**

Although the prompt is defined via get\_prompt\_s5\_surprise(), it is not directly used in generation (likely for documentation or comparison with LLM-based versions). The prompt itself emphasizes:

* Avoiding blockbusters.
* Embracing obscure and unconventional films.
* Staying genre-aware but not genre-constrained.

**Evaluation Approach**

The model is evaluated for the **first 10 test users**, using metrics designed to measure both **recommendation accuracy** and **diversity**:

* **Hit Rate** – Checks if the recommendation list contains any correct answers.
* **Average Rank** – Measures the average position of correct recommendations.
* **Recall@5** and **NDCG@5** – Focus on relevance among the top 5.
* **HHI & Entropy** – Gauge diversity and distribution fairness.
* **Gini Index** – Reflects inequality in recommendation exposure across all users.

Additionally, **system-level entropy** is calculated globally to assess overall diversity.

**Summary**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **Goal** | Generate surprising, obscure, and novel movie recommendations |
| **Penalty** | Strong downweighting of popularity |
| **Noise** | High randomness (σ = 0.6) to encourage novelty |
| **Genre usage** | Used softly (0.5 weight), not strictly enforced |
| **Candidate filtering** | Broad, with no genre filter |
| **Strengths** | High diversity, strong surprise potential |
| **Weaknesses** | Potentially lower hit rate and precision due to randomness |

S6 Motivate Reasoning

**Overview**

The **S6 Motivate Reasoning Model** enhances movie recommendations by generating **explanation-aware suggestions**. It not only predicts relevant movies for users but also provides **textual rationales** that clarify *why* each movie was chosen, using a **motivation-focused scoring formula**. This design introduces transparency and interpretability to the recommendation process.

**Design Motivation**

This model addresses the growing demand for **explainable AI** in recommender systems. Rather than treating the model as a black box, S6 uses **interpretable features** like user genre affinity, director familiarity, and even exploration incentives to construct both scores *and* justifications for recommendations.

**Recommendation Strategy**

The function get\_recommendations\_s6\_motivate() is the core of this model. It scores candidate movies based on multiple interpretable signals:

**1. Genre Affinity (60% weight)**

* Based on user ratings across genres.
* Ratings are **mean-centered** to emphasize preferences rather than absolute scores.
* Higher weights are given to genres where the user gave relatively higher ratings.

**2. Director Familiarity (20%)**

* Directors of previously watched films receive a preference boost if repeated in candidate movies.

**3. Genre Match Count (10%)**

* The raw number of genres in the candidate movie that overlap with the user's preferred genres.

**4. Exploration Bonus (10%)**

* Rewards movies containing genres the user rarely watches, promoting **exploration** and **discovery**.

**5. Score Aggregation**

A weighted sum of the above metrics forms the final movie score:

python

CopyEdit

score = 0.6 \* genre\_match\_score + 0.2 \* director\_score + 0.1 \* genre\_match\_count + 0.1 \* exploration\_bonus

**Explainability via Rationales**

Each recommendation is paired with a **natural language explanation**, automatically generated from the user’s history and the scoring components.

Example rationale formats:

* *“'Inception' aligns well with your favorite genres: Sci-Fi (score: 3.2), Thriller (score: 2.7).”*
* *“Based on your preferences in Drama (score: 2.5) and frequent director: Nolan (score: 1.5), we recommend 'Memento'.”*
* *“'Moonlight' is suggested to broaden your movie taste.”* (fallback for low-signal items)

This not only makes the model interpretable but also prepares it for **user-facing recommendation interfaces** or **integration with LLMs**.

**Candidate Pool and Sampling**

* Movies already seen by the user are excluded.
* A subset of up to 120 movies is randomly sampled for scoring to manage performance.
* Popularity normalization is included for potential future extension (though not directly used in scoring here).

**Evaluation Metrics**

The model evaluates recommendation quality using the following metrics over the **first 10 test users**:

|  |  |
| --- | --- |
| **Metric** | **Purpose** |
| Hit Rate | Measures how often ground-truth items appear in the top-3 recommendations |
| Average Rank | Position of correct predictions in the ranked list |
| Recall@5 | Proportion of true positives found in top-5 |
| NDCG@5 | Normalized Discounted Cumulative Gain for top-5 results |
| HHI | Diversity index — low values mean better distribution |
| Entropy | Measures overall recommendation spread and variety |
| Gini Index | Inequality in score distribution, used to analyze fairness |

In addition, **system-level entropy** is calculated to assess overall diversity of the model’s output.

**Summary**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **Goal** | Recommend relevant movies while providing clear, human-readable rationales |
| **Focus** | Interpretability, personalization, exploration |
| **Strengths** | Transparent reasoning, genre/director personalization, novelty support |
| **Weaknesses** | Requires genre and director metadata; slightly more computationally intense |

**Final Thoughts**

The S6 model is well-suited for **interactive recommenders**, education or curation tools, or situations where **user trust and transparency** are critical. It’s a step toward **human-AI collaboration** in content discovery, where users not only get suggestions but also understand the “why” behind them.

S7 Chain-of-Thought Reasoning (COT)

The model leverages a custom function, get\_recommendations\_s7\_cot, that formulates a detailed natural language prompt and parses the LLM's response:

**1. Prompt Construction**

A prompt is dynamically generated for each user:

sql

The user has watched the following movies: [Movie1, Movie2, ..., Movie5].

Please recommend 10 real movie titles that match their taste.

List them clearly like: 1. Title - Reason

**2. LLM Inference**

* The Phi-3 model generates a list of 10 recommendations in free-text format, including brief explanations for each.
* The output is parsed line-by-line to extract movie titles, discarding the numbering and explanations.

**3. Filtering and Validation**

* Only titles that exist in the candidate pool are retained.
* The final recommendation list is limited to 10 validated titles.

**Evaluation Metrics**

Each user’s output is evaluated using multiple quality and diversity metrics:

* **Hit Rate @3**: Checks if the top 3 recommended items contain any ground truth title.
* **Average Rank**: The average position of relevant ground truth items in the recommendation list.
* **Recall@5 / NDCG@5**: Accuracy-focused metrics computed over top-5 recommendations.
* **HHI / Entropy**: Diversity and distribution spread of recommendations.
* **Gini Index**: Measures inequality in exposure across all recommended items using a simulated rank-based attention score.

All metrics are saved to results/s7\_metrics.csv.

**Exposure Modeling**

To approximate user attention, the model assigns **exposure scores** to each recommended item based on its rank:

python

exposure\_score = 1.0 - 0.05 \* rank

These scores are used to compute a system-level **Gini Index**, indicating how equitably items are being recommended across users.

**Summary of Evaluation**

S7 represents the most **explainable and human-aligned** recommender in the system:

* **Strengths**:
  + Every recommendation comes with a textual justification.
  + The LLM can flexibly infer nuanced preferences from user history.
  + Combines personalization and interpretability.
* **Weaknesses**:
  + The output format can be fragile; improper parsing or hallucinated titles may occur.
  + Generative randomness introduces variance in outputs across runs.
  + Quality depends on prompt clarity and model configuration (e.g., temperature, top\_p).

# Evaluation Metrics

The models are evaluated on:

* Hit Rate: whether any ground-truth item appears in top-K
* Average Rank: average position of ground-truth items •Recall@5, NDCG@5: Top-K relevance
* Diversity:

oHHI (Herfindahl-Hirschman Index) oEntropy (user-level & system-level) oGini Index (distribution inequality)

All scores are aggregated across the first 10 users.

## Evaluation Metrics Table

Core Metrics (Averaged over First 10 Users)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Hit Rate** | **Avg. Rank** | **HHI** | **Entropy** | **Gini** |
| **S1** (Simple) | 1.0000 | 3.00 | 0.0104 | 6.6136 | 0.0285 |
| **S2** (Genre-Focused) | 0.8333 | 4.67 | 0.1000 | 3.3219 | 0.2557 |
| **S3** (Diversify + xLSTM) | 0.7667 | 4.00 | 0.1000 | 3.3219 | 0.4427 |
| **S4** (Diversify + Noise) | 0.1000 | 10.20 | 0.1000 | 3.3219 | 0.0483 |
| **S5** (Surprise) | 0.3000 | 9.40 | 0.1000 | 3.3219 | 0.0817 |
| **S6** (Motivate Reasoning) | 0.1000 | 10.60 | 0.1000 | 3.3219 | 0.1924 |
| **S7** (Chain-of-Thought) | 0.1000 | 2.10 | 0.1533 | 0.5907 | 0.0104 |

Recall and NDCG Results

|  |  |  |
| --- | --- | --- |
| **Model** | **Recall@5** | **NDCG@5** |
| **S1** (Simple) | 0.0208 | 0.2346 |
| **S2** (Genre-Focused) | 0.0208 | 0.1461 |
| **S3** (Diversify + xLSTM) | 0.0015 | 0.0301 |
| **S4** (Diversify + Noise) | 0.0008 | 0.0110 |
| **S5** (Surprise) | 0.0079 | 0.0362 |
| **S6** (Motivate Reasoning) | 0.0032 | 0.0073 |
| **S7** (Chain-of-Thought) | 0.0125 | 0.0098 |

System-Level Entropy (Across All Recommendations)

|  |  |
| --- | --- |
| **Model** | **System-Level Entropy** |
| **S5** (Surprise) | 5.2398 |
| **S6** (Motivate Reasoning) | 6.0461 |
| **S7** (Chain-of-Thought) | 3.5850 |

# Visualizations (Plot.ipynb)

The file Plot.ipynb produces comparative figures to support evaluation, including:

* Normalized Metric Comparison Across Models (Bar)
* Recall@5 and NDCG@5 Comparison (Bar)
* Diversity Metrics: HHI, Entropy, Gini, System-Level Entropy (Bar)
* Radar Plots (Per-model & Cross-model)
* Normalized Metric Line Plots
* Model Performance Heatmap
* Accuracy vs Diversity Scatter Plot
* Horizontal Bar: System-Level Entropy

These plots reveal trade-offs between precision and novelty across models.

# Results Summary

**Accuracy**

* S1 achieves the highest hit rate and recall but lacks diversity.
* S2 and S3 maintain balance between genre relevance and minor diversity.
* S7 performs best in ranking (Avg. Rank = 2.10) but still lacks top-K coverage.

**Diversity & Novelty**

* S5 and S6 improve system-level entropy by promoting obscure titles.
* S3 shows over-personalization (Gini = 0.44), overfitting to user niches.
* S7 demonstrates high ranking precision with moderate diversity.

**Explanation Power**

* S6 and S7 generate text explanations.
* These are suitable for trust-aware and human-centered applications.

# Conclusion

This practical work demonstrates how LLMs can enhance movie recommendation beyond traditional filtering. While S1 performs best in pure accuracy, LLM-based models (S6, S7) introduce meaningful explanations and S5–S7 promote novelty. Future work could explore fine-tuning, dialog-based interaction, and hybrid methods combining ranking and retrieval.

# References

[1] Yasmin Delbar, et al. "Benchmarking Large Language Models as Recommender Systems." In Proceedings of the ACM Web Conference 2024 (WWW '24), ACM, 2024. <https://dl.acm.org/doi/pdf/10.1145/3690655>

GitHub Reference: <https://github.com/yasdel/Benchmark_RecLLM_Fairness>