**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 uses the microsoft/Phi-3.5-mini-instruct large language model in a zero-shot setup to recommend movies based on a user’s previously watched titles. It does not require any training or fine-tuning and relies entirely on structured prompts to guide the LLM in selecting relevant recommendations from a predefined candidate list.

**Implementation**  
The code first loads user interaction data (test\_data), movie metadata (df\_items\_ml), and the pre-trained Phi-3.5 model and tokenizer using the Hugging Face Transformers library. The script selects 10 users with at least 6 watched movies and splits their history into input (first 3 titles) and ground truth (remaining titles). A candidate movie pool is created by sampling 120 movies the user has not seen. A prompt is constructed in natural language, embedding the 3 input movies and the candidate list, instructing the LLM to choose 10 titles. This prompt is tokenized and passed to the model on the best available device (MPS, CUDA, or CPU), and the output is decoded and parsed into movie titles. Only titles that exactly match the candidate list are kept. The system then calculates several evaluation metrics, including Hit Rate, Average Rank, Recall@5, NDCG@5, HHI, Entropy, and Gini Index, and stores the results in results/metrics\_s1.csv.

**Strengths & Limitations**  
S1 is easy to deploy and does not require training data or parameter tuning. It leverages the language model’s general understanding of movie-related content and user behavior. However, it depends heavily on the model correctly following the prompt, offers limited personalization beyond the short movie history, and may return popular or repetitive titles due to lack of deeper context.

S2 Genre-Focused

**Overview**  
The S2 model is a rule-based movie recommender that prioritizes strict genre matching between a user’s previously watched movies and potential candidates. It does not rely on any language model or machine learning; instead, it uses interpretable logic based on movie metadata—specifically genre overlap and normalized popularity—to rank and recommend items. This makes it a transparent, fast, and explainable alternative to generative LLM-based approaches like S1.

**Implementation**  
The system first loads movie metadata (df\_items\_ml) and full user interaction logs (test\_data\_ml1m\_fullInteraction) from CSV files. The evaluation loop selects 10 users, each with at least 6 known ratings, and splits their interaction history into two parts: the first 5 movies are used to identify preference patterns, while the rest form the ground truth. The function get\_recommendations\_genre\_strict extracts the user’s top 3 genres by counting frequency across the 5 input movies. It filters out all unseen candidate movies that share at least one of these genres and calculates a score for each based on the number of matching genres and a scaled popularity term (score = genre\_match\_count + 0.1 \* normalized\_popularity). Popularity is computed globally from interaction counts and perturbed with Gaussian noise to break score ties. The top 10 highest-scoring movies are selected as recommendations. While no model is invoked, a textual prompt is still generated internally to document the recommendation rationale. Finally, the recommendations are evaluated using multiple metrics (Hit Rate, Average Rank, HHI, Entropy, Recall@5, NDCG@5, Gini Index), and results are saved to results/s2\_metrics.csv.

**Strengths & Limitations**  
S2 is lightweight, explainable, and requires no deep model or training, making it ideal for interpretable baselines. It naturally promotes diversity through genre filtering and random sampling. However, its simplistic logic cannot capture complex or nuanced user preferences like themes, pacing, or tone, and it is limited by the quality of genre metadata. Its mild popularity bias, while useful for stability, may also reduce recommendation novelty.

S3 Genre-Contrast Diversity

**Overview**  
The S3 model introduces a diversification-focused recommendation strategy that blends genre-based user preferences with randomness and mild popularity bias. Its goal is to generate recommendations that reflect a user’s interests while also promoting novelty and exploration of less obvious genres. Instead of relying on strict matching rules, S3 leverages weighted genre signals, controlled noise, and scoring bias to produce a varied set of movie suggestions.

**Implementation**  
The code begins by loading user interaction logs and movie metadata from CSV files. Each user is processed if they have rated at least six movies: the first five are treated as input history, and the rest as ground truth. The core logic is implemented in the get\_recommendations\_s3\_diversify function. This function first constructs a user-specific genre profile by summing the user’s ratings per genre, producing a weighted preference vector. Then, for each candidate movie (sampled from the top user genres and excluding previously seen titles), a composite score is computed:  
score = genre\_score + 0.7 \* popularity + noise + bias,  
where genre\_score is derived from the user's weighted preferences, popularity is normalized from global interaction frequency, noise is sampled from a Gaussian distribution (N(0, 0.5)) to encourage variability, and bias is a tiny deterministic offset to break ties. The top 10 scoring titles are selected. If the score variance is extremely low (indicating uniformity), additional noise is injected to force diversity. A natural language prompt is also generated per user, documenting the recommendation context. Evaluation is performed using standard ranking and diversity metrics including Hit Rate, Average Rank, Recall@5, NDCG@5, HHI, Entropy, and Gini Index, and results are aggregated into a DataFrame.

**Strengths & Limitations**  
S3 offers a dynamic balance between personalization and discovery, enabling creative, genre-diverse recommendations while still reflecting user interests. It avoids deterministic output through score perturbation and is well-suited for recommendation settings that favor novelty. However, the randomness introduces potential evaluation instability, and genre-based logic may miss deeper semantic themes like mood or style. The ground truth may also not align well with diversity-focused outputs, slightly lowering precision in some cases.

S4 Quality + Novelty

**Overview**  
The S4 model is a hybrid, diversity-oriented recommender that encourages exploration of lesser-known movies while still respecting user preferences. It incorporates a genre-aware scoring mechanism but explicitly penalizes popularity to prevent over-recommendation of mainstream titles. Controlled randomness is also introduced to avoid recommendation stagnation and generate varied outputs. The model is particularly aimed at overcoming echo chambers by favoring novelty without completely discarding relevance.

**Implementation**  
The model starts by loading user interaction and item metadata from the MovieLens 1M dataset. For each of the 10 users with sufficient history, the interaction list is split into the first 5 movies (input) and the remaining (ground truth). Candidate movies are filtered to exclude previously seen titles and to include only those that share genres with the user’s top 3 genres. The function get\_recommendations\_s4\_diversify implements the scoring logic. First, a genre score is constructed by summing user ratings per genre from the input history, producing a personalized genre preference profile. For each candidate, a final score is computed:  
score = genre\_score + 0.8 \* popularity\_penalty + random\_noise + idx\_bias,  
where popularity\_penalty = -log(1 + normalized\_popularity) discourages overly popular items, random\_noise (sampled from N(0, 0.2)) adds stochasticity, and idx\_bias = idx \* 1e-4 ensures consistent ranking by breaking score ties. The top 10 scored titles are selected as recommendations. Metrics are then computed for each user, including ranking-based scores (Hit Rate, Average Rank, Recall@5, NDCG@5) and diversity-based scores (HHI, Entropy, Gini Index). Results are aggregated and stored in a DataFrame for analysis.

**Strengths & Limitations**  
S4 strategically reduces popularity bias, promoting fresh, less obvious recommendations. It achieves a good balance by anchoring recommendations in genre-based user preferences while penalizing mainstream content and adding score variability. This makes it particularly effective in use cases focused on discovery and novelty. However, its reliance on noise and penalty terms may reduce accuracy in terms of predicting ground truth, and subtle user interests may be overlooked if not captured in genre metadata. The model shines in scenarios where recommender fatigue or homogeneity are challenges.

S5 Surprise

**Overview**  
The S5 model is designed to generate surprising and unconventional movie recommendations by intentionally suppressing popular content and boosting unpredictability. Its primary goal is to challenge user expectations and escape the repetitive patterns found in typical recommender systems. By embracing randomness and penalizing mainstream appeal, the model favors lesser-known titles while still loosely referencing user genre preferences. This design aims to create a discovery-rich experience that is both fresh and engaging.

**Implementation**  
The system loads preprocessed user interaction data and movie metadata, filtering to include only users with sufficient history for evaluation. For each user, the first five watched titles are treated as history, while the remaining serve as ground truth. The recommendation generation is handled by get\_recommendations\_s5\_surprise, which begins by building a genre-weighted profile from the user's rated movies, summing the ratings per genre. Unlike previous strategies, candidate movies are not constrained by genre; instead, the pool spans the full movie list minus any titles the user has already seen.

For scoring, each candidate is evaluated using the formula:  
score = 0.5 \* genre\_score + (-2.0 \* log(1 + normalized\_popularity)) + strong\_noise + idx\_bias.  
Here, the genre score plays a supporting role (with weight 0.5), while a strong logarithmic penalty is applied to popular items to drastically reduce their ranking. A significant Gaussian noise term (standard deviation = 0.6) is introduced to maximize surprise, and a small index-based bias ensures deterministic tie-breaking. The top 10 scoring movies are returned as recommendations. Although a prompt is generated per user using get\_prompt\_s5\_surprise, it is primarily for documentation and not fed into a language model. Evaluation is then conducted across multiple diversity and relevance metrics, and global entropy is computed to assess system-wide novelty.

**Strengths & Limitations**  
The S5 model is uniquely positioned to generate recommendations that surprise users and break recommendation monotony. Its structure directly addresses the problem of over-personalization by diversifying exposure and encouraging genre and content exploration. This makes it especially suitable in contexts where users seek variety and unexpected suggestions. However, the high level of randomness and aggressive popularity penalties may result in lower alignment with user preferences and reduced hit rates. Additionally, the model depends on the assumption that surprise equates to value, which may not hold true for all user segments. Nonetheless, S5 is a valuable addition for creating exploration-heavy recommendation experiences.

S6 Motivate Reasoning

**Overview**  
The S6 model introduces a rationale-driven recommendation approach that prioritizes both relevance and transparency. Its key innovation lies in pairing each recommended movie with an explanatory justification that reflects the user’s preferences. By doing so, the model shifts from being a black-box predictor to a transparent decision-maker, helping users understand the “why” behind each suggestion. The scoring logic combines several human-interpretable factors like genre preferences, director familiarity, and exploration incentives, making the model suitable for trust-aware and educational recommendation environments.

**Implementation**  
The model begins by loading user interaction and movie metadata, filtering for users with at least six known ratings. Each user’s movie list is divided into five input movies and one or more ground truth titles. The function get\_recommendations\_s6\_motivate() handles the recommendation logic. It first computes a genre affinity profile from user-rated items, adjusting for individual bias by mean-centering the ratings. This forms the basis for the main scoring signal. Then, the model calculates director familiarity by identifying recurring directors across previously watched and candidate movies.

Next, it evaluates how many genres each candidate movie shares with the user's preferred genres, and computes an exploration bonus based on how rarely the user engages with a movie’s genres. All of these components are aggregated into a composite score using weighted contributions:  
score = 0.6 \* genre\_match\_score + 0.2 \* director\_score + 0.1 \* genre\_match\_count + 0.1 \* exploration\_bonus.

For each top-10 selected movie, a textual rationale is generated, dynamically constructed from the score components. These rationales follow intuitive templates such as:  
*“‘Inception’ aligns well with your favorite genres: Sci-Fi (score: 3.2), Thriller (score: 2.7).”*  
or  
*“We recommend ‘Memento’ due to your frequent interest in Drama (score: 2.5) and director Nolan (score: 1.5).”*

The candidate pool excludes previously seen movies and is downsampled to a maximum of 120 titles per user. Although popularity is calculated and normalized, it is not directly used in this version of the scoring function. Evaluation is performed for 10 users, using hit rate, average rank, recall@5, ndcg@5, entropy, HHI, and the Gini index. Global entropy is also computed to assess system-wide diversity.

**Strengths & Limitations**  
The S6 model delivers highly explainable recommendations by explicitly connecting user history to each suggested item. It is particularly valuable for applications where transparency and user trust are essential, such as educational platforms, content curation, or human-AI collaborative environments. The use of interpretable features and scoring decomposition makes the model easier to debug and refine. However, its reliance on genre and director metadata may limit its performance in capturing deeper thematic or emotional user preferences. Additionally, rationale generation, while useful, is currently rule-based and not adaptively optimized. Despite these constraints, S6 represents a meaningful step toward user-centered recommender systems that prioritize both clarity and utility.

S7 Chain-of-Thought Reasoning (COT)

**Overview**  
The S7 model introduces a Chain-of-Thought (CoT) recommendation strategy by prompting a language model to generate not only movie suggestions but also human-readable rationales for each. This design enables both personalization and transparency, with every recommendation accompanied by an explanation that contextualizes the match. By tapping into the reasoning capabilities of Phi-3, the system aims to produce suggestions that are both logically motivated and relevant to the user’s historical viewing behavior.

**Implementation**  
The model processes interaction and metadata from MovieLens, selecting users with at least six watched titles. For each user, the first five movies form the input history, while the remaining items serve as ground truth. The recommendation logic resides in the get\_recommendations\_s7\_cot() function, which performs three main operations.

First, a prompt is generated that mimics a natural conversation. It lists the user’s five known movies and asks the model to return ten recommendations in the format:  
1. Title – Reason, 2. Title – Reason, and so on. This open-ended structure allows the LLM to infer genre, thematic, or stylistic patterns from the input titles and elaborate on each selection.

Second, the Phi-3 model processes this prompt and returns a response that blends recommendations with justifications. The output is then parsed line-by-line to extract the movie titles, ignoring their explanations for evaluation purposes. Only titles that appear in the candidate pool—movies the user hasn't seen—are retained.

Third, the final top-10 list is built by validating and trimming the LLM’s output. Any hallucinated or invalid titles are discarded. Exposure scores are also computed based on each item’s rank using the formula:  
exposure\_score = 1.0 - 0.05 \* rank,  
which simulates how attention decays across ranked lists. These scores are later used to compute the Gini Index, capturing inequality in recommendation exposure across the system.

The model’s evaluation loop runs across 10 users and calculates metrics such as hit rate @3, average rank, recall@5, ndcg@5, entropy, HHI, and system-level Gini. All results are saved to results/s7\_metrics.csv.

**Strengths & Limitations**  
S7 stands out as the most human-aligned recommender in the system, offering high explainability through natural language rationales. It enables a new level of interaction and trust in recommendation interfaces, especially in applications where user transparency is essential. The LLM's reasoning flexibility can reveal subtle, abstract patterns in user preferences that rule-based models might miss. However, the generative nature of the output makes it fragile—parsing errors or hallucinated movie names can affect recommendation quality. Performance also varies depending on prompt phrasing and model parameters like temperature. Despite these challenges, S7 provides a compelling foundation for interpretable and dialog-friendly recommenders, bridging LLM reasoning with user-centric design.

# 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>