*Democritus University of Thrace • Department of Informatics*

# Comparative Evaluation

## of “Traditional” Recommender Systems

and Recommender Systems Based on Large Language Models

**Michail G. Selvesakis**

Supervisors: Dr. E. Mavridou • Dr. E. Vrochidou

*Kavala, 2026*

Recommender Systems | LLMs | Sustainability | Benchmark

# Introduction & Motivation

## The Role of Recommender Systems

Automated suggestion of relevant items to the user.

**Used in:**

- Netflix, Spotify, YouTube — content recommendation
- Amazon, e-shops — product recommendation

## Research Motivation

**Traditional methods vs LLMs:**
The rapid advance of Large Language Models (GPT, Gemini) raises questions of comparative performance.

**Contribution to research:**
Few studies combine accuracy + energy + cost in a single integrated framework.

**19 models evaluated across 4 axes:**

- Accuracy, Ranking, Sustainability, Cost

# Research Questions

| | |
|---|---|
| **RQ1** | How do traditional models (CF, Content-Based, Neural) compare with LLMs on accuracy, efficiency & sustainability? |
| **RQ2** | To what extent can LLMs in a zero-shot setting compete with fully trained traditional models? |
| **RQ3** | What are the trade-offs between recommendation quality and computational/environmental cost? |
| **RQ4** | How effectively do LLMs handle the cold-start problem for new users? |
| **RQ5** | How do the size, architecture and cost of different LLMs affect recommendation quality? |

# Theoretical Background

## Collaborative Filtering (CF)

- User-Based KNN
- Item-Based KNN
- Matrix Factorization (SVD, NMF)
- BPR-MF (Bayesian)

## Content-Based Filtering

- TF-IDF + cosine similarity (keywords converted into weighted vectors)
- 19 genre features per movie
- Does not depend on other users

## Neural Models

- NeuMF (Neural Matrix Factorization)
- LightGCN (Graph Neural Network)
- SASRec (Self-Attention)
- GRU4Rec (Recurrent NN)

## Large Language Models

- Zero-Shot Ranking
- GPT-5.2, GPT-4.1 (OpenAI/Azure)
- Gemini 3 & 3.1 (Google)
- Gemma-4-31B (Open Source)

# Methodology & Architecture

## Phase 1: Preparation

- Load config.json
- MovieLens-100K dataset
- Model definition

## Phase 2: Main Loop

- 10-fold Cross-Validation
- Training & Prediction
- CodeCarbon monitoring
- Metric computation

## Phase 3: Analysis & Export

- Cold-Start Evaluation
- Pareto analysis (7 fronts)
- Chart generation

**Dataset: MovieLens-100K**

**Validation: 10-fold CV**

**Models: 19 total**

**Infrastructure: Google Cloud n1-standard-32**

# Data & Implemented Models

## MovieLens-100K

| | |
|---|---|
| Ratings: | **100,000** |
| Users: | **943** |
| Movies: | **1,682** |
| Scale: | **1–5 (integers)** |

## 13 Traditional + 6 LLMs

**Traditional / Neural:**

- Random (baseline)
- SVD (Funk)
- NMF
- UserKNN
- ItemKNN
- ContentBased
- Hybrid
- BPR-MF
- SASRec
- NeuMF
- LightGCN
- GRU4Rec
- Pop (popularity)

**LLMs:**

- GPT-5.2-Chat
- GPT-4.1
- Gemini-3-Flash
- Gemini-3.1-Flash-Lite
- Gemini-3.1-Pro
- Gemma-4-31B

# Experimental Environment

Google Cloud

## Production VM — Google Cloud “n1-standard-32”

| | |
|---|---|
| Processor (CPU): | **Intel Xeon @ 2.30 GHz** |
| 32 logical cores | |
| Graphics Card (GPU): | **NVIDIA Tesla T4** |
| 16 GB VRAM • CUDA 12.4 | |
| RAM: | **120 GB** |
| OS: | **Linux (kernel 5.10.0-cloud-amd64)** |
| Python: | **3.11.14** |
| PyTorch: | **2.6.0+cu124** |
| Region: | **USA (us-central1, low $CO_2$)** |

GPU acceleration enabled via torch.cuda.is_available()

## Development — Local Machine

| | |
|---|---|
| CPU: | **AMD Ryzen 9 5950X** |
| GPU: | **NVIDIA GeForce RTX 3090 (24 GB)** |
| RAM: | **64 GB DDR4** |
| OS: | **Ubuntu Linux** |

## ⚙ Experiment Parameters

| | |
|---|---|
| Dataset: | **MovieLens ml-100k** |
| Validation: | **10-fold Cross-Validation** |
| Top-K: | **K = 10** |
| Total time: | **~5.8 hours (19 models + cold-start)** |
| Energy: | **CodeCarbon v3.2.5** |

# Evaluation Metrics

## Rating Prediction

RMSE: Root mean squared error. Penalises large deviations.

MAE: Mean absolute error. Weights all errors equally.

## Ranking Quality

NDCG@10: Relevant items should appear first (ideal list vs recommended list).

HR@10: Share of users who received at least 1 relevant item.

Precision / Recall / F1: Precision, recall and their harmonic mean.

## Beyond Accuracy

Coverage: Share of unique items recommended to ≥1 user.

Gini: Inequality of the recommendation distribution — Gini≈0: uniform, Gini≈1: a few items dominate.

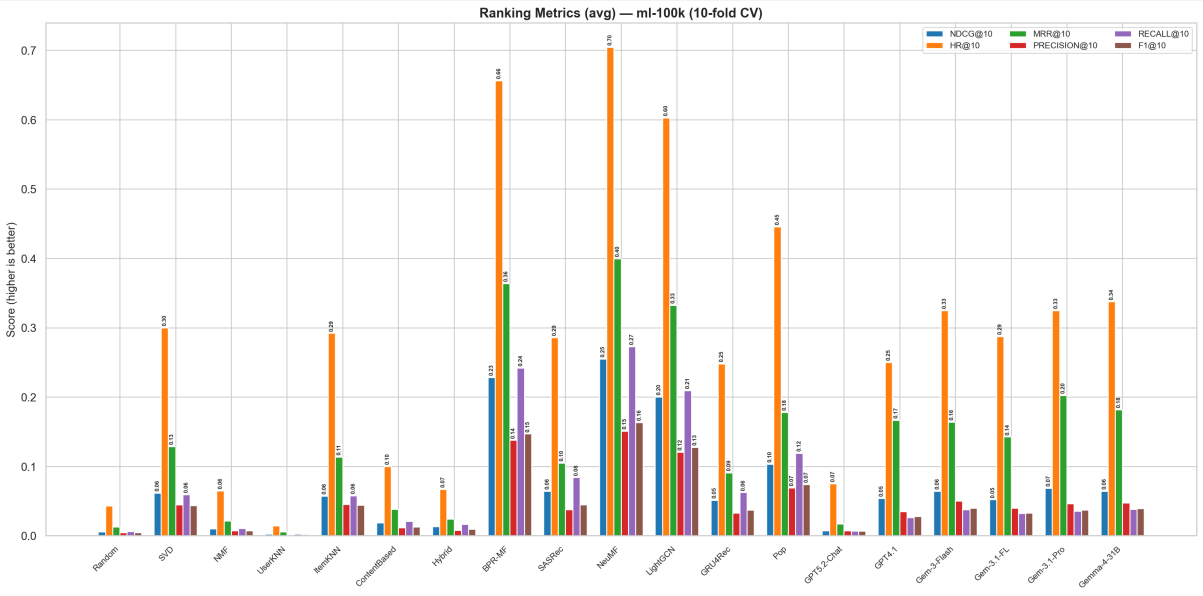
## Sustainability & Cost

Energy (Wh): Energy consumption measured with CodeCarbon (CPU+GPU+RAM).

$CO_2$ (g): Carbon emissions based on the US energy mix.

Cost ($): Cloud compute cost + LLM API cost.

# Results — Ranking Quality




## Key Findings (NDCG)

NeuMF (Best)

**0.255**

BPR-MF (2nd)

**0.228**

Pop (baseline)

**0.103**

Best LLM (Gemini 3.1 Pro)

**0.069**

GPT-5.2-Chat

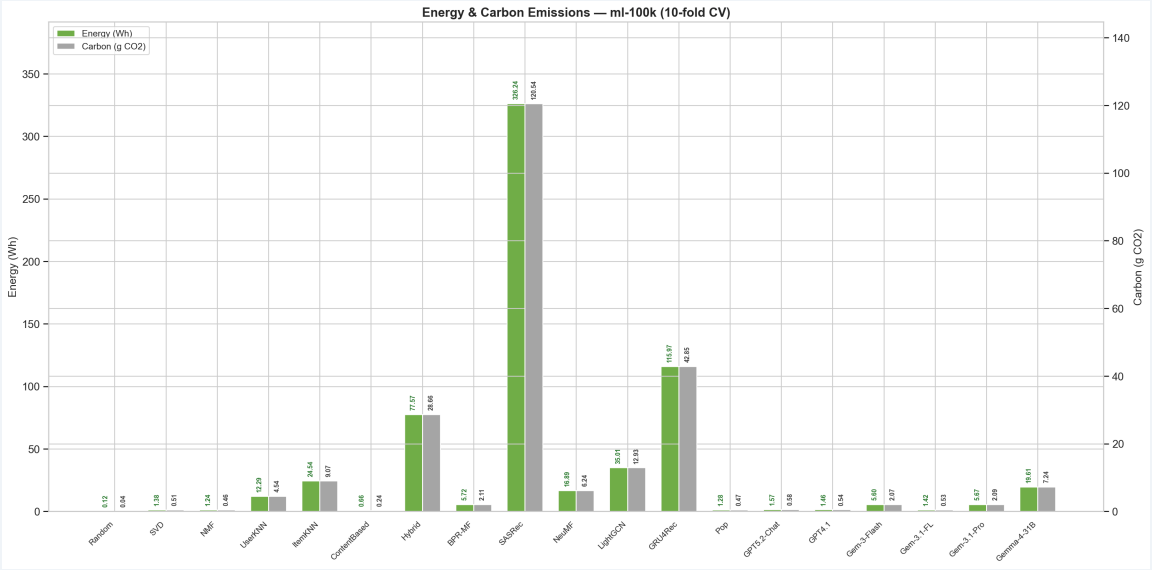
**0.007**

# Latency & Throughput

| Model | Latency Mean | P95 | Throughput (req/s) |
|---|---|---|---|
| Random | 0.41 ms | 0.46 ms | 2,451 |
| ContentBased | 1.03 ms | 1.45 ms | 971 |
| BPR-MF | 6.21 ms | 20.52 ms | 161 |
| SVD | 7.01 ms | 7.50 ms | 143 |
| NeuMF | 18.05 ms | 47.97 ms | 55 |
| ItemKNN | 139.49 ms | 263.68 ms | 7 |
| Gemini-3.1-Flash-Lite | 2,505 ms (2.5 s) | 3,831 ms | 0.40 |
| Gemma-4-31B | 177,388 ms (177 s) | 247,002 ms | 0.006 |

⚡ **Key finding: traditional models serve 5–2,451 responses/sec, while LLMs manage only 0.006–0.40!**

# Energy Footprint & Sustainability




## SVD vs SASRec Comparison

### SASRec

Highest consumption

**326 Wh**

*121 g $CO_2$*

### SVD

Similar NDCG to SASRec

**1.38 Wh**

*237× less energy*

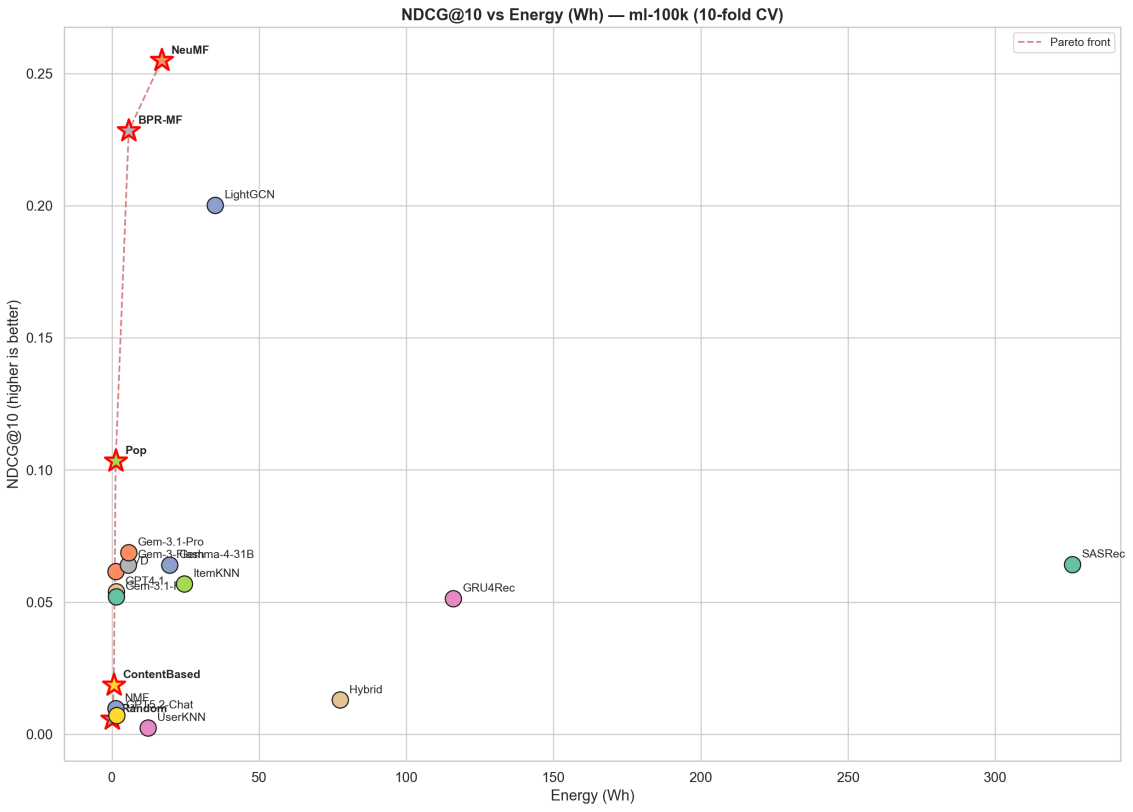
### Consumption Range

Between Random & SASRec

**0.12 → 326 Wh**

*2,750× difference*

# Pareto Analysis — Trade-offs




## Pareto-Optimal Models

**★ Random**
Extremely fast, low accuracy

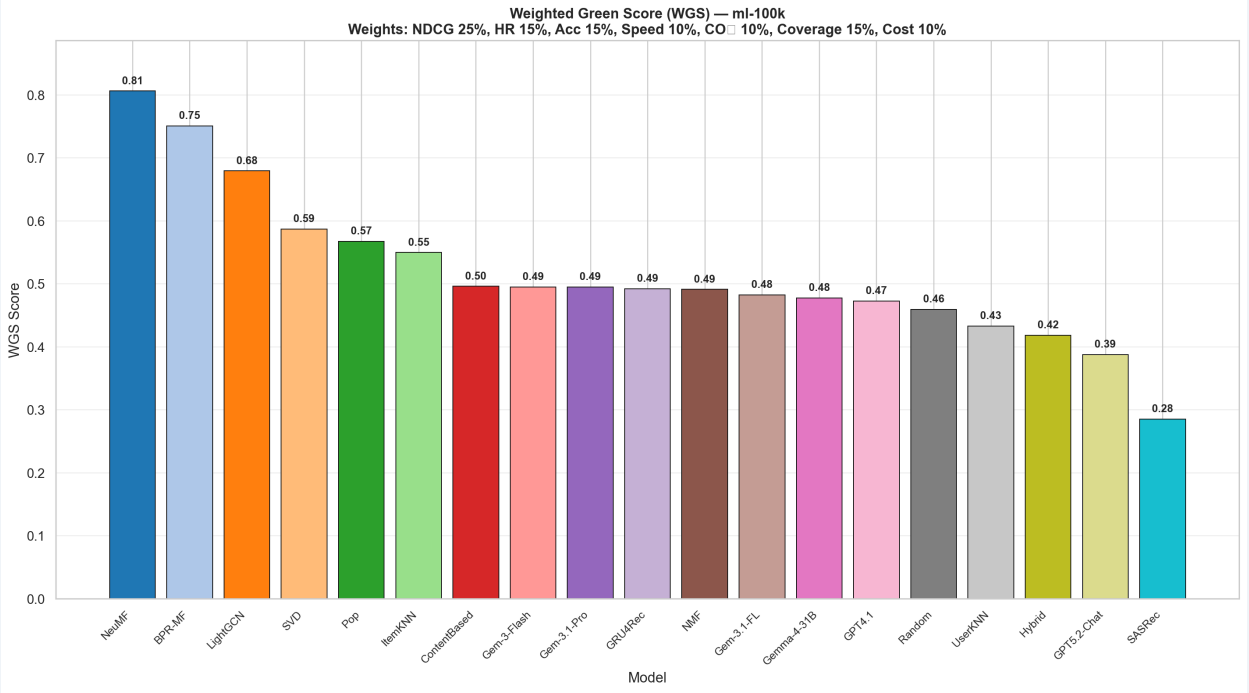
**★ ContentBased**
Lightweight, useful baseline

**★ Pop**
Good baseline, zero training

**★ BPR-MF**
Best NDCG-to-energy ratio

**★ NeuMF**
Highest NDCG overall

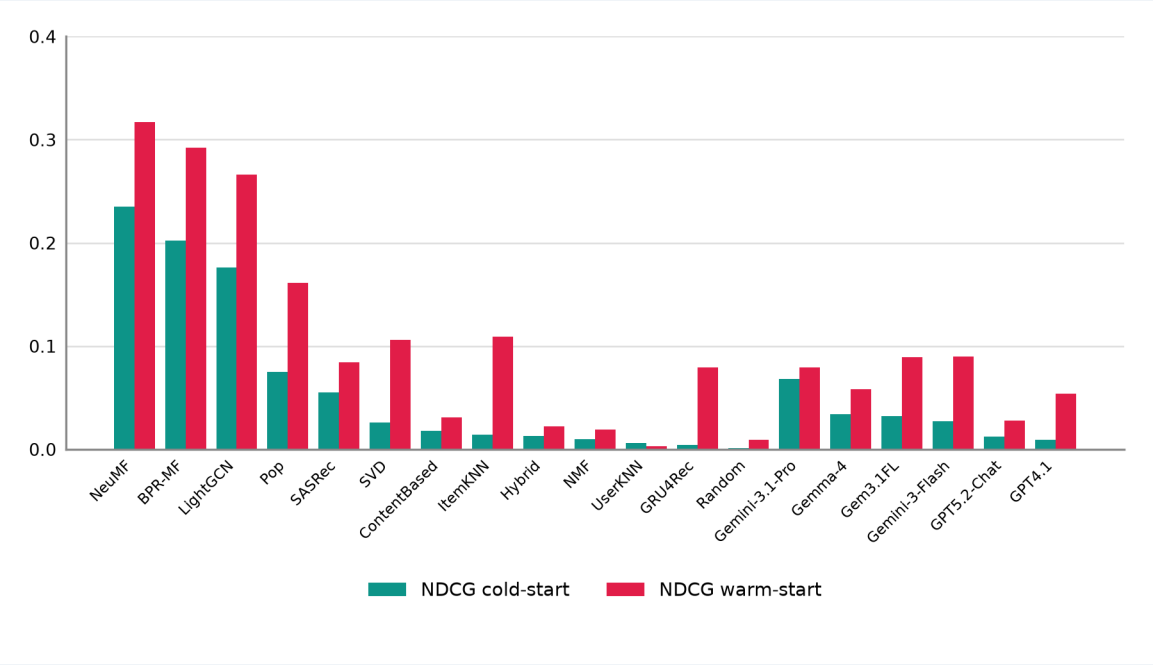
# Multi-Criteria Evaluation (Weighted Green Score)




*Weights: Quality 55% (NDCG 25%, HR 15%, RMSE 15%) • Diversity 15% • Efficiency 30% (Speed + $CO_2$ + Cost)*

# Cold-Start Evaluation

Definition of a “cold” user: ≤33 ratings → 246 users (26.1% of the total)




## Performance Degradation

**NeuMF** **−25.9%**
Most robust

**BPR-MF** **−30.8%**
2nd most robust

**SVD** **−75.5%**
Large drop

**GRU4Rec** **−94.9%**
Needs a lot of data

**Gemini-3.1-Pro** **−13.1%**
Best LLM at cold-start

# Conclusions

### 1. Neural models dominate ranking

NeuMF & BPR-MF lead. Prediction accuracy (RMSE) does not necessarily correlate with ranking quality — SVD excels at RMSE but lags at NDCG.

### 2. LLMs do not compete with traditional recommender systems

No LLM beat even the popularity baseline (Pop). The best LLM (Gemini 3.1 Pro) reached just 27% of NeuMF. GPT-5.2 performed worse than older models.

### 3. Complexity does not always deliver quality

SVD ≈ SASRec in ranking quality, but SASRec needs 126× more time and 237× more energy. A strong argument for simpler models.

### 4. Environmental Footprint — A Critical Criterion

Consumption ranges from 0.12 to 326.24 Wh. Algorithm choice must account for sustainability too. No LLM is Pareto-optimal.

# Future Work

### 01. Fine-tuning & Few-Shot LLMs

Evaluate LLMs with few-shot prompting or fine-tuning on recommendation data — replacing pure zero-shot.

### 02. Hybrid Use of LLMs

Use LLMs as feature-extraction tools feeding traditional models, or as conversational recommenders.

### 03. Larger Datasets

Extend to MovieLens-1M/10M, Amazon, Yelp. LLMs and SASRec may benefit more in other domains.

### 04. Real LLM Energy

Run open-source models (Qwen, Gemma) locally for complete energy measurement.

## Thank you for your attention!

*Michail G. Selvesakis • Department of Informatics, D.U.Th. • 2026*