Final Pproject Planning Document

Saloua Daouki

2025-07-03

Contents

Title

Comparing Traditional Recommender Systems with an Azure OpenAI Prototype: A Case Study Using MovieLens.

1. Motivation:

Personalized recommendation systems shape how we discover content in digital spaces — from movies and products to personalized learning. Throughout this course, I have implemented multiple recommendation models, from simple baselines to advanced matrix factorization, using the MovieLens dataset as a standard benchmark.

Inspired by these experiences, my long-term goal is to build an intelligent classroom recommender system for teachers. This future system will use survey data on students' learning styles, teacher inputs (such as class type: ICT, ELL, Honors, or General Education), and exam results to recommend or adapt lesson plans for each unique group. By understanding how different algorithms — traditional and modern LLM-based — generate recommendations on large datasets, I am preparing for that future work.

This final project ties all my learning together: comparing multiple traditional recommender systems and extending them with an LLM-based prototype using Azure OpenAI to explore opportunities and practical challenges for hybrid systems.

2. Dataset:

I will use the MovieLens 1M dataset, which includes 1 million ratings from 6,000 users on 4,000 movies. To add a unique dimension, I plan to enrich the dataset with additional content-based features — such as genres, simple poster metadata, or scraped movie taglines — to demonstrate how multi-source data can improve recommendations. I will scrape movie taglines from IMDb using Python's BeautifulSoup or use an existing CSV with poster info from The Movie Database (TMDb) API if time allows.

3. Research Question:

Which traditional recommender system — baseline predictor, content-based filter, memory-based collaborative filter, or model-based matrix factorization — produces the best results for rating prediction and top-N recommendation quality on the MovieLens dataset, and how does a small prototype LLM-based recommender using Azure OpenAI compare in generating relevant suggestions for the same users?

4. System Overview:

I will build and compare the following systems:

✓ *Baseline Predictors*

- Global average rating
- Bias-adjusted user/item effects

✓ *Content-Based Filtering*

- Uses genre information and additional scraped content features

✓ *Collaborative Filtering*

- User-based k-Nearest Neighbors
- Item-based k-Nearest Neighbors

✓ *Model-Based Collaborative Filtering*

- Matrix Factorization (SVD or ALS), with hyperparameter tuning to optimize prediction accuracy

✓ *Popularity-Based Recommender*

- Recommends the most popular unseen movies

✓ *LLM-Based Recommender (Azure OpenAI)*

- For a small sample of test users, I will craft prompts describing the user's known liked movies and ask Azure OpenAI to suggest additional movies.
- The generated titles will be mapped back to MovieLens items and compared to the actual test set.
- This demonstrates how LLMs can be used for human-like recommendation and content re-ranking.

5. Method & Evaluation:

- Use train/test splits to compare models on:
  - *Rating prediction accuracy: RMSE, MAE*
  - *Top-N recommendation quality: Precision@K, Recall@K, MAP*
  - *Business relevance: Diversity, Novelty, Serendipity*
- The LLM prototype will be evaluated by comparing the overlap between its generated titles and the actual items a user rated highly in the test set.
- I will use PySpark on Databricks to handle larger memory-based and model-based computations, and Azure OpenAI to run the LLM generation pipeline.

Then, I will compare LLM recommendations by calculating the precision and recall of overlap with held-out test items, and discuss qualitative impressions of novelty and relevance.

6. Unique Element:

As I mentioned above in section `2. Dataset`, to fulfill the unique data requirement, I will:

- Add additional content features (genres, poster sentiment, or scraped taglines)
- Implement an LLM-based recommendation prototype using Azure OpenAI to showcase hybrid system possibilities.

7. Long Term Vision:

This project is directly connected to my ultimate goal of creating an educational recommender for teachers. The future system would gather:

- Student survey data on learning styles,
- Teacher input about class type and goals,
- Student performance data (unit tests, state exams).

Based on these inputs, it would recommend personalized lesson plans or review modules to help students succeed. The lessons learned from blending traditional algorithms with an LLM layer will help me design an intelligent, explainable, and adaptable classroom recommender in the future.

8. Deliverables:

✓ A published notebook (RPubs) with:

- Full data prep, enrichment, and modeling pipeline
- Implementation and comparison of baseline, content-based, collaborative, and matrix factorization methods
- Small LLM prototype using Azure OpenAI
- Visualizations of performance metrics
- A reflection on the benefits, limitations, and practical trade-offs of LLM-based recommendations for real-world use cases like education.

The final notebook will clearly label which sections reuse methods from my previous projects, with improvements and unified evaluation.

My Next Steps:

1- Activate Azure for Students credits (done!).

2- Prepare the dataset and additional content features.

3- Build and tune traditional recommenders.

4- Design and test the Azure OpenAI prompt pipeline for a small test user group.

5- Compare all results and publish final analysis with clear visualizations.

6- Connect the results to the long-term vision for lesson plan personalization in schools.