Large Language Models as Movie Recommendation Systems

Chaitanya Mahajan
chma4223@colorado.edu

Rajath Shashikanth Vasisth
rava8988@colorado.edu

Rigved Thorat
rith8786@colorado.edu

Meghasrivardhan Pulakhandam
mepu1328@colorado.edu

Abstract

The rise of large language models (LLMs) has created a drastic surge in its application to various natural language problems, given their vast contextual knowledge. One such field is Recommendation Systems (RecSys), where systems previously suffering from data sparsity and cold start problems are now improved by integrating LLMs into the RecSys architecture.

In this paper, we attempt to evaluate the effectiveness of LLMs as standalone recommendation systems through the process of fine-tuning. Experiments are carried out for the movie recommendation task using a popular movie dataset, and results are compared for the baseline and fine-tuned versions of the LLM showing that fine-tuned LLMs show promise to sustain as standalone recommendation systems.

1 Introduction

In recent years, Recommendation Systems have become an important part of content and media houses such as Netflix and influence how content is consumed by a user accessing these mainstream media. To cater to the users interests, recommendation models have been built and reiterated to provide maximum satisfaction to the user (Tang et al., 2023). Recommendation engines mainly rely on two types of information (Sharma and Dutta, 2020): **(1) Content/Context knowledge:** This type of knowledge involves users history and activity to understand their likes and dislikes. **(2) Collaborative knowledge:** This knowledge utilizes similar preferences across a group of users to suggest recommendations to each other. The factors used to design such systems can be categorized into users U and items I. These variables helped model the initial recommendation systems, such as Collaborative Filtering (CF) models. These models utilized a method called Matrix Factorization, which involved optimizing a matrix of user and item interactions ($U \times I$) (Wang et al., 2022b). Some notable CF models, which form the baseline models of many research works are: SASRec (Kang and McAuley, 2018), NCF (He et al., 2017), and NextItNet (Yuan et al., 2018).

Soon, problems such as sparse data for these matrices posed a problem for new users and hence, questioned the effectiveness of these models. This problem started a search for building RecSys with a rich content base which is able to support the on-boarding of new users. Another problem with CF models is that their architecture is designed to consume only user-item interactions while there is number of multi modal auxiliary data which can improve RecSys to perform to their maximum potential (Wei et al., 2024). Thus, in recent systems, CFs are used in conjunction with other multimodal learning models to improve the overall performance of these recommendation engines.

LLMs are one such powerful contextual data houses which incorporate important information into the RecSys model. But, they have not been able to perform well as standalone systems for recommendation tasks due to their poor capability to capture collaborative information (He et al., 2023). Hence, the question arises, what if we are able to embed this collaborative information for a specific task, such as movie recommendation through the process of fine-tuning? Will the fine-tuned model perform better than a foundational model relying on techniques such as zero-shot learning to make movie recommendations?

In this work, we attempt to answer this question using the OpenAI model - GPT-4o mini fine-tuned for the movie recommendation task. We have used the Netflix Prize Dataset[1] for this task and the dataset is preprocessed to create prompts and inferences based on crude user-to-user interactions mined from the dataset. Finally, a compari-

[1] https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data

son study shows the ranking metrics applied on the recommendation for the fine-tuned model vs the foundational model in a zero-shot setting with both models based on the GPT-4o mini model.

2 Related Work

LLMs have been used in a wide array of applications in the RecSys model. In a zero-shot setting (He et al., 2023), they have been observed to better Conversational Recommendations Systems (CRS) than existing fine-tuned CRS models such as ReDIAL (Liang et al., 2024) and UniCRS (Wang et al., 2022a) when the performance was measured based on a conversation scraped from a Reddit movie discussion thread. Although, the performance is better than traditional CRS, LLMs have their own disadvantages. They suffer from poor collaborative knowledge and a variety of biases, such as popularity/geographical bias. The advantage of traditional CRS/RecSys is that the items are often encoded with item id's before being analyzed in a RecSys setting (Eg. Inception = MOVIE_123) (Zheng et al., 2024). Hence, they do not know if the movie is popular or not and rely heavily on statistics. LLMs on the other hand highly rely on natural language semantics and hence suffer from the aforementioned biases such as knowing how popular a movie is or what the ethnicity of the user is based on the region.

So, the next thing that might seem suitable is incorporating LLMs as a part of traditional RecSys models. For example, in Wei et al. (2024), they are used to process auxiliary or side information. Here, they can go through the users movie recommendation history and add more content to it such as the actor or genre of the movie i.e in a sense reinforce existing information with additional information. They also embed visual cues from posters, add additional information to the user profile and so on. After this, the data can be embedded and fine-tuned through a traditional RecSys model such as LATTICE (Zhang et al., 2021) - a very powerful Graph Neural Network (GNN) which can learn stronger user-item interactions for better recommendations.

In the previous case study, the LLM was providing side information which, along with the original information, is iterated through the traditional RecSys. But, there is a major inconsistency here. The data provided by the LLM is text tokens while the RecSys traditionally ingests item ID's (a unique code). How do we combine these two different data formats? Zheng et al. (2024) and Kim et al. (2024) answer this question through the method Alignment Tuning, where the item ID's and language semantics are combined and learned through a Multi Layered Perceptron (MLP) or the S-BERT (Reimers and Gurevych, 2019) techniques to create multi-level discrete codes which can now be processed through the RecSys. The Alignment tuning technique makes the hybrid RecSys system model agnostic, making it compatible across a wide variety of LLMs and traditional recommenders.

In Kim et al. (2024), a decent and light alternative to LLMs to encode multimodal information is modality encoders such as BERT (Devlin et al., 2019) or a Vision Transformer (Dosovitskiy et al., 2021) to encode multimodal data-additives to a traditional recommender. But, LLMs are always much better than these models. Even among LLMs, many publications have claimed that OpenAI's GPT models (4 or 3.5 turbo) perform exceptionally better than any other open-source models, such as Mistral-7B or Deepseek-7B (He et al. (2023), Raj et al. (2024)). Hence, we have sided with the GPT-4o mini model for our experiments.

It is also worth noting that the popular evaluation metrics used for recommendation tasks by He et al. (2023) and Raj et al. (2024) include Hit@K, Recall@K to evaluate recommendation items with ground truth items and the ROGUE metrics (Lin, 2004) to evaluate LLM-generated summaries. We will be using some of these ranking metrics in our own implementations.

3 Methodology

The proposed system solely depends on the contextual understanding of an LLM and fine-tuning the LLM, instead of the previously discussed hybrid settings involving LLMs and traditional RecSys models for suggesting movie recommendations. The architecture in Figure 1 can be mathematically modeled as a function that outputs the inference $\mathcal{I}$:

$$\mathcal{I} = \mathcal{F}(T, F, C). \tag{1}$$

If ϕ represents the fine-tuning loop, then:

$$\mathcal{F} = \phi(T, F, C). \tag{2}$$

The first task at hand was to prepare the dataset for fine tuning. The Netflix Prize dataset contains ratings from a scale of $1 - 5$ by nearly $50,000$ users for around $18,000$ movies, released in or before 2005. Keeping in mind the computational

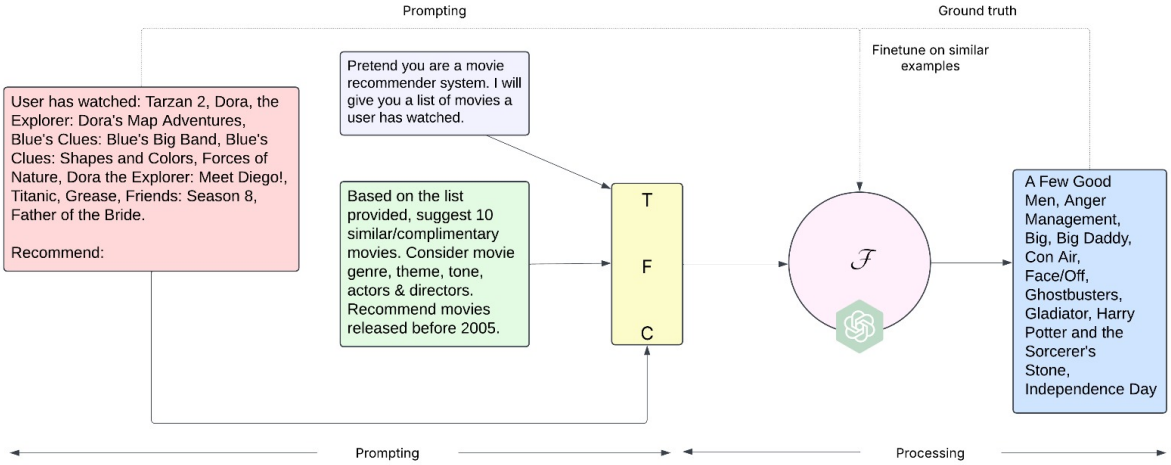

Figure 1: Brief architecture of our system with the fine tuned system $\mathcal{F}$, the task description T , preferred output format F and the prompt content C.

constraints, we randomly sampled $10,000$ users to learn their preferences. For each user k, we further sorted movie ratings (from 1 to 5) along with the timestamp of the rating. Then we sampled the top 10 movies of users (filtering for users having more than three 5-star reviews).

Based on the k^{th} user's top 10 movies, we randomly sampled $10,000$ neighbors. A neighbor is defined as a user who has given a 5-star rating to the k^{th} customer's top 10 movies. Finally, we extracted all the common movies watched by at least 50% of the neighbors to create our first dataset: **profileData50** - containing the ground truth or gold standard recommendations, if the k^{th} user has a list of top 10 most liked movies. Similarly, we also generated a **profileData60** having a watch factor of 60, i.e., consisting of 60% of the neighbors. For the 60% scenario instead of sampling neighbors who have given a 5 star rating for the 10 movies of the k^{th} user, we reduced the threshold to 4 star and above so as to obtain dense data.

It should be duly noted that this approach is completely rule-based to sample movies of neighbors with similar preferences, unlike machine learning methods that could have been used such as cosine similarity aggregation or a KNN based clustering (Kamble et al., 2025). By extracting the neighboring movies based on the top 10 movies the k^{th} user has watched, we set up our recommendation task.

The data is then split into train, validation and test sets, following a $70/20/10\%$ split respectively, reserving about 6000 users for training. Finally, we fine-tuned the GPT-4o-mini-2024-07-18 model from OpenAI. The aggregate cost of fine-tuning for the two datasets to generate two fine-tuned models came to about 16 USD at the rate of 3 USD per $1M$ tokens per epoch.

Hyperparameter	Value
Batch Size	32
Epochs	2
Learning Rate Multiplier	1
Other Parameters	Default

Table 1: Hyper-parameters used for the experiment

The format for fine-tuning involves: system, user, and assistant. The system's content involves the task to be performed by the system, i.e, the task definition and format $T + F$. The user content includes the prompt provided by the user (top 10 movies) C, and the assistant's content is the output provided by the model based on the contents of the system and the user i.e the Inference $\mathcal{I}$ (1). As the maximum number of suggested movies is 10 in the testing data, our fine-tuned model is also generating 10 recommendations.

4 Experiments and Results

The recommendations made by the finetuned GPT-4o mini model is compared against gold standard recommendations using metrics such as Precision@K, Recall@K and F1@K as shown in Figure 2. The rank metrics are further compared for the foundational baseline model of GPT 4o-mini using zero shot learning. The comparison of the ranking metrics for both the datasets of 50% and 60%

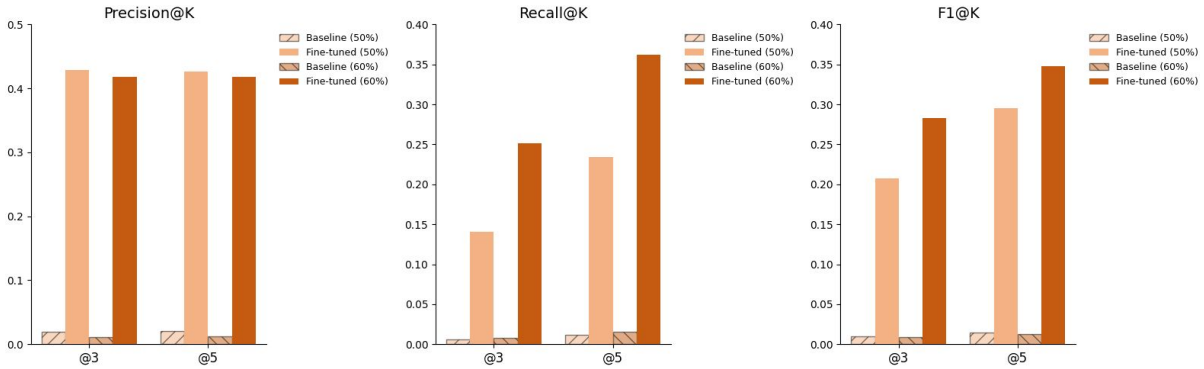

Figure 2: Precision, Recall and F1 @ K scores for $K = \{3, 5\}$ - Compared for the foundational model of GPT-4o mini and the finetuned model. Further separate comparison for the profileData50 and the profileData60 dataset.

Metric	Baseline	Fine-tuned
Precision@3	0.0198	0.4284
Recall@3	0.0063	0.1404
F1@3	0.0095	0.2075
Precision@5	0.0208	0.4268
Recall@5	0.0117	0.2339
F1@5	0.0146	0.2951
MRR	0.0505	0.5981
MAP	0.0060	0.2780

Table 2: Comparison of performance metrics between the baseline and fine-tuned models - profileData50

Metric	Baseline	Fine-tuned
Precision@3	0.0116	0.4184
Recall@3	0.0083	0.2509
F1@3	0.0089	0.2833
Precision@5	0.0125	0.4182
Recall@5	0.0155	0.3625
F1@5	0.0127	0.3476
MRR	0.0357	0.5526
MAP	0.0088	0.3009

Table 3: Comparison of performance metrics between the baseline and fine-tuned models - profileData60

watch factor can be found in the tables 2 and 3.

It is clear that fine-tuning an LLM has great prospects in improving collaborative knowledge of LLMs as opposed to just prompt engineering through zero-shot learning proposed in He et al. (2023). The results indicate the effectiveness of LLMs to be standalone systems in recommendation tasks without combining them with traditional recommendation systems.

5 Limitations and Future work

There are a few downsides to leveraging large language models for movie recommendation systems. First of all, LLMs tend to hallucinate (He et al., 2023). They also suffer from weak collaborative knowledge (when compared to CF models) and numerous biases, like popularity bias, geographical bias. Also, we have randomly sampled $10,000$ users from our dataset to meet our computational constraints. The sample size is not up to the standards of real-world sample sizes, which can be in millions for streaming applications like Netflix.

In the future, we plan to leverage machine learning models to generate ground truth labels. A more in-depth comparative analysis for fine-tuned LLMs vs traditional recommendation systems can also be performed to understand the upsides and downsides of fine-tuned LLMs. Finally, the intersection of LLMs and Reinforcement learning or Direct Preference Optimization (DPO) can also be studied to see how collaborative knowledge in LLMs improves.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. *Preprint*, arXiv:2010.11929.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie,

Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. *Preprint*, arXiv:1708.05031.

Zhankui He, Zhouhang Xie, Rahul Jha, Harald Steck, Dawen Liang, Yesu Feng, Bodhisattwa Prasad Majumder, Nathan Kallus, and Julian Mcauley. 2023. Large language models as zero-shot conversational recommenders. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, page 720–730, New York, NY, USA. Association for Computing Machinery.

Khushi Kamble, Pradnyanand Bhadarge, Shreejit Bhakte, Omkaresh Kulkarni, Rutuja Kadam, and Latika Pinjarkar. 2025. Movie recommendation system using hybrid based method. In *2025 International Conference on Automation and Computation (AUTOCOM)*, pages 119–126.

Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206.

Sein Kim, Hongseok Kang, Seungyoon Choi, Donghyun Kim, Minchul Yang, and Chanyoung Park. 2024. Large language models meet collaborative filtering: An efficient all-round llm-based recommender system. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 1395–1406, New York, NY, USA. Association for Computing Machinery.

Tingting Liang, Chenxin Jin, Lingzhi Wang, Wenqi Fan, Congying Xia, Kai Chen, and Yuyu Yin. 2024. LLM-REDIAL: A large-scale dataset for conversational recommender systems created from user behaviors with LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 8926–8939, Bangkok, Thailand. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Subham Raj, Anurag Sharma, Sriparna Saha, Brijraj Singh, and Niranjan Pedanekar. 2024. Transformative movie discovery: Large language models for recommendation and genre prediction. *IEEE Access*, 12:186626–186638.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Nisha Sharma and Mala Dutta. 2020. Movie recommendation systems: A brief overview. In *Proceedings of the 8th International Conference on Computer and Communications Management*, ICCCM '20, page 59–62, New York, NY, USA. Association for Computing Machinery.

Gary Tang, Jiangwei Pan, Henry Wang, and Justin Basilico. 2023. Reward innovation for long-term member satisfaction. In *Proceedings of the 17th ACM Conference on Recommender Systems*, RecSys '23, page 396–399, New York, NY, USA. Association for Computing Machinery.

Xiaolei Wang, Kun Zhou, Ji-Rong Wen, and Wayne Xin Zhao. 2022a. Towards unified conversational recommender systems via knowledge-enhanced prompt learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 1929–1937, New York, NY, USA. Association for Computing Machinery.

Zhu Wang, Honglong Chen, Zhe Li, Kai Lin, Nan Jiang, and Feng Xia. 2022b. Vrconvmf: Visual recurrent convolutional matrix factorization for movie recommendation. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(3):519–529.

Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, WSDM '24, page 806–815, New York, NY, USA. Association for Computing Machinery.

Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2018. A simple convolutional generative network for next item recommendation. *Preprint*, arXiv:1808.05163.

Jinghao Zhang, Yanqiao Zhu, Qiang Liu, Shu Wu, Shuhui Wang, and Liang Wang. 2021. Mining latent structures for multimedia recommendation. In *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, page 3872–3880, New York, NY, USA. Association for Computing Machinery.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. 2024. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 1435–1448.