Utilizing Users' Negative Preferences in LLM-Based Recommendation System

Xu, Yunfan
Kyushu University

Ushiama, Taketoshi
Kyushu University

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 270 (2025) 1846–1855

Procedia Computer Science

www.elsevier.com/locate/procedia

29th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2025)

Utilizing Users' Negative Preferences in LLM-Based Recommendation System

Yunfan Xu[a], Taketoshi Ushiama[a]

[a]*Kyushu University, 4-9-1 Shiobaru, Minami-ku, Fukuoka, 815-8540,Japan*

Abstract

Recommendation systems have significantly evolved with the integration of large language models (LLMs) which have attracted increasing attention due to their strong language understanding capabilities. In typical human-to-human recommendation processes, a recommender first presents a list of items and offers explanations based on the user's preferences. The user then provides feedback allowing the recommenders to refine future recommendations with great accuracy. To implement this refinement process effectively in automated systems, it is crucial to understand both positive and negative user preferences. This dual perspective highlights the discrepancies between the user's interests and recommended items.

In this study, we propose an LLM-based recommendation method that emphasizes the selection effects of recommended items by analyzing users' negative preferences. We evaluate the effectiveness of our approach through experiments using the LLaMA3.1 and MovieLens 1M dataset, comparing performance with existing baseline methods.

Keywords: recommendation system; large language model; negative preference

1. Introduction

Recommendation systems are one of the most important applications in artificial intelligence. They provide users with useful information derived from large-scale datasets. These systems typically analyze users' interaction history, predict their preferences, and generate personalized recommendations. For example, in the context of movie recommendations, the system infers user preferences from their viewing history and ratings, and suggests that match those preferences. It then recommends new movies that align with these inferred preferences.

Recently, large language models (LLMs) have attracted increasing attention due to their strong language understanding and generalization abilities across a wide range of downstream tasks. LLM-based methods have owned high

E-mail address: xu.yunfan.518@s.kyushu-u.ac.jp

performance in many natural language processing(NLP) applications and are now considered as a mainstream approach. In Particular, LLM-based recommendation systems are gaining attention for their ability to improve the analysis and interpretation of text data in tasks such as recommendation, information retrieval, and dialogue systems[1].

LLM-based recommendation systems generally face several challenges:

(1) Task specialization: A base LLM, without fine-tuning contains broad language knowledge, but lacks the ability to accurately apply task-specific knowledge accurately. As a result, such models often struggle to perform specific tasks with high precision in recommendation tasks and maybe biased by the popularity of recommended items. To address this, two main techniques are used: prompt engineering and instruction tuning[2]. Prompt engineering improves performance by designing the prompt of the model. For example, the Chain-of-Thought (CoT)[3] method divides a task into smaller reasoning steps, helping the model stay focused. Instruction tuning, on the other hand, involves fine-tuning the LLM model with a specific instruction dataset to teach task-specific behavior. Modern systems, such as ChatGPT[4], are typically based on pre-trained model that has been instruction-tuned using a general-purpose datasets like Alpaca[5]. In practice, a combination of both techniques often yields the best results.

(2) Output quality and hallucinations: LLMs may generate incomplete or incorrect responses, often referred to as hallucinations[6]. These occur when the model produces fluent and seemingly plausible outputs based on language patterns even when it lacks factual knowledge. To reduce hallucinations, additional task-specific knowledge can be incorporated through fine-tuning. However, full fine-tuning of large models is computationally expensive. Therefore, efficient method such as Low-Rank Adaptation (LoRA)[7] are often used to reduce the cost by updating only a small portion of the model's parameters.

(3) Prompt length limitations: Although recent advancements have extended the number of tokens that LLMs can handle, performance with long prompts remains suboptimal[8]. This is because longer prompts tend to include more noise, which negatively affects output quality. Additionally, long prompts increase processing time, reducing the overall efficiency of the model. Therefore, managing prompt length is an important consideration when optimizing LLM-based recommendation systems.

In this study, we address above challenges by designing recommendation process that mimics how humans make recommendations. Based on this flow, we propose a LLM-based recommendation system. The process is described below.

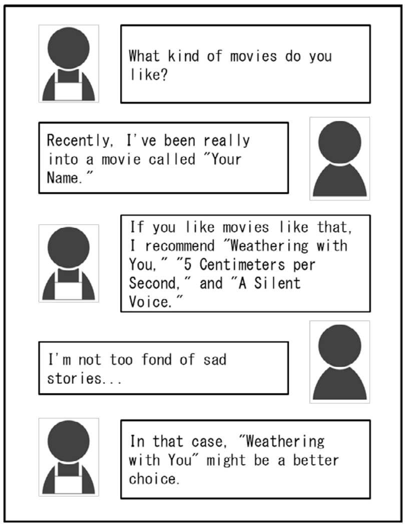
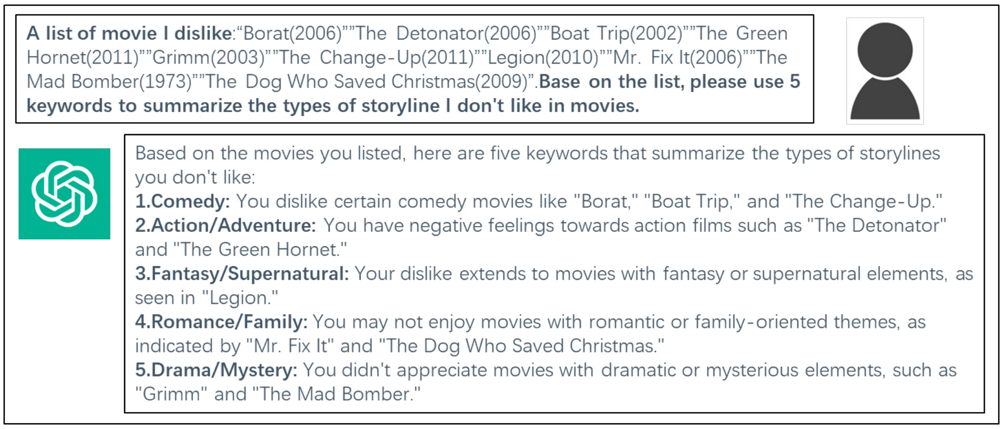

What kind of movies do you like?
Recently, I've been really into a movie called "Your Name."
If you like movies like that, I recommend "Weathering with You," "5 Centimeters per Second," and "A Silent Voice."
I'm not too fond of sad stories...
In that case, "Weathering with You" might be a better choice.

Fig. 1. Feedback In A Human-to-human Recommendation Processes

When people make recommendations, they generally follow a process similar to that illustrated in Figure 1. First, the recommender asks about the user's preferences. Then, they present recommended items along with explanations based on these preferences. The user responds with feedback taking into account both the explanations and their own preferences. Finally, the recommender uses this feedback to refine the list of recommended items, thereby improving recommendation accuracy.

In conventional methods, few studies focus on refining recommended items based on user feedback. A key challenge in this refinement process is capturing not only the users' positive preferences but also their negative preferences, which is essential for identifying the gap between users' interests and the recommended items. Balancing negative and positive samples in recommendation tasks is inherently difficult. Items rated poorly by users are known as hard negative samples[9]. Due to the scarcity of such samples, many traditional methods randomly treat items not included in the user's history as negative samples. However, this can introduce noise, which negatively impact on the accuracy of negative preference prediction. Additionally, analyzing preferences using only hard negative samples complicates model integration, as the score scales differ from those used in positive preferences and the original recommendation model.

We believe that the challenges inherent in traditional recommendation systems can be addressed by leveraging the strengths of LLM. LLMs perform exceptionally well in scenarios with limited data, such as zero-shot or few-shot learning, making them suitable for cold-start recommendation systems even when hard negative samples are scarce[10]. As illustrated in the Figure 2, ChatGPT's prediction of a user's negative preferences based on 10 movie viewing histories demonstrated that LLMs can effectively summarize negative preferences from a small number of negative samples. Furthermore, LLM-based methods process all data in natural language form, enabling unified processing by combining multiple model outputs.


A list of movie I dislike:"Borat(2006)""The Detonator(2006)""Boat Trip(2002)""The Green Hornet(2011)""Grimm(2003)""The Change-Up(2011)""Legion(2010)""Mr. Fix It(2006)""The Mad Bomber(1973)""The Dog Who Saved Christmas(2009)".**Base on the list, please use 5 keywords to summarize the types of storyline I don't like in movies.**

Based on the movies you listed, here are five keywords that summarize the types of storylines you don't like:
1.Comedy: You dislike certain comedy movies like "Borat," "Boat Trip," and "The Change-Up."
2.Action/Adventure: You have negative feelings towards action films such as "The Detonator" and "The Green Hornet."
3.Fantasy/Supernatural: Your dislike extends to movies with fantasy or supernatural elements, as seen in "Legion."
4.Romance/Family: You may not enjoy movies with romantic or family-oriented themes, as indicated by "Mr. Fix It" and "The Dog Who Saved Christmas."
5.Drama/Mystery: You didn't appreciate movies with dramatic or mysterious elements, such as "Grimm" and "The Mad Bomber."

Fig. 2. Prediction of negative preferences using LLM

Consequently, in this paper, we focus on refining recommended items based on user feedback, particularly exploring a recommendation method that predicts and utilizes negative preferences. The following diagram outlines the flow of proposed method, addresses key issues associated with LLM-based recommendation systems.

We divide the recommendation task into three steps and perform processing accordingly.

- We construct two types of models using the recommendation task dataset: a positive preference prediction model and a negative preference prediction model. To do this, we transform the dataset into a format suitable for instruction tuning and fine-tune the LLM using this data. Positive samples from the user's history are used for the positive model, while negative samples are used for the negative model.
- We generate prompts for the fine-tuned LLM based on the user history in the test data. To obtain recommendation candidates, we use a deep learning-based retrieval model. Then, we apply few-shot prompting with the LLM to generate both positive and negative preference lists.
- We combine the outputs of the two preference models with that of the retrieval model to perform weighted scoring. The final recommendation results, adjusted using preference prediction, are then combined with conventional methods.

In this study, we focus on movie recommendation as the target domain for the recommendation task. We use the MovieLens 1M dataset[11] and the LLaMA3.1-8B[12] language model. The performance of the proposed systems is evaluated by comparing it with conventional movie recommendation models.

The structure of this paper is organized as follows: Section 2 discusses related research on LLM-based recommendation models. Section 3 details the proposed method. Section 4 presents the experimental settings and results. Section 5 discusses the experimental findings, and Section 6 summarizes the study and outlines future work.

2. Related work

In recent years, recommendation systems utilizing generative AI have gained popularity. For example, the P5 model methodology[13] has been proposed as a representative approach. This method generates prompts for various types of recommendation-related tasks for the P5 language model, converts datasets into natural language, performs pre-training, and evaluates performance. This study outlines foundational methods for generative recommendation systems across recommendation tasks.

Unlike traditional recommendation systems that typically use feature vectors for user history and recommended items, LLM-based recommendation systems process user characteristics and item attributes in natural language form. Based on foundational LLMs, the task alignment with personalized recommendation tasks is typically achieved through the following processes.

Similar to traditional neural network-based recommendation techniques, LLMs can be aligned with downstream tasks for natural language processing through fine-tuning. However, due to the large scale of LLMs, full fine-tuning with complete data, as is done in traditional deep learning models, requires substantial computational resources. Therefore, LLM-based approaches in downstream tasks often adopt more efficient fine-tuning methods, such as LoRA[7] and Q-LoRA[21].

Additionally, the training data used for fine-tuning differs from that in traditional methods. While conventional approaches use structured data and require large datasets, LLM-based fine-tuning relies on instruction tuning with task-related prompts, requiring significantly less data[5]. Unlike traditional methods that input complete task knowledge, instruction tuning mainly draws out the model's inherent understanding about the task.

In the output phase, there are also methods to elicit the inherent knowledge of LLMs, such as in-context learning and prompting[14]. These methods mainly involve designing and controlling the context of model prompts given to the model. For example, by providing a small number of samples, the model can be guided to generate more accurate outputs. Leveraging the LLM's contextual understanding often results in improved textual output. Numerous studies on zero-shot and few-shot learning have also demonstrated the high feasibility of these approaches[15].

This study also employs the aforementioned methods in the system process but additionally introduces the concept of negative samples into the framework architecture. Below are some related studies that closely align with the objectives of this research.

In the study by Yang et al.[16], instruction data was created to predict users' positive preferences, and fine-tuning was performed using all parameters based on LLaMA-7B. Movie recommendations were generated from the candidates produced by a conventional retrieval model. The effectiveness of the LLM-based recommendation model was evaluated by comparing its performance with traditional recommendation models. Notably, this study focused primarily on methods centered on positive samples.

Bao et al.[17] employed LoRA to fine-tune LLaMA and evaluated performance using datasets from several popular recommendation tasks. In their study, the instruction specification included one positive item and one negative item from the user's history as input, and the output was a binary classification indicating whether the user liked or disliked a single test sample. Although this study handle negative samples as input, both positive and negative samples were limited in number. Additionally, the study focused on binary classification for individual samples, without considering addressing the refinement of recommendation lists based on user preferences.

Other methods for recommendation systems that apply LLMs often use LLM outputs for knowledge expansion within the recommendation model[19]. These approaches primarily aim to enhance recommendation accuracy by focusing on positive samples. They rarely address the prediction of negative preferences and the refinement of recommendation list based on such preferences. The impact of negative samples has been acknowledged in prior recommendation research, but the lack of hard negative samples has historically been a bottleneck. In this study, we aim to the generalization ability of LLMs to effectively handle negative samples. Our goal is to develop a recommendation system that considers both positive and negative user preferences.

3. Method

3.1. The framework of the proposed method

The framework of the proposed method is shown in Figure 3. First, based on the user's history, a candidate list is output by the retrieval model. Next, positive and negative samples are extracted from the user's viewing history, and preference prediction is performed using two fine-tuned models, which generate positive and negative preference lists from the candidate list. Finally, normalization is applied to each item in the candidate, positive, and negative lists, and the final recommendation list is produced using a weighted calculation.

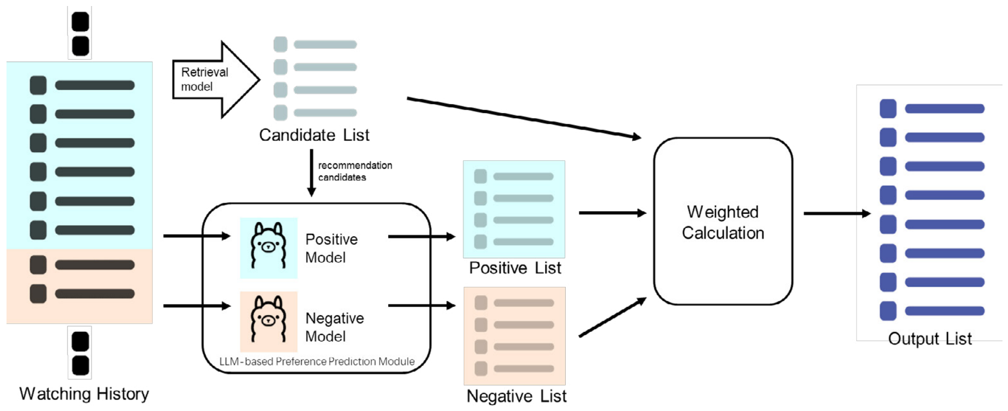


Fig. 3. The framework of our method

There are two reasons for using a retrieval model. The first is to serve as a base recommendation module. In human recommendation process, a preliminary recommendation list is typically presented first, and then refined based on user preferences. Similarly, in our system, the retrieval model is used to generate the base recommendation list. The second reason is to address the problem of prompt length. Prompts become too long if all items are treated as candidates. By limiting the number of candidates using the retrieval model, the prompt length can be reduced to a managed size. In this study, SASREC[20] is used as the retrieval model.

The LLM used for fine-tuning is LLaMA3.1-8B-Instruct, which was built with instruction tuning using general instruction data on LLaMA3.1-8B. Preliminary tests showed that LLaMA3.1-8B-Instruct outperformed the base LLaMA3.1-8B model. This is likely because LLaMA3.1-8B-Instruct has a better understanding of prompts. In related research, Bao et al.[17] and others also used LLaMA models tuned with general instruction data.

The instruction data used for fine-tuning is extracted from the training data of the MovieLens 1M dataset. Details of the fine-tuning process and the weight calculation will be discussed in the following section.

3.2. Fine-tuning

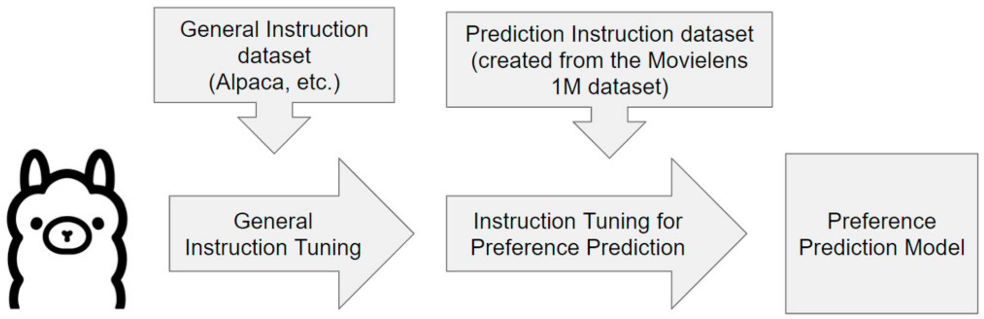

General Instruction dataset (Alpaca, etc.)
Prediction Instruction dataset (created from the Movielens 1M dataset)
General Instruction Tuning
Instruction Tuning for Preference Prediction
Preference Prediction Model

Fig. 4. The flow of fine-tuning

The flow of fine-tuning is depicted in Figure 4. The base model is tuned in two phases. First, the LLM's instruction comprehension ability is enhanced through general instruction tuning. Next, instruction data consisting instructions, input, and output is created from the MovieLens 1M dataset, and the dataset is organized accordingly. The organized dataset is then used to fine-tune the LLM into a preference prediction model.

From the MovieLens 1M dataset, the instruction data for the two types of preference prediction models are organized as follows.

Positive Model: The instruction data for the positive model is shown in Figure 5. The input consists of the user's 10 viewing history items, and the recommendation candidates are 20 other items not included in those 10 items. The output is the top 10 movies from those 20 candidates based on the user's rating.

```
Few-shot Example/instruction(positive):
Instruction: recommend 10 other movies and rank them by recommendation level based on user's
watching history from the candidate list , sorted by recommendation level.
input:User watched movies {watching history}.the candidates are {candidate list}.
output:{recommendation list}.
```

Fig. 5. The instruction data for the positive model

Negative Model: The instruction data for the negative model is shown in the figure 6. The input consists of 5 movies to which the user gave a low rating, and the recommendation candidates are 20 histories other than those 5 movies. The output is the bottom 10 movies from those 20 histories rated by the user. Because the number of negative ratings in the entire dataset is less than half that of the positive items, the history items in the negative model were set at half the size of those in the positive model.

```
Few-shot Example/instruction (negative):
Instruction: Predict 10 other movies that users will give low ratings to based on user's watching
history from the candidate list and rank them, for 1 is the worst one.
input:User gave low ratings to movies {watching history}.the candidates are {candidate list}.
output:{recommendation list}.
```

Fig. 6. The instruction data for the negative model

Fine-tuning on LLM requires a substantial amount of computation. Therefore, in this study, we used QLoRA[21] as the fine-tuning method. Additionally, given the LLM's strong generalization ability, we utilized only 20% of the training data for instruction tuning instead of the entire dataset.

3.3. Weighted Calculation

As shown in Figure 3, the model output consists of three ranking lists: (1) the base recommendation from the retrieval model(candidate list), (2) the positive list generated by the positive preference prediction model, and (3) the negative list from the negative preference prediction model. In the proposed method, the preference prediction models refine the base recommendation list by partitioning each item's score into two components: the base ranking score $Score_{rec}$ and the preference score $Score_{pref}$. These two scores are combined by using a weighting factor α, as expressed by the following formula:

$$Score_i = \alpha \times Score_{rec} + (1 - \alpha) \times Score_{pref} \tag{1}$$

The base ranking score is represented by the following formula:

$$Score_{rec} = \frac{1}{R_{base}} \tag{2}$$

R_{base} represents the rank in the base recommendation list.

The preference prediction score is calculated as the difference between the reciprocals of the ranks in the positive list and the negative list. Specifically, it is expressed by the following formula:

$$Score_{pref} = \frac{1}{R_{pos}} - \frac{1}{R_{neg}} \tag{3}$$

$Score_i$ for each item in the base recommendation list is calculated from the three output lists. A new ranking list is then generated from these scores and is treated as the output of the proposed method.

In this paper, considering the issue that LLMs are susceptible to general evaluations, we set the value of α to 0.6.

4. Evaluation

4.1. Experiment Setting

The dataset setting follows that of the baseline method, where items with interaction records are assigned a value of 1, and all other items are assigned a value of 0. The last interaction item for each user is considered the correct item, while all preceding items are treated as history. The training data for the model exclude the last two interaction records of each user; the penultimate record is used as validation data, and the final record is used as test data. The details of the MovieLens 1M dataset are shown in Table 1.

Table 1. The information of MovieLens 1M dataset

Users	item	interactions
6040	3883	1000209

The data for instruction tuning of the LLM was derived from the training dataset. To manage the length of the LLM prompt, the top 10 rated items were used as history for the positive preference prediction model, and the bottom 5 rated items were used as history for the negative preference prediction model.

4.2. Experiment Metrics

To verify the refinement effect of negative preferences, we used NDCG as an evaluation metric, which emphasizes the rank of correct items. The length of the output list was limited to 10 items.

4.3. *Baseline*

We compared the proposed method with the following three baseline models. GRU4REC[22] is an RNN-based sequential recommendation system that use GRU to predict the next item. SASREC[20] is a recommendation model predict the next item using a self-attention mechanism to account for changes in user preferences. Few-shot prompting is a method that directly inputs prompt data and outputs recommendation candidates without fine-tuning. It outputs results through few-shot prompting in the same format as the instruction data of the proposed method. The evaluation was conducted using the output results without performing weight calculations.

4.4. *Result*

Table 2. Comparison of performance with baseline approaches on Movielens 1M dataset

Model	NDCG@5	NDCG@10	NDCG@15
GRU4REC	0.4675	0.5095	0.5226
SASREC	0.5304	0.5602	0.5814
LLaMA3.1-8B-Instruct+few-shot	0.2481	0.2709	0.2709
Retrieval-positive	0.5375	0.5862	0.5887
Retrieval-positive+negative	**0.5432**	**0.5980**	**0.6005**

Table 2 shows the NDCG@5, NDCG@10, NDCG@15 results for each model. The proposed method achieved the highest performance across all metrics. In particular, the improvement over the few-shot model indicates that fine-tuning with instruction data reflecting user preferences including negative ones contributes to improved recommendation accuracy.

Conversely, performance was suboptimal when directly evaluating the output of the LLM without fine-tuning or weight calculation. Three primary factors contribute to this outcome. First, the LLM's recommendations are more influenced by public evaluations than those of traditional recommendation models. Second, numerous instances of hallucinations or generation failures were observed, likely due to prompt length or insufficient information. Finally, in the Top-N sequential recommendation task, items in the history are treated as positive samples and others as negative samples. Since rating information is not used, the correct data might include movies that the user rated poorly. As a result, because the LLM performs user preference predictions, this could negatively affect evaluation accuracy.

The proposed method mitigates the influence of public evaluations through fine-tuning. For items with numerous public evaluations, both high and low ratings increase, potentially causing them to be ranked highly in both the positive and negative preference prediction lists. By combining these two outputs and applying weight calculations, the influence of public evaluations can be reduced. Examination of the output of the fine-tuned LLM revealed a clear decrease in hallucination and generation failure instances. Additionally, adjusting the weights can partially mitigate the impact on correct samples. This issue will be discussed in detail in the next section.

5. Discussion

As mentioned earlier, in the Top-N recommendation task, the model predicts items of potential interest based on the user's viewing history. All previously watched movies are treated as positive samples, while unwatched movies are analyzed as negative samples. However, this approach does not take into account rating information.As a result, the ”correct” item in the test data may actually be a movie that the user rated poorly. In contrast, our preference prediction model generates recommendations based on the user's actual preferences. If the correct item is a poorly rated movie, it may act as noise and negatively affect the accuracy of recommendations. This may be one of the reasons why the performance of our method did not show a significant improvement over the baseline.

To examine the impact of this issue, we extracted 100 output samples from the proposed model. These are categorized into successful cases (where the rank of the correct item improved) and failure cases (where the rank declined). We then analyzed the user ratings of these items. The results are shown in the figures 7.

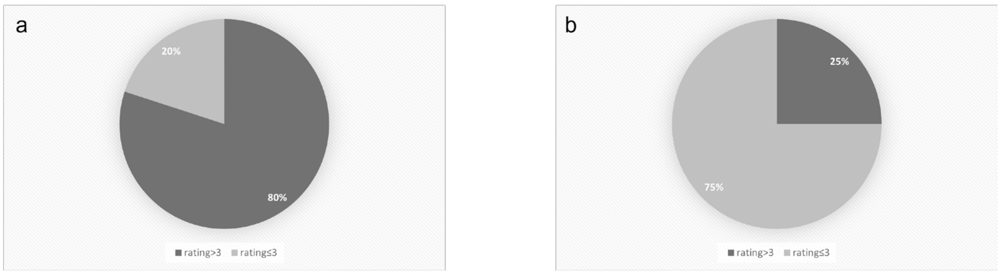
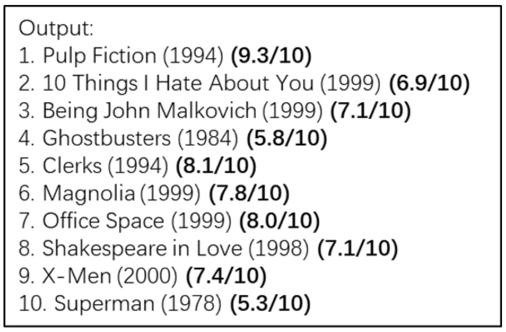


Fig. 7. (a) Rating distribution of successful examples of rank adjustment through preference prediction; (b) Rating distribution of failure examples of rank adjustment through preference prediction.

The results confirm that user preferences for the correct items have a significant impact on the evaluation results. Moreover, the experimental results support the effectiveness of the preference prediction model. This issue appears to seem from a mismatch between the evaluation task and the model's output. We believed that in a more suitable ranking task, the proposed method could achieve even better performance.

Figure 8 shows an example of a recommendation list generated by the proposed method. The score following each movie title represents its public rating. In general, movies with high public ratings tend to be ranked higher. However, the fact that some low-rated movies appear in the higher ranks demonstrates that fine-tuning has enabled the model to personalize recommendation based on individual user preferences.



```
Output:
1. Pulp Fiction (1994) (9.3/10)
2. 10 Things I Hate About You (1999) (6.9/10)
3. Being John Malkovich (1999) (7.1/10)
4. Ghostbusters (1984) (5.8/10)
5. Clerks (1994) (8.1/10)
6. Magnolia (1999) (7.8/10)
7. Office Space (1999) (8.0/10)
8. Shakespeare in Love (1998) (7.1/10)
9. X-Men (2000) (7.4/10)
10. Superman (1978) (5.3/10)
```

Fig. 8. An example of recommendations generated by the proposed method

Analyzing the overall recommendation output reveals that the LLM fine-tuned with general instruction data produces less stable results compared to the base LLM. Many outputs were generated in formats that deviated slightly from the intended instructional format. Although we attempted to reduce prompt length, it remained relatively long due to the need to include both viewing history and the candidate list. As a result, issues such as hallucinations and duplicated output still occurred. In future work, we plan to reduce prompt length further and improve output accuracy by using a multi-prompt output chain.

6. Conclusion

In this study, we proposed a recommendation method using an LLM, inspired by human recommendation process that refines recommendation lists based on user preference prediction. Since refinement involves comparing differences, it is essential to consider both positive and negative preferences. We utilized the LLM's strong generalization and language understanding capabilities to build the preference prediction model. The model was personalized through the fine-tuning and prompt length was controlled using a retrieval model. By combining the prediction results of positive and negative preferences through weighted calculation, we refined the recommendation list. Evaluation using the MovieLens 1M dataset, showed that the proposed method outperformed existing models, conforming its effectiveness.

Looking ahead, we aim to develop an LLM-based conversational recommendation system following the recommendation flow proposed in this study. Conversational recommendation systems typically involve four key tasks:

”Request”, ”Recommendation”, ”Explanation” and ”Response”[23]. This paper focused on the feedback process and the adjustment of recommendation item rankings. In the future work, we plan to leverage the LLM’s language understanding capabilities to address the remaining tasks. We will explore prompt-based approaches, integrate outputs across different models, and ultimately build a highly accurate conversational recommendation system.

References

[1] Fan, W., Zhao, Z., Li, J., Liu, Y., Mei, X., Wang, Y., ... Li, Q. (2023). Recommender systems in the era of large language models (llms). arXiv preprint arXiv:2307.02046.

[2] Wu, L., Zheng, Z., Qiu, Z., Wang, H., Gu, H., Shen, T., ... Chen, E. (2023). A Survey on Large Language Models for Recommendation. arXiv preprint arXiv:2305.19860.

[3] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems, 35, 24824-24837.

[4] https://chat.openai.com

[5] Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., ... Hashimoto, T. B. (2023). Stanford alpaca: An instruction-following llama model.

[6] Rawte, V., Sheth, A., Das, A. (2023). A survey of hallucination in large foundation models. arXiv preprint arXiv:2309.05922.

[7] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... Chen, W. (2021). Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685.

[8] Li, L., Zhang, Y., Chen, L. (2023, October). Prompt distillation for efficient llm-based recommendation. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (pp. 1348-1357).

[9] Ma, H., Xie, R., Meng, L., Chen, X., Zhang, X., Lin, L., Zhou, J. (2023, September). Exploring false hard negative sample in cross-domain recommendation. In Proceedings of the 17th ACM Conference on Recommender Systems (pp. 502-514).

[10] Sanner, S., Balog, K., Radlinski, F., Wedin, B., Dixon, L. (2023, September). Large language models are competitive near cold-start recommenders for language-and item-based preferences. In Proceedings of the 17th ACM conference on recommender systems (pp. 890-896).

[11] https://grouplens.org/datasets/movielens/1m/

[12] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Ganapathy, R. (2024). The llama 3 herd of models. arXiv preprint arXiv:2407.21783.

[13] Geng, S., Liu, S., Fu, Z., Ge, Y., Zhang, Y. (2022, September). Recommendation as language processing (rlp): A unified pretrain, personalized prompt predict paradigm (p5). In Proceedings of the 16th ACM Conference on Recommender Systems (pp. 299-315).

[14] Brown, T. B. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

[15] Wang, Lei, and Ee-Peng Lim. ”Zero-shot next-item recommendation using large pretrained language models.” arXiv preprint arXiv:2304.03153 (2023).

[16] Yang, F., Chen, Z., Jiang, Z., Cho, E., Huang, X., Lu, Y. (2023). Palr: Personalization aware llms for recommendation. arXiv preprint arXiv:2305.07622.

[17] Bao, K., Zhang, J., Zhang, Y., Wang, W., Feng, F., He, X. (2023). Tallrec: An effective and efficient tuning framework to align large language model with recommendation. arXiv preprint arXiv:2305.00447.

[18] Zhang, J., Xie, R., Hou, Y., Zhao, W. X., Lin, L., Wen, J. R. (2023). Recommendation as instruction following: A large language model empowered recommendation approach. arXiv preprint arXiv:2305.07001.

[19] Lyu, H., Jiang, S., Zeng, H., Xia, Y., Luo, J. (2023). Llm-rec: Personalized recommendation via prompting large language models. arXiv preprint arXiv:2307.15780.

[20] Kang, W. C., McAuley, J. (2018, November). Self-attentive sequential recommendation. In 2018 IEEE international conference on data mining (ICDM) (pp. 197-206). IEEE.

[21] Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms. arXiv preprint arXiv:2305.14314.

[22] Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D. (2015). Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939.

[23] Jannach, D., Manzoor, A., Cai, W., Chen, L. (2021). A survey on conversational recommender systems. ACM Computing Surveys (CSUR), 54(5), 1-36.