PepRec: Progressive Enhancement of Prompting for Recommendation

Yakun Yu
University of Alberta
yakun2@ualberta.ca

Shi-ang Qi
University of Alberta
shiang@ualberta.ca

Baochun Li
University of Toronto
bli@ece.toronto.edu

Di Niu
University of Alberta
dniu@ualberta.ca

Abstract

With large language models (LLMs) achieving remarkable breakthroughs in natural language processing (NLP) domains, recent researchers have actively explored the potential of LLMs for recommendation systems by converting the input data into textual sentences through prompt templates. Although semantic knowledge from LLMs can help enrich the content information of items, to date it is still hard for them to achieve comparable performance to traditional deep learning recommendation models, partly due to a lack of ability to leverage collaborative filtering. In this paper, we propose a novel training-free prompting framework, *PepRec*, which aims to capture knowledge from both content-based filtering and collaborative filtering to boost recommendation performance with LLMs, while providing interpretation for the recommendation. Experiments based on two real-world datasets from different domains show that *PepRec* significantly outperforms various traditional deep learning recommendation models and prompt-based recommendation systems.

1 Introduction

Recommendation systems, which aim to estimate the likelihood of a user interacting with an item, have been widely deployed in the real world, including e-commerce (Hussien et al., 2021), social media platforms (Yu et al., 2023), and news aggregators (Zhang and Wang, 2023). Deep learning-based recommendation models (DLRMs) have been the cornerstone and achieved state-of-the-art performance in recommendation, leveraging a wide array of user and item attributes, including both categorical and numerical features (Wang et al., 2022; Lin et al., 2022; Qu et al., 2022).

Recently, large language models (LLMs) have been particularly transformative, achieving unparalleled performance across various NLP tasks (Touvron et al., 2023; Brown et al., 2020; Bubeck et al., 2023; Wei et al., 2022; Besta et al., 2024). They are usually pretrained on a large corpus of text and thus can offer rich semantic knowledge to help improve recommendations via content-based filtering. For example, two movies "Oppenheimer" and "The Dark Knight" from a user's item-interacted history, despite having completely different film names and genres – where the first is a biographical thriller film and the latter is a superhero film – can have some hidden commonality as they are both directed by Christopher Nolan. This commonality, among other hidden common features, can be obtained from LLMs and be used to help recommend if the user may watch another new movie sharing the same content characteristic. In addition, LLMs can provide explanations for recommendations to enhance transparency and even do not require model training compared to DLRMs.

Recent research has explored the use of LLMs in recommendation systems (Dai et al., 2023; Liu et al., 2023; Zhang and Wang, 2023), usually converting the categorical features into textual sentences through prompt templates and feeding these sentences into LLMs to generate a prediction. For example, LLM4RS (Dai et al., 2023) proposes three types of prompts for point-wise, pair-wise, and list-wise ranking, respectively, to explore LLM capability in recommendation. However, prompts to LLMs are designed heuristically in existing works only based on a user's historical interactions with items, and are unable to achieve results comparable to traditional DLRMs. Part of the reason is that LLMs, while offering content analysis and filtering capability, do not perform any collaborative filtering between users (i.e., recommending a user the items on the basis of the preferences of other users with similar tastes) like conventional DLRMs would do. Though some works (Lin et al., 2023; Li et al., 2023) propose to train a DLRM and an LLM concurrently in a unified framework for alignment to solve this issue, they usually come with pretrain-

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 17941–17953

ing and fine-tuning, which are computational-costly and time-consuming.

In contrast, we propose a training-free Progressive Enhancement of Prompting (*PepRec*) framework to achieve outstanding recommendation performance with LLMs, by introducing a novel method to progressively discover a nuanced understanding of a user's likes and dislikes. *PepRec* distinguishes itself from prior prompt-based recommendations by enabling both fine-grained semantic knowledge mining from a user and collaborative filtering from other similar users to improve the performance of the recommendations. It iteratively recognizes and leverages the collective preferences of similar user cohorts, thereby enriching individual users' likes and dislikes with broader communal insights. On the other hand, compared to DLRM-based recommendation, *PepRec* inherits the strengths of LLMs, particularly their ability to extract and summarize hidden content traits from items, which not only helps boost the prediction but also makes recommendation outcome human-comprehensible. Our contributions through this work are threefold:

1. We design a preference generator that relies on LLMs themselves to generate short descriptions characterizing likes/dislikes based on the item-interaction histories of a user.

2. We further propose a method to enable both content filtering and collaborative filtering with LLMs, by progressively enhancing prompting with features of likes and dislikes discovered by reasoning other similar users. Specifically, we design a bootstrapping strategy to iteratively extract and aggregate preferences generated from the item-interaction histories of sampled similar users until no additional insights can be obtained.

3. We conduct extensive experimental evaluations of the *PepRec* framework on two real-world benchmark datasets. The results suggest that our approach significantly outperforms various traditional DLRMs and prompt-based recommendation systems on both datasets at a reasonable cost.

Our findings with *PepRec* underscore the potential of LLMs to significantly boost recommendation performance over various state-of-the-art recommendation methods, which to the best of our knowledge, is not achieved by the prior literature. In the meantime, *PepRec* provides intuitive and human-understood explanations for recommendation outcomes, thereby adding a layer of transparency to recommendation process, while maintaining operational efficiency (characterized by reduced training requirements and minimal training parameters).

2 Related Work

In this section, we review the existing related work on DLRMs and prompt learning.

2.1 Deep Learning Recommendation Models

DLRMs aim to estimate the likelihood of a user clicking on an item (e.g., an advertisement or link). They follow a common design framework: embedding layer, feature interaction layer, and prediction layer. Specifically, the embedding layer converts the categorical inputs to high-dimensional dense vectors via embedding tables. The feature interaction layer usually consists of a deep neural network (DNN) or factorization machine (FM) to learn the inner connection between users and items. For instance, DeepFM (Guo et al., 2017), and DCN (Wang et al., 2017) are proposed to capture high-order feature interactions by applying product operators to feature interactions. Autoint (Song et al., 2019) suggests adopting the attention mechanism to make features contribute differently to improve the final prediction.

Although the above methods have achieved remarkable progress in recommendation systems, they need lots of parameters to be trained and show no explainability for the prediction results. In contrast, our method is parameter-free, thus lowering the computation costs without using any feature selection mechanism (Yu et al., 2023; Wang et al., 2022; Lyu et al., 2023; Lin et al., 2022; Qu et al., 2022). Besides, our method can provide human-comprehensible explanations to solve the recommendation problem more intuitively; see the case study in Figure 3.

2.2 Prompt Engineering

LLMs, e.g., GPT (Brown et al., 2020; Bubeck et al., 2023), LLaMA (Touvron et al., 2023) and PaLM (Chowdhery et al., 2023) have led to remarkable success in the world of AI recently. Prompt engineering is a resource-efficient approach for LLM tasks (Besta et al., 2024). It first manually formulates a task description as a prompt, then sends the prompt to an LLM, and finally generates solutions using the LLM's autoregressive

token-based mechanism for the task. There are a few works that explore the prompting ability for recommendation systems. Different from traditional DLRMs, the categorical input is transformed into textual sentences as a prompt. For example, LLM4RS (Dai et al., 2023) proposes three kinds of prompts for point-wise, pair-wise, and list-wise ranking respectively, to uncover LLM capabilities in recommender systems. TALLRec (Bao et al., 2023) proposes to tune LLMs with their designed instruction data via pretraining and fine-tuning. DOKE (Yao et al., 2023) proposes prompts with domain knowledge, e.g., some co-occurred item descriptions from a user's item-interaction history and knowledge graphs, to rank items.

Our paper also focuses on prompt engineering for recommendation systems, however, we do not heuristically design a prompt template based on a user's item-interaction history or external knowledge graphs. Instead, we encourage our framework to behave as human thinking, i.e., automatically progressively understanding a user's preferences via carefully analyzing the commonality between items within a user and the commonality between similar users, without any pretraining or fine-tuning. Retrieval-Augmented Generation (RAG) typically relies on external knowledge sources (e.g., Wikipedia) to retrieve relevant documents, which are then used to improve generation (Gao et al., 2023). In contrast, our method is entirely self-contained, operating without reliance on external datasets or knowledge sources. Given this difference, comparisons with RAG-based methods may not be fully representative.

3 Method

In this section, we introduce the details of our proposed *PepRec* framework, including an in-depth overview, and its core components.

3.1 Overview

We focus on the click-through rate (CTR) prediction, which serves as the core component in recommendation systems to estimate a user's click probability toward a target item. A typical CTR dataset is formed by a series of triplets $\mathcal{D} = \{(U_i, I_j, Y_{i,j})\}_{i=1,j=1}^{N,M}$, where U_i represents the i-th user characterized by a unique ID and a set of meta-features such as age, gender, and occupation, I_j represents the j-th item possessing a unique ID and associated meta-features like title, $Y_{i,j}$ denotes the binary indicator which reflects user U_i's preference for item I_j. N and M are the number of users and items, respectively.

The methodology of the *PepRec* framework is segmented into two major phases: the training and the inference phases. The training phase is to train clusters based on the extensive user-item interaction histories available (with the detailed procedure introduced in Section 3.2). This critical step ensures that users are grouped into well-defined clusters, allowing the system to offer personalized recommendations that extend beyond the confines of an individual's history.

Once the clusters are fitted, the system transitions to the inference phase for evaluating new user-item pairs (U_i, I_j), as illustrated in Figure 1. It begins by locating the appropriate cluster for the test user, and therefore finding different similar user lists in the same cluster. Following this, *PepRec* employs a preference generator to distill the user's preferences as an initial prompt. Then it seeks to refine this initial prompt by incorporating insights from similar users within the same cluster. By iterative sampling similar users and summarizing their item interaction histories, *PepRec* enhances the initial prompt, ensuring this iterative enrichment continues until the prompt stabilizes (as no information gain), indicating a convergence of preferences. This enhanced prompt is then presented to the LLM along with the target item information to predict the probability of the user's affinity for the item. Two examples of the enhanced prompt in each iteration are shown in Figure 3.

3.2 User Clustering

The collective preferences and behaviors of a group can often provide more insights to help refine a single individual's potential interest. With this premise, our method first incorporates a comprehensive cluster analysis for users within the training dataset.

For any given user, U_i, the process begins with the searching of his/her historical interacted items, resulting in a collection of data tuples $(U_i, I_j, Y_{i,j})_{j=1}^{M}$, where I_j represents items and $Y_{i,j}$ is the associated interaction label. This step is followed by the information aggregation of the user's meta-features and item history into a cohesive textual representation (see Figure 4 in Appendix A). Subsequently, this aggregated text is transformed into a vector representation through an encoding layer using the sentence-transformer (Reimers and

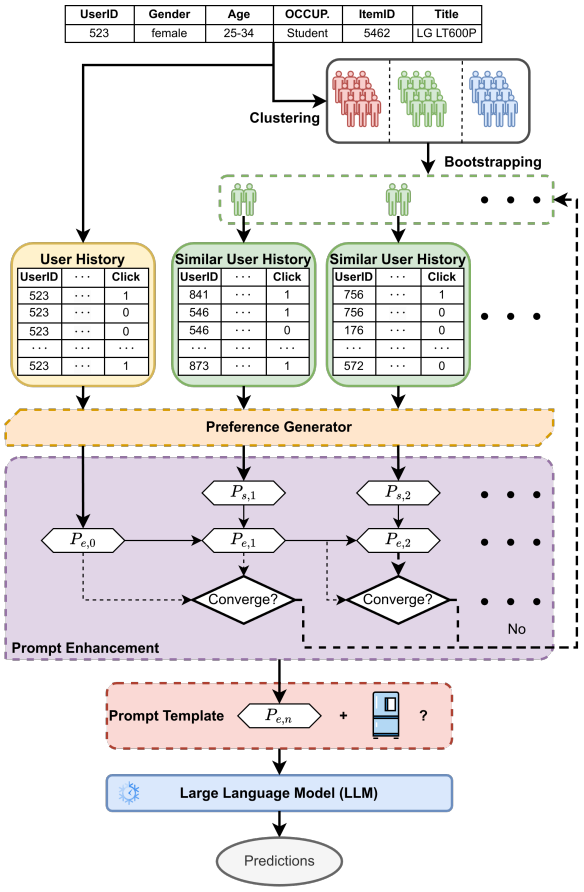

Figure 1: The detailed process of *PepRec* during the inference stage. The snowflake means the LLM is fixed (no need for training). And the ellipsis represents that the processing (bootstrapping, preference generator, and prompt enhancement) continues until convergence.

Gurevych, 2019). These contextualized vectors are then averaged to produce a fixed-size representation for each user.

Armed with these vectorized representations, the next step employs the K-Means clustering algorithm (Sculley, 2010), partitioning the user profiles into K distinct clusters based on their aggregated information. Once the clusters have been established, we can identify the set of users $\mathcal{S}_k = [U_{k,1}, U_{k,2}, \ldots]$ associated with the k-th cluster. The users within each cluster $\mathcal{S}_k$ are then ordered in ascending proximity to the cluster's centroid, facilitating a structured approach to identifying and leveraging similarities among users. This user clustering stage is summarized in Algorithm 2 in Appendix A.

3.3 Bootstrapping

Once the clustering process is completed and the sorted sets of users for each cluster have been established, we proceed to the inference stage on the test set. This stage involves identifying different subsets of similar users within a cluster whose interaction histories are regarded as closely resembling the preferences of the test user. The objective is to extrapolate the potential preferences of the test user by analyzing the aggregated interaction histories of these similar users. However, a practical challenge arises in maintaining a balance between the constraints of limited computational resources and the costs associated with prompt generation, as well as the goal of comprehensively incorporating the interaction histories of all similar users.

To address this challenge, we introduce a bootstrapping module that samples a fixed number of similar users each time to iteratively refine the target user's preferences, as shown in Figure 1. We first classify the target user into one of the predefined clusters, ensuring that they are grouped with similar users. Then we iteratively select α users from the sorted list within the identified cluster, denoted as $\mathcal{S}_k$. This selection is executed without replacement to maintain the uniqueness of each sampled user. Subsequently, from the collective user-item interactions of these α users, we extract the recent β interactions as the whole information of these users. This approach allows for the integration of diverse interaction histories from different users, enriching the information used for inferring the target user's preferences.

The bootstrapping process is iteratively conducted until a convergence criterion is met, indicating that the sampled interactions sufficiently represent the potential preferences of the target user (see detailed discussion in Section 3.5). This iterative refinement ensures that the model's predictions are closely aligned with the nuanced preferences of the target user, thereby enhancing the performance of the recommendation system.

3.4 Preference Generator

The detailed process of the preference generator is graphically depicted in Figure 2. Initially, the generator categorizes user-item interactions into positive (likes) and negative (dislikes) feedback. To systematically process the positive user history, we employ a structured prompt template $T_{like}(\cdot)$, which serves to enumerate these interactions with a human-readable format. Subsequently, this catalog of liked interactions is succinctly summarized as a short preference summary P_1' with the aid of the LLM, encapsulating the essence of user preferences. The template $T_{like}(\cdot)$ is also enriched with

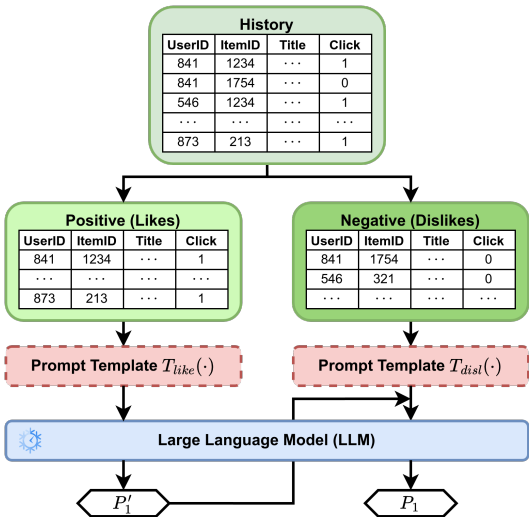

Figure 2: Demonstration of the preference generator, using a similar user history. The snowflake means the LLM is fixed (no need for training).

illustrative examples that serve as guiding beacons for LLM. For example, for beauty products, the illustrative cues can be "She likes beauty products for anti-aging.", or "He likes beauty products that contain fragrance." etc.

This summary, alongside the enumerated list of disliked interactions crafted using another prompt template $T_{disl}(\cdot)$, is then refined through further LLM processing to distill a refined summary of user preferences. It likes that LLM takes a second round of processing, wherein it refines the initial summary by critically analyzing and integrating the elements common to both liked and disliked interactions. For instance, if the initial summary highlights a feature that, upon further scrutiny, is found to be prevalent in disliked interactions as well, the refinement process will adjust the summary to negate this feature from the positive attributes. The refined summary will be used in subsequent stages to evolve with similar users' information. Notable, when extracting the preferences of users, due to the extensive user-item interaction history and the token limitation of the LLM, we select the recent interactions to infer the summary. We show the details of these prompt templates in Appendix B.

Remark LLMs can understand beyond mere brand and product names thus they possess the remarkable capability to distill and comprehend meta-information from user-item interactions. It is because, during the training phase, LLMs are trained using a comprehensive dataset encompassing a wide array of textual content across the websites. This exposure enables LLMs to capture and understand even the most nuanced product details (not listed in the datasets), such as energy efficiency ratings or cost considerations. An example of this capability is presented in Figure 3, where an LLM adeptly identifies critical attributes of an item, including its organic ingredient for the beauty product and the size as well as colors for appliances. Therefore, our prompt provides a layer of interpretability to the recommendations. This interpretability enhances the transparency and trustworthiness of the recommendation system, offering users not just personalized suggestions but also insightful explanations behind each recommendation.

3.5 Prompt Enhancement

Just as the brain assimilates a wealth of sensory data from its surroundings, our system undertakes a comprehensive collaborative filtering phase. This involves acquiring detailed information from both the target user and a cohort of similar users. Initially, we use the preference generator to distill the target user's interaction history into a preference prompt, denoted as $P_{e,0}$. This personal preference prompt is deemed crucial as it directly mirrors the user's interests and inclinations.

Parallel to this, we extend our collaborative filtering to encompass the experiences of similar users within the same cluster. Then we systematically bootstrap β user-item interact histories from the organized user list $\mathcal{S}_k$ and process these through the preference generator. This additional layer of information is particularly valuable when the primary user lacks direct interactions with similar users, allowing us to infer potential preferences based on the analogous experiences of similar users.

Following the above steps, the system combines the personal preference prompt $P_{e,0}$ with that of the first batch of similar users $P_{s,1}$ into an enhanced prompt $P_{e,1}$. This composite prompt undergoes a similarity analysis with the original user prompt $P_{e,0}$ to gauge the incremental value of the aggregated information. We will first compute the vector representation ($P_{e,n} \rightarrow \boldsymbol{V}_{e,n}$) for these two summaries using the same encoding layer we mentioned earlier. Then we compute the cosine similarity between the two vectors to quantify the extent of new insights gained through aggregation: $\cos(\boldsymbol{V}_{e,0}, \boldsymbol{V}_{e,1}) = \frac{\boldsymbol{V}_{e,0} \cdot \boldsymbol{V}_{e,1}}{\|\boldsymbol{V}_{e,0}\| \cdot \|\boldsymbol{V}_{e,1}\|}$. A high similar-

Algorithm 1 Progressive Enhancement of Prompting (Inference)

Require: Test user U_{test}, test item I_{test}, convergence threshold γ, K fitted clusters, sorted user sets $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_K\}$ for each cluster, the preference generator, the encoding layer, and an LLM model

Ensure: Prediction for the user's preference

procedure PEPREC($U_{\text{test}}, I_{\text{test}}, \gamma$)
  Extract user-item history for U_{test}
  Generate initial prompt $P_{e,0}$ using preference generator
  Assign U_{test} to the k-th cluster
  Initialize iteration counter $n = 0$, and similarity score $\sigma = 0$
  while $\sigma < \gamma$ **do** ▷ until convergence
    $n = n + 1$
    Sample (without replacement) β user-item interactions from the top α central users in $\mathcal{S}_k$
    Generate prompts $P_{s,n}$ from β interactions using preference generator
    Enhance prompt $P_{e,n} = P_{e,n-1} + P_{s,n}$
    $\sigma = \cos(\boldsymbol{V}_{e,n}, \boldsymbol{V}_{e,n-1})$
  end while
  With the final enhanced prompts $P_{e,n}$ and test item I_{test}, prompt an LLM to predict the user's preference probability
  return prediction
end procedure

ity score ($\cos(\boldsymbol{V}_{e,0}, \boldsymbol{V}_{e,1}) \to 1$) indicates minimal informational gain, potentially signaling the cessation of further data aggregation from similar users.

The iterative process of prompt enhancement is governed by a predefined threshold, serving as a regulatory mechanism for the informational gain in each cycle. Should the gain remain substantial, the system proceeds to incorporate additional batches of similar users' preferences, continuously refining the enhanced prompt until convergence is achieved or no further significant insights are obtained. This iterative refinement, akin to the human brain's feedback and learning cycle, ensures the recommendation system dynamically adapts and evolves based on the accumulation and integration of new information. The overall algorithm for the inference stage of the *PepRec* framework is described in Algorithm 1.

4 Experiment

In this section, we present the experimental details, including datasets, evaluation metrics, baseline models, implementation details, and experimental results in order to answer the following research questions:

- **RQ1**: How does our proposed method perform compared with current prompt-based models and traditional DLRMs?
- **RQ2**: What are the influences of different components for our proposed method?
- **RQ3**: How do different hyperparameter settings affect our proposed method?
- **RQ4**: Does our proposed method make interpretable and reasonable predictions?

4.1 Datasets

We conduct experiments on two real-world recommendation datasets (Ni et al., 2019), i.e., Beauty and Appliance[1]. They contain a collection of user-item interactions on Amazon spanning from 1996 to 2018. Table 3 offers an overview of the dataset statistics; see Appendix C.1 for preprocessing details. The datasets are split into training/validation/testing sets based on the proportions of 80%, 10%, and 10%. The validation set is used for the baseline models. All our experimental results are obtained from the testing set.

4.2 Evaluation Metrics

In line with prior research (Dai et al., 2023; Yu et al., 2023; Xi et al., 2023; Liu et al., 2023), we evaluate the performance of our method using two common metrics: AUC and Logloss. AUC, or the area under the ROC curve, measures the probability that a model rank a positive instance above a negative one, with higher values signifying superior performance. Logloss, aka binary cross-entropy loss, evaluates the accuracy of probability predictions, with a lower score denoting more accurate predictions.

4.3 Baselines and Implementation

We extensively compare our proposed method with 9 baseline methods, including 5 traditional DLRMs and 4 prompt-based approaches. Appendix C.2 introduces the details of these approaches and Appendix C.3 introduces the implementation details.

[1] https://jmcauley.ucsd.edu/data/amazon/

Dataset	Metrics	Methods									
		FM	DeepFM	WD	FNFM	AFN	LLM4RS	LLMRec	TALLRec	ReLLA	Ours
Beauty	AUC ↑	0.5485	0.5804	0.6746	0.5791	0.5009	0.5936	0.4348	0.5127	<u>0.6976</u>	**0.7318**
	Logloss ↓	0.9462	0.9105	0.7625	0.8386	0.7837	0.8892	<u>0.6907</u>	0.7649	0.7510	**0.5545**
Appliance	AUC ↑	0.5775	0.7167	0.6980	0.6259	0.6990	0.6302	0.4552	0.5512	<u>0.7386</u>	**0.7501**
	Logloss ↓	0.6389	0.5535	0.4950	0.4992	0.5390	0.5170	0.5972	<u>0.4905</u>	0.9532	**0.4871**

Table 1: Comparison of overall performance. The best results are given in bold, while the best baseline performance is underlined.

4.4 Overall Performance (RQ1)

Table 1 shows the overall performance, from which we can obtain the following observations:

- Our proposed method shows significantly better performance compared to the traditional DLRMs on both datasets. This indicates that when *PepRec* is equipped with collaborative filtering, similar to DLRMs, *PepRec* can make more accurate recommendations, thanks to the content-based filtering capability offered by the pretrained LLM.

- Our proposed method outperforms the prompt-based baselines, LLMRec and LLM4RS, where the LLM is fixed across all datasets. This indicates that the simple prompts designed based on a user's past item interactions may not be sufficient to capture their interests. This observation is further supported by the comparison of LLM4RS with WD, where LLM4RS's performance is worse than WD. Instead, by progressively refining the prompts through the integration of both the user's and similar users' information, as demonstrated in *PepRec*, we can more accurately and comprehensively infer the user's interests.

- Our proposed method performs better than both TALLRec and ReLLA, where the LLM is tuned by instruction data in recommendation. This again validates the effectiveness of our proposed method as it is capable of both content filtering and collaborative filtering yet these baselines are unable to achieve even with recommendation data fine-tuning the LLM.

- In addition to the superior performance, our proposed method needs no training and no parameters as compared to the baselines, which is very beneficial for recommendation systems where saving computation cost is highly required (Yu et al., 2023; Qu et al., 2022).

Metrics	Methods				
	PepRec-1	*PepRec*-2	*PepRec*-3	*PepRec*-4	Ours
AUC ↑	0.5740	0.6689	0.6856	0.7400	**0.7501**
Logloss ↓	1.1824	1.0265	0.7427	0.5479	**0.4871**

Table 2: The ablation study of different components on the Appliance dataset.

4.5 Ablation Study (RQ2)

In this section, we conduct the ablation study of key modules in our proposed method, as shown in Table 2. We derive four variants from *PepRec*: (i) *PepRec*-1: this variant only uses the positive item-interaction history of a user to form a preference prompt to infer the likelihood of the user liking a new item. (ii) *PepRec*-2: this variant only uses both the positive and negative item-interaction histories of a user, which are sequentially concatenated to form the preference prompt for inference. (iii) *PepRec*-3: this variant applies the preference generator only based on a user's own positive and negative item-interaction histories to generate the preference prompt for inference. (iv) *PepRec*-4: this variant applies the preference generator based on a user's own history and a similar user subset's history. Thus, it does not involve bootstrapping multiple similar user subsets to enhance the generated prompt.

From the table, we can find: (i) Using both positive and negative item-interaction histories is more beneficial than solely using the positive item-interaction history by comparing *PepRec*-1 to *PepRec*-2. (ii) *PepRec*-3 outperforms *PepRec*-2, indicating the necessity of the preference generator for boosting performance. (iii) *PepRec*-4 performs better than *PepRec*-3, therefore, the similar users' item-interaction history can be beneficial to enhance the user's preference prompt. (iv) *PepRec* outperforms *PepRec*-4, which shows that bootstrap-

	Item, I_j	No Enhancement, $P_{e,0}$	Enhanced after 1 iter, $P_{e,1}$	Enhanced after 2 iter, $P_{e,2}$
Case 1: Beauty	Aquaphor Healing Ointment,Advanced Therapy Skin Protectant Label: 1	- The user appreciates beauty products that are organic and made in the USA. - The user prefers Aloe Vera Gel from Nutrilab Naturals, especially for skin conditions like eczema and sunburn. (...) - The user is not a fan of beauty products from the brand Queen Helene. Pred: 0.5	Includes the following in addition to $P_{e,0}$: - They enjoy **skincare items** like Elbahya Regenerating Active 5 Serum and Clear Lift Revitalizing Serum. - They use hair and **body care products** like NuSkin Epoch Baby Hibiscus Hair & Body Wash and Aquaphor **Healing Ointment**. - They do not favor oral care products Pred: 0.8	N/A (converge at $P_{e,1}$)
Case 2: Appliance	Culligan IC 1 Icemaker and Refrigerator Filtration System Label: 1	- The user prefers white appliances. - The user prefers appliances that are 30 inches in size. (...) - The user has a preference for appliances that can be installed under cabinets. Pred: 0.28	Includes the following in addition to $P_{e,0}$: - They favor Frigidaire elements for range and switches for range. - They are interested in Broan range hoods and Whirlpool **refrigerator compatible** air filters. - They are inclined towards Whirlpool **icemakers** and drip pan kits. (...) Pred: 0.48	Includes the following in addition to $P_{e,1}$: - They are inclined towards purchasing **replacement parts** and aftermarket products for their appliances. - They also show preference for **refrigerator parts** such as crisper pans, shelves, and door racks. (...) Pred: 0.58

Figure 3: Examples of prompt enhancement process on two datasets where keywords in enhanced prompts are highlighted in blue. The eclipse (...) represents the omitted summaries that does not affect the conclusion.

ping multiple similar user item-interaction histories for prompt enhancement can further improve the performance.

4.6 Hyperparameter Analysis (RQ3)

PepRec is influenced by three key hyperparameters: (1) the number of clusters; (2) the number of users when bootstrapping a similar user subset, and (3) different LLMs. While Appendix C.4 provides complete experimental details, results, and summaries, the main findings are:

1. A few clusters and sampling users are enough to derive an enhanced prompt to maintain good performance.

2. The backbone LLM model has a significant impact on performance.

4.7 Case Study (RQ4)

In this subsection, we present a case study to demonstrate the effectiveness and interpretability of *PepRec*. We illustrate this with two examples, one from each dataset, as depicted in Figure 3.

The first example involves the recommendation of a beauty product (a skin repair item) shown in the first row of Figure 3. The initial user prompt, $P_{e,0}$, does not explicitly mention skin repair preferences, resulting in a highly uncertain initial prediction of 0.5. However, after one iteration of prompt enhancement, the revised prompt, $P_{e,1}$, incorporates broader preferences from similar users, such as an inclination towards skincare and body care products, and specifically mentions an interest in healing ointments. This adjustment increases the prediction accuracy to 0.8, aligning more with the true label (1).

The second example in the second row of Figure 3, focuses on a filtration system for an icemaker and refrigerator. The initial prompt, $P_{e,0}$, reflects a negative interest due to preferences for large appliances and those that fit under cabinets, leading to a low probability prediction of 0.28. After the first enhancement iteration, insights from similar users reveal a preference for refrigerator-compatible products and icemakers, which adjusts the prediction to 0.48. A subsequent iteration, incorporating preferences for replacement and refrigerator parts in $P_{e,2}$, further increases the predicted probability to 0.58. This gradual improvement in prediction more accurately reflects the true preference (1), showcasing the iterative refinement capability of our *PepRec*.

5 Conclusion

This study proposes a progressive enhancement-of-prompting framework for recommendation, *PepRec*, as a pioneering approach for making accurate and explainable recommendation predictions. *PepRec* can capture the user-item dynamics from both content-based filtering and collaborative filtering to emulate human-like decision-making in recommendations, which is achieved by progressively analyzing and synthesizing user preferences in conjunction with insights gathered from similar users' item-interaction histories. The extensive experimental results show that *PepRec* significantly outperforms various traditional DLRMs and prompt-based models. The enhanced prompt also demonstrates its capability to deliver personalized and

explainable recommendations. Furthermore, our proposed *PepRec* maintains operational efficiency (reduced training requirements and minimal training parameters) compared to existing baselines.

Limitations

The primary limitation of *PepRec* proposed in our study is its cost. We employ GPT4 as the LLM, and the generation of the enhanced prompt involves several rounds to query the LLM, which results in increased expenses. However, such costs are rapidly decreasing, thanks to the swift advancements in LLM technology. Another point of concern is the inference time compared to traditional DLRMs as we query from a third-party LLM. Besides, we only test our method on GPT variants, we will test on other LLMs like Llama in the future.

References

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1007–1014.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10.

Weiyu Cheng, Yanyan Shen, and Linpeng Huang. 2020. Adaptive factorization network: Learning adaptive-order feature interactions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3609–3616.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. 2023. Uncovering chatgpt's capabilities in recommender systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*, RecSys '23. ACM.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 1725–1731. AAAI Press.

Farah Tawfiq Abdul Hussien, Abdul Monem S Rahma, and Hala Bahjat Abdul Wahab. 2021. Recommendation systems for e-commerce systems an overview. In *Journal of Physics: Conference Series*, volume 1897, page 012024. IOP Publishing.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xiangyang Li, Bo Chen, Lu Hou, and Ruiming Tang. 2023. Ctrl: Connect tabular and language model for ctr prediction. *arXiv preprint arXiv:2306.02841*.

Jianghao Lin, Bo Chen, Hangyu Wang, Yunjia Xi, Yanru Qu, Xinyi Dai, Kangning Zhang, Ruiming Tang, Yong Yu, and Weinan Zhang. 2023. Clickprompt: Ctr models are strong prompt generators for adapting language models to ctr prediction. *arXiv preprint arXiv:2310.09234*.

Jianghao Lin, Rong Shan, Chenxu Zhu, Kounianhua Du, Bo Chen, Shigang Quan, Ruiming Tang, Yong Yu, and Weinan Zhang. 2024. Rella: Retrieval-enhanced large language models for lifelong sequential behavior comprehension in recommendation. In *Proceedings of the ACM on Web Conference 2024*, pages 3497–3508.

Weilin Lin, Xiangyu Zhao, Yejing Wang, Tong Xu, and Xian Wu. 2022. Adafs: Adaptive feature selection in deep recommender system. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3309–3317.

Junling Liu, Chao Liu, Renjie Lv, Kang Zhou, and Yan Zhang. 2023. Is chatgpt a good recommender? a preliminary study. *arXiv preprint arXiv:2304.10149*.

Fuyuan Lyu, Xing Tang, Dugang Liu, Liang Chen, Xiuqiang He, and Xue Liu. 2023. Optimizing feature set for click-through rate prediction. In *Proceedings of the ACM Web Conference 2023*, pages 3386–3395.

Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao, and Xiuqiang He. 2021. Simplex: A simple and strong baseline for collaborative filtering. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1243–1252.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. 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 188–197.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Liang Qu, Yonghong Ye, Ningzhi Tang, Lixin Zhang, Yuhui Shi, and Hongzhi Yin. 2022. Single-shot embedding dimension search in recommender system. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 513–522.

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*. Association for Computational Linguistics.

Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*, pages 995–1000. IEEE.

David Sculley. 2010. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178.

Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1161–1170.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, pages 1–7.

Yejing Wang, Xiangyu Zhao, Tong Xu, and Xian Wu. 2022. Autofield: Automating feature selection in deep recommender systems. In *Proceedings of the ACM Web Conference 2022*, pages 1977–1986.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Yunjia Xi, Weiwen Liu, Jianghao Lin, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, Rui Zhang, and Yong Yu. 2023. Towards open-world recommendation with knowledge augmentation from large language models. *arXiv preprint arXiv:2306.10933*.

Chen Xu, Jun Xu, Xu Chen, Zhenghua Dong, and Ji-Rong Wen. 2022. Dually enhanced propensity score estimation in sequential recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 2260–2269.

Jing Yao, Wei Xu, Jianxun Lian, Xiting Wang, Xiaoyuan Yi, and Xing Xie. 2023. Knowledge plugins: Enhancing large language models for domain-specific recommendations. *arXiv preprint arXiv:2311.10779*.

Yakun Yu, Shi-ang Qi, Jiuding Yang, Liyao Jiang, and Di Niu. 2023. ihas: Instance-wise hierarchical architecture search for deep learning recommendation models. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, page 3030–3039, New York, NY, USA. Association for Computing Machinery.

Li Zhang, Weichen Shen, Jianhang Huang, Shijian Li, and Gang Pan. 2019. Field-aware neural factorization machine for click-through rate prediction. *IEEE Access*, 7:75032–75040.

Zizhuo Zhang and Bang Wang. 2023. Prompt learning for news recommendation. *arXiv preprint arXiv:2304.05263*.

A Cluster Training Details

This section provides additional materials for training the clusters. Specifically, Figure 4 shows an overview of the training process and Algorithm 2 shows the detailed steps.

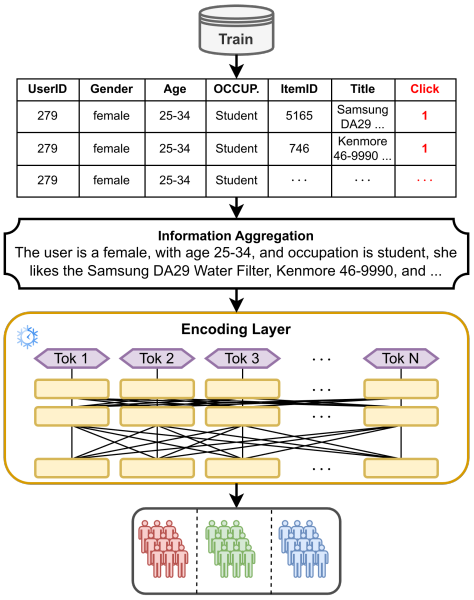

Figure 4: Overview of training user clusters. The snowflake means that the encoding layer is pretrained and no need for training.

B Prompt Templates

This section presents the basic prompt templates used in the preference generator and the final inference step. Specifically, Figure 5 shows the prompt template to generate the user's preference summary (P_1' in Figure 2) given the positive item-interaction history (input). Figure 6 shows the prompt template to refine the user's preference summary as P_1 in Figure 2 given P_1' (input1) and the negative item-interaction history (input2). Figure 7 shows the prompt template to ask the LLM for the prediction score of the test user towards the test item given the enhanced preference prompt $P_{e,n}$ (input1) and the test item title (input2).

C Experimental Details

C.1 Datasets

We show the dataset statistics in Table 3. Following the common practices (Xu et al., 2022; Mao et al., 2021; Dai et al., 2023), we convert the ratings into binary labels, considering ratings of 4 and 5 as positive and others as negative. We only include users and items with at least five interactions and exclude

Algorithm 2 Clustering

Require: Training dataset $\mathcal{D}_{\text{train}}$, a K-Means cluster with predefined K

Ensure: Set of sorted users for each cluster $\mathcal{S}$

procedure USER_CLUSTERING($\mathcal{D}_{\text{train}}$, K)
  for each user U_i in $\mathcal{D}_{\text{train}}$ **do**
    Extract user-item interactions $\{(U_i, I_j, Y_{i,j})\}_j$ in $\mathcal{D}_{\text{train}}$
    Aggregate interactions to form user information H_i
    Encode H_i by the encoding layer
  end for
  Feed all the encoded user representations into K clusters
  for each cluster $k = 1, \ldots, K$ **do**
    Collect all users in this cluster: $\mathcal{S}_k = [U_{k,1}, U_{k,2}, \ldots]$
    Sort $\mathcal{S}_k$ in the ascending order of the distance to the cluster's centroid.
  end for
  return $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_K\}$
end procedure

T_{like} = '''
Briefly summarize the user's appliance preference based on the following information:
{input}

Only output several short sentences in the following format without any additional text or thoughts! And each sentence has no redundancy with other sentences:
- The user likes appliances that consume less energy while maintaining high performance.
- The user likes appliances that are easy to clean and maintain.
- The user likes appliances with user-friendly interfaces and intuitive controls.
- The user likes quieter appliances.
'''

Figure 5: Prompt template of T_{like}

items lacking titles. We filter users and items with fewer than five interactions for both datasets. Items with missing titles are also discarded.

Datasets	# Users	# Items	# Interactions
Beauty	324,038	32,586	371,345
Appliance	515,650	30,252	602,777

Table 3: The dataset statistics.

C.2 Baselines

In this section, we will describe the details of the baseline models utilized in the performance comparison. Specifically, we include 5 traditional DLRMs and 4 prompt-based models. The 5 traditional DLRMs include:

- **FM** (Rendle, 2010): It linearly models all in-

T_{disl} = ‘‘‘
Further summarize the user's appliance preference based on the following two information:

The user's appliance preference we have already:
{input1}

Appliance history:
{input2}

Only output several short sentences in the following format without any additional text or thoughts! Each sentence has no redundancy with other sentences, and do not repeat the above preference we have already:
- The user doesn't like cheap appliances.
- The user is neutral to appliances that come with some noise.
- The user doesn't like appliances occupying too much space.
’’’

Figure 6: Prompt template of T_{disl}

score_prompt = ‘‘‘
Given the user's preferences:
{input1}

Given an appliance titled:
{input2}

Conclude how likely the user likes the appliance with an exact probability number, the number should range from 0 to 1. 0 means 'totally dislike' and 1 means 'totally like', do not explain the reason and include any other words.
’’’

Figure 7: Prompt template for predictions

teractions between variables using factorized parameters, which can estimate interactions even in problems with huge sparsity (like recommender systems).

- **DeepFM** (Guo et al., 2017): It combines the power of factorization machines and deep learning for feature learning in a neural network architecture, enabling to learn both low- and high-level feature interactions.

- **WD** (Cheng et al., 2016): It jointly trains wide linear models and deep neural networks to combine the benefits of memorization and generalization for recommender systems.

- **FNFM** (Zhang et al., 2019): It combines traditional feature combination methods and deep neural networks to automate feature combinations to improve the accuracy of click-through rate prediction.

- **AFN** (Cheng et al., 2020): It introduces a logarithmic transformation layer that converts the power of each feature in a feature combination into the coefficient to be learned.

And the four prompt-based models are:

- **LLM4RS** (Dai et al., 2023): It designs a prompt based on historical interacted items of a user to uncover the LLM's recommendation capabilities.

- **LLMRec** (Liu et al., 2023): It predicts user ratings based on the prompt including item titles and few-shot information.

- **TALLRec** (Bao et al., 2023): It first tunes the LLM with publicly available self-instruct data, then further finetunes the LLM with the recommendation-related instruction data.

- **ReLLA** (Lin et al., 2024): It constructs instruction data based on a user's relevant item-interaction history instead of recent history, and uses the data to fine-tune the LLM.

C.3 Implementation Details

In our experiments, we use all-MiniLM-L6-v2[2] as the sentence-transformers model which maps users' information to 384-dimensional dense vectors for clustering. We choose GPT4[3] (OpenAI, 2023) as the LLM backbone for LLM4RS, LLMRec, and our proposed framework. We set the temperature of GPT4 to 0 to ensure consistent outputs when generating preference summaries. A temperature of 0 means the model will always select the highest probability word during the query, ensuring the highest possible reproducibility for our work.

We set the number of clusters to $k = 7$ for the Beauty dataset and $k = 5$ for the Appliance dataset. The number of bootstrapping samples for each iteration α is set to be 5. The threshold for convergence γ is 0.9 for Beauty and 0.95 for Appliance. Different users reach the convergence threshold at different iterations, and we set the maximum bootstrapping iterations to 5. We use the LLM to generate five prediction probabilities and average them as the final probability to compute the AUC and Logloss, further enhancing reproducibility.

We implement the traditional DLRM baselines using a public library [4] that includes various commonly used DLRMs, with a batch size of 2048. We use the Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.001 and a weight decay of 1e-6. The best DLRM is selected based on the validation set's performance. For consistency, prompt-based baselines without fine-tuning produce probabilities rather than discrete ratings (between 1 and 5). Baselines with LLM tuning are implemented using their provided codes. All ex-

[2]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[3]https://platform.openai.com/docs/guides/text-generation

[4]https://github.com/rixwew/pytorch-fm

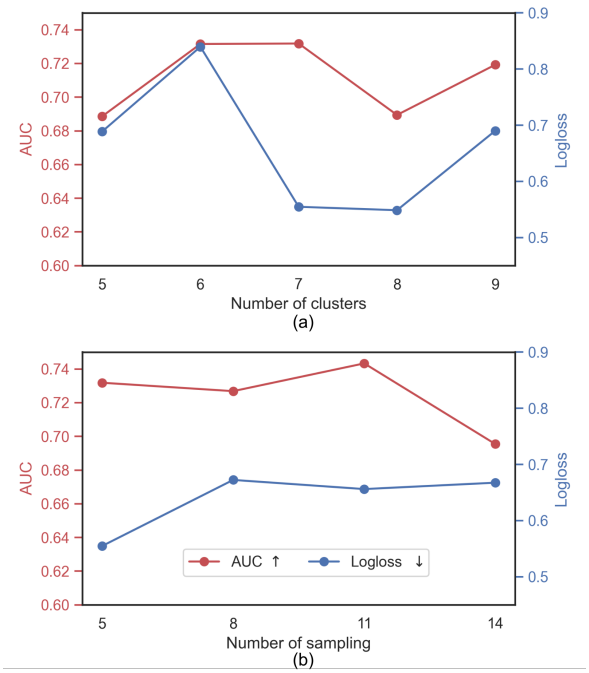

Figure 8: Hyperparameter analysis of (a) Impact of the number of clusters and (b) Impact of the number of sampling users.

periments are conducted on a single NVIDIA RTX 3090 GPU.

C.4 Hyperparameter Analysis (RQ3)

The core hyperparameters of our proposed method are the number of clusters and the number of users when bootstrapping a similar user subset. We notice that there are so many combinations of them taking different values, even if their values vary only in a small range. So we adopt a coarse hyperparameter tuning strategy, that is, we first find the best number of clusters and then determine the best number of users inside a similar user subset.

In particular, we vary the number of clusters in $\{5, 6, 7, 8, 9\}$, where we notice that even adding one more cluster can make a difference to the performance, as shown in Figure 8 (a). We can observe from this figure that all the AUCs are higher while most Loglosses are lower than the baselines, which again indicates the effectiveness of our proposed framework. Furthermore, the results achieve the best when dividing users into seven clusters. Figure 8 (b) plots the performance against different numbers of sampling users. We find sampling five users per similar user subset leads to relatively best results. If we sample more users, the performance may decrease due to the noise when aggregating their item-interaction histories. Therefore, we can

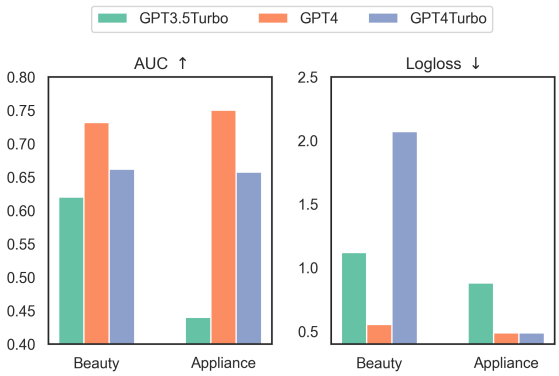

Figure 9: Impact of GPT backbones.

conclude that only a few clusters and sampling users are enough to derive an enhanced prompt.

Our proposed method can be compatible with various LLMs. We mainly focus on OpenAI API [5] in our work. We investigate the impact of using different GPT backbones, i.e., GPT3.5 Turbo, GPT4, and GPT4 Turbo. Figure 9 plots the results on both datasets. We can observe that GPT4 outperforms both GPT3.5 Turbo and GPT4 Turbo. The Logloss of GPT4 Turbo on Beauty shows an outlier trend, which might indicate that the GPT4 Turbo model has calibration issues despite its ability to correctly discriminate and rank instances. Besides, we will explore our framework with other LLMs like Llama (Touvron et al., 2023) in our future work.

[5] https://platform.openai.com/docs/models/overview