Aligning Large Language Models for Controllable Recommendations

Wensheng Lu[1] **Jianxun Lian**[2] **Wei Zhang**[1] **Guanghua Li**[1]
Mingyang Zhou[1] **Hao Liao**[1*] **Xing Xie**[2]
College of Computer Science and Software Engineering, Shenzhen University, China[1]
Microsoft Research Asia[2]
2210273060@email.szu.edu.cn jianxun.lian@outlook.com
{2210275010, 2210275050}@email.szu.edu.cn
{zmy, haoliao}@szu.edu.cn
xingx@microsoft.com

Abstract

Inspired by the exceptional general intelligence of Large Language Models (LLMs), researchers have begun to explore their application in pioneering the next generation of recommender systems — systems that are conversational, explainable, and controllable. However, existing literature primarily concentrates on integrating domain-specific knowledge into LLMs to enhance accuracy using a fixed task template, often overlooking the diversity of recommendation tasks and the ability of LLMs to follow recommendation-specific instructions. To address this gap, we first introduce a collection of supervised learning tasks, augmented with labels derived from a conventional recommender model, aimed at explicitly improving LLMs' proficiency in adhering to recommendation-specific instructions. Next, we propose a reinforcement learning-based alignment procedure to enhance LLMs' generalization ability. Extensive experiments on two real-world datasets demonstrate that our approach significantly improves the capability of LLMs to respond to instructions within recommender systems, reducing formatting errors while maintaining a high level of accuracy.

1 Introduction

Recommender systems are designed to identify and suggest the most appropriate items to users from a vast array of candidates, based on the users' profiles, past interactions, and present intentions. Witnessing the impressive capabilities of Large Language Models (LLMs), such as knowledge retention, reasoning, and problem-solving, researchers are now exploring the integration of LLMs into the next wave of intelligent recommender systems, which aim to be conversational, explainable, and controllable. Bridging the gap between the general-purpose LLMs and the specific requirements of recommendation tasks poses a challenge. In this context, fine-tuning LLMs with domain-specific knowledge and recommendation-focused tasks emerges as a promising strategy to harness their potential for advanced recommendation purposes (Bao et al., 2023; Zhang et al., 2023; Chen, 2023).

Typical approaches in the literature involve reformatting recommendation tasks — such as item reranking or click-through rate (CTR) prediction — into natural language constructs to facilitate the fine-tuning of LLMs. However, we have observed that LLMs fine-tuned through this straightforward method, albeit enhancing accuracy in offline evaluations, frequently generate outputs with domain-specific formatting errors. These errors may manifest as repeated items in the top-k recommendations or the inclusion of items previously interacted with by the user. Additionally, these LLMs exhibit a limited ability to adhere to diverse recommendation-specific instructions. This compromises their effectiveness as interactive agents in real-world recommender systems. A vivid example can be found in Section 3.4.

In this paper, we investigate the alignment of an LLM for recommender systems. Our objective extends beyond merely improving the recommendation accuracy of an original LLM; we aim to significantly enhance controllability and reduce formatting errors. Drawing inspiration from the Reinforcement Learning from Human Feedback (RLHF) framework (Ouyang et al., 2022), our methodology is structured into two phases: the supervised learning (SL) stage and the reinforcement learning (RL) stage. To inject domain-specific knowledge and foster recommendation-relevant control capabilities within the LLM, we devise a series of fine-tuning tasks, including item recommendation, item search, category control, and category proportion control. These tasks often necessitate generating a list of items that not only meet users' instructions but also maintain high quality, despite the typically sparse ground-truth signals found in user behavior history. To tackle this issue, we pro-

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 8159–8172
August 11-16, 2024 ©2024 Association for Computational Linguistics

pose augmenting supervised labels with predictions from a traditional recommender model, such as SASRec (Kang and McAuley, 2018). These augmented labels can help distill knowledge from the traditionally trained recommender model and meet the dynamic requirements of recommendation instructions.

After the SL stage, the LLM exhibits a significantly enhanced ability to follow recommendation-related instructions, surpassing existing approaches that solely fine-tune the LLM on item recommendation and search tasks. Nevertheless, the SL stage's data generation process inherently provides only positive examples for each instruction. To address scenarios where the LLM generates suboptimal responses, we introduce an RL stage with carefully crafted reward signals to further refine the LLM's capacity to follow instructions. To the best of our knowledge, this is the first study that employs both SL and RL stages to align LLMs for controllable recommendation purposes. We conduct comprehensive experiments on two real-world datasets, Steam and Amazon Movie, with results demonstrating that our method markedly improves the LLM's ability to follow instructions while simultaneously reducing formatting errors. Our major contributions are summarized as follows:

- We introduce a novel supervised learning stage, which encompasses a suite of tasks designed for enhancing controllability and label-augmentation by a teacher recommender model, to align an LLM into an interactive recommender agent.
- To mitigate formatting errors and improve the instruction-following generalization, We further design an alignment stage based on reinforcement learning with a variety of rewards that are tailored for the nuances of the controllable recommendation task.
- Experiment results validate that our model markedly surpasses existing LLM-based recommendation models, and exhibits a robust capacity to follow users' instructions while maintaining a high level of recommendation precision. Source code is available at https://github.com/microsoft/RecAI/tree/main/RecLM-gen.

2 Methodology

2.1 Intention Categories

This paper aims to enhance the instruction-following capabilities of LLMs for recommendation tasks. We categorize recommendation instructions into three distinct types:

Implicit intention. This is the assumed default where the prompt describes the user's profile (such as attributes and past favored items). The LLM is tasked with recommending items that align with the user's preferences.

Item-wise intention. In addition to the profile, users may express specific desires, such that the recommended items should either exhibit particular characteristics ("I wish to watch an action movie") or exclude them ("Please avoid suggesting any action movies").

List-wise intention. Users may have requirements for the entire list of recommended items; hence, evaluating individual items' attributes is insufficient. For example, if all items in a recommendation list belong to the same category A, the user may be disappointed by the lack of diversity. Consequently, the user might request the recommender system to ensure that the proportion of category A is below a certain threshold.

To effectively train LLMs as recommender agents capable of adhering to these three types of instructions, we introduce a novel two-stage fine-tuning approach: a supervised learning (SL) stage (Section 2.2) followed by a reinforcement learning (RL) stage (Section 2.3). The overall framework is illustrated in Figure 1.

2.2 The SL Stage

We represent each data sample in the recommendation task with natural language text, adopting the format "Instruction: [Prompt Content]. Output: [Response Content]", where [Prompt Content] includes detailed instructions such as the user's profile and intention, and [Response Content] contains the expected item recommendations that fulfill the instructions. Different from traditional recommender systems that utilize item IDs, we employ only item titles to represent items in both [Prompt Content] and [Response Content] to fully leverage LLMs' general abilities and ensure a smooth interaction between users and our LLM-based recommender. Data samples are generated according to the following tasks:

2.2.1 Data Generation Tasks

Sequential Recommendation Instructions (I_0) This task represents the traditional sequential recommendations: given a user's previously interacted items, the goal is to predict future interactions.

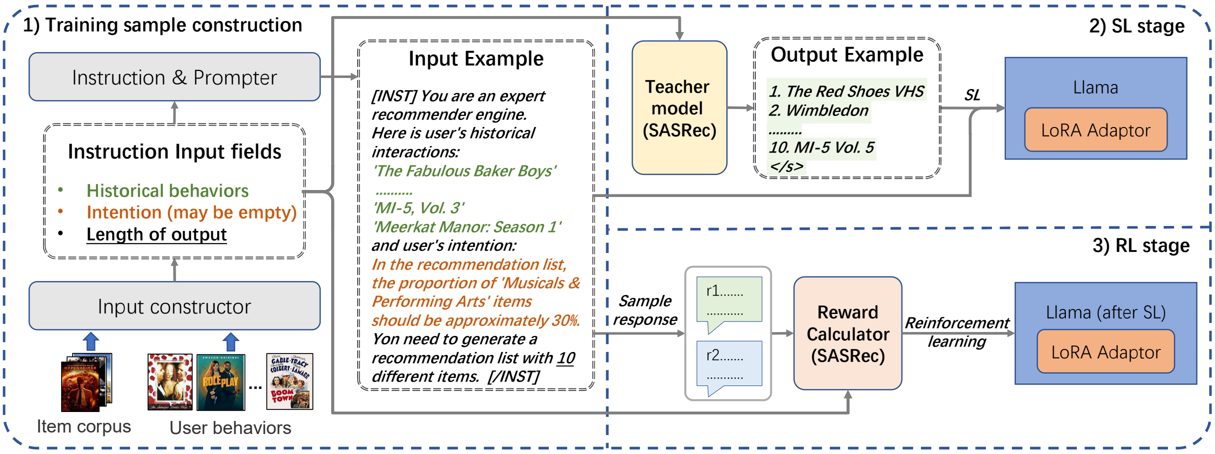
1) Training sample construction
Instruction & Prompter
Instruction Input fields
• Historical behaviors
• Intention (may be empty)
• Length of output
Input constructor
Item corpus
User behaviors
Input Example
[INST] You are an expert recommender engine. Here is user's historical interactions: 'The Fabulous Baker Boys' .... 'MI-5, Vol. 3' 'Meerkat Manor: Season 1' and user's intention: In the recommendation list, the proportion of 'Musicals & Performing Arts' items should be approximately 30%. Yon need to generate a recommendation list with 10 different items. [/INST]
2) SL stage
Teacher model (SASRec)
Output Example
1. The Red Shoes VHS
2. Wimbledon
....
10. MI-5 Vol. 5
</s>
SL
Llama
LoRA Adaptor
3) RL stage
Sample response
r1.......
....
r2.......
....
Reward Calculator (SASRec)
Reinforcement learning
Llama (after SL)
LoRA Adaptor

Figure 1: Overview of the proposed method.

Specifically, we use the first $n-1$ items to construct the user's behavioral profile, while the n^{th} item serves as the ground-truth label. The LLM is instructed to recommend k items; if the ground-truth item is among these k suggestions, the recommendation is considered a successful hit. The value of k for each data sample is randomly selected from 1 to 10.

Category Control Instructions (I_1) This task corresponds to instructions with item-wise intention, which we implement into two distinct types: 1) Positive control (I_1^{+C}), where the user hopes to receive more items in the recommendation list that match the specific category C_{target}. 2) Negative control (I_1^{-C}), where users indicate a weariness towards a certain category C_{target} and wish to reduce the inclusion of such items in their received recommendations as much as possible.

Category Proportion Control Instructions (I_2) We implement the list-wise intention into three distinct types: 1) $I_2^{CP\leq x}$, in this case, the user hopes to have the proportion of the item of C_{target} less than a certain percentage x. 2) $I_2^{CP\approx x}$, in this case, the user hopes that the proportion of items in C_{target} will be approximately a certain percentage. 3) $I_2^{CP\geq x}$, in this case, the user hopes that the proportion of items in C_{target} is greater than a certain percentage.

Item Search Instructions (I_3) To aid the LLM in memorizing in-domain item attributes (in this paper, we use item category as the key attribute for illustration), we introduce an item search task: the objective is to retrieve k items belonging to a target category C_{target}. For this purpose, we randomly select k items from C_{target} to serve as the ground truth for the response.

ShareGPT (I_4) To avoid catastrophic forgetting and preserve the general intelligence capabilities of the LLM, we follow Zeng et al. (2023) and integrate a certain proportion of ShareGPT[1] training data into the SL stage. The ShareGPT data includes a diversity of real-world tasks in the user-assistant conversation format, which helps LLM revisit past knowledge during the in-domain fine-tuning stage.

2.2.2 Label Augmentation

Instructions I_0, I_1, and I_2 require the inclusion of k items within the response text. However, due to the typically sparse nature of user historical behavior, it is often impractical to construct a ground-truth response based solely on this information. To overcome this limitation, we employ the sequential recommender model SASRec (Kang and McAuley, 2018) as a teacher model to generate a set of top recommendations, $\boldsymbol{P}_{SASRec}$, for each data sample. We then curate the top-k list by selecting items from $\boldsymbol{P}_{SASRec}$ that align with the given instructions. The top-k list is assembled as follows: the first item is the ground-truth item (i.e., the n^{th} item in the user's history). The method for compiling the remaining $k-1$ items varies slightly: for I_0, we fill the list with SASRec's top predictions; for I_1 and I_2, we filter SASRec's predictions to ensure the final recommendation list adhere to the specified instruction.

2.3 The RL Stage

Following previous works (Touvron et al., 2023; Ouyang et al., 2022), we employ the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017) to further fine-tune the model after the SL stage. Different from (Ouyang et al., 2022), scores

[1] https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered

are not produced by a reward model; instead, they are derived from reward rules that are specifically tailored for I_0, I_1, and I_2. Fundamentally, rewards consist of two components: item-level rewards and list-level rewards.

2.3.1 Item-level Reward

The item-level reward assigns a score to each item generated by LLM, serving as an immediate reward of reinforcement.

For each $item_i$ in LLM's recommendation list, let $Rank_i$ refer to the rank of $item_i$ in the teacher model (SASRec)'s prediction list. We can then calculate the preference scores for $item_i$ by:

$$Scores_i = \begin{cases} -1, & \text{if } item_i \text{ is illegal} \\ +1, & \text{if } item_i \text{ is } item_{target} \\ \frac{1}{log_2(Rank_i+3)}, & \text{else} \end{cases} \tag{1}$$

$item_i$ is considered illegal if it meets any of the following conditions: $item_i$ does not exist, it is a duplicate of any item within the preceding set $item_{[1,\ldots,i-1]}$, it is identical to an item in the user's history, or its index i exceeds k.

Beyond the preference score, $Scores$, a control effect score, $Scores^{ctl}$, is required to gauge the extent to which an item corresponds with user intentions. Essentially, a generated item that adheres to the given instruction is awarded a high score, signifying positive reinforcement, while a non-conforming item incurs a low score, serving as negative feedback. The complete calculation is described in algorithm 1 in the Appendix.

Finally, we get item-level reward R_{item}, which measures the overall merits of each item:

$$R_{item} = 0.5 * Scores + 0.5 * Scores^{ctl} \tag{2}$$

2.3.2 List-level Reward

The list-wise reward, which measures how well the entire list of recommendations matches the user's preferences and control intentions, is added to the last token of output, serving as a termination reward. To encourage the LLM to rank ground-truth items in top positions, we adjust the $Scores$ as per Equation 3 to derive $Score^*$:

$$Scores_i^* = \begin{cases} -1, & \text{if } item_i \text{ is illegal} \\ \frac{Scores_i}{log_2(i+2)}, & \text{else} \end{cases} \tag{3}$$

In addition to this, we need to measure how well the output matches the control intention. Let $Count_{in}, Count_{out}$ denote the number of items that belong/do not belong to the target category. For different control intentions, we use different calculation methods to get $Score_{list}^{ctl}$:

$$Score_{list}^{ctl} = \begin{cases} \text{sum}(Scores^*), & \text{if } I_0 \\ \frac{1}{log_2((k-Count_{in})+2)}, & \text{if } I_1^{+C} \\ \frac{1}{log_2((k-Count_{out})+2)}, & \text{if } I_1^{-C} \\ \frac{1}{log_2(max(Count_{in}-k*m,0)+2)}, & \text{if } I_2^{CP\leq m} \\ \frac{1}{log_2(max(k*m-Count_{in},0)+2)}, & \text{if } I_2^{CP\geq m} \\ \frac{1}{log_2(abs(Count_{in}-k*m)+2)}, & \text{if } I_2^{CP\approx m} \end{cases} \tag{4}$$

Finally, we get R_{list}, which measures the overall merits of the recommendation list:

$$R_{list} = 0.5 * \text{sum}(Scores^*) + 0.5 * Score_{list}^{ctl} \tag{5}$$

2.3.3 RL Implementation Notes

We follow the *Transformer Reinforcement Learning*[2] package to implement the reinforcement learning stage. We set up LoRA layers (Hu et al., 2022) for both the policy network and the critic network, with a LoRA rank of 4 and a LoRA alpha of 2. During the RL sampling phase, we sample 2 responses with a temperature of 0.7 for each instruction. The final reward is calculated by combining item-level reward, list-level reward, and a KL penalty:

$$\boldsymbol{R}_{\text{final}}[\boldsymbol{y}] = \boldsymbol{R}_{item}[\boldsymbol{y}] + \boldsymbol{R}_{list}[\boldsymbol{y}] - \eta \mathbf{KL}(\pi_\theta^{\text{RL}}, \pi^{\text{SFT}})[\boldsymbol{y}] \tag{6}$$

where η is set to 0.3. $\boldsymbol{y}$ represents a generated response (which is a token sequence) for an instruction sample. $\mathbf{KL}[\boldsymbol{y}]$ represents there is a KL penalty on each token in $\boldsymbol{y}$. $\boldsymbol{R}_{item}[\boldsymbol{y}]$ means there is an item-level reward at the last token of each item title. $\boldsymbol{R}_{list}[\boldsymbol{y}]$ represents that there is a list-level reward on the ending token of $\boldsymbol{y}$. To ensure that the information in R_{list} is not overshadowed by R_{item}, we amplify R_{list} by a factor of 10. Additionally, we employ reward whitening to enhance the stability of training. We use Generalized Advantage Estimation (GAE) to estimate the advantage values, with hyperparameters $\gamma = 0.99, \lambda = 0.95$. During the loss calculation phase, we set the clipping range for the probability ratio to $[1-\epsilon, 1+\epsilon]$, where $\epsilon = 0.2$. The loss weight for the critic network is 0.5.

3 Experiments

3.1 Experiment Setting

3.1.1 Dataset

We experiment with two popular datasets in the recommender system domain: the Amazon Movies

[2] https://github.com/huggingface/trl

Dataset	#Users	#Items	#Inters	#Sparsity
Movie	$13,218$	$18,744$	$744,313$	99.70%
Steam	$12,658$	$8,572$	$632,900$	99.42%

Table 1: General Statistics of the Two Datasets

& TVs [3] dataset (**Movie** for short) and the Steam dataset (Kang and McAuley, 2018). Both datasets include item category information, which aids in constructing the user's control intentions. Basic statistics of datasets are summarized in Table 1. We employ the leave-one-out approach (Kang and McAuley, 2018) to split the dataset. Therefore, the size of test set is consistent with the number of users in each dataset. To accelerate the validation process, we only include four types of instructions in the valid set: I_0, I_1^{+C}, I_1^{-C}, and $I_2^{CP \approx 50\%}$, and each type of instruction has 320 instances. Upon observing multiple metrics on the validation set, we find that SL typically converged around 30 epochs. Full details of the training set on each instruction task can be found at Table 8 in the Appendix.

3.1.2 Implementation Details

We choose Llama-2-7b-chat as the foundational model for our research (Touvron et al., 2023). We set the model's maximum sequence length to 1024 tokens. User behavior sequences are truncated to incorporate no more than 10 items, and excessively lengthy item titles are condensed to a maximum of 64 tokens. Furthermore, to accommodate the complete recommendation list within the output, we reserve a larger token count, setting the maximum output length of our model to 512 tokens. Instructions and responses are formatted using the official prompt template in Touvron et al. (2023).

We use LoRA (Hu et al., 2022) to fine-tune all linear layers in Llama2-7b-chat, with trainable parameters accounting for about 0.6% of the total parameters in SL and 0.3% in RL. The optimizer is Adam. In the SL stage, the learning rate is set to 0.001, the LoRA dimension is set to 16, the LoRA alpha to 8, and the batch size is 64. The ShareGPT data accounted for 50% of the total training data. In the RL stage, the learning rate is 5×10^{-6}. We set separate LoRA layers for actor and critic networks, with LoRA dimension at 4 and LoRA alpha at 2. To encourage model exploration, we set the temperature to 0.7 and the weight of entropy loss to 0.01. We release the source code at: https://github.com/microsoft/RecAI/tree/main/RecLM-gen.

3.1.3 Metrics

For all models, we use Top-k hit ratio (**HR@K**) and Top-k NDCG (**NDCG@K**) to evaluate the accuracy of recommendations. Meanwhile, we use some additional indicators to evaluate the model's ability to follow user instructions: 1). For category control, we use the Top-k target category proportion (**TCP@K**) metric to evaluate the proportion of target categories in the recommendation list. The calculation of TCP@K is shown in Equation 7. 2). For category proportion control, we use the target category proportion accuracy (**CPA**), defined in Equation 8, to measure whether the responding item distribution complies with the instruction. Note that k indicates the number of recommended items and m indicates the proportion requirement in instructions.

$$TCP@K = \frac{1}{K}\sum_{i=1}^{K}\mathbb{1}(item_i \in C_{target}) \tag{7}$$

$$CPA = \begin{cases} \mathbb{1}(Count_{in} \leq k * m), & \text{if } I_2^{CP \leq m} \\ \mathbb{1}(Count_{in} \geq k * m), & \text{if } I_2^{CP \geq m} \\ \mathbb{1}(\text{abs}(Count_{in} - k * m) \leq 1), & \text{if } I_2^{CP \approx m} \end{cases} \tag{8}$$

3.1.4 Baselines

We divide the compared methods into 3 classes: general LLM, fine-tuned LLM, and our method and its variants. For sequential recommendation, we additionally compared with **SASRec** (Kang and McAuley, 2018) that we used as the teacher model.

General LLM: We use a closed-source LLM **GPT-3.5-turbo-0613** [4] (GPT-3.5 for short) and an open-source LLM **Llama2-7b-chat** (Touvron et al., 2023). Due to the high cost, all GPT-3.5 results are obtained from the first 1,000 samples of the test set.

Fine-tuned LLM: **InstructRec** (Zhang et al., 2023) uses 39 types of recommendation-related instructions constructed from user history and review data. It uses Beam search to obtain the top-k recommendation list during the test phase. Its backbone is 3B Flan-T5-XL. **PALR** (Chen, 2023) also uses Llama2-7b-chat as the backbone. We fine-tune both models using our datasets, employing data sample generation techniques as described in their respective papers. For instance, a key distinction between PALR and our supervised instruction tuning task, I_0, is that PALR does not utilize a teacher

[3] https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/

[4] https://platform.openai.com/docs/models/gpt-3-5-turbo

recommender model for label augmentation; instead, it relies solely on user behavior history to derive label responses.

Our method and variants: $\mathbf{Ours_{v1}}$ is a model that only uses sequential recommendation instructions (I_0) in SL. $\mathbf{Ours_{v2}}$ is a model uses all instructions ($I_{\{0,1,2,3,4\}}$) in SL. $\mathbf{Ours_{v3}}$ is a model obtained by further fine-tuning $\mathrm{Ours_{v2}}$ with RL, but without the item-level reward. $\mathbf{Ours_{full}}$ is the complete version of our method. We compare different variants of our method as an ablation study to verify the effect of different components.

3.2 Overall Performance

3.2.1 Sequential Recommendation

We begin by assessing model performance under standard sequential recommendation scenarios, where users have no explicit intentions. It is important to clarify that our objective in this paper is to improve the ability of LLMs to follow recommendation-related instructions, not to outperform the teacher model, SASRec. Achieving recommendation accuracy on par with the teacher model is considered satisfactory for our purposes. Table 2 presents the comprehensive results. Initially, all fine-tuned models surpass the performance of general LLMs such as GPT-3.5 and Llama2-7b, affirming the necessity of fine-tuning for domain-specific tasks. Additionally, our approach significantly outperforms the fine-tuned benchmarks (InstructRec and PALR), validating the efficacy of our Supervised Learning (SL) stage. Lastly, our completed model, $\mathbf{Ours_{full}}$, achieves results comparable to the teacher model, SASRec, reinforcing the premise that an LLM must first grasp user preferences before its instruction-following capabilities can be further evaluated.

3.2.2 Category Control

To evaluate the Category Control Instructions ($\boldsymbol{I_1}$), we use the following two settings:

Positive (I_1^{+C}): We utilize the target item's category as the control signal C_{target} and incorporate a positive descriptor of C_{target} into the input instruction to represent the user's explicit intent to receive more items within that category.

Negative (I_1^{-C}): We analyze the output statistics of top-k recommendations by SASRec to identify the category with the largest proportion — excluding the target item's category — as C_{target} for simulated control. A negative descriptor of C_{target} is then embedded in the instruction to reflect the user's intent to minimize items from this category.

Dataset	Method	HR@10	NDCG@10	HR@5	NDCG@5
Movie	SASRec	**0.1229**	<u>0.0913</u>	<u>0.1018</u>	<u>0.0844</u>
	GPT − 3.5	0.0050	0.0025	0.0030	0.0019
	Llama2 − 7b	0.0120	0.0056	0.0133	0.0072
	InstructRec	0.0524	0.0381	0.0406	0.0343
	PALR	0.0868	0.0787	0.0832	0.0775
	$\mathrm{Ours_{v1}}$	0.1108	0.0861	0.0991	0.0827
	$\mathrm{Ours_{v2}}$	<u>0.1211</u>	**0.0927**	**0.1056**	**0.0880**
	$\mathrm{Ours_{v3}}$	0.1150	0.0858	0.1010	0.0824
	$\mathrm{Ours_{full}}$	0.1148	0.0867	0.1001	0.0825
Steam	SASRec	**0.1121**	**0.0648**	**0.0778**	**0.0538**
	GPT − 3.5	0.0160	0.0079	0.0090	0.0055
	Llama2 − 7b	0.0052	0.0028	0.0016	0.0012
	InstructRec	0.0220	0.0113	0.0122	0.0082
	PALR	0.0408	0.0320	0.0373	0.0308
	$\mathrm{Ours_{v1}}$	0.0930	0.0535	0.0666	0.0451
	$\mathrm{Ours_{v2}}$	<u>0.1036</u>	<u>0.0583</u>	<u>0.0717</u>	<u>0.0479</u>
	$\mathrm{Ours_{v3}}$	0.1014	0.0557	0.0695	0.0454
	$\mathrm{Ours_{full}}$	0.1001	0.0551	0.0688	0.0449

Table 2: Results of (I_0). The best result is highlighted in **boldface** and the runner-up is denoted with <u>underline</u>.

Table 3 presents the results, using the TCP@10 metric to assess conformity to control signals, while HR@10 and NDCG@10 indicate recommendation accuracy when explicit intentions are additionally included in the prompt. Values in parentheses show the outcomes of each corresponding model without control signals. Two key observations emerge: Firstly, differentiating supervised learning tasks by intention is essential. Evidence for this includes InstructRec's improved TCP performance in the $I_1{+}C$ task, which aligns with its training, versus no effect in the $I_1{-}C$ task not covered by its training. Similarly, $\mathbf{Ours_{v1}}$, trained solely on recommendation tasks, underperforms in TCP compared to $\mathbf{Ours_{v2}}$. Secondly, $\mathbf{Ours_{v2}}$, $\mathbf{Ours_{v3}}$, and $\mathbf{Ours_{full}}$ show notable TCP enhancements, with $\mathbf{Ours_{full}}$ leading. Additionally, HR and NDCG metrics also see a significant lift compared to Table 2, confirming our method's effectiveness in adhering to category control instructions.

3.2.3 Category Proportion Control

To evaluate the Category Control Instructions ($\boldsymbol{I_1}$), we use the following three settings:

$\boldsymbol{I_2^{CP\leq 20\%}}$: In this case, we select the C_{target} of I_1^{-C} in the same way to simulate control. The purpose is to control the proportion of C_{target} to be no greater than 20%.

$\boldsymbol{I_2^{CP\approx 30\%}}$: In this case, we select the category of the target item as the C_{target} of control simulating. The purpose is to control the proportion of items in C_{target} to be approximately 30%.

$\boldsymbol{Control}$		$\boldsymbol{I_1^{+C}}$			$\boldsymbol{I_1^{-C}}$		
Dataset	Model	HR@10	NDCG@10	TCP@10(%) ↑	HR@10	NDCG@10	TCP@10(%) ↓
Movie	GPT − 3.5	0.0200	0.0111	7.39(2.23)	0.0150	0.0074	**7.78**(19.72)
	Llama2 − 7b	0.0238	0.0109	4.89(2.80)	0.0259	0.0123	11.61(18.52)
	InstructRec	0.1045	0.0687	37.39(7.16)	0.0362	0.0242	56.61(41.82)
	PALR	0.0832	0.0746	8.70(7.64)	0.0809	0.0728	46.64(61.46)
	Ours_{v1}	0.1054	0.0788	8.69(8.07)	$\underline{0.1023}$	$\underline{0.0778}$	35.99(40.83)
	Ours_{v2}	**0.2620**	**0.1814**	81.38(8.05)	**0.1099**	**0.0832**	19.35(39.80)
	Ours_{v3}	$\underline{0.2383}$	$\underline{0.1609}$	$\underline{89.06}$(7.89)	0.1017	0.0772	15.80(39.04)
	$\text{Ours}_{\text{full}}$	0.2336	0.1574	**93.52**(7.81)	0.0999	0.0756	$\underline{11.49}$(38.41)
Steam	GPT − 3.5	0.0360	0.0187	37.62(25.59)	0.0090	0.0044	19.67(43.59)
	Llama2 − 7b	0.0073	0.0044	21.68(19.94)	0.0047	0.0024	32.57(50.69)
	InstructRec	0.0494	0.0247	51.70(29.47)	0.0114	0.0059	70.39(53.49)
	PALR	0.0392	0.0301	13.04(12.45)	0.0350	0.0275	47.01(52.60)
	Ours_{v1}	0.0922	0.0509	27.23(26.41)	0.0887	0.0488	51.38(56.63)
	Ours_{v2}	**0.3484**	**0.2110**	92.25(28.02)	**0.1236**	**0.0724**	12.18(55.67)
	Ours_{v3}	$\underline{0.3422}$	$\underline{0.2049}$	$\underline{95.13}$(29.75)	$\underline{0.1200}$	$\underline{0.0696}$	$\underline{6.42}$(54.37)
	$\text{Ours}_{\text{full}}$	0.3395	0.2036	**95.80**(29.92)	0.1167	0.0676	**5.51**(54.00)

Table 3: Results of I_1 control. The best result is highlighted in **boldface** and the runner-up is denoted with $\underline{\text{underline}}$. Values in parentheses show outcomes of the corresponding model without control signals.

$\boldsymbol{I_2^{CP \geq 30\%}}$: We select the category of the target item as the C_{target}. The purpose is to control the proportion of items in C_{target} to be no less than 30%.

Results are reported in Table 4. In this case, the primary metric is CPA. Overall, Ours_{full} achieves the best performance, outperforming all other baselines and its own variants.

3.3 Formatting and General Evaluation

Finally, we assess the formatting ability and overall linguistic capabilities of LLMs in generating structured recommendations. We measure formatting ability for the recommendation domain in four dimensions: 1) **CorrectCount**: The accuracy of the recommended item count against the given k in the instruction. 2) **RepeatItem**: The frequency of repeated items in the recommendation list. 3) **NonExist**: The occurrence of non-existent (hallucinated) items in the list. 4) **InHistory**: The rate of recommended items already present in the user's history. For a more challenging test, LLMs are tasked with recommending k items where k is a random number between 11 and 15, despite training only on ranges from 1 to 10. To evaluate the general language ability of LLMs, we utilize two standard tasks, **MMLU** with five examples (5-shot) and **GSM8K** with eight examples (8-shot), leveraging the unified framework provided by https://github.com/EleutherAI/lm-evaluation-harness.

Table 5 presents the comprehensive outcomes. Notably, nearly all models, with the exception of PALR on the Movie dataset, excel in the CorrectCount metric. For other formatting metrics such as RepeatItem, NonExist, and InHistory, a progressive enhancement is evident from Ours_{v1} through Ours_{v3}, culminating in Ours_{full}. In Table 5, the primary focus is on *Formatting Quality*, while *Precision* metrics serve as supplementary references. The *Generalization* metrics indicate the extent of catastrophic forgetting. When compared to the fine-tuned baseline PALR, Ours_{full} shows the least performance deterioration from its underlying Llama2-7b architecture. Owing to the 512-token input limitation of InstructRec, its performance could not be assessed on the *Generalization* tasks.

3.4 Case Study

For clarity, we present two illustrative cases in Table 6. The first is about the $I_2^{CP \approx 50\%}$ instruction on the Movie dataset. A user requests that roughly 50% of the top-5 recommendations feature "Art House & International, By Original Language, Chinese" characteristics. Our method successfully generates a list where 3 out of 5 items possess these attributes, while three baseline methods fail to meet this criterion. Llama-2-7b simply repeats items mentioned in the user history.

In the second case, concerning the I_1^{-C} instruction, the user specifies to exclude "Shooter" games from recommendations. Our method effectively adheres to this restriction by omitting any items from the blacklisted category and successfully includes the ground-truth item in the recommendation list. Conversely, Llama-2-7b incorrectly suggests "Far Cry 5", a Shooter game, while InstructRec not only

Control		$I_2^{CP\leq 20\%}$			$I_2^{CP\approx 30\%}$			$I_2^{CP\geq 30\%}$		
Dataset	Model	HR@10	NDCG@10	CPA(%) ↑	HR@10	NDCG@10	CPA(%) ↑	HR@10	NDCG@10	CPA(%) ↑
Movie	GPT – 3.5	0.0050	0.0021	1.80(0.50)	0.0210	0.0124	9.71(0.70)	0.0250	0.0167	6.61(1.10)
	Llama2 – 7b	0.0198	0.0089	16.30(9.83)	0.0270	0.0114	9.71(1.14)	0.0279	0.0119	3.45(2.09)
	InstructRec	0.0372	0.0252	9.75(15.87)	0.1031	0.0684	31.24(3.96)	0.1041	0.0692	57.58(9.60)
	PALR	0.0792	0.0711	29.76(6.70)	0.0813	0.0724	10.28(3.16)	0.0825	0.0733	11.17(9.84)
	Ours$_{v1}$	0.1018	**0.0742**	30.97(14.34)	0.1048	0.0766	16.14(4.43)	0.1061	0.0776	10.46(10.19)
	Ours$_{v2}$	**0.1027**	0.0688	29.18(14.93)	**0.2034**	**0.1417**	58.71(4.61)	0.2055	0.1433	45.89(9.96)
	Ours$_{v3}$	0.0943	0.0594	33.74(15.03)	0.1948	0.1319	65.78(4.58)	0.1994	0.1340	48.19(9.21)
	Ours$_{full}$	0.0981	0.0642	**45.90**(16.33)	0.2005	0.1354	**69.81**(4.68)	**0.2134**	**0.1447**	**61.98**(9.15)
Steam	GPT – 3.5	0.0170	0.0074	29.80(14.40)	0.0290	0.0152	25.60(3.40)	0.0360	0.0202	51.10(39.60)
	Llama2 – 7b	0.0041	0.0022	14.65(7.24)	0.0024	0.0011	23.06(6.79)	0.0026	0.0013	32.97(34.65)
	InstructRec	0.0109	0.0056	15.86(21.39)	0.0442	0.0223	13.22(3.15)	0.0435	0.0220	55.90(40.30)
	PALR	0.0333	0.0257	6.18(2.47)	0.0374	0.0287	13.39(4.13)	0.0388	0.0296	20.11(18.96)
	Ours$_{v1}$	0.0863	0.0478	20.51(12.25)	0.0897	0.0496	12.23(3.96)	0.0903	0.0502	39.22(38.73)
	Ours$_{v2}$	0.1144	0.0647	41.26(14.90)	0.2072	0.1191	71.97(3.26)	0.2184	0.1238	70.79(39.79)
	Ours$_{v3}$	0.1172	0.0663	65.41(18.04)	0.2105	0.1206	**77.51**(2.14)	0.2283	0.1286	82.07(40.69)
	Ours$_{full}$	**0.1183**	**0.0663**	**70.58**(18.15)	**0.2225**	**0.1295**	74.29(2.77)	**0.2382**	**0.1365**	**88.40**(40.98)

Table 4: Results of I_2 control. The best result is highlighted in **boldface** and the runner-up is denoted with underline. Values in parentheses show outcomes of the corresponding model without control signals.

Dataset	Method	Formatting Quality(%)				Precision		Generalization	
		CorrectCount ↑	RepeatItem@K ↓	NonExist@K ↓	InHistory@K ↓	HR@K ↑	NDCG@K ↑	MMLU ↑	GSM8K ↑
Movie	GPT – 3.5	100.00	2.45	66.10	4.22	0.0100	0.0034	0.700	0.7460
	Llama2 – 7b	99.75	5.64	49.47	29.74	0.0183	0.0075	0.440	**0.2858**
	InstructRec	100.00	10.01	8.52	15.35	0.0546	0.0378	–	–
	PALR	77.27	65.92	4.06	9.31	0.0869	0.0787	0.377	0.1099
	Ours$_{v1}$	92.96	13.63	7.03	5.76	0.1164	0.0876	0.341	0.1318
	Ours$_{v2}$	100.00	9.61	5.27	4.02	**0.1285**	**0.0950**	0.450	0.1842
	Ours$_{v3}$	100.00	2.37	1.14	1.36	0.1214	0.0886	0.453	0.1789
	Ours$_{full}$	**100.00**	**1.14**	**0.95**	**1.24**	0.1220	0.0890	**0.455**	0.1782
Steam	GPT – 3.5	99.90	2.44	26.76	4.50	0.0200	0.0094	0.700	0.7460
	Llama2 – 7b	99.79	5.88	20.78	43.90	0.0074	0.0030	0.440	**0.2858**
	InstructRec	98.41	0.99	4.60	7.54	0.0270	0.0130	–	–
	PALR	97.53	17.67	46.78	2.88	0.0404	0.0316	0.417	0.1327
	Ours$_{v1}$	95.79	3.95	3.00	1.49	0.1029	0.0559	0.327	0.0819
	Ours$_{v2}$	100.00	2.68	1.59	2.55	**0.1152**	**0.0612**	**0.458**	0.2146
	Ours$_{v3}$	100.00	0.37	1.04	0.22	0.1149	0.0593	0.457	0.2039
	Ours$_{full}$	**100.00**	**0.23**	**0.78**	**0.17**	0.1149	0.0589	0.457	0.2123

Table 5: Results of formatting and general evaluation. The best result (excluding GPT-3.5) is highlighted in **boldface** and the runner-up is denoted with underline.

makes aching historical mistakes but also recommends a lot of games from the shooter genre.

3.5 Extend to combinatorial Controls

Considering LLMs' strong generalization capabilities, we further examine the performance of LLMs on complex, combinatorial instructions not encountered during training, as shown in Table 7. Here, $TC_{1\neg 2}P@10(\%)$ means the percentage of items belong to C_1 but does not belong to C_2. The results confirm our expectations that while our methods surpass baseline models, they achieve lower compliance with user instructions, as measured by TCP and CPA, compared to simpler single-control instructions. This motivates us to include some complex instructions in the alignment process in future work.

4 Related Work

In recent years, the remarkable natural language processing capabilities of LLMs have inspired researchers to leverage them for recommendation tasks (Fan et al., 2023). Early implementations utilized language models as feature extractors, creating knowledge-aware recommendation embeddings exemplified by models such as U-BERT (Qiu et al., 2021) and UserBERT (Wu et al., 2022). With the advent of generative models like GPT, the focus has shifted towards generative recommendation models that frame recommendations as natural language generation tasks (Wu et al., 2023).

Initially, adaptation of LLMs to recommendation scenarios relied heavily on techniques like prompt engineering (Gao et al., 2023; Sun et al., 2023) and contextual learning (Dai et al., 2023; Liu et al., 2023). However, these LLMs often underperform compared to traditional recommendation models trained on task-specific data, prompting the neces-

Instruction $\mathbf{I_2^{CP\approx 50\%}}$**:** You are an expert recommender engine. You need to generate a recommendation list simultaneously considering user's preference and user's intention. Here are the user's historical interactions: 'The Blind Swordsman: Zatoichi Takeshi Kitano; Tadanobu Asano japan', 'Buffy the Vampire Slayer - The Complete Seventh Season', 'Number Ones', '24: Season 3', 'Madonna - The Immaculate Collection', 'Spider-Man 2', 'Heavy Metal 2000 VHS', 'MI-5, Vol. 1', 'MI-5, Vol. 2', 'Beyond Re Animator VHS', and user's intention: In the recommendation list, the proportion of **'Art House & International, By Original Language, Chinese'** items should be approximately 50%. You need to generate a recommendation list with 5 different items.	
$\mathbf{GPT-3.5}$	1. Oldboy 2. Chungking Express 3. Battle Royale 4. In the Mood for Love 5. Memories of Murder
$\mathbf{Llama2-7b}$	1. The Blind Swordsman: Zatoichi 2. Buffy the Vampire Slayer 3. Number Ones 4. 24: Season 3 5. Madonna - The Immaculate Collection
InstructRec	1. The Magnificent Seven VHS 2. Shanghai Noon 3. X-Men: First Class 4. X-Men VHS 5. Akira VHS
$\mathbf{Ours_{full}}$	1. **Five Deadly Venoms VHS** 2. Ip Man 2 3. The Village: Vista Series 4. Dawn of the Dead 5. Fong Sai Yuk VHS
Instruction $\mathbf{I_1^{-C}}$**:** You are an expert recommender engine. You need to generate a recommendation list simultaneously considering user's preference and user's intention. Here are the user's historical interactions: 'Yet Another Zombie Defense HD', 'Battlefield™ V', 'Tom Clancy's Ghost Recon® Wildlands', 'Agents of Mayhem', 'Beholder', 'HELLDIVERS™ Dive Harder Edition', 'SUPERHOT', 'Watch_Dogs® 2', 'Deus Ex: Mankind Divided', 'POSTAL 2', and user's intention: Don't recommend me any **'Shooter'** products. You need to generate a recommendation list with 5 different items.	
$\mathbf{GPT-3.5}$	1. Stardew Valley 2. Hollow Knight 3. Cities: Skylines 4. Divinity: Original Sin 2 5. The Witness
$\mathbf{Llama2-7b}$	1. Assassin's Creed Odyssey 2. The Witcher 3: Wild Hunt 3. Horizon Zero Dawn 4. Far Cry 5 5. Life is Strange 2
InstructRec	1. POSTAL 2 2. POSTAL Redux 3. Half-Life 2 4. Resident Evil 0 5. POSTAL
$\mathbf{Ours_{full}}$	1. The Witcher® 3: Wild Hunt 2. Human: Fall Flat 3. **The Forest** 4. Outlast 5. Middle-earth™: Shadow of War™

Table 6: Two real examples for case study. We highlight some texts for Repeat with history item, In target category item and **target item**.

Control		$I_1^{+C_1}$ & $I_1^{-C_2}$		$I_2^{C_1P\leq20\%}$ & $I_1^{-C_2}$		
Dataset	Model	HR@10	TC$_{1-2}$P@10(%) ↑	HR@10	C_1PA ↑	TC$_2$P@10(%) ↓
Movie	$\mathrm{GPT-3.5}$	0.0176	7.15(2.20)	0.0132	9.84(6.37)	**12.51**(19.45)
	Llama2	0.0155	4.51(2.79)	0.0226	16.50(10.34)	17.04(23.82)
	InstructRec	0.0944	27.23(7.14)	0.0333	12.83(9.47)	51.81(31.91)
	PALR	0.0762	7.99(7.60)	0.0692	33.21(6.79)	38.08(53.46)
	Ours$_{\mathrm{v1}}$	0.1002	8.21(8.05)	0.0965	28.40(11.94)	28.69(31.27)
	Ours$_{\mathrm{v2}}$	0.2031	34.63(8.04)	0.1012	29.13(13.21)	25.28(30.40)
	Ours$_{\mathrm{v3}}$	0.2033	44.16(7.88)	0.0927	36.25(15.07)	22.01(28.88)
	Ours$_{\mathrm{full}}$	**0.2064**	**49.61**(7.89)	**0.1018**	**45.14**(15.95)	16.65(28.50)
Steam	$\mathrm{GPT-3.5}$	0.0194	34.15(21.85)	0.0183	15.38(4.21)	22.13(34.18)
	Llama2	0.0093	18.14(16.04)	0.0073	12.49(4.99)	30.40(38.49)
	InstructRec	0.0432	27.94(18.63)	0.0133	13.84(20.37)	38.23(34.63)
	PALR	0.0353	10.52(11.09)	0.0294	7.04(2.53)	41.31(47.31)
	Ours$_{\mathrm{v1}}$	0.0859	25.07(25.31)	0.0842	21.58(12.25)	36.18(41.81)
	Ours$_{\mathrm{v2}}$	0.2620	42.39(26.95)	0.1187	54.06(14.58)	29.63(39.34)
	Ours$_{\mathrm{v3}}$	0.2729	48.27(28.83)	0.1183	75.56(17.74)	22.38(35.74)
	Ours$_{\mathrm{full}}$	**0.2807**	**56.00**(28.99)	**0.1204**	**77.26**(17.81)	**18.82**(35.56)

Table 7: Results of two combinatorial control instructions: (1) $I_1^{+C_1}$ & $I_1^{-C_2}$; (2) $I_2^{C_1P\leq20\%}$ & $I_1^{-C_2}$. The best result is highlighted in **boldface** and the runner-up is denoted with underline. Values in parentheses show outcomes of the corresponding model without control signals.

sity to fine-tune LLMs for better alignment with recommendation tasks. P5 (Geng et al., 2022) introduced a unified framework that integrates 5 recommendation tasks through fine-tuning on the FLAN-T5 model (Raffel et al., 2020). Subsequently, InstructRec (Zhang et al., 2023) tailored FLAN-T5 for various downstream recommendation tasks using instruction tuning. TALLRec (Bao et al., 2023) fine-tunes LLaMA for recommendations with very few training samples, but it focuses on the binary classification task. PALR (Chen, 2023) employs two types of instructions for instruction tuning to facilitate list-level recommendation generation.

Despite these advancements, current studies have not fully explored the potential of LLMs to enhance the interactivity of recommender systems. We aim to harness the instruction-following prowess of LLMs for recommendation tasks through fine-tuning, to create a conversational, controllable, and interactive recommender agent.

5 Conclusion

In conclusion, our paper presents a new approach to tailor LLMs for interactive recommender systems, combining a supervised learning phase with innovative control tasks and a reinforcement learning stage with specialized reward signals. Our method successfully meets the detailed demands of recommendation contexts, enhancing LLMs' performance. Experimental results show our approach exceeds current LLM-based systems in precision, controllability, and presentation, offering a significant step towards refined and reliable recommendation services.

6 Limitation

This paper's primary constraints are as follows: (1) The emphasis is placed on improving the LLM's ability to follow recommendation-related instructions. This focus may inadvertently compromise the LLM's broader intellectual capabilities, as revealed in Table 5. How to further reduce catastrophic forgetting remains a big challenge. (2) In real-world scenarios, users often have diverse control intentions, including intricate blends of various instructions (as discussed in Section 3.5) and new instructions beyond category control. As foundational research, this paper addresses only the most critical elements, specifically category control and formatting control. More diverse and complicated instructions are yet to be explored.

7 Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant Nos. 62276171 and 62072311), Guangdong Basic and Applied Basic Research Foundation (Grant Nos. 2024A1515011938, 2020B1515120028), Guangdong Provincial Key Laboratory of Popular High Performance Computers (Grant No. 2017B030314073), Guangdong Province Engineering Center of China-made High Performance Data Computing System, Shenzhen Fundamental Research-General Project (Grant Nos. 20220811155803001, 20210324094402008). Hao Liao is the corresponding author.

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*, RecSys '23, page 1007–1014, New York, NY, USA. Association for Computing Machinery.

Zheng Chen. 2023. Palr: Personalization aware llms for recommendation. *arXiv preprint arXiv:2305.07622*.

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, page 1126–1132, New York, NY, USA. Association for Computing Machinery.

Wenqi Fan, Zihuai Zhao, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Jiliang Tang, and Qing Li. 2023. Recommender systems in the era of large language models (llms). *arXiv preprint arXiv:2307.02046*.

Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524*.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, RecSys '22, page 299–315, New York, NY, USA. Association for Computing Machinery.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Wang-Cheng Kang and Julian J. McAuley. 2018. Self-attentive sequential recommendation. In *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*, pages 197–206. IEEE Computer Society.

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*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Zhaopeng Qiu, Xian Wu, Jingyue Gao, and Wei Fan. 2021. U-bert: Pre-training user representations for improved recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4320–4327.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *EMNLP*, pages 14918–14937. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2022. Userbert: Pre-training user model with contrastive self-supervision. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2087–2092, New York, NY, USA. Association for Computing Machinery.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2023. A survey on large language models for recommendation. *arXiv preprint arXiv:2305.19860*.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023. Agenttuning: Enabling generalized agent abilities for llms. *arXiv preprint arXiv:2310.12823*.

Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023. Recommendation as instruction following: A large language model empowered recommendation approach. *arXiv preprint arXiv:2305.07001*.

A Appendix

A.1 Dataset detail

We present detailed information about the instruction set we construct in Table 8, including the quantity of each type of instruction.

Instruct type	Movie	Steam
I_0	13218	12658
(I_1^{+C}, I_1^{-C})	$(6609, 6609)$	$(6329, 6329)$
$(I_2^{CP\leq *}, I_2^{CP\geq *}, I_2^{CP\approx *})$	$(4406, 4406, 4406)$	$(4219, 4219, 4220)$
I_3	18744	8572
I_4	29199	23273
ALL	87597	69819

Table 8: Constructed instruction set

Algorithm 1: Calculation of $Scores^{ctl}$

```
input  : $items = [item_1, ..., item_N], k, C_{target}$
output : Each item's score $Scores^{ctl}$
init   : $Scores^{ctl} = \mathbf{0}^{1 \times N}$
init   : $Count_{in} = Count_{out} = 0$
for $i \leftarrow 1$ to $N$ do
    if $item_i$ is illegal then
        $Scores_i^{ctl} = -1$
        continue
    if $item_i \in C_{target}$ then $in = 1, out = 0$;
    else $in = 0, out = 1$;
    $Count_{in} += in, Count_{out} += out$
    if $I_0$ then $s = Scores_i$;
    if $I_1^{+C}$ then $s = in$;
    if $I_1^{-C}$ then $s = out$;
    if $I_2^{CP\leq m}$ then
        if $Count_{out} > (k - k * m)$ then $s = 0.5$;
        else if $out$ then $s = 1.0$;
        else if $Count_{in} < k * m$ then $s = 0.5$;
        else $s = 0.0$;
    if $I_2^{CP\geq m}$ then
        if $Count_{in} > k$ then $s = 0.5$;
        else if $in$ then $s = 1.0$;
        else if $Count_{out} < (k - k * m)$ then
          $s = 0.5$;
        else $s = 0.0$;
    if $I_2^{CP\approx m}$ then
        if $in$ then
            if $Count_{in} \leq k * m$ then $s = 1.0$;
            else $s = 0.0$;
        else if $Count_{in} \geq k * m$ then $s = 1.0$;
        else if $Count_{out} \leq (k - k * m)$ then
          $s = 0.5$;
        else $s = 0.0$;
    $Scores_i^{ctl} = s$
```

A.2 Computational Cost

Finetuning LLMs for recommendations is a little bit more expensive than training traditional recommender models like SASRec, but the resource and cost are not a big barrier even for small organizations or researchers. Here we take the Movie dataset for illustration. The dataset statistics can be found in Table 1 and Table 7.

For the SL stage, we use 4 A100 GPUs (40GB GPU memory) for training. The batch size on each GPU is 1 and the gradient accumulation step of 16. The maximum sequence length is set to 1024. Under this setting, each epoch costs about 70 minutes. Usually, models can fully converge in 30 epochs. So, the total training time for SFT is about 35 hours.

For the RL stage, we use 2 A100 GPUs (40GB GPU memory) for training. The batch size on each GPU is 1 and the gradient accumulation step of 2. For each batch, instruction sampling time + 2 candidate response generation time + model training time counts about 40 seconds, and the maximum training steps are set to 3k. Thus, the total RL training time is $3000 * 40/60/60 = 33$ hours.

Once the training process is finished, we use vllm[5] for inference and serving. On a single A100 GPU (40GB memory), when responding to the top-10 recommendation request, our implementation can process 20 requests per second (on average each request involves newly generated 112 tokens).

A.3 Prompts

In this section, we show the prompt for the instruction used in the experiment. The prompt of I_0, I_1, I_2, I_3 is illustrated in Listing 1 to Listing 4 respectively. The prompt of positive and negative category control intentions is illustrated in Listing 5 to Listing 6 respectively

[5] https://github.com/vllm-project/vllm

Listing 1: Prompts of I_0

```
Instruction: You are an expert recommender engine. You need to generate a
    recommendation list considering user’s preference from historical interactions.
    The historical interactions are provided as follows: {history}. You need to
    generate a recommendation list with {item_count} different items.
Output: {item_list}

Instruction: You are an expert recommender engine. You need to select a
    recommendation list considering user’s preference from historical interactions.
    The historical interactions are provided as follows: {history}. The candidate
    items are: {candidate_titles}. You need to select a recommendation list with
    {item_count} different items from candidate items.
Output: {item_list}
```

Listing 2: Prompts of I_1

```
Instruction: You are an expert recommender engine. You need to generate a
    recommendation list simultaneously considering user’s preference inferred from
    historical interactions and user’s intention. If user’s preference conflicts
    with his intention, you should comply with his intention. Here are user’s
    historical interactions: {history}, and user’s intention:
    {synthetic_intention}. You need to generate a recommendation list with
    {item_count} different items.
Output: {item_list}

Instruction: You are an expert recommender engine. You need to select a
    recommendation list from candidate items simultaneously considering user’s
    preference inferred from historical interactions and user’s intention. If
    user’s preference conflicts with his intention, you should comply with his
    intention. Here are user’s historical interactions: {history}, and user’s
    intention: {synthetic_intention}. The candidate items are: {candidate_titles}.
    You need to select a recommendation list with {item_count} different items from
    candidate items.
Output: {item_list}
```

Listing 3: Prompts of I_2

```
Instruction: You are an expert recommender engine. You need to generate a
    recommendation list simultaneously considering user’s preference inferred from
    historical interactions and user’s intention. Here are user’s historical
    interactions: {history}, and user’s intention: In the recommendation list, the
    proportion of ’{target_category}’ items should be less than or equal to
    {category_proportion}. You need to generate a recommendation list with
    {item_count} different items.
Output: {item_list}

Instruction: You are an expert recommender engine. You need to generate a
    recommendation list simultaneously considering user’s preference inferred from
    historical interactions and user’s intention. Here are user’s historical
    interactions: {history}, and user’s intention: In the recommendation list, the
    proportion of ’{target_category}’ items should be more than or equal to
    {category_proportion}. You need to generate a recommendation list with
    {item_count} different items.

Instruction: You are an expert recommender engine. You need to generate a
    recommendation list simultaneously considering user’s preference inferred from
    historical interactions and user’s intention. Here are user’s historical
    interactions: {history}, and user’s intention: In the recommendation list, the
    proportion of ’{target_category}’ items should be approximately
    {category_proportion}. You need to generate a recommendation list with
    {item_count} different items.
Output: {item_list}
```

Listing 4: Prompts of I_3

```
You are an expert recommender engine. You need to generate a recommendation list
    complying user’s intention. Here is user’s intention: {synthetic_intention}.
    Please generate a recommendation list with {item_count} different items.
```

```
Output: {item_list}

You are an expert recommender engine. You need to select a recommendation list
    complying user’s intention from candidate items. Here is user’s intention:
    {synthetic_intention}. The candidate items are: {candidate_titles}. Please
    select a recommendation list with {item_count} different items from candidate
    items.
Output: {item_list}
```

Listing 5: Prompts of positive intention in I_1 and I_3

```
I like ’{target_category}’ products
Please recommend some ’{target_category}’ items
I’m interested in ’{target_category}’
I would like to buy some ’{target_category}’ products
I would like to browse some ’{target_category}’ products
I prefer in ’{target_category}’ item
```

Listing 6: Prompts of negative intention in I_1 and I_3

```
I don’t like ’{target_category}’ products
Please exclude any ’{target_category}’ item
I’m not interested in ’{target_category}’
Don’t recommend me any ’{target_category}’ products
I don’t want to browse any ’{target_category}’ product
I hate ’{target_category}’ items
```