Logit Space Constrained Fine-Tuning for Mitigating Hallucinations in LLM-Based Recommender Systems

Jianfeng Deng[1], **Qingfeng Chen**[1*], **Debo Cheng**[2*], **Jiuyong Li**[3], **Lin Liu**[3]
[1]Guangxi University, Nanning, China
[2]Hainan University, Haikou, China
[3]University of South Australia, Adelaide, Australia
jianfeng_web@163.com, qingfeng@gxu.edu.cn
chedy055@mymail.unisa.edu.au
{jiuyong.li, lin.liu}@unisa.edu.au

Abstract

Large language models (LLMs) have gained increasing attention in recommender systems, but their inherent hallucination issues significantly compromise the accuracy and reliability of recommendation results. Existing LLM-based recommender systems predominantly rely on standard fine-tuning methodologies, often ignoring hallucination issues during the fine-tuning process. To address this challenge, we propose Logit Space Constraints Fine-Tuning (LCFT), a novel fine-tuning framework designed to mitigate hallucination in LLM-based recommenders. Specifically, LCFT takes as input semantically positive and negative instruction pairs and incorporates Kullback–Leibler (KL) divergence into the training objective to explicitly maximise their distributional disparity in the logit space. By conducting such logit space-constrained fine-tuning, LCFT encourages more distinguishable and semantically grounded representations, thereby reducing the model's susceptibility to hallucination. Extensive experiments on two recommendation models with distinct LLM backbones and four real-world datasets demonstrate that LCFT consistently reduces hallucination and enhances recommendation performance. Our source code is available at: https://github.com/djf-web/LCFT.

1 Introduction

Large language models (LLMs), such as GPT-3 (Ouyang et al., 2022; Brown et al., 2020) and LLaMA (Touvron et al., 2023), are pretrained on massive datasets and demonstrate exceptional capabilities in contextual understanding, knowledge reasoning, and compositional generalisation. Their utility has expanded beyond traditional natural language processing (NLP) (Liang et al., 2022; Chowdhery et al., 2022; Wei et al., 2022) to fields such as robotics (Shah et al., 2022; Xiao et al., 2022; Driess et al., 2023) and information retrieval (Jeronymo et al., 2023; Wang et al., 2023; Jin et al., 2023; Li et al., 2025; Liu et al., 2025). By parsing complex instructions and generating predictive, context-aware outputs, LLMs are redefining the landscape of machine learning research. In particular, recommender systems, critical for personalised content delivery, stand to benefit significantly (Chen et al., 2024; Zhang et al.; Deng et al., 2024). LLMs can mitigate cold-start problems for users and items by leveraging their extensive pretrained knowledge (Sanner et al., 2023), support dynamic modelling through contextual reasoning (Dai et al., 2023a; Gao et al., 2023), and introduce novel mechanisms to transcend the limitations of traditional collaborative filtering approaches.

Recommendation methods based on LLMs have demonstrated strong effectiveness across various recommendation scenarios. In recent years, a wide range of LLM-based recommendation algorithms have been proposed. One line of work directly employs LLMs as generators for recommender systems (Gao et al., 2023; Liu et al., 2023; Dai et al., 2023b; Sanner et al., 2023). For example, (Sanner et al., 2023) validates the effectiveness of using pretrained LLMs directly as generators in cold-start settings; (Gao et al., 2023) constructs prompts based on users' historical interactions and interacts with the LLM to generate recommendations; (Dai et al., 2023b) integrates LLMs with traditional recommendation models to enhance performance. However, due to the discrepancy between the pretraining objectives of LLMs and the specific goals of recommendation tasks, using LLMs directly as recommendation generators still faces performance limitations in practical applications.

To mitigate the discrepancy between the pretraining objectives of LLMs and the goals of recommendation tasks, several studies have explored instruction tuning of LLMs using recommendation datasets to enhance their performance in recom-

*Corresponding authors

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 29311–29324
November 4-9, 2025

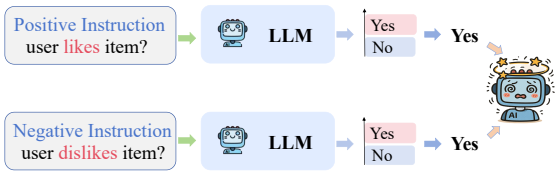

Figure 1: Hallucinations in LLM-based recommender systems can seriously affect the accuracy of recommendations. This makes the LLM-based recommender system behave like it is guessing, reducing the reliability and usefulness of its recommendations.

mendation scenarios. Among them, one group of studies focuses on fine-tuning LLMs using ID information (Bao et al., 2023; Li et al., 2023; Feng et al., 2023). For example, (Bao et al., 2023) constructs instruction-tuning datasets from recommendation datasets to fine-tune LLMs and improve their recommendation capabilities; (Li et al., 2023) combines LLMs with traditional recommendation models and applies joint fine-tuning to improve efficiency; and (Feng et al., 2023) adopts reinforcement learning to optimise LLMs, significantly enhancing their conversational recommendation ability. Meanwhile, another line of research incorporates collaborative information from recommendation datasets into the fine-tuning process to further boost performance (Zhang et al., 2023; Zheng et al., 2024; Liao et al., 2024; Zhu et al., 2024; Hong et al., 2025; Zhang et al., 2025). For instance, (Zhang et al., 2023) fine-tunes LLMs to generate semantically rich item tags, which are then used to improve ranking accuracy; (Zheng et al., 2024) integrates both linguistic and collaborative semantics during fine-tuning to enhance performance; (Liao et al., 2024) proposes a hybrid item representation method that combines the strengths of traditional recommender systems and LLMs; and (Zhang et al., 2025) incorporates collaborative information into LLM-based recommenders, significantly improving recommendation performance. However, these methods often rely on conventional fine-tuning techniques such as LoRA (Hu et al., 2022). Such techniques remain vulnerable to hallucinations generated by LLMs, which may compromise the accuracy and reliability of recommendation results.

In natural language processing (NLP), hallucination typically refers to a model generating content that is not grounded in the input or real-world facts (Ji et al., 2023). In recommender systems, hallucination refers to cases where the preference is fabricated. However, since the ground truth of preferences is often unknown until interaction occurs, we use contradictions to identify instances of hallucination in preference reasoning. Specifically, we propose an interpretable and verifiable definition of hallucination in the recommendation setting: if a model produces the same or logically inconsistent responses to semantically opposite instructions (e.g., "Will the user like this item?" vs. "Will the user dislike this item?"), it can be considered as hallucinating. From a commonsense reasoning perspective, a user cannot simultaneously like and dislike the same item. This contradiction signals a breakdown in the model's internal coherence and motivates our approach to mitigate such inconsistencies through contrastive instruction tuning.

As shown in Figure 1, the hallucination problem of LLM-based recommender systems can significantly impact the accuracy of recommender systems. Ideally, if an LLM-based recommender system does not suffer from hallucination and possesses strong understanding capabilities, it should generate different recommendation results when given semantically opposite instructions. However, LLM-based recommender systems are often affected by hallucinations and tend to produce identical recommendations even for contradictory prompts. In such cases, the model's behaviour resembles random guessing, which severely undermines the reliability and practical utility of the recommendations.

To address the hallucination problem in LLM-based recommender systems, we propose a novel fine-tuning framework—Logit Space Constraints Fine-Tuning (LCFT). Unlike traditional methods that rely on explicit negative feedback or modifications to original user behavior data, LCFT leverages contrastive instruction tuning. The LCFT framework begins by constructing semantically opposing instruction pairs, such as "Will the user click on this item?" (positive) and "Will the user not click on this item?" (negative). It then fine-tunes the LLM using LoRA (Hu et al., 2022) by maximising the Kullback–Leibler (KL) divergence between the logit distributions of these instruction pairs. This fine-tuning objective encourages the model to produce distinguishable logit distributions in response to semantically contrasting inputs, reinforcing its ability to differentiate instruction semantics. As a result, the model becomes more sensitive to instruction differences, generating appropriately contrasting responses and reducing the likelihood

of hallucinations. Importantly, LCFT is a general fine-tuning framework that can be integrated into existing LLM-based recommender systems. Extensive experiments on two LLM-based recommender systems (LLama-7B and Vicuna-7B) and four real-world datasets demonstrate that LCFT consistently mitigates hallucinations and improves recommendation accuracy across different LLM backbones.

In summary, the main contributions of our work include:

- We identify the hallucination problem in current LLM-based recommender systems, where the model generates identical recommendations even when given semantically opposite instructions. This behaviour resembles "guessing" rather than true "understanding", severely undermining the accuracy and reliability of the recommendations.
- We propose a novel fine-tuning framework for LLM, Logit Space Constraints Fine-Tuning (LCFT), which effectively mitigates the hallucination problem in current LLM-based recommender systems, thereby improving the accuracy and reliability of recommendation results.
- Extensive experiments on LLM-based recommender systems using two distinct LLM backbones and four real-world datasets demonstrate that LCFT consistently mitigates hallucinations and enhances recommendation performance across diverse scenarios.

2 Related Work

In this section, we review several studies related to our work. Traditional recommender systems primarily rely on collaborative filtering, content-based methods, or hybrid models (Koren et al., 2009; Guo et al., 2017; Wang et al., 2017; Zhou et al., 2018; He et al., 2020; Hidasi et al., 2016; Cui et al., 2020; Tang and Wang, 2018; Yuan et al., 2019). These approaches typically make use of users' historical interactions to generate recommendations. While they have achieved widespread success in practice, they still show limitations in modelling complex semantic relationships.

In recent years, recommendation methods based on LLMs have gained increasing attention. These methods can generally be categorised into two types: one directly applies pre-trained LLMs to recommendation tasks, while the other fine-tunes LLMs on recommendation datasets before deployment.

2.1 Zero-tuning

With the rapid development of LLMs, there have been an increasing number of studies to explore their potential in recommender systems. A representative line of work involves directly applying pre-trained LLMs to recommendation tasks (Gao et al., 2023; Liu et al., 2023; Dai et al., 2023b; Sanner et al., 2023). For example, (Liu et al., 2023) systematically evaluates the recommendation performance of ChatGPT across multiple application scenarios; (Gao et al., 2023) proposes a method that incorporates user profiles and historical interactions into prompts for LLMs, generating recommendations through interaction with the model; (Dai et al., 2023b) combines ChatGPT with traditional information retrieval techniques to enhance recommendation quality; (Sanner et al., 2023) focuses on evaluating the performance of LLM-based recommendation methods in cold-start settings. However, since these LLMs are not specifically designed or trained for recommendation tasks, their performance remains limited.

2.2 Fine-tuning

To overcome the limitations of directly using LLMs in recommendation tasks, recent research has explored instruction tuning of LLMs based on recommendation datasets to enhance their recommendation capabilities. Some studies focus on fine-tuning LLMs using ID information (Bao et al., 2023; Li et al., 2023; Feng et al., 2023). For instance, (Bao et al., 2023) fine-tunes LLMs using recommendation scenario-specific instructions to improve recommendation performance; (Li et al., 2023) combines LLMs with traditional recommendation models for fine-tuning to enhance efficiency; and (Feng et al., 2023) adopts reinforcement learning strategies to improve LLM performance in conversational recommendation settings.

Another line of works integrates collaborative information to further improve fine-tuning outcomes (Zhang et al., 2023; Zheng et al., 2024; Liao et al., 2024; Zhu et al., 2024; Hong et al., 2025; Zhang et al., 2025). For example, (Liao et al., 2024) proposes a hybrid item representation approach that effectively combines the strengths of traditional and LLM-based recommendation; (Zhang et al., 2023) fine-tunes LLMs to generate high-quality labels for items, enhancing ranking performance;

(Zheng et al., 2024) incorporates both linguistic and collaborative semantics during fine-tuning to boost model performance; and (Zhang et al., 2025) fuses collaborative information into the fine-tuning process to significantly improve recommendation performance. Although the above methods have improved LLM-based recommendation to some extent, they often rely on conventional fine-tuning techniques (e.g., LoRA (Hu et al., 2022)) and are still affected by the hallucination problem in LLMs, which reduces recommendation accuracy.

In contrast, we propose a fine-tuning framework designed to mitigate the hallucination issue in LLMs, which can be seamlessly integrated into LLM-based recommender systems. This framework significantly enhances both the accuracy and reliability of recommendations.

3 Preliminary

3.1 Problem Statement

Traditional instruction fine-tuning approaches for LLM-based recommender systems primarily focus on maximising the accuracy of user preference prediction. Their objective function can be formulated as: $\max_{\theta} \sum_{(u,i)\in\mathcal{D}} \log P_\theta(Y \mid X)$, where X denotes the instruction generated based on the interaction history between user u and item i, Y is the output of the LLM, and θ represents the model parameters. However, these methods often neglect the hallucination problem—when presented with semantically contradictory instructions, LLMs may produce nearly identical outputs, indicating a lack of semantic sensitivity and undermining the reliability of the recommendations.

To address this limitation, LCFT introduces a contrastive fine-tuning strategy with an objective that maximises the KL divergence between the logit distributions of positive and negative instruction pairs. This contrastive training encourages the model to produce distinguishable outputs for contrasting inputs, enhancing its ability to capture fine-grained instruction semantics. As a result, LCFT effectively mitigates hallucinations.

3.2 Instruction Tuning

Although LLMs (such as GPT, LLaMA, etc.) possess powerful language understanding and generation capabilities, they are often better at handling text than performing specific tasks (such as recommendation tasks) without specialised training. To enhance their ability to carry out specific tasks, these models typically require instruction tuning. Instruction tuning is a method of retraining LLMs using a large number of human-annotated instruction-response pairs. Its core objective is to improve the model's ability to understand and follow natural language instructions, thereby enhancing its generalisation and practicality in specific situations.

In recommendation tasks, instruction tuning typically involves the following four steps:

Step 1 (Define the Recommendation Task Instruction): Clarify the objective of the recommendation task and convert it into a natural language instruction.

Step 2 (Construct Input-Output Pairs): Build the task inputs and outputs based on the user's historical behaviour.

Step 3 (Form Instruction Tuning Samples): Combine the task instruction with the task input to create an instruction input, and use the task output as the instruction output.

Step 4 (Construct the Instruction Dataset): Organise the formatted samples into a standard format (*Instruction Input, Instruction Output*), which is then used for instruction tuning of the model.

An example of formatting recommendation data for instruction tuning is shown below.

Instruction Input
Task Instruction: Given the user's historical interactions, please determine whether the user will like the target new book by answering "Yes" or "No". **Task Input:** User's liked items: Sula. User's disliked items: Pigs in Heaven. Target new book: The Bean Trees.
Instruction Output
Task Output: No.

4 Methodology

This section provides a detailed description of our proposed LCFT framework, including the overview of LCFT and the construction of pairs of semantically opposite instructions. Further, we introduce how to incorporate a KL divergence term into the loss function during the fine-tuning process.

4.1 Overview of the LCFT Method

As shown in Figure 2, our overall approach consists of two parts: prompt construction and the

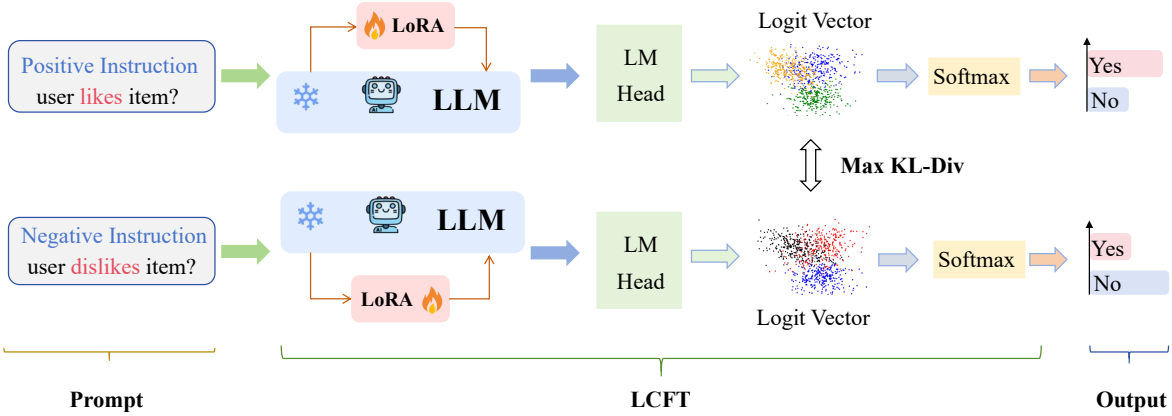

Figure 2: The overview of our proposed LCFT method. LCFT fine-tunes the LLM using LoRA by leveraging pairs of semantically opposite instructions—positive and negative. During fine-tuning, it maximises the KL divergence between the logit distributions of these paired inputs, encouraging the model to generate clearly distinguishable responses.

LCFT fine-tuning method. (1) Prompt Construction: We first construct semantically opposite instruction pairs—positive instruction X_{pos} and negative instruction X_{neg}—to form contrastive prompts for training. (2) LCFT (Logit Constrained Fine-Tuning): Based on these constructed prompts, the LLM is fine-tuned via LoRA to maximise the KL divergence between their output logit distributions, encouraging the model to distinguish between positive and negative instruction, thereby guiding it to generate differentiated responses when given positive versus negative instructions.

4.2 Prompt Construction

Our fine-tuning framework relies on a pair of positive and negative instruction inputs, where the negative instruction is generated by semantically inverting the positive instruction.

Existing LLM-based recommender systems typically use fixed prompt templates for positive instructions. For example, TALLRec adopts the following fixed prompt template structure:

> • **Positive instruction prompt template.** #
> Task Instruction: Given the user's historical interactions, please determine whether the user will likes the target new movie by answering "Yes" or "No".
> Task Input: User's liked items: GodFather. User's disliked items: Star Wars. Target new movie: Iron Man
> Task Output: No.

Based on the existing positive prompt templates used in LLM-based recommender systems, we construct corresponding negative prompt templates. For example, given the fixed prompt structure used by TALLRec, we generate the following corresponding negative prompt template:

> • **Negative instruction prompt template.** #
> Task Instruction: Given the user's historical interactions, please determine whether the user will dislikes the target new movie by answering "Yes" or "No".
> Task Input: User's liked items: GodFather. User's disliked items: Star Wars. Target new movie: Iron Man
> Task Output: Yes.

We construct negative instruction prompts via semantic inversion to enhance fine-tuning. This creates a clear contrast between positive and negative outputs, promoting better separation in the model's response distributions and improving KL divergence optimisation. By altering only the instruction while keeping the input fixed, the model learns to focus more precisely on task intent, reducing hallucinations.

4.3 Logit Constrained Fine-Tuning

LLMs have a vast number of parameters, making full-parameter fine-tuning computationally expensive. Since most of the knowledge in LLMs lies in a low-dimensional subspace (Hu et al., 2022), LCFT adopts the efficient LoRA approach (Hu et al., 2022), which improves performance by updating only a small subset of parameters (Li and Liang, 2021; Houlsby et al., 2019; Lester et al., 2021). Specifically, LCFT keeps all pretrained

LLM parameters frozen and trains only additional low-rank matrices to efficiently incorporate task-specific information. The original LoRA training objective in LCFT is:

$$\mathcal{L}_{LoRA} = \min_{\Theta} - \sum_{(X,Y)\in\mathcal{Z}} \sum_{t=1}^{|Y|} \log P_{\Phi+\Theta}(Y_t|X, Y_{<t}) \quad (1)$$

where Φ denotes the original parameters of the LLM $\mathcal{M}$, which are kept frozen during training. Θ represents the LoRA parameters, which are updated during training (accounting for only about 0.1% of the original LLM parameters (Hu et al., 2022)). X denotes the instruction input of an arbitrary single sample, including the positive instruction and the negative instruction, and Y is the instruction output. Y_t denotes the t-th token of Y, and $Y_{<t}$ refers to the tokens preceding Y_t. $\mathcal{Z}$ represents the training dataset.

In addition, LCFT takes both positive and negative instruction samples as input, constructs different logit distributions, and maximises the KL divergence between the logit distributions of the positive and negative instruction samples, thereby guiding the model to generate clearly distinguishable instruction responses. The KL divergence objective in LCFT is formulated as:

$$\mathcal{L}_{KL} = \min_{\Theta} - \sum_{(X_{\text{pos}},X_{\text{neg}},Y)\in\mathcal{Z}} \sum_{t=1}^{|Y|} \left[D_{KL}\left(P_{\Phi+\Theta}(Y_t|X_{\text{pos}}, Y_{<t}) \parallel P_{\Phi+\Theta}(Y_t|X_{\text{neg}}, Y_{<t})\right)\right] \quad (2)$$

where X_{pos} denotes the positive instruction input, and X_{neg} denotes the negative instruction input. $D_{KL}(\cdot\|\cdot)$ represents the Kullback-Leibler divergence, which measures the discrepancy between the logit representations distribution generated from the positive instruction input and that generated from the negative instruction input.

In summary, the final learning objective of LCFT can be described as:

$$\mathcal{L}_{LCFT} = \mathcal{L}_{LoRA} + \lambda \cdot \mathcal{L}_{KL} \quad (3)$$

where $\mathcal{L}_{LoRA}$ denotes the original LoRA fine-tuning objective, and $\mathcal{L}_{KL}$ represents the training objective for KL divergence optimisation.

5 Experiments

In this section, we conduct experiments to answer the following research questions:

RQ1: Compared to the fine-tuning strategies used by existing LLM-based recommender systems, does the LCFT perform better under few-shot training and full-shot training conditions?

RQ2: When integrated into LLM-based recommender systems with different LLM backbones, how does the performance of the fine-tuning framework we propose compare?

RQ3: How does the performance of LCFT change when the KL divergence term is removed during fine-tuning, no longer constraining the logit space of the LLM?

RQ4: How do hyperparameters influence the performance of the LCFT framework?

5.1 Experimental Settings

We first introduce the datasets and few-shot training setting, then describe the baseline methods and evaluation metrics, and finally present the parameter settings for the LCFT model.

Datasets. We validate the effectiveness of the proposed model on four publicly available benchmark datasets: movie (F.Maxwell and Konstan, 2015), book (Ziegler et al., 2005), ML-1M (Harper and Konstan, 2015) and Amazon-Book (He and McAuley, 2016). Details of the datasets can be found in Appendix A.

Few-shot training setting. In real-world applications, data from specific domains is often sparse. Therefore, few-shot training is of great importance under data-limited scenarios. In the few-shot setting, the model is fine-tuned using only a small number of samples from the training set, where "shot" refers to the number of training samples used. By setting the shot value to a small number (e.g., 16), we can evaluate whether the method can still effectively leverage LLMs to acquire recommendation capabilities under extremely limited training data. To assess the effectiveness of our proposed fine-tuning framework under the few-shot scenario, we conduct few-shot training experiments on the movie and book datasets. The goal is to evaluate whether the framework can still effectively mitigate hallucination phenomena in recommendation tasks under sample-scarce conditions.

Baselines. To verify the effectiveness of our proposed fine-tuning framework, we integrate it into

Table 1: Under the few-shot training setting, the AUC performance of all methods on the two datasets is shown below. We use bold to indicate the best result and underline to indicate the second-best result. "Rel. Imp." denotes the average relative improvement of LCFT-TALLRec over baseline methods across the three different shot settings.

Dataset	movie				book			
Few-shot / Methods	16	64	256	Rel. Imp.	16	64	256	Rel. Imp.
GRU4Rec	0.4907	0.4987	0.5289	42.3%	0.4895	0.4964	0.4986	26.3%
Caser	0.4968	0.5106	0.5420	39.4%	0.4984	0.4972	0.4957	25.7%
SASRec	0.5043	0.5048	0.5225	41.1%	0.4948	0.5006	0.5020	25.2%
DROS	0.5076	0.5154	0.5407	38.2%	0.4928	0.4913	0.4913	27.1%
GRU-BERT	0.5085	0.5165	0.5344	38.6%	0.5007	0.4964	0.4979	25.4%
DROS-BERT	0.5021	0.5171	0.5394	38.5%	0.5007	0.4898	0.5020	25.6%
TALLRec	0.6724	0.6748	0.7198	4.5%	0.5636	0.6039	0.6438	3.5%
LCFT-TALLRec	**0.7013**	**0.7157**	**0.7436**	-	**0.6021**	**0.6195**	**0.6531**	-

two existing LLM-based recommender systems and compare it with the following baseline methods: (1) **GRU4Rec** (Hidasi et al., 2016): GRU4Rec applies recurrent neural networks (RNNs) to session-based recommendation by modelling the complete user interaction sequence to enhance recommendation performance. (2) **Caser** (Tang and Wang, 2018): Caser employs convolutional neural networks (CNNs) to capture users' short-term interests and sequential patterns, achieving a unified modelling of both long-term and short-term user preferences. (3) **SASRec** (Kang and McAuley, 2018): SASRec is based on a self-attention mechanism that dynamically selects items from user history most relevant to the current recommendation, integrating short-term interest with long-term semantic information to improve performance. (4) **DROS** (Yang et al., 2023): DROS introduces a distributional adaptation mechanism within the empirical risk minimisation (ERM) framework to simulate potential distribution shifts between training and testing phases, thereby enhancing model robustness and generalisation on unseen data. (5) **GRU-BERT**: GRU-BERT enhances GRU4Rec by incorporating the pretrained language model BERT (Devlin et al., 2019), combining item ID embeddings with textual semantic embeddings to improve item representation and recommendation quality. (6) **DROS-BERT**: DROS-BERT integrates BERT's text modelling capabilities into the DROS framework, combining distributional adaptation with semantic information to further improve performance under distribution mismatch scenarios. (7) **MF** (Koren et al., 2009): MF decomposes the rating matrix into two low-dimensional matrices representing users and items, modelling the latent preference between users and items via inner product. (8) **LightGCN** (He et al., 2020): LightGCN uses a lightweight graph convolutional network to capture high-order connections in the user-item interaction graph, thereby improving user interest modelling. (9) **ICL** (Dai et al., 2023a): ICL leverages the contextual learning capabilities of LLMs to align recommendation strategies with prompts by interacting with ChatGPT through natural language instructions. (10) **Prompt4NR** (Zhang and Wang, 2023): Prompt4NR designs various prompt templates (including fixed and soft prompts) and enhances recommendation performance by integrating predictions from multiple templates. Prompt4NR-V uses Vicuna-7B as the backbone LLM. (11) **TALLRec** (Bao et al., 2023): TALLRec reformulates the recommendation task as an instruction input for LLMs, using instruction tuning to enhance performance, especially in few-shot settings. Following the few-shot training setup of TALLRec (using LLaMA-7B as the backbone LLM), we integrate our fine-tuning framework into it and name the variant LCFT-TALLRec. (12) **CoLLM** (Zhang et al., 2025): To improve the performance of LLM-based recommender systems, CoLLM integrates collaborative information into the input of LLMs. CoLLM-MF uses MF as the collaborative modelling module. Following its setup (using Vicuna-7B as the backbone LLM), we integrate our fine-tuning framework into CoLLM-MF and name the variant LCFT-CoLMF.

Evaluation metrics. We employ AUC and UAUC (Liu et al., 2021) as evaluation metrics to assess the performance of recommendation methods. To ensure the stability of the results, each method

Table 2: Under the full-shot training setting, the performance results of all methods on the two datasets are presented below. Similarly, we use bold to indicate the best result and underline to indicate the second-best result. "Rel. Imp." denotes the average relative improvement of LCFT-CoLMF over the baseline methods on the two evaluation metrics.

Dataset	ML-1M			Amazon-Book		
Methods	AUC	UAUC	Rel. Imp.	AUC	UAUC	Rel. Imp.
MF	0.6482	0.6361	12.7%	0.7134	0.5565	14.5%
LightGCN	0.5959	0.6499	16.2%	0.7103	0.5639	14.1%
SASRec	0.7078	0.6884	3.6%	0.6887	0.5714	15.4%
ICL	0.5320	0.5268	36.7%	0.4820	0.4856	50.3%
Prompt4NR-V	0.7071	0.6739	4.8%	0.7224	0.5881	11.0%
CoLLM-MF	<u>0.7295</u>	<u>0.6875</u>	2.1%	<u>0.8109</u>	<u>0.6225</u>	1.5%
LCFT-CoLMF	**0.7451**	**0.7019**	-	**0.8221**	**0.6323**	-

is run five times with different random seeds, and the average performance is reported.

Implementation and Parameter Settings. We implement LCFT using PyTorch and optimise the LLM with the AdamW optimiser (Kingma and Ba, 2015). Details of the parameter settings can be found in Appendix B.

5.2 Performance Comparison

5.2.1 Few-shot training setting (RQ1)

To evaluate the effectiveness of our proposed fine-tuning framework in mitigating hallucination under limited training data conditions, we conducted an analysis of its performance in a few-shot setting. The table 1 presents a comparison between our method and approaches without hallucination mitigation strategies. Based on the experimental results, we have the following observations: (1) Compared with various baseline models, our fine-tuning framework significantly improves the performance of LLM-based recommender systems under a few-shot training setting, indicating that the framework remains effective in mitigating hallucinations and enhancing recommendation quality even in data-scarce scenarios. (2) Traditional recommendation models do poorly with few-shot training because they struggle to learn from limited data and can't match LLM-based systems in semantic understanding and generalisation. (3) LLM-based recommender systems trained using conventional fine-tuning methods achieve suboptimal performance. This can be attributed to the inability of traditional fine-tuning to effectively address the inherent hallucination problem of LLMs, which negatively impacts the accuracy and stability of the recommendations.

5.2.2 Full-shot training setting (RQ1)

To further validate the effectiveness of our fine-tuning framework under sufficient training data conditions (full-shot), we evaluated its performance on the ML-1M and Amazon-Book datasets. The corresponding results are shown in the table 2, leading to the following key findings: (1) Our method achieves the best performance among all baselines, indicating that by constraining the logit representation space of LLMs, our framework effectively suppresses hallucination, thereby enhancing the recommendation performance of LLM-based systems. (2) Overall, LLM-based recommender systems perform better than traditional models because they can capture complex semantic relationships. In contrast, traditional models struggle to represent rich semantic information. (3) Although LLM-based recommender systems trained with traditional fine-tuning strategies perform relatively well, they still fall short of our method. This suggests that conventional fine-tuning fails to effectively mitigate hallucination in LLMs, thus limiting the potential improvement in recommendation quality.

5.2.3 Overall Performance Analysis (RQ2)

Across different LLM backbones (LLaMA-7B and Vicuna-7B), our hallucination-mitigation fine-tuning framework integrated into two LLM-based recommender systems (TALLRec and CoLLM) consistently outperforms traditional fine-tuning strategies. Moreover, under both few-shot and full-shot training settings, LCFT demonstrates stable and strong performance. These results collectively indicate that our proposed hallucination-mitigation fine-tuning framework is highly generalisable and can be broadly integrated into existing LLM-based recommender systems to improve recommendation accuracy and reduce hallucination interference.

Table 3: Two user cases of LCFT method.

User	Historical records	Traditional fine-tune methods	LCFT
User1	User Preference: "GoodFellas (1990)", "Dead Poets Society (1989)", "Die Hard (1988)". User Unpreference: "Mulholland Falls (1996)", "First Wives Club, The (1996)", "Stargate (1994)".	User likes movie "The Good, The Bad and The Ugly (1966)"?: **No.** User dislikes movie "The Good, The Bad and The Ugly (1966)"?: **No.**	User likes movie "The Good, The Bad and The Ugly (1966)"?: **No.** User dislikes movie "The Good, The Bad and The Ugly (1966)"?: **Yes.**
User2	User Preference: "Reality Bites (1994)", "Billy Madison (1995)". User Unpreference: "Young Guns II (1990)", "Son in Law (1993)", "An Unforgettable Summer (1994)".	User likes movie "Willy Wonka and the Chocolate Factory (1971)"?: **No.** User dislikes movie "Willy Wonka and the Chocolate Factory (1971)"?: **No.**	User likes movie "Willy Wonka and the Chocolate Factory (1971)"?: **No.** User dislikes movie "Willy Wonka and the Chocolate Factory (1971)"?: **Yes.**

Additionally, we conducted an ablation study (**RQ3**) and hyperparameter analyses (**RQ4**), with detailed analysis provided in Appendices C and D, respectively.

5.3 Case Studies of LCFT

We present case studies from the MovieLens-100K dataset (a movie watching scenario) in Table 3. These examples demonstrate that traditional fine-tuning methods can yield logically inconsistent or contradictory outputs. For example, in the case of user1, a baseline model simultaneously infers that the user both likes and dislikes the movie "The Good, The Bad and The Ugly (1966)", which is clearly inconsistent with the facts. In contrast, LCFT employs a contrastive objective that maximises the KL divergence between logit distributions of semantically opposite instructions, explicitly encouraging the model to produce consistent and distinguishable outputs.

This approach helps the model learn a more coherent representation of user preferences, thereby reducing the risk of contradiction and factual fabrication. These qualitative examples provide compelling evidence that LCFT enhances coherence and effectively mitigates hallucinations.

6 Conclusion

In this work, we introduced LCFT, a novel fine-tuning framework designed to mitigate hallucination in LLM-based recommender systems. LCFT mitigates hallucination by taking semantically opposite instruction pairs as input and incorporates a KL divergence term into the training objective to enlarge their distributional differences in the logit space. Extensive experiments conducted on two LLM-based recommender systems built on different LLM backbones and across four real-world datasets demonstrate that LCFT effectively reduces hallucinations and improves recommendation performance. In the future, we plan to integrate causal inference methods (Cheng et al., 2024; Huang et al., 2025) with the LCFT framework to further alleviate the impact of hallucinations on recommendation performance.

Acknowledgment

This work was partially supported by the Specific Research Project of Guangxi for Research Bases and Talents (GuiKe AD24010011), the Key Research & Development Program Project of Guangxi (GuiKe AB25069095) and the Australian Research Council (under grant DP230101122).

Limitations

LCFT relies on a contrast between positive and negative instructions. In some cases, such a clear contrast may not exist. Users should be cautious when using LCFT for recommendations in these situations. Maximising the KL divergence between positive and negative instructions carries the risk of overconfidence or unstable outputs for certain inputs. Although this risk does not affect the overall performance of LCFT compared to other SOTA methods, users should be aware of it and interpret LCFT's output confidence with caution. Our current evaluation is restricted to English-language datasets. The generalizability of LCFT to other languages, domains, or underrepresented user groups remains an open question and a valuable direction for future work.

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*, page 1007–1014.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, pages 1877–1901.

Qingfeng Chen, Boquan Wei, Debo Cheng, Jiuyong Li, Lin Liu, and Shichao Zhang. 2024. A novel shadow variable catcher for addressing selection bias in recommendation systems. In *2024 IEEE International Conference on Data Mining (ICDM)*, pages 71–80. IEEE.

Debo Cheng, Jiuyong Li, Lin Liu, Jixue Liu, and Thuc Duy Le. 2024. Data-driven causal effect estimation based on graphical causal modelling: A survey. *ACM Computing Surveys*, 56(5):1–37.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, and 1 others. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Qiang Cui, Shu Wu, Qiang Liu, Wen Zhong, and Liang Wang. 2020. MV-RNN: A multi-view recurrent neural network for sequential recommendation. *IEEE Trans. Knowl. Data Eng.*, 32(2):317–331.

Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. 2023a. Uncovering chatgpt's capabilities in recommender systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*, page 1126–1132.

Sunhao Dai and 1 others. 2023b. Uncovering chatgpt's capabilities in recommender systems. In *RecSys*, pages 1126–1132.

Jianfeng Deng, Qingfeng Chen, Debo Cheng, Jiuyong Li, Lin Liu, and Xiaojing Du. 2024. Mitigating dual latent confounding biases in recommender systems. *arXiv preprint arXiv:2410.12451*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186. Association for Computational Linguistics.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, and 1 others. 2023. Palm-e: An embodied multimodal language model. *arXiv:2303.03378*, abs/2303.03378.

Yue Feng, Shuchang Liu, Zhenghai Xue, Qingpeng Cai, Lantao Hu, Peng Jiang, Kun Gai, and Fei Sun. 2023. A large language model enhanced conversational recommender system. *CoRR*, abs/2308.06212.

F.Maxwell and Joseph A. Konstan. 2015. The movielens datasets: History and context. In *ACM Transactions on Interactive Intelligent Systems (TiiS)*.

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*.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*.

F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648.

Balázs Hidasi and 1 others. 2016. Session-based recommendations with recurrent neural networks. In *ICLR*.

Minjie Hong, Yan Xia, Zehan Wang, Jieming Zhu, Ye Wang, Sihang Cai, Xiaoda Yang, Quanyu Dai, Zhenhua Dong, Zhimeng Zhang, and 1 others. 2025. Eager-llm: Enhancing large language models as recommenders through exogenous behavior-semantic integration. In *Proceedings of the ACM on Web Conference 2025*, pages 2754–2762.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, and 1 others. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019*, volume 97, pages 2790–2799.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Zhirong Huang, Shichao Zhang, Debo Cheng, Jiuyong Li, Lin Liu, Guangquan Lu, and Guixian Zhang. 2025. Learning instrumental variable representation for debiasing in recommender systems. *Neural Networks*, page 107977.

Vitor Jeronymo, Luiz Henrique Bonifacio, Hugo Abonizio, Marzieh Fadaee, Roberto de Alencar Lotufo, Jakub Zavrel, and Rodrigo Frassetto Nogueira. 2023. Inpars-v2: Large language models as efficient dataset generators for information retrieval. *arXiv:2301.01820*, abs/2301.01820.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38.

Matthew Jin, Syed Shahriar, Michele Tufano, Xin Shi, Shuai Lu, Neel Sundaresan, and Alexey Svyatkovskiy. 2023. Inferfix: End-to-end program repair with llms. *arXiv:2303.07263*.

Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*, pages 197–206.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP,7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.

Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. 2025. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. *arXiv preprint arXiv:2507.18224*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL/IJCNLP 2021, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Xinhang Li, Chong Chen, Xiangyu Zhao, Yong Zhang, and Chunxiao Xing. 2023. E4srec: An elegant effective efficient extensible solution of large language models for sequential recommendation. *arXiv preprint arXiv:2312.02443*.

Percy Liang, Rishi Bommasani, Tony Lee, and 1 others. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. Llara: Large language-recommendation assistant. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1785–1795.

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

Yixin Liu, Guibin Zhang, Kun Wang, Shiyuan Li, and Shirui Pan. 2025. Graph-augmented large language model agents: Current progress and future prospects. *arXiv preprint arXiv:2507.21407*.

Yiyu Liu, Qian Liu, Yu Tian, Changping Wang, Yanan Niu, Yang Song, and Chenliang Li. 2021. Concept-aware denoising graph neural network for micro-video recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1099–1108.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, pages 27730–27744.

Scott Sanner, Krisztian Balog, Filip Radlinski, Ben Wedin, and Lucas Dixon. 2023. Large language models are competitive near cold-start recommenders for language-and item-based preferences. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 890–896.

Dhruv Shah, Blazej Osinski, Brian Ichter, and Sergey Levine. 2022. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *CoRL 2022, 14-18 December 2022*, volume 205 of *Proceedings of Machine Learning Research*, pages 492–504. PMLR.

Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, pages 565–573.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv:2302.13971*.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. *arXiv:2303.07678*.

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.

Jason Wei, Yi Tay, Rishi Bommasani, and 1 others. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Ted Xiao, Harris Chan, Pierre Sermanet, Ayzaan Wahid, Anthony Brohan, Karol Hausman, Sergey Levine, and Jonathan Tompson. 2022. Robotic skill acquisition via instruction augmentation with vision-language models. *arXiv:2211.11736*, abs/2211.11736.

Zhengyi Yang and 1 others. 2023. A generic learning framework for sequential recommendation with distribution shifts.

Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *WSDM 2019, February 11-15, 2019*, pages 582–590. ACM.

Guixian Zhang, Guan Yuan, Debo Cheng, Lin Liu, Jiuyong Li, and Shichao Zhang. Mitigating propensity bias of large language models for recommender systems. *ACM Transactions on Information Systems*, pages 1–27.

Jizhi Zhang, Keqin Bao, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Is chatgpt fair for recommendation? evaluating fairness in large language model recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 993–999.

Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. 2025. Collm: Integrating collaborative embeddings into large language models for recommendation. *IEEE Transactions on Knowledge and Data Engineering*.

Zizhuo Zhang and Bang Wang. 2023. Prompt learning for news recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 227–237. ACM.

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

Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1059–1068.

Yaochen Zhu, Liang Wu, Qi Guo, Liangjie Hong, and Jundong Li. 2024. Collaborative large language model for recommender systems. In *Proceedings of the ACM on Web Conference 2024*, pages 3162–3172.

Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, page 22–32. Association for Computing Machinery.

A Datasets Details.

The construction and usage details for each dataset are introduced as follows:

movie. To conduct experiments under the few-shot training setting, we follow the preprocessing approach used in TALLREC (Bao et al., 2023) and adopt the processed MovieLens-100K dataset (F.Maxwell and Konstan, 2015). We select the latest 10,000 user interaction records from the original dataset and split them into training, validation, and test sets in a ratio of 8:1:1. For each sample, the 10 interactions prior to the target item are retained as the user's historical behaviour. Following the TALLREC setting, interactions with ratings higher than 3 are regarded as "liked".

book. We adopt the preprocessed BookCrossing dataset (Ziegler et al., 2005) used in TALLREC. For each user, one interacted item is randomly selected as the prediction target, and 10 others are randomly chosen from the remaining items as historical interactions. The dataset is split into training, validation, and test sets in an 8:1:1 ratio, consistent with TALLREC. In this dataset, which contains ratings from 1 to 10, those above 5 are considered as "liked".

ML-1M. MovieLens-1M (Harper and Konstan, 2015) is a standard dataset widely used in recommendation system research, consisting of user ratings on movies ranging from 1 to 5. We preprocess the data according to the settings in CoLLM (Zhang et al., 2025), treating ratings above 3 as "liked" interactions. The training, validation, and test sets are split based on CoLLM's temporal segmentation: the most recent 20 months of interaction records are used, where the first 10 months serve as the training set, the next 5 months as the validation set, and the remaining 5 months as the test set. ML-1M dataset contains 33,891 training, 10,401 validation, and 7,331 test interactions, involving 839 users and 3,256 items.

Amazon-Book. Amazon-Book (He and McAuley, 2016) is a book subset of the Amazon Product Review dataset, which is widely used in the recommendation field. It contains user ratings on books ranging from 1 to 5. Following the CoLLM setting, interactions with ratings greater than or equal to 4 are considered as "liked". The data split also follows CoLLM: interactions from the year 2017 are used, with the first 11 months allocated to the training set and the last month evenly divided between the validation and test sets. Amazon-Book dataset includes 727,468 training, 25,747 validation, and 25,747 test interactions, covering 22,967 users and 34,154 items.

B Parameter Settings.

The LLama-7B and Vicuna-7B we use contain approximately 7 billion parameters. The training process was conducted on a single NVIDIA RTX 4090 GPU (24GB VRAM) running the Ubuntu 22.04 operating system. Parameter-efficient fine-tuning was performed using LoRA, with a total training time of approximately 16 GPU hours. The learning rate is selected from [1e-2, 1e-3, 1e-4], and the weight decay is set to 1e-3. For LCFT-TALLRec, we follow the design of TALLRec and use the LLaMA-7B. The LoRA module is configured with the following hyperparameters: $r = 8$, $alpha = 16$, $dropout = 0.005$, and $target\_modules$ is set to "[q_proj, v_proj]". For LCFT-CoLMF, we follow the design of CoLLM and adopt the Vicuna-7B. The MLP hidden layer dimension in the CIE module is set to 10 times the input dimension. The LoRA module uses the same configuration as in LCFT-TALLRec. For the parameter λ in Eq.(3), we set it to 0.0001 for movie and ML-1M and 0.00001 for book and Amazon-Book.

C Ablation Study.

To verify the effectiveness of our proposed LCFT fine-tuning framework, we conducted ablation studies under the k-shot training setting. Specifically, we compared the performance of the full LCFT-TALLRec model with its variant (w/o LCFT), where "w/o LCFT" refers to the removal of the logit space constraint imposed by the negative instruction. In this case, the model degenerates into a traditional fine-tuning approach. As shown in the table 4, the experimental results demonstrate that:

(1) The performance of the model without the logit space constraint (w/o LCFT) is inferior to that of the complete LCFT-TALLRec model. This indicates that without this constraint, the large language model struggles to effectively mitigate hallucination during fine-tuning, thereby impairing recommendation performance. This result validates the effectiveness of our proposed fine-tuning framework in reducing hallucination and enhancing the performance of LLM-based recommender systems.

(2) Across two datasets and three different k-shot training settings, LCFT-TALLRec consistently outperforms the "w/o LCFT" variant. This demonstrates that the LCFT framework possesses strong generalisation capabilities and can be applied to real-world scenarios with varying degrees of data scarcity. It effectively alleviates the hallucination issue in LLM-based recommender systems and enhances their recommendation performance even with only a small number of training samples.

Table 4: The ablation study of our LCFT method on two real-world datasets. The best results are in boldface.

dataset	few-shot	method	AUC
movie	16	w/o LCFT	0.6701
		LCFT-TALLRec	**0.7013**
	64	w/o LCFT	0.6713
		LCFT-TALLRec	**0.7157**
	256	w/o LCFT	0.7142
		LCFT-TALLRec	**0.7436**
book	16	w/o LCFT	0.5598
		LCFT-TALLRec	**0.6021**
	64	w/o LCFT	0.5996
		LCFT-TALLRec	**0.6195**
	256	w/o LCFT	0.6401
		LCFT-TALLRec	**0.6531**

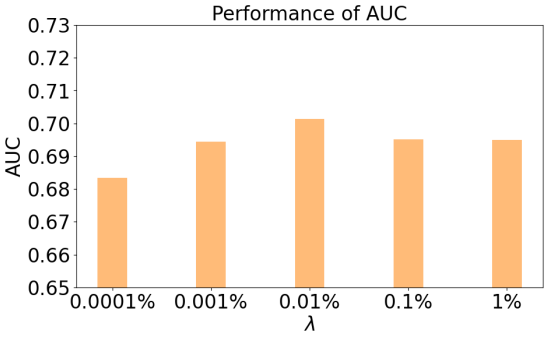

Figure 3: Effect of the λ selection. We show the results of AUC on the movie datasets.

D Hyper-parameter Analysis.

To evaluate the impact of the hyperparameter on the performance of the LCFT framework, we performed a sensitivity analysis on the movie and book datasets under a few-shot setting (with 16 samples). In this experiment, we fixed all other parameters

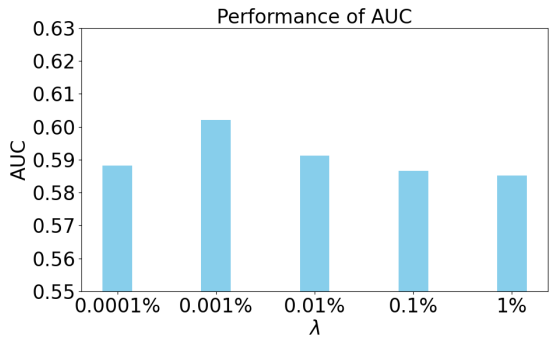

Figure 4: Effect of the λ selection. We show the results of AUC on the book datasets.

and varied the hyperparameter λ in Eq. (3) within the range $[0, 0.01]$ for testing.

As shown in the figure 3 and figure 4, the experimental results indicate the following:

(1) On the movie dataset, as λ increases, the AUC first rises and then falls, reaching its peak at $\lambda = 0.0001$ (0.01%). This suggests that a moderate logit space constraint helps mitigate hallucination issues in LLM-based recommender systems and improves recommendation performance. When λ is too small, the influence of the KL divergence term is weak, and the training process is dominated by the log-likelihood term—i.e., the traditional objective of large language models—potentially failing to leverage negative instruction samples to adjust logit representations effectively. This may lead to insufficient separation in logit space between positive and negative instruction samples. Conversely, when λ is too large, the KL term dominates the optimisation objective, causing the model to overly emphasise the separation of positive and negative instruction sample distributions, at the expense of output accuracy, thereby resulting in performance degradation.

(2) On the book dataset, the AUC also exhibits a similar trend of first increasing and then decreasing, with the best performance achieved at $\lambda = 0.00001$ (0.01%). This may be due to factors such as differences in dataset noise; the optimal hyperparameter value may vary slightly across datasets.

E Societal Impact

(1) Positive impacts: Given the predictive nature of recommendation tasks, it is inherently difficult to directly determine whether a model's response constitutes factual fabrication. To address this, LCFT is the first to propose an interpretable and verifiable definition of hallucination in the recommendation setting: if a model produces the same or logically inconsistent responses to semantically opposite instructions, it can be considered as hallucinating. LCFT effectively mitigates hallucination in LLM-based recommender systems, providing more reliable responses and thereby improving overall model performance.

(2) Negative impacts: As with most machine learning methods, LCFT may be affected by biases present in the training data. We have not yet examined whether LCFT introduces or amplifies bias with respect to sensitive attributes such as gender or race. Investigating the fairness implications of LCFT in comparison with existing SOTA methods is an important direction we leave for future work.

F GenAI Usage Disclosure

We disclose the following use of generative AI tools in the preparation of this paper:

- Writing Assistance: We used OpenAI's ChatGPT (GPT-4) to support language refinement, proofreading, and academic tone improvement in the abstract, introduction, and methodology sections. All generated content was reviewed, edited, and verified by the authors to ensure accuracy and originality.
- Code Development: No generative AI tools were used in the development, debugging, or implementation of the research code.
- Data Processing and Analysis: No generative AI tools were used for data generation, preprocessing, or analysis.

All intellectual contributions, research design, experimental evaluation, and interpretations presented in this paper are the result of the authors' own work.