RecCocktail: A Generalizable and Efficient Framework for LLM-Based Recommendation

Min Hou[1], Chenxi Bai[1], Le Wu[1*], Hao Liu[1], Kai Zhang[2], Weiwen Liu[3], Richang Hong[1], Ruiming Tang[4], Meng Wang[1]

[1]Hefei University of Technology, Hefei, China
[2]University of Science and Technology of China, Hefei, China
[3]Shanghai Jiao Tong University, Shanghai, China
[4]Kuaishou Inc., Beijing, China

hmhoumin@gmail.com, bcx@mail.hfut.edu.cn, lewu.ustc@gmail.com, haoliu0723@gmail.com, kkzhang08@ustc.edu.cn, wwliu@sjtu.edu.cn, hongrc.hfut@gmail.com, tangruiming@kuaishou.com, eric.mengwang@gmail.com

Abstract

Large Language Models (LLMs) have achieved remarkable success in recent years, owing to their impressive generalization capabilities and rich world knowledge. To capitalize on the potential of using LLMs as recommender systems, mainstream approaches typically focus on two paradigms. The first paradigm designs multi-domain or multi-task instruction data for generalizable recommendation, so as to align LLMs with general recommendation areas and deal with cold-start recommendation. The second paradigm focuses on enhancing domain-specific recommendation tasks, improving performance in warm recommendation scenarios. While most previous works treat these two paradigms separately, we argue that they have complementary advantages, and combining them can yield better results. In this paper, we propose a generalizable and efficient LLM-based recommendation framework *RecCocktail*. Our approach begins with fine-tuning a "base spirit" LoRA module using domain-general recommendation instruction data to align LLM with recommendation knowledge. Next, given users' behavior of a specific domain, we construct a domain-specific "ingredient" LoRA module. We then provide an entropy-guided adaptive merging method to mix the "base spirit" and the "ingredient" in the weight space. Please note that, *RecCocktail* combines the advantages of the existing two paradigms without introducing additional time or space overhead during the inference phase. Moreover, *RecCocktail* is efficient with plug and play, as the "base spirit" LoRA is trained only once, and any domain-specific "ingredient" can be efficiently mixed with only domain-specific fine-tuning. Extensive experiments on multiple datasets under both warm and cold-start recommendation scenarios validate the effectiveness and generality of the proposed *RecCocktail*.

Code — https://github.com/1149722739/RecCocktail

1 Introduction

Large Language Models (LLMs) have demonstrated significant success across diverse fields, driven by their emergent capabilities such as world knowledge, language understanding, and complex reasoning. Recently, LLMs have introduced transformative advancements to recommendation tasks (Xi et al. 2024; Deng et al. 2022; Geng et al. 2022; Hou et al. 2025). Along this line, the emergence of ChatGPT and its remarkable reasoning capabilities have catalyzed early studies (Dai et al. 2023; Sanner et al. 2023; Wang et al. 2023). These works focus on the zero-shot/few-shot recommendation potential of LLMs through in-context learning. However, the intrinsic gap between the pre-training general text corpus of LLMs and the requirements of recommendation tasks results in suboptimal performance when relying solely on in-context learning. Consequently, the key to developing an effective LLM-based recommender system lies in bridging this gap, enabling the model to truly "understand" how to recommend.

To address this challenge, researchers have proposed a variety of approaches. We classify them into two paradigms, each tackling the problem from a distinct perspective. As shown in Figure 1(1), the first one is summarized as the **breadth-oriented paradigm**. These works integrate multi-domain (Tang et al. 2024) or multi-task (Geng et al. 2022; Zhang et al. 2024; Cui et al. 2022) recommendation data to construct extensive recommendation world knowledge, paving the way for developing a generalizable LLM-based recommender. The key focus of this paradigm is the integration of multi-source data to build instruction-tuning datasets (Peng et al. 2024; Jin et al. 2023) and the design of instruction templates (Zhang et al. 2024; Geng et al. 2022) tailored to various tasks. The second paradigm is termed the **depth-oriented paradigm**, illustrated in Figure 1(2). This line of research seeks to enable LLMs to deeply comprehend recommendation tasks within specific domains. Key areas of focus include: the in-depth extraction of domain-specific recommendation knowledge, such as collaborative filtering information (Lin et al. 2024a; Kim et al. 2024; Liao et al. 2024; Ren et al. 2024; Kong et al. 2024), and the development of efficient and effective alignment methods between LLMs and recommendation tasks. Specifically, compared with enormous parameters in LLMs, downstream tasks do not have sufficient data for tuning all parameters. Therefore, parameter-efficient fine-tuning methods become optimal for applying LLMs, in which lightweight Low-Rank

*Corresponding Author.

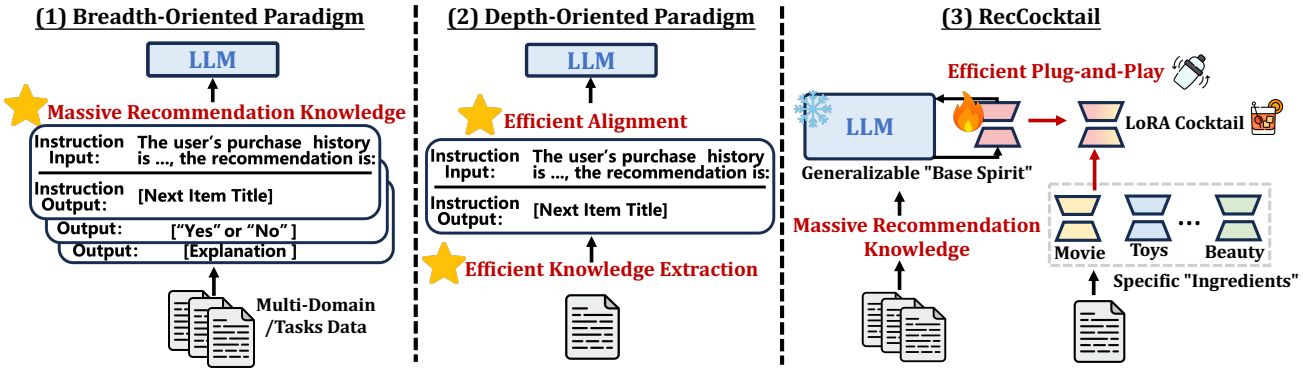


Figure 1: Illustration of different LLM-based recommendation paradigms. (1) Breadth-oriented paradigm. (2) Depth-oriented paradigm. (3) Our proposed *RecCocktail*.

Adapter (LoRA) is one representative work (Hu et al. 2022). By borrowing ideas of LLMs, these methods include leveraging (Bao et al. 2023) or enhancing (Kong et al. 2024) LoRA fine-tuning techniques and designing data-efficient fine-tuning strategies (Lin et al. 2024b).

These works make significant advancements in recommendation research. Nevertheless, we argue that these two paradigms have complementary advantages. Both generalizable recommendation knowledge and efficient domain-specific understanding are essential for recommender systems. Relying solely on one aspect risks falling short in addressing the diverse challenges in real-world recommendation scenarios. The breadth-oriented paradigm may underperform in specific domains. Conversely, the depth-oriented paradigm struggles with distribution shifts between training and test data. It faces challenges when new users or items appear or when training data are sparse.

To this end, we investigate how to integrate the advantages of both paradigms to simultaneously enhance the model's generalization ability and domain-specific performance. The task creates significant obstacles: (1) Efficiency. Integrating two paradigms may introduce model complexity, and finding an efficient integrating method without excessive computational overhead is a critical challenge. (2) Generalizability. We need to preserve the model's generalization ability to a large extent, enabling it to quickly scale to new domains, new items, and other new recommendation scenarios. In this paper, we propose a generalizable and efficient recommendation framework named *RecCocktail*, inspired by the cocktail preparation process, as shown in Figure 1(3). Specifically, to align LLM with any recommendation task, *RecCocktail* constructs a general recommendation instruction dataset from multiple recommendation domains, and fine-tunes LLM to get a domain-general LoRA module as "base spirit". Secondly, to tailor the framework for specific domains, *RecCocktail* constructs domain-specific instruction datasets, and fine-tunes LLM to get a domain-specific LoRA module as "ingredient". After that, *RecCocktail* performs a highly efficient and effective linear arithmetic operation to merge the "base spirit" and the "ingredient" LoRA within the weight space, allowing *RecCocktail* to maintain strong recommendation performance across both specific domains and out-of-distribution scenarios. To further enhance the merging process, we also introduce an adaptive merging method guided by entropy minimization during test time. Importantly, *RecCocktail* does not introduce additional time or space overhead during the inference phase. Furthermore, *RecCocktail* is designed for ease of use, allowing for a plug-and-play integration where the "base spirit" is trained once, and domain-specific "ingredients" are incorporated through minimal fine-tuning. Finally, extensive experiments conducted on various datasets demonstrate the effectiveness and generalizability of the framework in multiple recommendation scenarios, highlighting its potential for broad application.

2 Preliminary

2.1 LLM-Based Recommendation

• **Task Formulation.** Let $\mathcal{U}$ and $\mathcal{I}$ represent the sets of users and items, respectively. The historical interaction sequence of a user $u \in \mathcal{U}$ is denoted as $\mathcal{S}_u = \left[i_u^1, i_u^2, \ldots, i_u^L\right]$, arranged in chronological order, where $i_u \in \mathcal{I}$ and $L = |\mathcal{S}_u|$. The goal is to predict this user's next liked item $i_u^{L+1} \in \mathcal{I}$ based on the historical interactions.

• **Instruction Tuning for LLM-Based Recommendation.** For LLM-based recommendation, instruction tuning (Wei et al. 2022) is the key step to bridge the gap between the next-word prediction objective of LLMs and the recommendation task (Bao et al. 2023; Kong et al. 2024; Lin et al. 2024b). Formally, instruction tuning involves fine-tuning LLMs using training data organized as explicit instruction pairs $\{(\mathbf{x}_u, \mathbf{y}_u)|u \in \mathcal{U}\}$. Here, $\mathbf{x}_u$ represents a detailed textual instruction that encapsulates the interaction sequences $\mathcal{S}_u$ and the recommendation task, while $\mathbf{y}_u$ corresponds to the textual description of the predicted item i_u^{L+1}. The instruction fine-tuning process is guided by minimizing the following autoregressive loss function:

$$\mathcal{L}_\Theta^{LLM} = -\sum_u \sum_{t=1}^{|\mathbf{y}_u|} \log P_\Theta\left(y_u^t \mid \mathbf{y}_u^{<t}, \mathbf{x}_u\right), \quad (1)$$

where y_u^t denotes the t-th token of the output sequence $\mathbf{y}_u$,

$\mathbf{y}_u^{<t}$ is the token sequence preceding y_u^t, and Θ is the LLM's model parameters.

2.2 Instruction Tuning with LoRA

In traditional fine-tuning as described in Eqn. (1), updating all parameters makes the process highly computationally intensive, particularly for LLMs. To address this issue, parameter-efficient methods are designed to fine-tune LLMs while updating only a small subset of parameters. Low-Rank Adaptation (LoRA) (Hu et al. 2022) is the mainstream approach. LoRA addresses this issue by introducing low-rank matrices that are trained alongside the frozen original model weights. This allows the model to adapt to specific tasks by learning a small number of additional parameters, without requiring modifications to the entire model.

Specifically, for any pre-trained weight matrics $\boldsymbol{W}_0 \in \mathbb{R}^{d\times k}$ in the transformer block of the LLM, which takes an input $\boldsymbol{x} \in \mathbb{R}^k$ and output $\boldsymbol{h}$. LoRA modifies $\boldsymbol{h} = \boldsymbol{W}_0\boldsymbol{x}$ to:

$$\boldsymbol{h} = \boldsymbol{W}_0\boldsymbol{x} + \boldsymbol{B}\boldsymbol{A}\boldsymbol{x}, \tag{2}$$

where $\boldsymbol{B} \in \mathbb{R}^{d\times r}$, $\boldsymbol{A} \in \mathbb{R}^{r\times k}$ are the low-rank projection matrices. Notably, the rank $r \ll \min(d, k)$, ensuring that the number of parameters introduced by $\boldsymbol{B}\boldsymbol{A}$ is significantly fewer than those of $\boldsymbol{W}_0$, as $dr + rk \ll dk$. During fine-tuning, only $\boldsymbol{A}$ and $\boldsymbol{B}$ are updated, while $\boldsymbol{W}_0$ remains fixed. In a similar way, LoRA adapter is generally applicable to any LLM layer desired for updating. The training objective of LoRA fine-tuning can be formulated as:

$$\max_{\Delta\Theta} \sum_{u} \sum_{t=1}^{|\mathbf{y}_u|} \log P_{\Theta_{\text{pre}}+\Delta\Theta}\left(\mathbf{y}_u^t \mid \mathbf{y}_u^{<t}, \mathbf{x}_u\right). \tag{3}$$

Here, Θ_{pre} is the parameters of the pre-trained LLM. $\Delta\Theta = \{\boldsymbol{A}^l, \boldsymbol{B}^l\}_{l=1}^L$ denotes the set of parameters of LoRA fine-tuning, and L represents the number of LoRA modules.

3 Methodology

In this section, we propose *RecCocktail*, a generalizable, effective, and efficient LLM-based recommendation framework. As shown in Figure 2, *RecCocktail* operates through three key stages. We start by preparing the base spirit. We align the LLM with any recommendation task and fine-tune a domain-general LoRA module as the base spirit (Section 3.1). Secondly, to adapt the framework to a specific domain, we fine-tune the domain-specific LoRA module (Section 3.2) as an ingredient. Subsequently, *RecCocktail* performs an efficient linear arithmetic operation to merge the base spirit LoRA and the ingredient LoRA into a cocktail within the weight space (Section 3.3). We further provide a test-time entropy-guided adaptive merging method to quickly construct cocktail LoRA tailored to different inference scenarios. The merged cocktail LoRA collaborative enhances performance across both specific domains and out-of-distribution scenarios without introducing additional time or space overhead.

3.1 Constructing Generalizable Base Spirit

In this subsection, we construct a generalizable base spirit equipped with general recommendation knowledge. As illustrated in Figure 2(a), we first create a large-scale instruction dataset by aggregating user behavior data from multiple domains. We then fine-tune a pre-trained LLM on this dataset using the LoRA technique, resulting in a domain-general LoRA module, which serves as our base spirit.

• **Instruction Dataset Construction.** Given N recommendation domains (i.e., $\mathcal{D}^1$, $\mathcal{D}^2$,..., $\mathcal{D}^N$), let $\mathcal{U}^n$, $\mathcal{I}^n$, and $\mathcal{S}^n$ denote the user set, item set, and user interaction sequence set of domain n, respectively. To provide general user modeling and recommendation knowledge, we aggregate data from all N domains and design instruction templates to convert them into text format. Note that both the choice of recommendation domains and the design of instruction templates are flexible. In this paper, we adopt the template shown in Figure 2(a). We transform the multi-domain recommendation data into an instruction dataset $\mathcal{D}_g = \{(\mathbf{x}, \mathbf{y})\}$, where $\mathbf{x}$ and $\mathbf{y}$ represent the instruction input and output, respectively. The input a task description, the user's historical interactions, and a set of candidate items, all expressed in text. Each item is represented by its title. The candidate set includes one ground-truth item and several randomly sampled negative samples. The output is designed to rank the user's next most likely item.

• **Tuning Domain-General LoRA Module.** Given the instruction dataset $\mathcal{D}_g$, we apply LoRA fine-tuning to adapt the pre-trained LLM for general recommendation tasks. The pre-trained model parameters are kept frozen, while trainable low-rank decomposition matrices are introduced into each layer of the Transformer architecture, enabling efficient and lightweight tuning. Formally,

$$\max_{\Delta\Theta_g} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}_g} \sum_{t=1}^{|\mathbf{y}|} \log P_{\Theta_{\text{pre}}+\Delta\Theta_g}\left(\mathbf{y}^t \mid \mathbf{y}^{<t}, \mathbf{x}\right), \tag{4}$$

where Θ_{pre} is the parameters of the pre-trained LLM, and $\Delta\Theta_g$ denotes the set of parameters of LoRA fine-tuning. By undergoing this fine-tuning step, $\Delta\Theta_g$ is now enriched with extensive general recommendation knowledge.

3.2 Constructing Domain-Specific Ingredient

Each recommendation domain exhibits unique user behavior patterns, making the acquisition of domain-specific knowledge essential for delivering accurate recommendations. To address these domain-specific characteristics, we construct an instruction dataset $\mathcal{D}_s$ tailored to the target domain s, and fine-tune the pre-trained LLM using the LoRA technique. The domain-specific LoRA module serves as a ingredient of a cocktail. As shown in Figure 2(b), the instruction template and LoRA fine-tuning procedure follow a similar approach to that described in Section 3.1. Formally, the training objective for the domain-specific LoRA $\Delta\Theta_s$ is defined as:

$$\max_{\Delta\Theta_s} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}_s} \sum_{t=1}^{|\mathbf{y}|} \log P_{\Theta_{\text{pre}}+\Delta\Theta_s}\left(\mathbf{y}^t \mid \mathbf{y}^{<t}, \mathbf{x}\right). \tag{5}$$

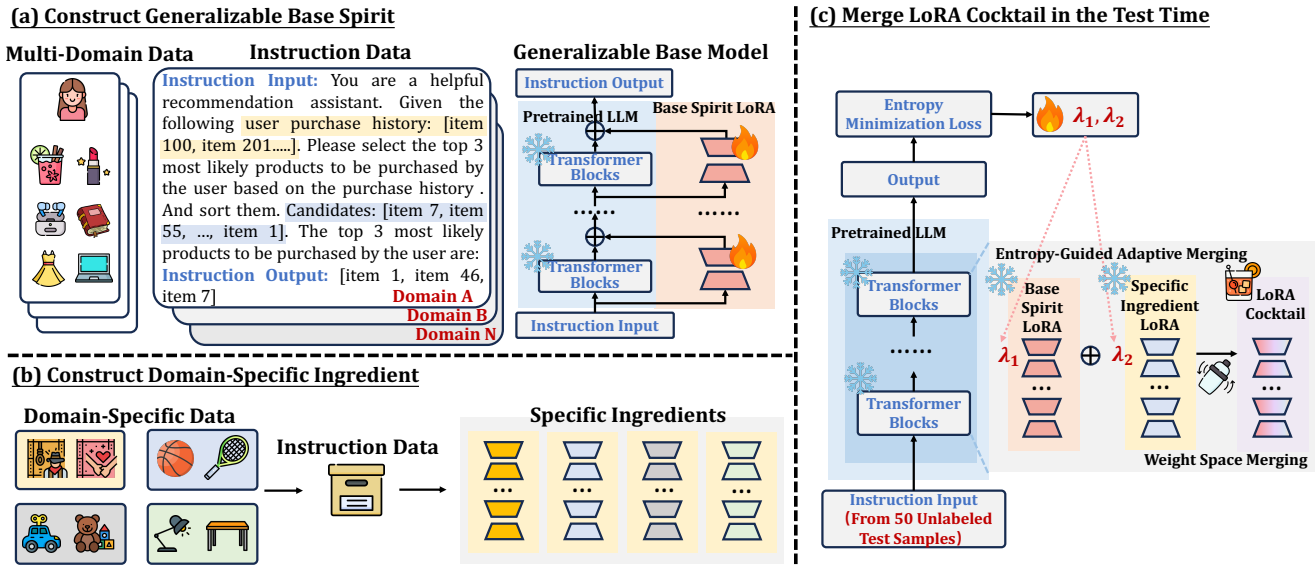


Figure 2: Illustration of our proposed *RecCocktail* framework.

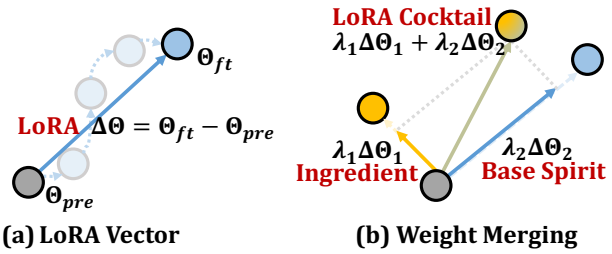


Figure 3: Illustration of task arithmetic (Ilharco et al. 2023). (a) Task vectors are obtained by subtracting pre-trained weights from fine-tuned weights. (b) Adding task vectors improves performance on multiple tasks.

3.3 LoRA Cocktail for Plug-and-Play

After preparing the base spirit and the ingredient, we propose integrating these two parts to improve recommendation accuracy and enhance generalization capabilities simultaneously. Natural questions arise: could this goal be achieved by applying traditional ensemble learning methods that combine the outputs of multiple models? Or could we integrate general and domain-specific knowledge by directly performing second-round fine-tuning on the domain-general LoRA module with the domain-specific dataset? Unfortunately, the answer is no. Since LLMs generate natural language text, ensembling their outputs can introduce semantic inconsistencies or ambiguities, while also increasing inference time and GPU memory usage. Meanwhile, performing a second round of fine-tuning risks catastrophic forgetting (Kirkpatrick et al. 2017), causing the model to collapse.

• **LoRA Cocktail.** We propose a simple yet effective method called LoRA cocktail, which linearly merges the model parameters of the domain-general base spirit LoRA $\Delta\Theta_g$ and the domain-specific ingredient LoRA $\Delta\Theta_s$. We draw an analogy to cocktail making, where mixing a base spirit with an additional ingredient results in a drink that combines the flavors of both parts. Formally, given the base spirit $\Delta\Theta_g = \{\boldsymbol{A}_g^l, \boldsymbol{B}_g^l\}_{l=1}^L$ and a ingredient $\Delta\Theta_s = \{\boldsymbol{A}_s^l, \boldsymbol{B}_s^l\}_{l=1}^L$, we define the LoRA cocktail operator $\oplus$ as:

$$\Delta\Theta_m = (\lambda_1\Delta\Theta_g) \oplus (\lambda_2\Delta\Theta_s) = \{\boldsymbol{A}_m^l, \boldsymbol{B}_m^l\}_{l=1}^L, \quad (6)$$

$$\boldsymbol{A}_m^l = \lambda_1\boldsymbol{A}_g^l + \lambda_2\boldsymbol{A}_s^l, \quad \boldsymbol{B}_m^l = \lambda_1\boldsymbol{B}_g^l + \lambda_2\boldsymbol{B}_s^l, \quad (7)$$

where the coefficients λ_1 and λ_2 represents the importance of merging. We constraint $\lambda_1+\lambda_2 = 1$ and $0 <= \lambda_1, \lambda_2 <= 1$. They can be considered hyperparameters and selected using the validation data. Please note that the mixed $\Delta\Theta_m$ maintains the same total number of parameters as one standard LoRA, making our LoRA Cocktail method simple, fast, and effective. There is no extra cost at inference time in terms of memory or compute, since we only do element-wise operations on model weights. In addition, the domain-general LoRA module is reusable. When facing a new domain, it is only necessary to retrain a domain-specific LoRA.

LoRA cocktail is the weight merging of the domain-general LoRA module and the domain-specific one. Here we explain how it works as shown in Figure 3. We find that after fine-tuning, a LoRA module can be interpreted as a task-specific weight update vector: $\Delta\Theta = \Theta_{\text{ft}} - \Theta_{\text{pre}}$, which defines a direction in the pre-trained model's weight space. Shifting the model's weights along this direction enhances its performance on the corresponding task. Intuitively, a LoRA vector encodes all of the information needed to solve a task that is learned via fine-tuning. Adding two directions allows the model to move in a way that incorporates both general knowledge and domain-specific knowledge. This idea aligns with the concept of "task vectors" proposed in recent work (Ilharco et al. 2023), where weight differences from pre-trained to fine-tuned models are shown

to encode task-specific information. These vectors, residing in the same parameter space, can be combined to equip the model with multiple capabilities. This is further supported by findings that models fine-tuned from the same initialization often lie in the same error basin (Neyshabur, Sedghi, and Zhang 2020; Zhang et al. 2023), making such linear merging feasible and effective. In this light, the LoRA cocktail approach leverages LoRA as task vectors, enabling efficient knowledge integration through simple vector addition.

• **Entropy-Guided Adaptive Merging.** As shown in Figure 2(c), we provide an efficient and automatic way to better choose merging coefficients λ_1 and λ_2. As discussed in the previous subsection, λ_1 and λ_2 can be chosen by employing the grid-search in the validation data. Nevertheless, (1) it is still lacking a guiding principle. (2) When the distribution of the inference data differs significantly from that of the validation set, the chosen coefficients may perform poorly. To this end, we introduce entropy minimization on the unlabeled test samples as an optimization surrogate objective to update λ_1 and λ_2. Specifically, the Shannon Entropy (Shannon 1948) is a well-known measure of uncertainty. For a sample $\mathbf{x}_i$, the predicted output of a neural network $\mathcal{F}_\theta(\mathbf{x}_i)$ is $\hat{\mathbf{y}}_i$, the Shannon entropy is calculated as $H(\hat{\mathbf{y}}_i) = -\sum_c^C p\left(\hat{\mathbf{y}}_{i,c}\right) \log p\left(\hat{\mathbf{y}}_{i,c}\right)$, where $p\left(\hat{\mathbf{y}}_{i,c}\right)$ denotes the probability that the input $\mathbf{x}_i$ is predicted to be the c-th class. Lower entropy indicates that the model has lower uncertainty about its predictions, meaning the model is more confident in its outputs. Therefore, the intuition behind our method is that the good coefficients λ_1 and λ_2 for the test inputs should make the mixed model more confident in its prediction, that is, it should lead to lower model entropy over the input (Wang et al. 2021a,b; Yang et al. 2024). Formally, we collect a set of unlabeled test samples $\mathcal{D}_t$, i.e., some instruction inputs in the test time. We fix the $\Delta\Theta_g$, $\Delta\Theta_s$, Θ_{pre}, and using the following entropy minimization loss to update coefficients λ_1 and λ_2:

$$\min_{\lambda_1,\lambda_2} \sum_{\mathbf{x}_i \in \mathcal{D}_t} H\left(\mathcal{F}_{\Theta_{\text{Cocktail}}}\left(\mathbf{x}_i\right)\right), \tag{8}$$

$$\text{where } \Theta_{\text{Cocktail}} = \Theta_{\text{pre}} + \left(\lambda_1 \Delta\Theta_g\right) \oplus \left(\lambda_2 \Delta\Theta_s\right). \tag{9}$$

For the LLM, the output of $\mathcal{F}_{\Theta_{\text{Cocktail}}}$ is a sentence. Since our instruction is to select a title from a given candidate set, the first few tokens output by the model are more important because after deciding on them, the subsequent tokens are more certain. So in practice, we can only calculate the average entropy of the first few tokens in the sentence to represent $H\left(\mathcal{F}_{\Theta_{\text{Cocktail}}}\right)$. Good performance can be achieved with just 3 tokens. Besides, we do not need all test data to be available. Even with only 50 unlabeled tests data, our method can have significant performance improvements.

3.4 Discussion

• **Key Advantages of *RecCocktail*.** 1) Generalization. *RecCocktail* is generalizable to various recommendation scenarios as it can adaptively determine how to fuse domain-general and domain-specific knowledge based on the distribution of the test data. Even in extreme cases where no training data is available for the new domain, *RecCocktail* can still work using the generalizable base spirit. Additionally, leveraging the in-context learning capabilities of LLMs, *RecCocktail* naturally exhibits task generalization. E.g., it can generate explainable recommendation results. 2) Efficiency. After adjusting the merging coefficients using a few unlabeled test data, we retains only a single LoRA module. As a result, there is no additional memory or computational overhead during inference. *RecCocktail* offers plug-and-play integration, where the domain-general module is trained once, and the domain-specific plugin is incorporated with minimal fine-tuning.

• **Comparison to Existing Methods.** Traditional sequential recommendation models (e.g., GRU4Rec (Hidasi et al. 2016), SASRec (Kang and McAuley 2018)) typically rely on explicit item IDs for modeling, limiting new domains or platforms generalization. Transferable methods address this by unifying multi-domain data in the input space. E.g., VQ-Rec (Hou et al. 2023) and UniSRec (Hou et al. 2022) use text representations with contrastive pre-training for enhanced transferability. Cross-domain sequential recommenders transfer knowledge between specific domain pairs, typically requiring shared users or items. Transferable methods build general-purpose models transferable to any new domain usually via pretraining-finetuning framework without requiring shared users or items. RecCocktail belongs to transferable methods. With LLMs' emergence, breadth-oriented methods leverage their generalization capabilities. P5 (Geng et al. 2022) unifies five recommendation tasks through a text-to-text paradigm. These methods enhance generalization by aggregating data from multiple domains/tasks. In contrast, *RecCocktail* takes a parameter merging approach, integrating knowledge directly in the parameter space—offering greater flexibility and scalability. Depth-oriented methods focus on domain-specific alignment. TallRec (Bao et al. 2023) uses LoRA for efficient adaptation, while LLaRA (Liao et al. 2024), iLoRA (Kong et al. 2024), and AlphaRec (Sheng et al. 2025) align collaborative signals with LLMs. Although this significantly enhances performance in warm-start scenarios, it reduces generalization for new domains and cold-start scenarios.

4 Experiments

4.1 Experimental Settings

• **Datasets.** We conduct experiments on e-commerce and movie recommendation scenarios. For the e-commerce recommendation scenario, the domain-general instruction tuning dataset is conducted using eight e-commerce domains in Amazon[1] (Clothing, Cell, Grocery, Health, Home, Pet, Tools, Videos) and validated on three domain-specific datasets in Amazon (Beauty, Toys, Sports). For the movie recommendation scenario, the domain-general dataset is built using MovieLens-10M[2] and validated on the domain-specific dataset MovieLens-1M.

For all datasets, items are represented using their textual "title" information. To prevent data leakage, we care-

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

[2]https://grouplens.org/datasets/movielens/

Methods		Beauty		Toys		Sports		MovieLens-1M	
		NDCG@1	NDCG@3	NDCG@1	NDCG@3	NDCG@1	NDCG@3	NDCG@1	NDCG@3
Traditional	**BPR-MF**	0.1630	0.2588	0.1276	0.2056	0.1496	0.2338	0.1724	0.4185
	GRU4Rec	0.1672	0.2752	0.1320	0.2243	0.1787	0.2829	0.1724	0.4423
	SASRec	0.2410	0.3284	0.2223	0.3105	0.1957	0.2967	0.2257	0.4708
	FMLP-Rec	0.2988	0.4000	0.2994	0.3990	0.2645	0.3812	0.2410	0.5515
Transferable	**UniSRec**	0.2654	0.4089	0.2612	0.3998	0.2341	0.3721	0.2615	0.5594
	VQ-Rec	0.2714	0.4157	0.2715	0.4119	0.2476	0.3944	0.2805	0.5745
	One Model for All	0.3130	0.4177	0.3233	0.4070	0.2986	0.3765	0.4421	0.5317
LLM-Based	**Qwen2-7B-zeroshot**	0.0300	0.0394	0.0843	0.1062	0.0170	0.0242	0.0814	0.1057
	RecFormer	0.2858	0.3840	0.3001	0.3880	0.2667	0.3885	0.2743	0.5701
	P5	0.1775	0.2482	0.1171	0.1709	0.1860	0.2674	0.2046	0.2947
	TALLRec	0.3347	0.3593	0.3746	0.3993	0.3585	0.3826	0.5392	0.5661
	AlphaRec	0.2489	0.3456	0.2264	0.3193	0.2418	0.3538	0.2318	0.3661
Ours	**RecCocktail-WA**	0.3722	0.3959	0.3722	0.3961	0.3410	0.3613	0.5738	0.5982
	RecCocktail-G	0.3081	0.3316	0.2957	0.3209	0.2750	0.2998	0.5680	0.5918
	RecCocktail-S	0.4079	0.4291	0.4076	0.4314	0.3735	0.3925	0.5460	0.5703
	RecCocktail	**0.4132***	**0.4350***	**0.4097***	**0.4334***	**0.3754***	**0.3944***	**0.5783***	**0.6023***

Table 1: Performance Comparison in Warm Start I.I.D Scenario (Beauty, Toys, Sports, and MovieLens-1M).

Methods		Beauty		Toys		Sports		MovieLens-1M	
		NDCG@1	NDCG@3	NDCG@1	NDCG@3	NDCG@1	NDCG@3	NDCG@1	NDCG@3
Traditional	**BPR-MF**	0.0306	0.0688	0.0333	0.0765	0.0350	0.0739	0.0723	0.1421
	GRU4Rec	0.0562	0.1063	0.0447	0.0926	0.0640	0.0996	0.0798	0.1489
	SASRec	0.0656	0.1368	0.0670	0.1210	0.0547	0.1203	0.0912	0.1891
	FMLP-Rec	0.0587	0.1229	0.0537	0.1117	0.0545	0.1236	0.1145	0.1947
Transferable	**UniSRec**	0.0957	0.1457	0.0814	0.1559	0.0832	0.1408	0.0985	0.1343
	VQ-Rec	0.1189	0.1589	0.0957	0.1603	0.0985	0.1463	0.1025	0.1412
	One Model for All	0.1085	0.1638	0.1120	0.1515	0.0969	0.1406	0.1264	0.1699
LLM-Based	**Qwen2-7B-zeroshot**	0.0187	0.0260	0.0293	0.0356	0.0213	0.0273	0.0318	0.0407
	RecFormer	0.1051	0.1687	0.0913	0.1592	0.0922	0.1489	0.1108	0.1547
	P5	0.0871	0.1466	0.0755	0.1358	0.0758	0.1355	0.0957	0.1319
	TALLRec	0.1480	0.1783	0.1624	0.1831	0.1251	0.1524	0.1458	0.1668
	AlphaRec	0.0588	0.1235	0.0553	0.1223	0.0565	0.1358	0.0923	0.1981
Ours	**RecCocktail-WA**	0.1766	0.2077	0.1589	0.1810	0.1610	0.1903	0.1636	0.2597
	RecCocktail-G	0.1746	0.2072	0.1474	0.1710	0.1581	0.1868	0.1455	0.1890
	RecCocktail-S	0.1603	0.1863	0.1504	0.1764	0.1564	0.1867	0.1636	0.2597
	RecCocktail	**0.1825***	**0.2145***	**0.1614***	**0.1821***	**0.1646***	**0.1921***	**0.1818***	**0.2755***

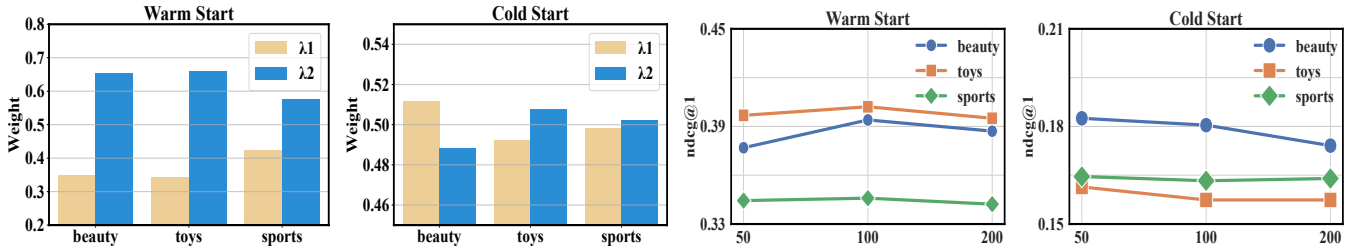
Table 2: Performance Comparison in Cold-Start Item O.O.D Scenario (Beauty, Toys, Sports, and MovieLens-1M).

fully removed the overlapping portions between the domain-general dataset and the domain-specific datasets. We consider two recommendation settings: 1) **Warm-Start Setting** keeps the five-core dataset and filters users and items with fewer than five interactions for all datasets. Following (Geng et al. 2022; Lin et al. 2024a), we adopt the leave-one-out strategy to split the filtered dataset. More concretely, we split the last interaction of each user into the test set, the second-to-last one into the validation set, and the rest into the training data. 2) **New-Item Setting** uses the same training and validation sets as the warm-start setting, but replaces the items in the test set with those that never appear in the training or validation sets as ground-truth.

• **Baselines.** Baselines include traditional recommendation methods (BPR-MF (Rendle et al. 2009), GRU4Rec (Hidasi et al. 2016), SASRec (Kang and McAuley 2018), and FMLP-Rec) (Zhou et al. 2022), transferable sequential recommenders (UniSRec (Hou et al. 2022), VQ-Rec (Hou et al. 2023)) and cross-domain recommender (One Model for All ()), LLM-based recommenders (Qwen2-7B-zeroshot [3], RecFormer (Li et al. 2023), P5 (Geng et al. 2022), TALLRec (Bao et al. 2023), AlphaRec (Sheng et al. 2025)) and our ablation counterparts *RecCocktail*-WA (weight average), *RecCocktail*-G (domain general LoRA only), *RecCocktail*-S (domain-specific LoRA only).

• **Evaluation Setting.** Following LLM-based recommendation works (Zhang et al. 2024; Kim et al. 2024), we add 29 randomly selected non-interacted items to the candidate set, so that for each user it contains 1 positive item and 29 negative items. For quantitative comparison, we employ widely used ranking-based metrics, NDCG@1 and NDCG@3 for all experiments. All metrics are "the higher, the better". For all tables in the following, **bold*** numbers refer to the best performance, while underlined numbers indicate the second-best performance.

[3] https://huggingface.co/Qwen/Qwen2-7B-Instruct



(a) Warm-Start Scenario (b) Cold-Start Scenario (c) Warm-Start Scenario (d) Cold-Start Scenario

Figure 4: (a) and (b) are the coefficients λ_1 and λ_2 calculated by entropy-guided adaptive merging. (c) and (d) are the impact of the number of unlabeled test data.

• **Implementation Details.** To ensure fair comparison, experimental settings are standardized as follows: Traditional methods (BPR-MF, GRU4Rec, SASRec, FMLP-Rec) use learning rate 0.001, Adam optimizer, batch size 256, and embedding dimension 64. Transferable models (UniSRec, VQ-Rec, RecFormer) fine-tune pre-trained models from original authors on our dataset, with UniSRec and VQ-Rec using BERT for text processing. P5 and TALLRec use identical instruction fine-tuning templates: P5 with full fine-tuning, TALLRec with LoRA (rank 16). For *RecCocktail*, we use Qwen2-7B backbone with LoRA (rank 16, alpha 32, dropout 0.05) for both general and specific LoRA on NVIDIA RTX 4090 GPUs. Learning rate is selected from {1e-4, 2e-4} with batch size 128. For adapter fusion on two GPUs, batch size is 60, with test samples grid-searched from {50, 100}. Title token counts are 3 for Toys/Sports and 5 for Beauty. We employ gradient checkpointing and VLLM inference acceleration.

4.2 Overall Performance

We comprehensively compare *RecCocktail* against traditional, transferable, and LLM-based recommenders. Tables 1 and 2 present results for warm-start I.I.D and new-item O.O.D settings, respectively. Results indicate that:

- The proposed *RecCocktail* consistently achieves the best performance across all I.I.D and O.O.D scenarios on the four datasets, with a t-test at $p<0.05$ level. The strong performance of *RecCocktail* demonstrates its ability to efficiently capture domain-specific knowledge while maintaining excellent generalization capabilities.
- Traditional recommendation methods and the ID-based LLM recommendation method P5, AlphaRec perform poorly in cold-start scenarios. Relying heavily on collaborative filtering information reduces the model's generalization capability.
- We observe that *RecCocktail* consistently outperforms its ablation counterparts across all scenarios. This highlights the importance of integrating both general recommendation knowledge and domain-specific insights, which are complementary. The results also confirm the effectiveness of our LoRA cocktail method for knowledge fusion. Note that the *RecCocktail*-G has not been exposed to training data from the Beauty, Toys, Sports, or Movielens-1M domains. It still achieves commendable performance in such zero-shot settings, demonstrating that it has effectively learned generalizable recommendation knowledge. We also find that simple average merging (*RecCocktail*-WA) sometimes trigger negative transfer, demonstrating the effectiveness and importance of our proposed Entropy-Guided Adaptive Merging strategy for selecting appropriate merging coefficients.

Scenario	Sample	TallRec	2nd Finetune	RecCocktail
Warm	**10%**	0.3957	0.0704	0.5353
	20%	0.4298	0.0790	0.5454
	30%	0.4563	0.0540	0.5498
Cold	**10%**	0.1091	0.0000	0.1455
	20%	0.1091	0.0182	0.1636
	30%	0.1273	0.0000	0.1636

Table 3: NDCG@1 performance in the few-shot training setting on Movielens-1M Dataset.

4.3 In-Depth Analysis

• **Analysis of the Coefficients λ_1 and λ_2.** In Figure 4a and 4b, we investigate the coefficients λ_1 and λ_2 calculated by entropy-guided adaptive LoRA cocktail in the warm-start scenario and the cold-start scenario, respectively. λ_1 and λ_2 represent the respective weights assigned to the base spirit and the ingredient LoRA module during their merging. We observe that in the warm-start scenario, λ_2 is relatively large, reflecting a greater reliance on domain-specific LoRA. Conversely, in the cold-start scenario, the weight of λ_1 increases significantly, emphasizing the importance of domain-general knowledge. This result is reasonable and aligns with the differing requirements of these two scenarios. This result also demonstrates the effectiveness of our entropy-guided adaptive LoRA cocktail method.

• **Analysis of the Number of Unlabeled Test Data.** The number of unlabeled test data is one of the hyperparameters. Figure 4c and 4d illustrates the impact of different numbers on the NDCG@1 performance. The entropy-guided learning method converges rapidly, experimental results indicate that setting the number to 50 or 100 achieves a model fusion weight with optimal performance, this configuration proves effective across the majority of experiments

Methods	Beauty		Toys		Sports	
	NDCG@1	NDCG@3	NDCG@1	NDCG@3	NDCG@1	NDCG@3
RecCocktail-G	0.1786	0.2083	0.1554	0.1721	0.1538	0.1814
RecCocktail-S	0.1663	0.1911	0.1534	0.1723	0.1509	0.1809
RecCocktail	**0.1801***	**0.2086***	**0.1683***	**0.1882***	**0.1587***	**0.1864***

Table 4: Ablation Study on Llama-3.1-8b Across Datasets in Cold-Start Item O.O.D Scenario (Beauty, Toys, and Sports).

conducted on the Beauty, Toys, Sports and MovieLens-1M datasets. Therefore, a limited amount of unlabeled test data is sufficient to learn suitable model fusion weights, thereby significantly reducing the data requirements and computation resource costs.

• **Performance in Few-Shot Training Setting.** We further conduct experiments in scenarios with limited domain-specific training data. Specifically, we adopt a few-shot training setup on MovieLens-1M, where only a small percentage of samples are randomly selected from the training set for model training. We compare *RecCocktail* with TallRec and the results of second-round fine-tuning on the generalizable base model. The experimental results are shown in Table 3. We find that the optimization approach of second-round fine-tuning led to catastrophic forgetting. It fails to generate output in the specified instruction format. The experimental results demonstrate that *RecCocktail* maintains strong performance even in few-shot scenarios.

• **Experiments on More LLM Backbones.** To evaluate the robustness of the RecCocktail framework to different LLM backbones, we also utilize Llama-3.1-8B[4] as LLM backbone and observe the results. As shown in Table 4, when switching the LLM backbone to LLaMA-3.1-8B, *RecCocktail* improves over both *RecCocktail*-G and *RecCocktail*-S on all datasets, indicating the effectiveness of adaptive merging.

• **Case Study.** We conduct a case study to examine *RecCocktail*'s recommendation results. We randomly selected a user from the Movielens-1M test set, provided their viewing history and a candidate set, and asked both our model and ChatGPT to make recommendations with explanations (Figure 5). *RecCocktail* successfully generalizes to the explainable recommendation task, accurately capturing the user's preference for action movies from their historical viewing records and leveraging world knowledge to interpret Die Hard. In contrast, GPT-4 lacks domain-specific knowledge, incorrectly associating Die Hard with science fiction films like Star Wars and Terminator. This highlights *RecCocktail*'s task generalization capability and deep understanding of recommendation knowledge.

Instruction Input:

I am a movies fan. I watched [Star Wars: Episode VI - Return of the Jedi (1983)||E.T. the Extra-Terrestrial (1982) || Predator (1987) || Jurassic Park (1993) ||, The Hustler (1961) || Star Wars: Episode IV - A New Hope (1977) || Raiders of the Lost Ark (1981) || Star Wars: Episode V - The Empire Strikes Back (1980)||Jaws (1975)||King Kong (1933) || Goldfinger (1964) || Good, The Bad and The Ugly, The (1966) || Saving Private Ryan (1998) || Run Lola Run (Lola rennt) (1998) || Terminator, The (1984) || Alien (1979) || Thelma & Louise (1991)||Mad Max (1979)||Fistful of Dollars, A (1964) || Rocky (1976)]. Please recommend me only one most likely movies from [Hollywood Knights, The (1980) || Rocky IV (1985) || South Park: Bigger, Longer and Uncut (1999) || Kicking and Screaming (1995) || Pushing Hands (1992) || Enchanted April (1991) || We're No Angels (1989) || Winslow Boy, The (1998) || Spanking the Monkey (1994) || Soul Man (1986) || Girl, Interrupted (1999) || Headless Body in Topless Bar (1995) || Dying Young (1991) || Mafia! (1998) || Champagne (1928) || Young Guns (1988) || Grumpy Old Men (1993) || Boys of St. Vincent, The (1993) || Shaft (1971) || Die Hard (1988)] and give me the reasons.

RecCocktail's Output:

I recommend you to watch Die Hard (1988) because it is a classic action movie that has been highly acclaimed for its thrilling plot, intense action sequences, and memorable characters. The film features a gripping storyline where a police officer has to save a skyscraper from a group of terrorists, showcasing his skills and bravery. It has set a benchmark for action movies and has influenced many films in the genre.

GPT-4's Output:

I recommend you to watch Die Hard (1988) because it features a thrilling blend of action and suspense that aligns well with your taste for iconic films like Star Wars, Alien, and Terminator. The movie's clever storyline, memorable characters, and intense plot twists provide a captivating experience similar to those of the classic adventures you already enjoy. Plus, Die Hard is often regarded as a quintessential action film, making it a must-watch for any movie fan.

Figure 5: Case study of *RecCocktail* and GPT-4.

5 Conclusion

We proposed a generalizable and efficient LLM-based recommendation framework *RecCocktail*. As the recommendation data for a specific domain is limited, *RecCocktail* is designed to combine domain-general and domain-specific recommendation knowledge. Specifically, *RecCocktail* first constructs domain-general LoRA as base spirit and domain-specific LoRA as ingredient. Then the combination is achieved by merging two fine-tuning model parameters. *RecCocktail* is generalizable as it injects recommendation general knowledge to any domain-specific recommendation tasks. Besides, *RecCocktail* is efficient not only because it chooses parameter-efficient fine-tuning, but also the plug-and-play nature of any domain-specific recommendation task and domain-general task. Extensive experimental results on four recommendation datasets under both the warm scenario and cold-start scenario show the effectiveness of our proposed framework.

Acknowledgments

This work was supported in part by grants from the National Key Research and Development Program of China (Grant No. 2021ZD0111802), the National Natural Science Foundation of China (Grant No. 62402159, U23B2031, and 72188101), and the Fundamental Research Funds for the Central Universities (Grant No. JZ2025HGPB0248, PA2025IISL0105). The computation was completed on the HPC Platform of Hefei University of Technology.

[4]https://huggingface.co/meta-llama/Llama-3.1-8B

References

Bao, K.; Zhang, J.; Zhang, Y.; Wang, W.; Feng, F.; and He, X. 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*, 1007–1014.

Cui, Z.; Ma, J.; Zhou, C.; Zhou, J.; and Yang, H. 2022. M6-rec: Generative pretrained language models are open-ended recommender systems. *arXiv preprint arXiv:2205.08084*.

Dai, S.; Shao, N.; Zhao, H.; Yu, W.; Si, Z.; Xu, C.; Sun, Z.; Zhang, X.; and Xu, J. 2023. Uncovering chatgpt's capabilities in recommender systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*, 1126–1132.

Deng, Y.; Li, Y.; Zhang, W.; Ding, B.; and Lam, W. 2022. Toward personalized answer generation in e-commerce via multi-perspective preference modeling. *ACM Transactions on Information Systems (TOIS)*, 40(4): 1–28.

Geng, S.; Liu, S.; Fu, Z.; Ge, Y.; and Zhang, Y. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, 299–315.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.

Hou, M.; Wu, L.; Liao, Y.; Yang, Y.; Zhang, Z.; Zheng, C.; Wu, H.; and Hong, R. 2025. A Survey on Generative Recommendation: Data, Model, and Tasks. arXiv:2510.27157.

Hou, Y.; He, Z.; McAuley, J.; and Zhao, W. X. 2023. Learning Vector-Quantized Item Representation for Transferable Sequential Recommenders. In *Proceedings of the ACM Web Conference 2023*, 1162–1171. New York, NY, USA.

Hou, Y.; Mu, S.; Zhao, W. X.; Li, Y.; Ding, B.; and Wen, J.-R. 2022. Towards universal sequence representation learning for recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 585–593.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Ilharco, G.; Ribeiro, M. T.; Wortsman, M.; Schmidt, L.; Hajishirzi, H.; and Farhadi, A. 2023. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

Jin, W.; Mao, H.; Li, Z.; Jiang, H.; Luo, C.; Wen, H.; Han, H.; Lu, H.; Wang, Z.; Li, R.; Li, Z.; Cheng, M. X.; Goutam, R.; Zhang, H.; Subbian, K.; Wang, S.; Sun, Y.; Tang, J.; Yin, B.; and Tang, X. 2023. Amazon-M2: A Multilingual Multilocale Shopping Session Dataset for Recommendation and Text Generation. In *37th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Kang, W.-C.; and McAuley, J. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, 197–206. IEEE.

Kim, S.; Kang, H.; Choi, S.; Kim, D.; Yang, M.; and Park, C. 2024. Large language models meet collaborative filtering: An efficient all-round llm-based recommender system. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1395–1406.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13): 3521–3526.

Kong, X.; Wu, J.; Zhang, A.; Sheng, L.; Lin, H.; Wang, X.; and He, X. 2024. Customizing Language Models with Instance-wise LoRA for Sequential Recommendation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Li, J.; Wang, M.; Li, J.; Fu, J.; Shen, X.; Shang, J.; and McAuley, J. 2023. Text Is All You Need: Learning Language Representations for Sequential Recommendation. In *KDD*, 1258–1267.

Liao, J.; Li, S.; Yang, Z.; Wu, J.; Yuan, Y.; Wang, X.; and He, X. 2024. LLaRA: Large Language-Recommendation Assistant. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, 1785–1795.

Lin, X.; Wang, W.; Li, Y.; Feng, F.; Ng, S.-K.; and Chua, T.-S. 2024a. Bridging items and language: A transition paradigm for large language model-based recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1816–1826.

Lin, X.; Wang, W.; Li, Y.; Yang, S.; Feng, F.; Wei, Y.; and Chua, T.-S. 2024b. Data-efficient Fine-tuning for LLM-based Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, 365–374.

Neyshabur, B.; Sedghi, H.; and Zhang, C. 2020. What is being transferred in transfer learning? *Advances in neural information processing systems*, 33: 512–523.

Peng, B.; Ling, X.; Chen, Z.; Sun, H.; and Ning, X. 2024. eCeLLM: Generalizing Large Language Models for E-commerce from Large-scale, High-quality Instruction Data. In *41st International Conference on Machine Learning*.

Ren, X.; Wei, W.; Xia, L.; Su, L.; Cheng, S.; Wang, J.; Yin, D.; and Huang, C. 2024. Representation Learning with Large Language Models for Recommendation. In *Proceedings of the ACM Web Conference 2024*, 3464–3475.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, 452–461. Arlington, Virginia, USA.

Sanner, S.; Balog, K.; Radlinski, F.; Wedin, B.; and Dixon, L. 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*, 890–896.

Shannon, C. E. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3): 379–423.

Sheng, L.; Zhang, A.; Zhang, Y.; Chen, Y.; Wang, X.; and Chua, T.-S. 2025. Language Representations Can be What Recommenders Need: Findings and Potentials. In *The 13th International Conference on Learning Representations*.

Tang, Z.; Huan, Z.; Li, Z.; Zhang, X.; Hu, J.; Fu, C.; Zhou, J.; Zou, L.; and Li, C. 2024. One Model for All: Large Language Models are Domain-Agnostic Recommendation Systems. *ACM Trans. Inf. Syst.* Just Accepted.

Wang, D.; Shelhamer, E.; Liu, S.; Olshausen, B.; and Darrell, T. 2021a. Tent: Fully Test-Time Adaptation by Entropy Minimization. In *ICLR*.

Wang, X.; Tang, X.; Zhao, X.; Wang, J.; and Wen, J.-R. 2023. Rethinking the Evaluation for Conversational Recommendation in the Era of Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 10052–10065.

Wang, X.; Tsvetkov, Y.; Ruder, S.; and Neubig, G. 2021b. Efficient Test Time Adapter Ensembling for Low-resource Language Varieties. In *EMNLP: Findings*.

Wei, J.; Bosma, M.; Zhao, V.; Guu, K.; Yu, A. W.; Lester, B.; Du, N.; Dai, A. M.; and Le, Q. V. 2022. Finetuned Language Models are Zero-Shot Learners. In *ICLR*.

Xi, Y.; Liu, W.; Lin, J.; Cai, X.; Zhu, H.; Zhu, J.; Chen, B.; Tang, R.; Zhang, W.; and Yu, Y. 2024. Towards Open-World Recommendation with Knowledge Augmentation from Large Language Models. In *Proceedings of the 18th ACM Conference on Recommender Systems*, 12–22.

Yang, E.; Wang, Z.; Shen, L.; Liu, S.; Guo, G.; Wang, X.; and Tao, D. 2024. AdaMerging: Adaptive Model Merging for Multi-Task Learning. In *The Twelfth International Conference on Learning Representations*.

Zhang, J.; Liu, J.; He, J.; et al. 2023. Composing parameter-efficient modules with arithmetic operation. *Advances in Neural Information Processing Systems*, 36: 12589–12610.

Zhang, J.; Xie, R.; Hou, Y.; Zhao, X.; Lin, L.; and Wen, J.-R. 2024. Recommendation as Instruction Following: A Large Language Model Empowered Recommendation Approach. *ACM Trans. Inf. Syst.* Just Accepted.

Zhou, K.; Yu, H.; Zhao, W. X.; and Wen, J.-R. 2022. Filter-enhanced MLP is All You Need for Sequential Recommendation. In *Proceedings of the ACM Web Conference 2022*, WWW '22, 2388–2399.