CoRA: Collaborative Information Perception by Large Language Model's Weights for Recommendation

Yuting Liu[1*], **Jinghao Zhang**[3*], **Yizhou Dang**[1], **Yuliang Liang**[1], **Qiang Liu**[3†], **Guibing Guo**[1†], **Jianzhe Zhao**[1], **Xingwei Wang**[2]

[1]Software College, Northeastern University, China
[2]School of Computer Science and Engineering, Northeastern University, China
[3]New Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, China
{liuyuting, yizhoudang, liangyuliang}@stumail.neu.edu.cn, jinghao.zhang@cripac.ia.ac.cn, qiang.liu@nlpr.ia.ac.cn,{guogb, zhaojz}@swc.neu.edu.cn, wangxw@mail.neu.edu.cn

Abstract

Involving collaborative information in Large Language Models (LLMs) is a promising technique for adapting LLMs for recommendation. Existing methods achieve this by concatenating collaborative features with text tokens into a unified sequence input and then fine-tuning to align these features with LLM's input space. Although effective, in this work, we identify two limitations when adapting LLMs to recommendation tasks, which hinder the integration of general knowledge and collaborative information, resulting in sub-optimal recommendation performance. (1) Fine-tuning LLM with recommendation data can undermine its inherent world knowledge and fundamental competencies, which are crucial for interpreting and inferring recommendation text. (2) Incorporating collaborative features into textual prompts disrupts the semantics of the original prompts, preventing LLM from generating appropriate outputs. In this paper, we propose a new paradigm, Collaborative LoRA (CoRA), with a collaborative query generator. Rather than input space alignment, this method aligns collaborative information with LLM's parameter space, representing them as incremental weights to update LLM's output. This way, LLM perceives collaborative information without altering its general knowledge and text inference capabilities. Specifically, we employ a collaborative filtering model to extract user and item embeddings and inject them into a set number of learnable queries. We then convert collaborative queries into collaborative weights with low-rank properties and merge the collaborative weights into LLM's weights, enabling LLM to perceive the collaborative signals and generate personalized recommendations without fine-tuning or extra collaborative tokens in prompts. Extensive experiments confirm that CoRA effectively integrates collaborative information into LLM, enhancing recommendation performance.

Code — https://github.com/VanillaCreamer/CoRA

[*]These authors contributed equally.
[†]Guibing Guo and Qiang Liu are corresponding authors.

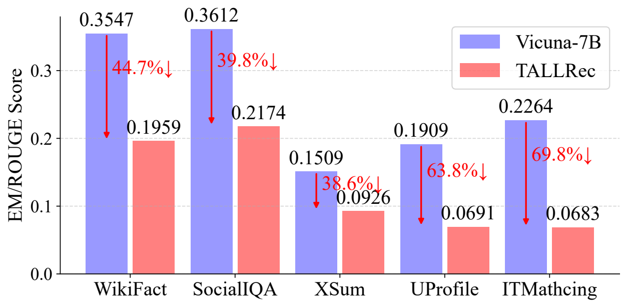
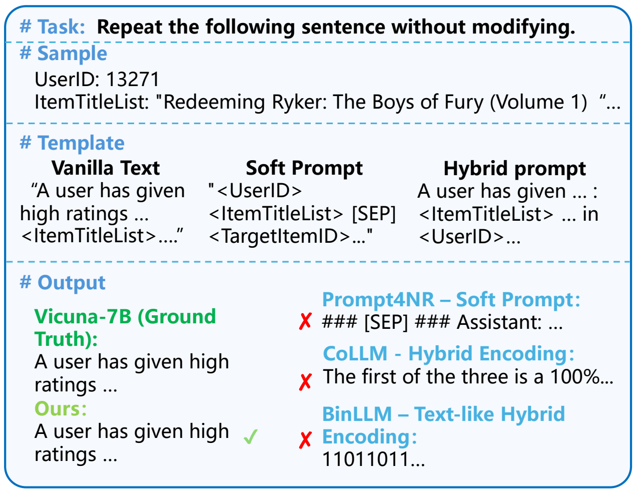


Figure 1: The performance of Vicuna-7B before and after fine-tuning on Amazon-Book using the prompt in TALLRec. The EM/ROUGE-L scores of generated answers on datasets represent various general and recommendation abilities.

Introduction

Large language models (LLMs) have showcased remarkable performance in a wide range of natural language processing tasks and demonstrated excellent generalization capabilities, offering a promising solution to real-world problems. To leverage the extensive competencies of LLMs, an increasing number of studies are investigating ways to frame recommendation problems in natural language, allowing LLMs to tackle these queries (LLMRec) (Li et al. 2023b; Zhang et al. 2024a; Dai et al. 2023a). Collaborative information, which delineates the co-occurrence patterns in user-item interactions (Zhang et al. 2024b), is critical for the success of conventional collaborative filtering methods. Regardless, in LLM-based recommendation approaches, collaborative information cannot be directly interpreted as text. LLMs' ineptitude in perceiving this information hampers their performance compared to traditional recommendation models. Consequently, enabling LLMs to comprehend and utilize collaborative information presents a significant challenge.

Some approaches are beginning to focus on this issue (Li et al. 2023c; Li, Zhang, and Chen 2022). For instance, Prompt4NR (Zhang and Wang 2023) makes the first trial of *prompt learning* paradigm to convert the News Recommen-


# Task: **Repeat the following sentence without modifying.**

# Sample

UserID: 13271

ItemTitleList: "Redeeming Ryker: The Boys of Fury (Volume 1) "...

# Template

Vanilla Text	Soft Prompt	Hybrid prompt
"A user has given high ratings ... <ItemTitleList>...."	"<UserID> <ItemTitleList> [SEP] <TargetItemID>..."	A user has given ... : <ItemTitleList> ... in <UserID>...

# Output

Vicuna-7B (Ground Truth):
A user has given high ratings ...

Ours:
A user has given high ratings ... ✓

Prompt4NR – Soft Prompt:
✗ ### [SEP] ### Assistant: ...

CoLLM - Hybrid Encoding:
✗ The first of the three is a 100%...

BinLLM – Text-like Hybrid Encoding:
✗ 11011011...

Figure 2: Collaborative features interfering with LLM's understanding of textual prompts. We use pre-trained Vicuna-7B as the ground truth. Our method avoids this interference.

dation task as a language prediction task, where user and item data are wrapped into textual prompts. CoLLM (Zhang et al. 2023) captures collaborative information through an external traditional model and maps it to the input space of LLM, forming user and item embeddings as unique tokens for LLM usage. BinLLM (Zhang et al. 2024b) converts collaborative embeddings into binary value sequences that LLMs can operate on directly, facilitating LLMs' direct usage of collaborative information in text-like format. LlaRA (Liao et al. 2023) uses a hybrid prompting method that integrates ID embeddings with textual features.

Despite effectiveness, these techniques fail to preserve the LLM's inherent comprehension and inference for universal and recommendation information when adapting LLMs to recommendation. Our experiments reveal that LLM's capabilities have been weakened from two perspectives.

(1) Researchers have discovered that supervised fine-tuning on instructions they have never seen will encourage them to produce hallucinations (Li et al. 2024). We evaluate LLM's interpreting and inferring capabilities for general and recommendation textual descriptions. For general knowledge, we use Vicuna-7B(Chiang et al. 2023) to generate answers on *WikiFact* (Goodrich et al. 2019), *SocialIQA* (Sap et al. 2019), and *XSum* (Narayan, Cohen, and Lapata 2018) datasets. For recommendation knowledge, we constructed subsets for user profiling (UProfile) and item title matching (ITMatching) based on the *Amazon-Book* (McAuley et al. 2015) to evaluate LLM's comprehension of user and item textual description. As shown in Fig. 1, the LLM's performance on all five datasets significantly decreased after fine-tuning on *Amazon-Book* using the prompt in TALLRec (Bao et al. 2023), indicating that its capacities for utilizing, reasoning, summarizing, profiling, and understanding recommendation texts have been compromised during fine-tuning.

(2) Existing methods endeavor to align collaborative information with the input space of LLMs by embedding collaborative features of users and items into textual prompts. Nevertheless, this practice can disrupt the LLM's comprehension of the original text semantics. As shown in Fig. 2, while we expect pre-trained Vicuna-7B to repeat user and item description, the revised prompts fail to guide the LLM to generate the correct responses. Overall, existing studies limit general knowledge and collaborative information integration when adapting LLM to recommendation tasks.

To address this issue, we propose aligning collaborative features with the LLM's parameter space by transforming them into plug-in weights to update LLM's output. Unlike input space alignment, this parameter space alignment approach enables general LLM to directly leverage collaborative information as an adaption of each user-item pair for recommendation tasks, eliminating the need for fine-tuning or modifying the text prompts.

Specifically, the prediction for each user-item pair is divided into two parts: collaborative weights generation and text inference with collaborative weights. First, we utilize a pre-trained collaborative filtering model to extract user and item embeddings, which are processed by a collaborative query generator. The generator receives collaborative information and injects it into learnable query embeddings. Inspired by VLoRA (Ma et al. 2024), we transform collaborative queries into collaborative perceptual weights similar to "low-rank adaption" (LoRA (Hu et al. 2022)). We incorporate the collaborative weights into the LLM's pre-trained weights to endow it with the capability to perceive collaborative information between the user and the item, allowing it to generate personalized predictions for each user-item pair with general text prompts. Importantly, our technique does not alter the LLM's understanding of the original text, as it updates LLM's output with incremental weights instead of introducing new tokens. For tasks that do not involve collaborative information, our method switches to utilizing the frozen backbone to infer textual prompts. To sum up, our contributions are summarized as follows:

- We explore the issues present when integrating textual prompts and collaborative features in LLM's input space. To resolve these matters, we propose to equip LLMs with collaborative perception capability by merging collaborative weights into LLM's pre-trained weights. It can be seen as **Co**llaborative Lo**RA** (CoRA), facilitating the cooperation of text and collaborative information.
- Following the CoRA method, we propose a collaborative query generator that injects collaborative features extracted from conventional recommenders into learnable query embeddings and then transforms them into LLM weights space with a low-rank property.
- Experimental results demonstrate the effectiveness of our approach, showing significant improvements over state-of-the-art LLMRec methods and traditional collaborative filtering methods on real-world datasets.

Related Work

Collaborative Filtering Models

Collaborative information is essential in the existing recommendation literature(Yuan et al. 2023; Liu et al. 2023b). In traditional personalized recommendation, collaborative filtering (CF) models are prevalent, leveraging collaborative information of users and items to generate predictions (Rendle et al. 2009; Guo, Zhang, and Yorke-Smith 2015; Zhang et al. 2021). In these models, users and items are represented as latent factors fed into neural networks to model their interactions (He et al. 2020; Tang and Wang 2018; He et al. 2017). These studies achieved remarkable success in academia and industry, inspiring further exploration into collaborative information perception for LLMRec.

LLM for Recommendation

Given the impressive capabilities exhibited by LLMs, there is an increasing focus on exploring their potential applications in recommender systems. Techniques in this domain involve translating recommendation tasks into natural language tasks and adapting LLMs to generate recommendation results directly. These generative approaches can be divided into two paradigms based on whether parameters are tuned: non-tuning and tuning paradigms. The non-tuning paradigm assumes LLMs already possess the recommendation abilities and attempts to leverage their strong zero/few-shot abilities by introducing specific prompts (Liu et al. 2023a; Dai et al. 2023a; Mysore, McCallum, and Zamani 2023; Wang et al. 2023; Hou et al. 2023). In contrast, the tuning paradigm uses prompt learning or instruction tuning (Kang et al. 2023; Wang et al. 2022; Cui et al. 2022) to enhance LLM's recommendation capacities.

A new trend is surfacing to include collaborative information in LLMs. Some studies focus on discovering user and item encoding methods to introduce new tokens through vocabulary expansion(Zheng et al. 2023; Hua et al. 2023; Rajput et al. 2023; Zheng et al. 2024). Others explore extracting collaborative information using latent factor models and aligning it with the input space of LLMs (Zhang et al. 2023, 2024b; Liao et al. 2023; Zhang and Wang 2023; Li et al. 2023d). While these methods exhibit strong performance, they constrain specific general capabilities of LLMs. Moreover, due to the hybrid prompting methods, collaborative features can disrupt original textual semantics. Recent works (Zhu et al. 2024; Kong et al. 2024) propose MoE-LoRA frameworks since they suggested that a single LoRA module cannot serve all users. In this work, we translate the collaborative data of users and items into the LLM's weight space, enabling LLMs to perceive personalized information while minimizing complexity.

Preliminaries

In this section, we introduce the problem definition, the basic concepts, and the notations in this paper.

Problem Definition

We represent the interaction dataset as $\mathcal{D} = \{(u,i,y)|u \in \mathcal{U}, i \in \mathcal{I}\}$ where $\mathcal{U}$ and $\mathcal{I}$ denote the set of users and items, respectively, with $y_{ui} = \{0,1\}$ indicating the interaction label. For an item i, there is a unique identifier and textual description t_i. For a user u, we construct the textual description from its historical interactions $t_u = \{t_i | i \in \mathcal{I}_u\}$. In this study, we aim to enable LLMs to leverage both collaborative embeddings e_u, e_i and textual descriptions t_u, t_i to predict whether a user u will enjoy an item i (i.e. y_{ui}).

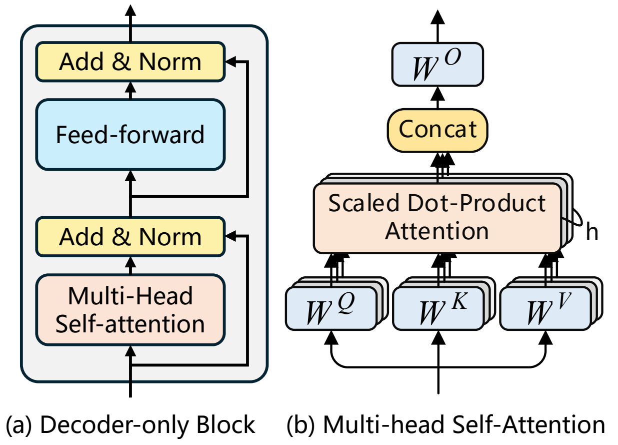

Add & Norm
Feed-forward
Add & Norm
Multi-Head Self-attention
(a) Decoder-only Block
W^O
Concat
Scaled Dot-Product Attention
h
W^Q
W^K
W^V
(b) Multi-head Self-Attention

Figure 3: (a) Architecture of the LLM's Decoder Block. (b) Details of the multi-head self-attention module.

Collaborative Filtering Models

To gather collaborative information, we are looking into utilizing CF methods, which represent users and items by latent factors (a.k.a. embeddings). User and item embeddings are variously calculated to capture collaborating relations precisely. The formulation of encoding a sample $(u,i,y) \in \mathcal{D}$ can be written as follows:

$$\mathbf{e}_u = f_\psi(u;\mathcal{D});\ \mathbf{e}_i = f_\psi(i;\mathcal{D}), \tag{1}$$

where $\mathbf{e}_u, \mathbf{e}_i \in \mathbb{R}^{d_c}$ denote the user's and item's representations with dimension d_c, $f_\psi(\cdot)$ denotes the encoding process, and ψ represents model parameters. The user and item embeddings are then fed into the interaction prediction.

Large Language Model

LLMs refer to a class of language models equipped with billions of parameters. Due to its superiority in performance and training efficiency in generative tasks, LLMs with decoder-only architecture have become mainstream. As depicted in Fig. 3, their decoder block contains a multi-head self-attention module, two add-and-norm modules, and a feed-forward network (Vaswani et al. 2017).

Multi-Head Self-Attention module consists of four types of weights: W^Q, W^K, W^V, and $W^O \in \mathbb{R}^{d_{\text{model}} \times d_{\text{head}}}$, where d_{model} and d_{head} are dimensions of the input embeddings and attention heads, and d_{head}=$d_{\text{model}}/N_{\text{head}}$. For an input sequence with L tokens $X \in \mathbb{R}^{L \times d_{\text{model}}}$, the calculation of the multi-head self-attention layer is as follows:

$$\begin{aligned} \text{MultiHead}(Q,K,V) &= \text{Concat}(\text{head}_1,\ldots,\text{head}_{N_{\text{head}}})W^O \\ \text{where head}_i &= \text{Attention}(Q_i,K_i,V_i) \\ &= \text{softmax}(\frac{XW_i^Q XW_i^{K\top}}{\sqrt{d_{\text{head}}}})XW_i^V. \end{aligned} \tag{2}$$

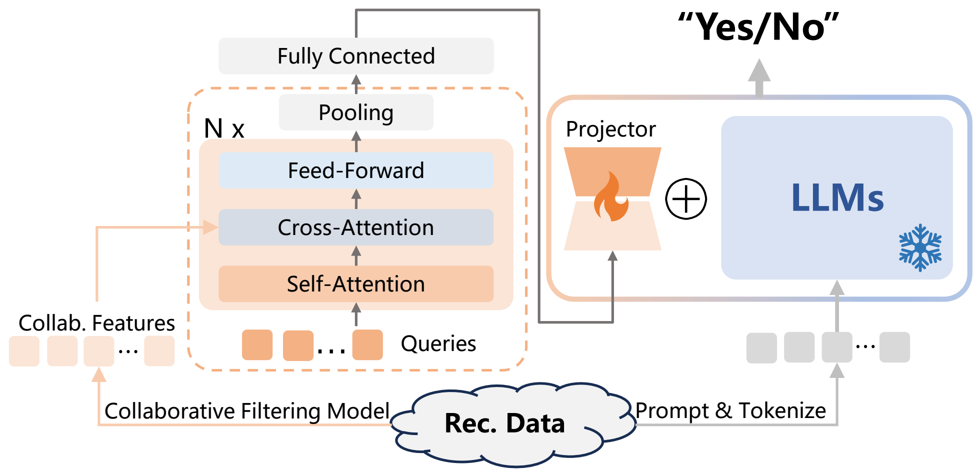


Figure 4: Model architecture overview of our CoRA. The left path extracts user and item embeddings using a CF model and generates collaborative queries. The right path fills the text fields in the prompt template, introducing textual descriptions for inference. Finally, the collaborative queries are projected into the LLM's parameter space and merged into the LLM's weights, enabling the LLM to perceive collaborative information without any fine-tuning or extra tokens in textual prompts.

Feed-forward Network is an MLP with two linear layers and one non-linear activation function as follows:

$$\mathrm{FFN}(X) = \sigma(XW_{\text{up}})W_{\text{down}}, \tag{3}$$

where W_{up} and W_{down} are the weights of linear layers, and σ is an activation function. Overall, there are six types of linear weights in a decoder block: W^Q, W^K, W^V, W^O in the multi-head self-attention module, and W_{up}, W_{down} in the feed-forward module.

Methodology

In this section, we introduce our collaborative weights generator and CoRA method, which effectively integrates collaborative information into LLMs in the parameter space. The overall framework is shown in Fig. 4. First, we explain how to obtain collaborative queries from the pre-trained user and item embeddings through a collaborative query generator. Then, we elaborate on how to equip LLM with collaborative perception capability with the help of generated collaborative queries, followed by a description of the prediction and training strategy.

Generating Collaborative Queries

For a user-item pair (u, i), we first obtain user and item embeddings $\mathbf{e}_u, \mathbf{e}_i \in \mathbb{R}^{d_c}$ from a pre-trained CF model and concatenate them as $[\mathbf{e}_u, \mathbf{e}_i]$. Then, we fed them into a collaborative query generator to obtain user-item-specific collaborative queries, which bridge the collaborative information and the LLM's parameter space. Similar to the Q-Former in BLIP-2 (Li et al. 2023a) and the perceptual weights generator in VLoRA (Ma et al. 2024), we initialize k learnable query embeddings and input them into N decoder blocks with cross-attention modules, to absorb collaborative information from different k semantic sub-spaces.

Specifically, as shown in Fig. 4, k query embeddings first interact with each other in the self-attention module, capturing the underlying relationship between different representation sub-spaces. Then, these k query embeddings perceive and comprehend collaborative information in pre-trained user and item embeddings with the help of the cross-attention module. Finally, the k query embeddings are transformed into k deep collaborative features after passing through the feed-forward network. Afterward, we adopt a pooling operation to aggregate collaborative information from k sub-spaces, obtaining the eventual collaborative information-aware query embedding $\mathbf{q}_c \in \mathbb{R}^{2d_c}$, where d_c is the dimension of pre-trained user/item embeddings. In BLIP-2, the output query embedding is linearly projected into the same dimension as the text embedding of the LLM. However, to avoid the collision between textual prompts and collaborative information, we utilize it to generate an incremental weight to guide LLM in recognizing collaborative information and generating appropriate results.

Collaborative Perception in LLM

Existing LLMRec methods project user and item embeddings into the LLM's input space to integrate collaborative and textual information. These approaches interfere with the LLM's comprehension of the original textual prompts using its general capabilities, resulting in sub-optimal performance. Consequently, we suggest avoiding the incorporation of user/item embeddings and textual prompts. Motivated by VLoRA (Ma et al. 2024), which enables LLMs to perceive visual features by model weights, we address this issue by treating the collaborative information as incremental weights of pre-trained LLMs.

By establishing a bridge between collaborative information and the LLM's parameter space through query embedding $\mathbf{q}_c$, our objective is to generate collaborative weights

#Question: A user has given high ratings to the following movies: ⟨ItemTitleList⟩. Leverage the information to predict whether the user would enjoy the movie titled ⟨TargetItemTitle⟩? Answer with “Yes” or “No”. \n#Answer:

Table 1: Example of the used prompt template, using the same format as TALLRec (Bao et al. 2023).

W_c that can be seamlessly integrated into the pre-trained LLM weights. As we mentioned in the preliminaries, the shape of LLM weights should be $d_{\text{model}} \times d_{\text{model}}$. With transforming $\mathbf{q}_c \in \mathbb{R}^{2d_c}$ into $\Delta W \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ directly, we will introduce a transformation matrix $W \in \mathbb{R}^{2d_c \times d_{\text{model}} \cdot d_{\text{model}}}$, whose parameter cost is unacceptable. Accordingly, a low-rank computation is considered in this process to reduce the computation costs. Inspired by LoRA (Hu et al. 2022), we utilize a Fully Connected layer to map the output query embedding $\mathbf{q}_c$ into LLM's parameter space as $\Delta W_A \in \mathbb{R}^{d_{\text{model}} \times r}$ in LoRA, and ascend the dimension with a final linear projector $W_{\text{proj}} \in \mathbb{R}^{r \times d_{\text{model}}}$, which is equivalent to ΔW_B in LoRA. Eventually, the user-item-specific collaborative weight is calculated by multiplying ΔW_A and ΔW_B, which is formulated as:

$$W_c = \Delta W_A \Delta W_B = \mathrm{R}(\mathbf{q}_c W_{\text{FC}}) W_{\text{proj}}, \tag{4}$$

where $\mathrm{R}(\cdot)$ represent a *reshape* operator, which reshape the product of $\mathbf{q}_c$ and W_{FC} from $\mathbb{R}^{d_{\text{model}} \cdot r}$ into $\mathbb{R}^{d_{\text{model}} \times r}$.

Generally, for an input sample consisting of a user, an item, and their textual descriptions, we extract their collaborative features through a pre-trained CF model and inject the features into a set number of learnable queries with the cross-attention module. Then, we map the collaborative queries into LLM's parameter space and transform them into incremental LLM weights with low-rank properties. Finally, the collaborative weights can be directly merged into pre-trained LLM weights:

$$\hat{W} = W + W_c = W + \mathrm{R}(\mathbf{q}_c W_{\text{FC}}) W_{\text{proj}}, \tag{5}$$

where $W \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ denotes the overall weights of LLM. By absorbing collaborative weights, the LLM inherently perceives collaborative information without altering the LLM's understanding of the original textual prompts.

LLM Prediction and Training Method

Once LLM is equipped with collaborative perceptual capabilities, it can predict recommendations without additional fine-tuning. For an input sample $s = (u, i, t_u, t_i)$, the prediction generation can be formulated as:

$$\begin{aligned} \hat{y} &= \mathrm{LLM}(s) = \mathrm{LLM}_{W+\Delta W}(p) \\ &= \mathrm{LLM}_{W+g_\Theta([f_\psi(u), f_\psi(i)])}(p) \end{aligned} \tag{6}$$

where u, i, and t_u, t_i represent the identifier and textual description of the user and item, respectively. p is the prompt constructed with textual information t_u and t_i as shown in Tab. 1. $g_\Theta(\cdot)$ denotes the collaborative weights generator with parameters Θ.

Dataset	#Train	#Valid	#Test	#User	#Item
ML-1M	33,891	10,401	7,331	839	3,256
Amazon-Book	727,468	25,747	25,747	22,967	34,154

Table 2: Statistics of the processed datasets.

The only module that requires training is the collaborative weights generator, which converts pre-trained collaborative features to collaborative weights. We minimize prediction errors to optimize the parameters of the generator Θ:

$$\hat{\Theta} = argmin_\Theta \sum_{(u,i,y) \in \mathcal{D}} \ell(\hat{y}, y), \tag{7}$$

where $\ell(\cdot)$ denotes the loss function, which is implemented as the binary cross-entropy (BCE) loss.

Experiments

Experimental Settings

Datasets. We adopt two widely-used public datasets for evaluation: ML-1M[1] and Amazon-Book[2]. For dataset processing, we adhere entirely to the setup of CoLLM (Zhang et al. 2023). The detailed statistics of the datasets are summarized in Tab. 2.

Baselines. To assess the effectiveness of CoRA, we compare it with three types of methods: conventional collaborative filtering methods (MF (Rendle et al. 2009), LightGCN (He et al. 2020), and SASRec (Kang and McAuley 2018)), LLM-Rec methods without collaborative information (ICL (Dai et al. 2023b), Prompt4NR (Zhang and Wang 2023), and TALLRec (Bao et al. 2023)), and LLMRec methods that consider collaborative information(PersonPrompt (Li, Zhang, and Chen 2022), CoLLM (Zhang et al. 2023), and BinLLM (Zhang et al. 2024b)).

Implementation Details. We use Vicuna-7B as the backbone LLM to implement our method. Binary Cross-Entropy (BCE) is employed as the objective function for optimizing all methods. AdamW (Loshchilov and Hutter 2017) optimizer is adopted for optimizing LLM, and Adam (Kingma and Ba 2014) optimizer for other methods. The detailed hyper-parameter settings and exploration can be found in the Appendix and the source code. We tune hyper-parameters according to the AUC metric on the validation dataset.

Regarding the collaborative weights generator, we set the hidden dimension and the embedding size of the collaborative model d_c as 256 and the number of blocks N as 8. The rank r of perceptual weights is 16. The number of perceptual queries k is 4. We insert collaborative weights ΔW on every decoder block of LLM. For better collaborative perceptual ability, we explore to insert ΔW for different types of weights in LLM, including $[W_Q, W_K, W_V, W_O, W_{FFN}]$. It is worth noting that the last linear layer W_{proj} of the collaborative weights generator is zero-initialized for training stability as it is equivalent to the ΔW_B of LoRA weights.

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

[2]https://jmcauley.ucsd.edu/data/amazon/index_2014.html

Dataset		Amazon-Book			ML-1M		
Method		AUC	UAUC	Improve	AUC	UAUC	Improve
Collab.	MF	0.7105	0.5543	14.04%	0.6486	0.6396	10.56%
	LightGCN	0.7026	0.5619	13.93%	0.5858	0.6512	15.68%
	SASRec	0.6675	0.5614	17.04%	0.7005	0.6734	3.65%
LLMRec	ICL	0.5180	0.5043	51.61%	0.5119	0.5178	38.37%
	Prompt4NR	0.6527	0.5011	25.10%	0.7027	0.6713	3.28%
	TALLRec	0.6583	0.4971	25.11%	0.7044	0.6741	3.31%
LLMRec w/ Collab.	PersonPrompt	0.7113	0.5596	13.44%	0.7014	0.6503	5.40%
	CoLLM-MF	0.8021	0.5782	5.14%	0.7028	0.6714	3.64%
	CoLLM-LGCN	0.7835	0.5663	7.48%	0.7164	0.6842	4.68%
	CoLLM-SAS	0.7538	0.5874	7.55%	0.7059	0.6531	4.84%
	BinLLM	0.8157	0.5724	4.83%	0.7132	0.6815	2.11%
Ours	CoRA-MF	**0.8179**	**0.6262**	-	**0.7361**	**0.6884**	-
	CoRA-LGCN	0.7886	0.5689	-	0.7128	0.6966	-
	CoRA-SAS	0.7677	0.5961	-	0.7019	0.6517	-

Table 3: Performance comparison of various models on Amazon-Book and ML-1M. "Collab." denotes collaborative recommendation methods. "Improve" denotes the relative improvement of CoRA compared to baselines, averaged over the two metrics. All improvements are statistically significant, as determined by a paired t-test with $p \leq 0.05$.

For our CoRA, we implement it across three collaborative filtering methods to validate the effectiveness of our method in utilizing different types of collaborative information, denoted as CoRA-MF, CoRA-SASRec, and CoRA-LightGCN.

Evaluation Metrics. We employ two widely used metrics for click/rating prediction: AUC (Area under the ROC Curve, which measures the overall prediction accuracy) and UAUC (AUC averaged over users) (Liu et al. 2021) to evaluate the performance of our method and baselines.

Hyperparameter Settings. Regarding hyperparameter tuning, we explore the learning rate within the range of $[1e-2, 1e-3, 1e-4]$ and the weight decay within the range of $[1e-2, \ldots, 1e-6]$. Finally, we train the collaborative weights generator with a learning rate of $1e-2$ on Amazon-Book and $1e-3$ on ML-1M. On two datasets, the warm-up learning rate is set to $1e-5$. The rank r of the collaborative weights is set to 16, and we insert collaborative weights into the weights of four types of LLMs, namely $[W_Q, W_K, W_V, W_O]$ after comparing the performances. For all pre-training collaborative filtering models, we set the embedding size to 256. The learning rate is explored within the range of $[1e-1, \ldots, 1e-5]$, and the weight decay is explored within the range of $[1e-2, \ldots, 1e-6]$.

Besides, we adopt an early-stop strategy if AUC on the validation set no longer increases after 20 epochs to avoid overfitting and achieve the best performance. Our source code included in the Supplementary material provides detailed hyper-parameter settings.

Performance Comparison

Overall Performance. The performance is summarized in Tab. 3, from which we can find that our CoRA outperforms both collaborative filtering and LLMRec baselines, confirming the superiority of CoRA in leveraging collaborative information from traditional CF methods and general knowledge from LLM to make better predictions.

LLMRec with collaborative information surpasses traditional collaborative filtering methods, revealing that LLM's extensive knowledge can be effectively generalized to recommendation systems. Furthermore, LLMRec methods that incorporate collaborative information show superior performance compared to those that do not, highlighting the significance of incorporating collaborative data into LLM for enhanced recommendations. Overall, integrating collaborative information with LLM's general abilities represents a promising advancement in recommendation technology.

Our approach retains its advantages compared to LLMRec with collaborative information methods. When comparing the variants of CoRA and CoLLM based on different collaborative filtering models, CoRA consistently beats CoLLM. This indicates that aligning collaborative information with LLM's parameter space is more effective in helping LLM perceive collaborative information and integrate it with linguistic competence.

Warm and Cold Performance. The main purpose of introducing collaborative information into LLMRec is to enhance the performance of LLMRec methods in warm-start scenarios. We divide the test data into warm and cold subsets based on the number of interactions. Without loss of generality, we compare the testing performance of MF, TALLRec, CoLLM, BinLLM, and CoRA on both warm and cold test subsets, as shown in Fig. 5.

In the warm scenario, TALLRec performs worse than MF because it does not consider collaborative information. Conversely, LLMRec methods that integrate collaborative infor-

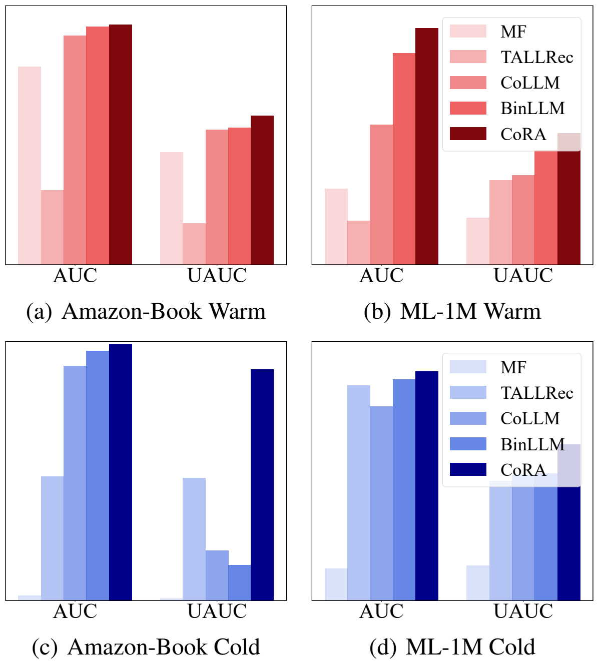


Figure 5: Performance comparison in warm and cold scenarios on Amazon-Book and ML-1M. The left and right y-axis are AUC and UAUC, respectively.

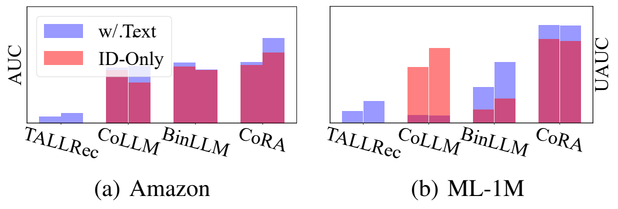


Figure 6: Performance of various variants. "ID-Only" refers to the removal of the item text. "w/ Text" represents adding item textual descriptions.

Weight Type	Amazon-Book		ML-1M	
	AUC	UAUC	AUC	UAUC
qkvof	0.8141	0.6068	0.7312	0.6801
qkvo	**0.8179**	**0.6262**	**0.7361**	**0.6884**
qkv	0.7741	0.5747	0.6947	0.5933
qko	0.8091	0.5949	0.7111	0.5973
qk	0.7685	0.5644	0.6784	0.5887

Table 4: The impact of perceptual weighs type. q, k, v, and o denote query, key, value, and output linear weights in the self-attention module, respectively. f denotes the weights of feed-forward networks.

mation perform better, with CoRA achieving the best. This demonstrates that collaborative information is crucial for recommendations in warm-start scenarios, and our method is more effective in utilizing it.

In the cold start scenario, all LLMRec methods outperform MF, indicating that LLM's universal capabilities can effectively alleviate the cold start problem by utilizing item textual information. Moreover, CoRA enhances the cold-start performance, suggesting that our method is superior to existing methods in integrating the collaborative knowledge of CF models and the general language knowledge of LLM.

Ablation Study

The effectiveness of integrating collaborative and textual information. To further validate the superiority of our method in enabling LLM to incorporate collaborative information, we construct variants of different LLMRec methods only with ID embeddings (i.e., removing textual descriptions of items). We treat TALLRec as a "Text-Only" variant of the three methods as a reference. The results are shown in Fig. 6, from which we can observe that our method outperforms the baselines when using only ID embeddings, confirming that our method can significantly enhance the collaborative perception ability of LLM. On this basis, the performance of all methods improved after involving item textual descriptions in most cases. Notably, the performance of CoLLM on ML-1M significantly declined after introducing textual information, indicating that aligning ID embeddings with LLM's input space interferes with textual semantics and fails to combine their respective advantages. Our CoRA, on the other hand, shows a greater enhancement after incorporating text information, which we attribute to introducing collaborative information in LLM's parameter space instead of the input space, fundamentally avoiding the interference problem.

The type of equipped collaborative weights. As mentioned in **preliminaries**, there are six types of weights in an LLM's decoder block, which are **q**uery, **k**ey, **v**alue, **o**utput, and up&down (**f**eed-forward). We explore the impact of inserting collaborative weights for different types of LLM weights. As shown in Tab. 4, LLM equipped with collaborative weights for all types except feed-forward of weights performs best. Moreover, we observe that the performance of *qkvo* and *qko* is much better than *qkv*, suggesting that the output weights are essential for collaborative perception.

Conclusion

In this paper, we first explore the issues arising from aligning collaborative information in the input space of LLMs. To address them, we introduce CoRA, a novel paradigm that enables LLM to perceive collaborative information without fine-tuning or extra collaborative tokens. Our method converts collaborative information into LLM's incremental weights through a collaborative weights generator, effectively integrating collaborative and textual information. Extensive experiments demonstrate the superiority of CoRA. For future work, we will expand experiments on other LLM backbones and recommendation tasks. Besides, we aim to extend our method to device-cloud collaborative learning.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China under Grant (No. 62032013, 62102074), the Science and technology projects in Liaoning Province (No. 2023JH3/10200005).

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. *Proceedings of the 17th ACM Conference on Recommender Systems*.

Chiang, W.-L.; Li, Z.; Lin, Z.; Sheng, Y.; Wu, Z.; Zhang, H.; Zheng, L.; Zhuang, S.; Zhuang, Y.; Gonzalez, J. E.; Stoica, I.; and Xing, E. P. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality.

Cui, Z.; Ma, J.; Zhou, C.; Zhou, J.; and Yang, H. 2022. M6-Rec: Generative Pretrained Language Models are Open-Ended Recommender Systems. *ArXiv*.

Dai, S.; Shao, N.; Zhao, H.; Yu, W.; Si, Z.; Xu, C.; Sun, Z.; Zhang, X.; and Xu, J. 2023a. Uncovering ChatGPT's Capabilities in Recommender Systems. *Proceedings of the 17th ACM Conference on Recommender Systems*.

Dai, S.; Shao, N.; Zhao, H.; Yu, W.; Si, Z.; Xu, C.; Sun, Z.; Zhang, X.; and Xu, J. 2023b. Uncovering ChatGPT's Capabilities in Recommender Systems. *Proceedings of the 17th ACM Conference on Recommender Systems*.

Goodrich, B.; Rao, V.; Liu, P. J.; and Saleh, M. 2019. Assessing The Factual Accuracy of Generated Text. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Guo, G.; Zhang, J.; and Yorke-Smith, N. 2015. TrustSVD: Collaborative Filtering with Both the Explicit and Implicit Influence of User Trust and of Item Ratings. In *AAAI Conference on Artificial Intelligence*.

He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; and Wang, M. 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*.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural Collaborative Filtering. *Proceedings of the 26th International Conference on World Wide Web*.

Hou, Y.; Zhang, J.; Lin, Z.; Lu, H.; Xie, R.; McAuley, J.; and Zhao, W. X. 2023. Large Language Models are Zero-Shot Rankers for Recommender Systems. *ArXiv*, abs/2305.08845.

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*.

Hua, W.; Xu, S.; Ge, Y.; and Zhang, Y. 2023. How to Index Item IDs for Recommendation Foundation Models. *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*.

Kang, W.; and McAuley, J. 2018. Self-Attentive Sequential Recommendation. *2018 IEEE International Conference on Data Mining (ICDM)*, 197–206.

Kang, W.-C.; Ni, J.; Mehta, N.; Sathiamoorthy, M.; Hong, L.; Chi, E. H.; and Cheng, D. Z. 2023. Do LLMs Understand User Preferences? Evaluating LLMs On User Rating Prediction. *ArXiv*, abs/2305.06474.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

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 *Neural Information Processing Systems*.

Li, J.; Chen, J.; Ren, R.; Cheng, X.; Zhao, W. X.; Nie, J.-Y.; and Wen, J.-R. 2024. The Dawn After the Dark: An Empirical Study on Factuality Hallucination in Large Language Models. *ArXiv*, abs/2401.03205.

Li, J.; Li, D.; Savarese, S.; and Hoi, S. 2023a. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In *International Conference on Machine Learning*.

Li, J.; Wang, M.; Li, J.; Fu, J.; Shen, X.; Shang, J.; and McAuley, J. 2023b. Text Is All You Need: Learning Language Representations for Sequential Recommendation. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Li, L.; Zhang, Y.; and Chen, L. 2022. Personalized Prompt Learning for Explainable Recommendation. *ACM Transactions on Information Systems*, 41: 1 – 26.

Li, X.; Chen, B.; Hou, L.; and Tang, R. 2023c. CTRL: Connect Tabular and Language Model for CTR Prediction. *ArXiv*, abs/2306.02841.

Li, X.; Chen, C.; Zhao, X.; Zhang, Y.; and Xing, C. 2023d. E4SRec: An Elegant Effective Efficient Extensible Solution of Large Language Models for Sequential Recommendation. *ArXiv*, abs/2312.02443.

Liao, J.; Li, S.; Yang, Z.; Wu, J.; Yuan, Y.; Wang, X.; and He, X. 2023. LLaRA: Large Language-Recommendation Assistant. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Liu, J.; Liu, C.; Lv, R.; Zhou, K.; and Zhang, Y. B. 2023a. Is ChatGPT a Good Recommender? A Preliminary Study. *ArXiv*.

Liu, Y.; Liu, Q.; Tian, Y.; Wang, C.; Niu, Y.; Song, Y.; and Li, C. 2021. Concept-Aware Denoising Graph Neural Network for Micro-Video Recommendation. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*.

Liu, Y.; Yang, E.; Dang, Y.; Guo, G.; Liu, Q.; Liang, Y.; Jiang, L.; and Wang, X. 2023b. ID Embedding as Subtle Features of Content and Structure for Multimodal Recommendation. *ArXiv*, abs/2311.05956.

Loshchilov, I.; and Hutter, F. 2017. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

Ma, F.; Xue, H.; Wang, G.; Zhou, Y.; Rao, F.; Yan, S.; Zhang, Y.; Wu, S.; Shou, M. Z.; and Sun, X. 2024. Visual Perception by Large Language Model's Weights. In *Neural Information Processing Systems*.

McAuley, J.; Targett, C.; Shi, Q.; and van den Hengel, A. 2015. Image-Based Recommendations on Styles and Substitutes. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 43–52.

Mysore, S.; McCallum, A.; and Zamani, H. 2023. Large Language Model Augmented Narrative Driven Recommendations. *Proceedings of the 17th ACM Conference on Recommender Systems*.

Narayan, S.; Cohen, S. B.; and Lapata, M. 2018. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. *ArXiv*, abs/1808.08745.

Rajput, S.; Mehta, N.; Singh, A.; Keshavan, R. H.; Vu, T. H.; Heldt, L.; Hong, L.; Tay, Y.; Tran, V. Q.; Samost, J.; Kula, M.; Chi, E. H.; and Sathiamoorthy, M. 2023. Recommender Systems with Generative Retrieval. *ArXiv*, abs/2305.05065.

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*, 452––461.

Sap, M.; Rashkin, H.; Chen, D.; Le Bras, R.; and Choi, Y. 2019. Social IQa: Commonsense Reasoning about Social Interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

Tang, J.; and Wang, K. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*.

Vaswani, A.; Shazeer, N. M.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Neural Information Processing Systems*.

Wang, X.; Tang, X.; Zhao, W. X.; Wang, J.; and rong Wen, J. 2023. Rethinking the Evaluation for Conversational Recommendation in the Era of Large Language Models. *ArXiv*, abs/2305.13112.

Wang, X.; Zhou, K.; rong Wen, J.; and Zhao, W. X. 2022. Towards Unified Conversational Recommender Systems via Knowledge-Enhanced Prompt Learning. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Yuan, Z.; Yuan, F.; Song, Y.; Li, Y.; Fu, J.; Yang, F.; Pan, Y.; and Ni, Y. 2023. Where to Go Next for Recommender Systems? ID- vs. Modality-based Recommender Models Revisited. *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1–11.

Zhang, J.; Liu, Y.; Liu, Q.; Wu, S.; Guo, G.; and Wang, L. 2024a. Stealthy Attack on Large Language Model based Recommendation. *ArXiv*, abs/2402.14836.

Zhang, J.; Zhu, Y.; Liu, Q.; Wu, S.; Wang, S.; and Wang, L. 2021. Mining Latent Structures for Multimedia Recommendation. In *Proceedings of the 29th ACM International Conference on Multimedia*, 3872–3880.

Zhang, Y.; Bao, K.; Yang, M.; Wang, W.; Feng, F.; and He, X. 2024b. Text-like Encoding of Collaborative Information in Large Language Models for Recommendation. *ArXiv*, abs/2406.03210.

Zhang, Y.; Feng, F.; Zhang, J.; Bao, K.; Wang, Q.; and He, X. 2023. CoLLM: Integrating Collaborative Embeddings into Large Language Models for Recommendation. *ArXiv*, abs/2310.19488.

Zhang, Z.; and Wang, B. 2023. Prompt Learning for News Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Zheng, B.; Hou, Y.; Lu, H.; Chen, Y.; Zhao, W. X.; and rong Wen, J. 2023. Adapting Large Language Models by Integrating Collaborative Semantics for Recommendation. *2024 IEEE 40th International Conference on Data Engineering*, 1435–1448.

Zheng, Z.; Chao, W.; Qiu, Z.; Zhu, H.; and Xiong, H. 2024. Harnessing Large Language Models for Text-Rich Sequential Recommendation. *Proceedings of the ACM on Web Conference 2024*.

Zhu, J.; Lin, J.; Dai, X.; Chen, B.; Shan, R.; Zhu, J.; Tang, R.; Yu, Y.; and Zhang, W. 2024. Lifelong Personalized Low-Rank Adaptation of Large Language Models for Recommendation. *ArXiv*.