CoLLM: Integrating Collaborative Embeddings Into Large Language Models for Recommendation

Yang Zhang ⓘ, Fuli Feng ⓘ, Jizhi Zhang, Keqin Bao, Qifan Wang ⓘ, and Xiangnan He ⓘ, *Senior Member, IEEE*

Abstract—Leveraging Large Language Models as recommenders, referred to as LLMRec, is gaining traction and brings novel dynamics for modeling user preferences, particularly for cold-start users. However, existing LLMRec approaches primarily focus on text semantics and overlook the crucial aspect of incorporating collaborative information from user-item interactions, leading to potentially sub-optimal performance in warm-start scenarios. To ensure superior recommendations across both warm and cold scenarios, we introduce *CoLLM*, an innovative LLMRec approach that explicitly integrates collaborative information for recommendations. CoLLM treats collaborative information as a distinct modality, directly encoding it from well-established traditional collaborative models, and then tunes a mapping module to align this collaborative information with the LLM's input text token space for recommendations. By externally integrating traditional models, CoLLM ensures effective collaborative information modeling without modifying the LLM itself, providing the flexibility to adopt diverse collaborative information modeling mechanisms. Extensive experimentation validates that CoLLM adeptly integrates collaborative information into LLMs, resulting in enhanced recommendation performance.

Index Terms—Recommender system, large language model, collaborative information.

I. INTRODUCTION

LARGE Language Models (LLMs) like GPT3 [1], [2] and LLaMA [3] have made rapid advancements, showcasing remarkable capabilities in context comprehension, reasoning, generalization, and modeling world knowledge, among others [4]. These exceptional proficiencies have sparked intense interest and enthusiasm for exploring and utilizing LLMs across diverse fields and disciplines [5], [6], [7], [8]. Recommender systems, as a core engine for personalized information filtering

Yang Zhang, Fuli Feng, Jizhi Zhang, and Keqin Bao are with the University of Science and Technology of China, Hefei 230026, China (e-mail: zyang1580@gmail.com; fulifeng93@gmail.com; cdzhangjizhi@mail.ustc.edu.cn; baokq@mail.ustc.edu.cn).

Qifan Wang is with the Meta AI, New York, NY 10001 USA (e-mail: wqfcr618@gmail.com).

Xiangnan He is with the MoE Key Lab of Brain-inspired Intelligent Perception and Cognition, University of Science and Technology of China, Anhui 230026, China (e-mail: xiangnanhe@gmail.com).

Our implementations are available in Github: https://github.com/zyang1580/CoLLM.
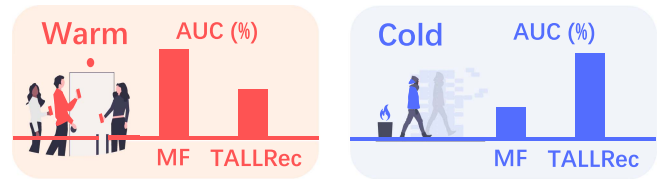
Digital Object Identifier 10.1109/TKDE.2025.3540912

Fig. 1.　A demonstration of LLMRec method (TALLRec [12]) performance compared to conventional methods (MF [14]) in warm and cold scenarios on Amazon-Book [15] data.

on the web, are also anticipated to reap significant benefits from the development of LLMs. For instance, the world knowledge and context comprehension abilities of LLMs could enhance item understanding and user modeling, particularly for cold items/users [9]. This anticipation opens up an exciting new direction: leveraging LLMs as recommenders (LLMRec) [10], [11], which exhibits the potential to become a transformative paradigm for recommendation [10], [12], [13].

To leverage LLMs as recommenders, pioneering studies have relied on In-Context Learning [2], which involves directly asking LLMs to make recommendations by using natural language-based prompts [16], [17], [18], [19]. However, most empirical findings indicate that the original LLMs themselves struggle to provide accurate recommendations, often due to a lack of specific recommendation task training [12], [20], [21]. To address this challenge, increasing efforts have been devoted to further fine-tuning LLMs using relevant recommendation data [12], [19], [20], [21]. Nevertheless, despite incorporating tuning to learn the recommendation task, these methods could still fall short of surpassing well-trained conventional recommender models, particularly for warm users/items, as demonstrated in recent works [22] and Fig. 1.

We argue that the primary limitation of existing LLMRec methods is their inadequate modeling of local collaborative information implied within the co-occurrence patterns in user-item interactions. These methods represent users and items using text tokens, relying predominantly on text semantics for recommendations, which inherently fall short of capturing collaborative information. For example, two items with similar text descriptions might possess distinct collaborative information if consumed by different users, yet this difference often goes unaccounted for due to the textual similarity. Nevertheless, collaborative information between users and items often proves beneficial for recommendations, especially for ones with rich interactions (i.e., warm ones) [23]. Ignoring such information can lead to suboptimal performance. Hence, we introduce a novel research

problem: how can we efficiently integrate collaborative information into LLMs to optimize their performance for both warm and cold users/items?

To solve the issue, we propose explicitly modeling collaborative information in LLMs. Drawing from the experience of classic collaborative filtering with latent factor models (e.g., Matrix Factorization) [14], [24], [25], a straightforward solution is introducing additional tokens and corresponding embeddings in LLMs to represent users/items, akin to the roles played by user/item embeddings in latent factor models. This enables the possibility of encoding collaborative information when using these embeddings to fit interaction data, similar to MF. However, directly adding token embeddings would decrease scalability for large-scale recommendations and increase LLMs' tokenization redundancy, resulting in a lower overall information compression rate [26]. This reduced compression rate is particularly significant considering that collaborative information is typically low-rank, and it could ultimately make prediction tasks (including recommendation) more challenging for LLMs [26], [27]. Moreover, this method lacks the flexibility to incorporate more advanced modeling mechanisms, such as explicitly capturing high-order collaborative relationships like LightGCN [28].

In this light, to effectively enhance LLM with collaborative information in a lightweight and flexible manner, we propose *CoLLM*, a new method that treats collaborative information as a separate modality and introduces it into LLM by directly mapping it from a (well-trained) conventional collaborative model using a Multilayer Perceptron (MLP). CoLLM employs a two-step tuning procedure: first, fine-tuning LLM in the LoRA manner using language information exclusively to learn the recommendation task, and then specially tuning the mapping module to make the mapped collaborative information understandable and usable for LLM's recommendation via considering the information when fitting recommendation data. By aligning knowledge from the conventional model with LLMs, CoLLM effectively incorporates collaborative information into LLMs. This approach maintains scalability comparable to the original LLMs while also providing flexibility in implementing various collaborative information modeling mechanisms by adapting the choice of conventional models.

The main contributions are summarized as follows:

- We highlight the significance of incorporating collaborative information modeling into LLMs for recommendation, so as to make LLMRec perform well in both warm and cold users/items.
- We introduce CoLLM, a novel method that effectively integrates collaborative information into LLMs by harnessing the capability of external traditional models to capture the information.
- We conduct extensive experiments on two real-world datasets. Extensive results demonstrate the effectiveness of our proposal.

II. RELATED WORK

In this section, we first discuss the related work on LLM-Rec. Subsequently, given our focus on integrating collaborative

information into LLMs as an additional modality, we would briefly discuss the related work on multimodal LLM.

A. LLMRec

Recently, with the remarkable emergence of LLMs, there has been a gradual exploration of integrating these sophisticated models with recommender systems [10], [11], [29], [30]. Some researchers employ the methodology of In-context Learning, relying on the natural language comprehension and generation capabilities of LLMs for recommendation purposes [18]. Among them, Chat-Rec [18] facilitates the recommendation process by harnessing the conversational capabilities of Chat-GPT. Besides, researchers are also endeavoring to facilitate the acquisition of recommendation capabilities in LLMs through in-context learning approaches [17], [31]. However, because the objective of LLM pre-training is not geared towards recommendation, these methods often exhibit suboptimal performance. To alleviate this issue, some researchers have employed instruction tuning using empirical recommendation data to enhance the recommendation capabilities of the LLM [12], [21], and achieved commendable performance. Additionally, [32], [33] either employed fine-tuning techniques or utilized prompting on the LLM to enable it to acquire proficiency in using diverse tools for facilitating conversational recommendations.

Although the above works demonstrate the feasibility of tuning LLMs using recommendation data, they still fall short in certain settings compared to traditional models [22]. This can be attributed to the fact that LLMs often heavily rely on semantic priors and tend to overlook collaborative information [34]. To our knowledge, BIGRec [20] is the only work that addresses this issue beyond ours. However, BIGRec tackles the problem by ensembling LLMs with collaborative models, rather than integrating collaborative information into the LLM generation process, which limits the full potential of LLMs. Besides, two concurrent works [35], [36] explore the use of collaborative embeddings; however, they focus on directly learning ID embeddings in the LLM space. In contrast, we focus on mapping collaborative embeddings into the LLM space. When extending to the field of language models (LMs) for recommendation, some works have concentrated on combining LMs with collaborative models [23], [37]. Typically, they use the LM's semantic information as a feature for the collaborative model. However, these approaches may face issues related to forgetting semantic information. In contrast, our approach centers around LLM and still relies on the LLM itself to seamlessly integrate semantic and collaborative information, rather than the reverse.

B. Multimodal LLM

Among the progress in the field of LLM, the endeavors that resonate most closely with our work involve the exploration of multimodal LLMs [38], [39], [40], [41]. For instance, MiniGPT4 [39] combines a frozen visual encoder with a frozen advanced language model, revealing that aligning visual features with large language models enables advanced multi-modal capabilities like generating detailed image descriptions and creating

websites from hand-drawn drafts. Palm-E [41] aims to integrate real-world continuous sensor inputs into language models, creating a connection between words and sensory information. This integration enables the development of embodied language models capable of addressing robotic challenges. These works aim to leverage the robust generation and comprehension capabilities of LLM to process textual data and map information from other modalities such as vision and audio to the textual modality, thereby achieving a large multimodal model with language as its primary carrier, which is similar to our motivations.

III. PRELIMINARIES

To begin with, we briefly present the problem definition, the basic concepts of LLMs, and the collaborative models used in our framework.

Problem Definition: Let $\mathcal{D}$ denote the historical interaction dataset. Each data point within $\mathcal{D}$ is represented as (u, i, y), where u and i correspond to a user and an item, respectively, with $y \in \{1, 0\}$ indicating the interaction label. Furthermore, there is additional textual information available for items, primarily in the form of item titles. In this study, we explore the utilization of both the interaction data and textual information to fine-tune an LLM for recommendation purposes. Our goal is to enable the LLM to effectively leverage collaborative information beyond text information, achieving superior performance in both warm and cold recommendation scenarios.

Large Language Model: LLMs refer to a class of language models equipped with at least several billion parameters and trained on massive text datasets, showcasing remarkable emergent capabilities [4]. LLMs demonstrate strong proficiency in general natural language understanding and generation, as well as various other aspects such as world knowledge modeling, enabling them to excel at handling a wide range of complex tasks as long as they can be described in language. Typically, LLMs process input text through the following two key steps: 1) tokenization and embedding lookup: in this step, the input text is transformed into meaningful lexical tokens, which are then embedded into a vector space; 2) contextual modeling and output generation (LLM Prediction): LLMs utilize their neural networks, primarily based on a decoder-only transformer architecture, to process the token embeddings obtained in the previous step, generating coherent and contextually relevant output. In this work, we use Vicuna-7B [42] for recommendation.

Conventional Collaborative Recommender: We mainly consider the latent factor models, such as MF and LightGCN, for encoding collaborative information. These approaches typically represent users/items using latent factors, also known as embeddings. Subsequently, they form latent user/item representations through various operations, e.g., neighborhood aggregation in LightGCN, to better model collaborative information. Formally, for each sample $(u, i, y) \in \mathcal{D}$,

$$\boldsymbol{u} = f_\psi(u; \mathcal{D}); \quad \boldsymbol{i} = f_\psi(i; \mathcal{D}), \tag{1}$$

where $\boldsymbol{u} \in \mathcal{R}^{1 \times d_1}$ denotes the user's representation with dimension d_1, $f_\psi(u; \mathcal{D})$ denotes the process used to obtain this representation, similarly for i, and ψ denotes model parameters.

The user and item representations are then fed into an interaction module to generate predictions. By minimizing the prediction errors against the actual interaction labels, the latent representations would learn to encode collaborative information within the interaction data.

IV. METHODOLOGY

Collaborative information can be viewed as a distinct information modality, capturing user and item co-occurrence relationships in the interaction data. The LLM itself lacks a dedicated mechanism to extract this modality beyond text modality. To overcome the limitations, rather than modifying the LLMs directly, we continue to extract collaborative information using conventional models and then transform the extracted results into a format that the LLM can comprehend and utilize for recommendations, inspired by recent advancements in multimodal LLMs [5], [39]. This concept forms the foundation of our CoLLM methods. In the following, we provide a detailed overview of CoLLM, beginning with a description of the model architecture designed to connect the conventional models and LLM. We then outline the training strategy that enables the effective integration of collaborative information into LLMs.

A. Model Architecture

Fig. 2 illustrates the model architecture of CoLLM, which consists of three components: prompt construction, hybrid encoding, and LLM prediction. Similar to existing approaches, CoLLM starts by converting recommendation data into language prompts (prompt construction), which are then encoded and inputted into an LLM to generate recommendations (hybrid encoding and LLM prediction). Differently, our approach introduces an innovative aspect by incorporating collaborative information to enhance the LLMs' recommendations. This is achieved through the specific designs in the two former components:

- When constructing prompts, we add user/item ID fields in addition to text descriptions to represent collaborative information.
- When encoding prompts, alongside the LLMs' tokenization and embedding look-up for encoding textual information, we employ a conventional collaborative model to generate user/item representations that capture collaborative information, and map them into the token embedding space of the LLM.

After representing textual and collaborative information within the token embedding space, the LLM could leverage both types of information to perform recommendations. Next, we delve into the specific details of each component.

1) Prompt Construction: We utilize fixed prompt templates for prompt generation. Similar to TALLRec [12], we describe items using their titles and describe users by the item titles from their historical interactions. Uniquely, in order to incorporate collaborative information, we introduce additional user and item ID-related fields that do not carry meaningful semantics but serve as placeholders for the collaborative information within
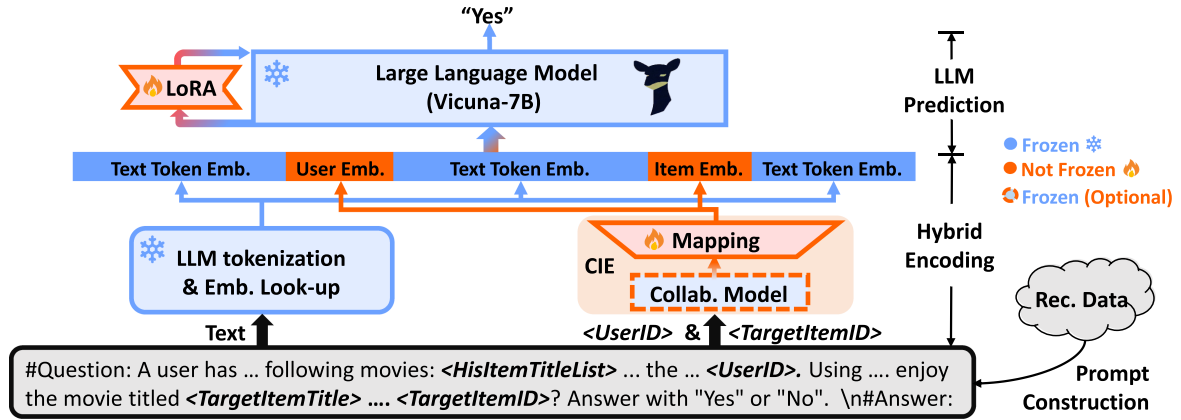
Fig. 2. Model architecture overview of CoLLM, comprising three key components: prompt construction, hybrid encoding, and LLM prediction. "Collab." is short for "collaborative", "Rec." for "recommendation", and "Emb." for "embedding".

the prompt. Ultimately, our fixed prompt template is structured as follows:

> • **Prompt template.** #Question: A user has given high ratings to the following items: $\langle HisItemTitleList\rangle$. Additionally, we have information about the user's preferen ces encoded in the feature $\langle UserID\rangle$. Using all available information, make a prediction about whether the user would enjoy the item titled $\langle TargetItemTitle\rangle$ with the feature $\langle TargetItemID\rangle$? Answer with "Yes" or "No". #Answer:

In this template, "$\langle HisItemTitleList\rangle$" represents a list of item titles that a user has interacted with, ordered by their interaction timestamps, serving as textual descriptions of the user's preferences. "$\langle TargetItemTitle\rangle$" refers to the title of the target item to be predicted. The "$\langle UserID\rangle$" and "$\langle TargetItemID\rangle$" fields are utilized to incorporate user and item IDs, respectively, for injecting collaborative information. To maintain semantic coherence while integrating user/item IDs, we treat them as a type of feature for users/items within the prompt, as indicated by the underlined content in the template. For each recommendation sample, we populate the four fields with the corresponding values of the sample to construct the sample-specific prompt.

2) Hybrid Encoding: The hybrid encoding component is utilized to convert the input prompt into latent vectors, i.e., embeddings suitable for LLM processing. In general, we employ a hybrid encoding approach. As shown in Fig. 2, for all textual content, we make use of the LLM's built-in tokenization and embedding mechanism to convert it into tokens and subsequent token embeddings. In contrast, when dealing with the "$\langle UserID\rangle$" and "$\langle TargetItemID\rangle$" fields, we leverage a Collaborative Information Encoding (CIE) module built with a conventional collaborative recommender, aiming at extracting collaborative information for the LLM to utilize.

Formally, for a prompt corresponding to the sample $(u, i, y) \in \mathcal{D}$, we initiate the process by using the LLM Tokenizer to tokenize its textual content. The tokenization result is denoted as

$P = [t_1, t_2, \ldots, t_k, u, t_{t+1}, \ldots, i, \ldots, t_K]$, where t_k represents a text token, and u/i signifies the user/item (ID) placed within the "$\langle UserID\rangle$"/ "$\langle TargetItemID\rangle$" field. We then further encode the prompt into a sequence of embeddings E:

$$E = \left[\boldsymbol{e}_{t_1}, \ldots, \boldsymbol{e}_{t_k}, \boldsymbol{e}_u, \boldsymbol{e}_{t_{k+1}}, \ldots, \boldsymbol{e}_i, \ldots, \boldsymbol{e}_{t_K} \right], \quad (2)$$

where $\boldsymbol{e}_{t_k} \in \mathcal{R}^{1 \times d_2}$ denotes the token embedding for t_k in the LLM with dimension d_2, obtained via embedding lookup, i.e., $\boldsymbol{e}_{t_k} = Embedding_{LLM}(t_k)$; while $\boldsymbol{e}_u / \boldsymbol{e}_i \in \mathcal{R}^{1 \times d_2}$ denotes the collaborative information embeddings (i.e., collaborative embeddings) for the user u/item i, obtained via the following CIE module.

CIE module: The CIE module consists of a conventional collaborative model ($f_\psi(\cdot)$ in (1)) and a mapping layer $g_\phi(\cdot)$ parameterized by ϕ, to extract collaborative information for LLM usage. When provided with the user u and item i, the conventional collaborative model generates user and item ID representations ($\boldsymbol{u}$ and $\boldsymbol{i}$) encoding collaborative information. Subsequently, the mapping layer maps these representations to the LLM token embedding space, creating the final latent collaborative embeddings ($\boldsymbol{e}_u$ and $\boldsymbol{e}_i$) used by the LLM. Formally, we have:

$$\begin{aligned} \boldsymbol{e}_u &= g_\phi(\boldsymbol{u}), \quad \boldsymbol{u} = f_\psi(u; \mathcal{D}), \\ \boldsymbol{e}_i &= g_\phi(\boldsymbol{i}), \quad \boldsymbol{i} = f_\psi(i; \mathcal{D}), \end{aligned} \quad (3)$$

where $\boldsymbol{u} = f_\psi(u; \mathcal{D}) \in \mathcal{R}^{1 \times d_1}$ denotes the user representation obtained by f_ψ following (1) for $\mathcal{D}$, similarly for $\boldsymbol{i}$. The collaborative model can be implemented as any conventional collaborative recommender as described in Section III. As for the mapping layer, we implement it as a Multilayer Perceptron (MLP), maintaining an input size equal to the dimension d_1 of $\boldsymbol{u}/\boldsymbol{i}$ and an output size equal to the LLM embedding size d_2 (usually $d_1 < d_2$).

3) LLM Prediction: Once the inputted prompt has been converted into a sequence of embeddings E (in (2)), the LLM can utilize it to generate predictions. However, due to the absence of specific recommendation training in LLM, instead of relying solely on the LLM, we introduce an additional LoRA module [43] to perform recommendation predictions, as depicted

in Fig. 2. The LoRA module entails adding pairs of rank-decomposition weight matrices to the original weights of the LLM in a plug-in manner for specifically learning new tasks (recommendation) while just introducing a few parameters. Then, the prediction can be formulated as follows:

$$\hat{y} = h_{\hat{\Theta}+\Theta'}(E),$$

where $\hat{\Theta}$ denotes the fixed model parameters of the pre-trained LLM $h(\cdot)$, and Θ' denotes the learnable LoRA parameters for the recommendation task. $\hat{y}$ represents the prediction probability for the label being 1, i.e., the likelihood of answering "Yes" for LLM. The consideration of using LoRA here is that with the plug-in approach, we only need to update the LoRA weights to learn the recommendation task, enabling parameter-efficient learning.

B. Tuning Method

We now consider how to train the model parameters. To expedite the tuning process, we freeze the LLM, including its embedding layer, and focus on tuning the plug-in LoRA and CIE module. Functionality speaking, the CIE module is responsible for extracting collaborative information and making it usable for LLM in recommendation, while the LoRA module assists the LLM in learning the recommendation task. To tune them, one intuitive approach is to directly train them simultaneously. However, because of the significant reliance on collaborative information, training them both from scratch concurrently may negatively impact LLM recommendations in cold scenarios. To address this, we propose a two-step tuning approach, tuning each component separately as follows:

• *Step 1. Tuning the LoRA Module:* To endow the cold-start recommendation capabilities of LLM, our initial focus is on fine-tuning the LoRA module to learn recommendation tasks independently of collaborative information. During this step, we exclude the collaborative information-related portions of the prompt, which are denoted by the content with underlines in the prompt template. Instead, we solely utilize the remaining text-only segment of the prompt to generate predictions and minimize prediction errors for tuning the LoRA module to learning recommendation. Formally, this can be expressed as:

$$\hat{\Theta}' = \operatorname{argmin}_{\Theta'} \sum_{(u,i,y)\in\mathcal{D}} \ell\left(h_{\hat{\Theta}+\Theta'}(E_t), y\right), \quad (4)$$

where E_t represents the sequence of embeddings for the text-only prompt, fully obtained through the tokenization and embedding lookup in the LLM; ℓ denotes the recommendation loss, which is implemented as the binary cross-entropy (BCE) loss; $h_{\hat{\Theta}+\Theta'}(E_t)$ represents the LLM's prediction using E_t. and $\hat{\Theta}'$ denotes the learned parameters for the LoRA module.

• *Step 2. Tuning the CIE Module:* In this step, we tune the CIE module while keeping all other components frozen. The objective of this tuning step is to enable the CIE module to learn how to extract and map collaborative information effectively for LLM usage in recommendations. To achieve this, we utilize

prompts containing collaborative information, which are constructed using the full template, to generate predictions and tune the CIE model to minimize prediction errors. Formally, we solve the following optimization problem:

$$\min_{\Omega} \sum_{(u,i,y)\in\mathcal{D}} \ell\left(h_{\hat{\Theta}+\hat{\Theta}'}(E), y\right), \quad (5)$$

where E represents the sequence of embeddings for the full prompt, obtained through both the CIE module and the LLM's embedding lookup as shown in Equation (2). Ω denotes the model parameters of the CIE module that we aim to train, for which, we consider two different choices:

• $\Omega = \phi$, implying that we only tune the mapping layer g_ϕ while utilizing a well-trained collaborative model $f_{\psi=\hat{\psi}}$ in the CIE, where $\hat{\psi}$ represents pre-trained parameters for f_ψ (with BCE).

• $\Omega = \{\phi, \psi\}$, meaning we train both the conventional collaborative model f_ψ and the mapping layer g_ϕ within the CIE module.

We believe both choices are viable. The first option may be faster since it focuses solely on tuning the mapping function. However, the second option has the potential to lead to better performance as it can more seamlessly integrate collaborative information into LLM with fewer constraints from the collaborative model.

The above two steps are executed only once. It's worth noting that in step 2, we exclusively tune the CIE module without fine-tuning the LoRA to utilize collaborative information. The rationale behind this is as follows: after step 1, the LLM has already acquired the capability to perform recommendation tasks, i.e., inferring the matching between users and items within the token embedding space. Collaborative information would be also leveraged based on inferring the matching. Once it is mapped into the token embedding space, we believe it can be effectively used by the LLM for recommendations without further tuning the LoRA module.

C. Discussion

Relation to Soft Prompt Tuning: When considering our method without the LoRA module, it can be seen as a variant of soft prompt tuning in recommendation systems, with collaborative embeddings serving as the soft prompts. However, two distinct differences set our approach apart: 1) the soft prompt utilized by the LLM retains a low-rank characteristic, as it is derived from low-rank representations of conventional collaborative models; and 2) the collaborative model can provide valuable constraints and priors for learning the soft prompt, offering additional guidance regarding the collaborative information. These two factors enhance the efficacy of our method in capturing collaborative information and encoding personalized information more effectively.

Inference Efficiency: We acknowledge that a significant challenge for LLMRec, including our CoLLM, is its relatively high computational cost, posing impediments to practical applications. However, a range of acceleration techniques tailored for LLMs has emerged, showcasing promising outcomes, such as

caching and reusing [44]. LLMRec, including our CoLLM, could potentially benefit from these techniques to enhance their inference efficiency. Given that our objective is to incorporate collaborative information for the improvement of recommendation quality, the exploration of acceleration methods is deferred to future research. Additionally, it is noteworthy that, in comparison to existing LLMRec methods (such as TALLRec), our CoLLM just introduces the CIE module. The module is much smaller when compared to LLMs, and its training does not involve the updating of the LLM. Consequently, CoLLM would not introduce excessive additional overhead in both training and inference, as demonstrated in Section V-C3.

V. EXPERIMENTS

In this section, we perform experiments to answer two research questions: **RQ1**: Can our proposed CoLLM effectively augment LLMs with collaborative information to improve recommendation, in comparison to existing methods? **RQ2**: What impact do our design choices have on the performance/efficiency of the proposed method? How does the method perform on other datasets and LLM backbones?

A. Experimental Settings

We conduct our experiments on two datasets:
- *ML-1 M* [45] refers to the well-known movie recommendation dataset — MovieLens-1M.[1] This dataset contains user ratings on movies, collected between 2000 and 2003, with ratings on a scale of 1 to 5. We convert these ratings into binary labels using a threshold [46] of 3. Ratings greater than 3 are labeled as "positive" ($y = 1$), while the rest are labeled as "negative" ($y = 0$).
- *Amazon-Book* [15] refers to a book recommendation dataset, the "book" subset of the famous Amazon Product Review dataset.[2] It compiles user reviews of books from Amazon, collected between 1996 and 2018, with review scores ranging from 1 to 5. We transform these review scores into binary labels using a threshold[3] of 4.

To better simulate real-world recommendation scenarios and prevent data leakage [47], [48], we split the dataset into training, validation, and testing sets based on the interaction timestamp. Specifically, for ML-1 M, we preserve the interactions from the most recent twenty months, using the first 10 months for training, the middle 5 months for validation, and the last 5 months for testing. As for the Amazon-Book dataset, given its large scale, we just preserve interactions from the year 2017 (including about 4 million interactions), allocating the first 11 months for training, and the remaining two half months for validation and testing, respectively. Given the sparse nature of the Amazon-Book dataset, we filtered out users and items with fewer than 20 interactions to ensure data quality for measuring warm-start performance. The statistical information of the processed dataset is available in Table I.

---

[1]https://grouplens.org/datasets/movielens/1m/
[2]https://cseweb.ucsd.edu/~jmcauley/datasets.html#amazon_reviews
[3]A higher value is utilized to prevent significant imbalance between positive and negative data.

TABLE I
STATISTICS OF THE EVALUATION DATASETS

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

1) Evaluation Metrics: We employ two commonly used metrics for explicit recommendation to assess the performance of studied methods: AUC, UAUC [49] and NDCG [50]. AUC is the area under the ROC curve that quantifyes the overall prediction accuracy. UAUC is derived by first computing the AUC individually for each user over the exposed items and then averaging these results across all users. NDCG denotes the Normalized Discounted Cumulative Gain metrics. AUC evaluates the overall ranking quality. UAUC and NDCG provide insights into user-level ranking quality.

2) Compared Methods: To fully evaluate the proposed method CoLLM, we compare it with three types of methods: conventional collaborative methods, combining language model and collaborative model methods, and LLMRec methods. Specifically, we select the following methods as baselines:
- *MF [14]:* This refers to Matrix Factorization, one representative latent factor-based collaborative filtering method.
- *LightGCN [28]:* This is a representative graph-based collaborative filtering method, which utilizes a simplified graph convolutional neural network to enhance user interest modeling.
- *SASRec [51]:* This is a representative sequential recommendation method, which uses the self-attention network to encode sequential patterns for modeling user interest. It could be thought of as a collaborative method considering sequential information.
- *DIN [52]:* This is a representative collaborative CTR model, which employs an attention mechanism to activate the most relevant user behavior for adaptively learning user interest with respect to a certain item.
- *CTRL (DIN) [23]:* This is a state-of-the-art (SOTA) method for combining language and collaborative models through knowledge distillation. We utilize a DIN [52] as the collaborative model.
- *ICL [17]:* This is a LLMRec method based on the In-Context Learning ability of LLM. It directly queries the original LLM for recommendations using prompts.
- *Prompt4NR (Vicuna) [53]:* This is a SOTA method that uses both fixed and soft prompts to utilize traditional Language Models (LM), such as BERT [54], for recommendation purposes. We extend this method to the LLM Vicuna-7B for a fair comparison and tune the LLM with LoRA to manage computational costs.
- *TALLRec [12]:* This is a state-of-the-art LLMRec method that aligns LLM with recommendations through instruction tuning. We implement it on Vicuna-7B.

Apart from CTRL, there are other language model-based recommender models such as P5 [55] and CTR-BERT [37]. However, these models have demonstrated weaker performance when compared to CTRL [23]. Therefore, we have chosen

Dataset		ML-1M				Amazon-Book			
Methods		AUC	UAUC	NDCG	Rel. Imp.	AUC	UAUC	NDCG	Rel. Imp.
Collab.	MF	0.6482	0.6361	0.8447	10.3%	0.7134	0.5565	0.8194	12.8%
	LightGCN	0.5959	0.6499	0.8564	13.2%	0.7103	0.5639	0.8245	11.0%
	SASRec	0.7078	0.6884	0.8612	1.9%	0.6887	0.5714	0.8244	8.4%
	DIN	0.7166	0.6459	0.8496	4.9%	0.8163	0.6145	0.8419	3.2%
LM+Collab.	CTRL (DIN)	0.7159	0.6492	0.8559	4.6%	0.8202	0.5996	0.8363	4.2%
LLMRec	ICL	0.5320	0.5268	0.8114	33.8%	0.4820	0.4856	0.7917	48.2%
	Prompt4NR (Vicuna)	0.7071	0.6739	0.8663	2.7%	0.7224	0.5881	0.8346	10.4%
	TALLRec	0.7097	0.6818	0.8711	1.8%	0.7375	0.5983	0.8361	8.2%
Ours	CoLLM-MF	0.7295	0.6875	0.8714	-	0.8109	0.6225	0.8457	-
	CoLLM-LightGCN	0.7100	0.6967	0.8740	-	0.8026	0.6149	0.8411	-
	CoLLM-SASRec	0.7235	0.6990	0.8765	-	0.7746	0.5962	0.8361	-
	CoLLM-DIN	0.7353	0.6923	0.8735	-	0.8245	0.6474	0.8550	-

"Collab." denotes collaborative methods. "rel. Imp." denotes the relative improvement of CoLLM compared to baselines, averaged over the AUC and UAUC metrics. For a collaborative method "x", the rel. imp. is computed using CoLLM-X; for LLMRec methods, it is determined by comparing them to CoLLM-MF.

not to include them in our comparative analysis. Regarding BIGRec [20], it is not suitable for comparison in our setting, as it does not optimize prediction accuracy and would yield poor performance in our setting[4] Regarding our own methods, we have implemented them across all four types of collaborative models, denoted as CoLLM-MF, CoLLM-LightGCN, CoLLM-SASRec, and CoLLM-DIN, respectively. Specifically, for collaborative models (SASRec and DIN) that incorporate historical sequences, we incorporate their sequence representation as a part of the user representation within our CIE module (u in (3)).

3) Implementation Details: We implement all the compared methods using PyTorch 2.0. When not specified by the original paper, we employ Binary Cross-Entropy (BCE) as the optimization loss for all methods. For (large) language models, we use the AdamW optimizer, and for other methods, we use the Adam optimizer [56]. Regarding hyperparameter tuning, we explore the learning rate within the range of [1e-2, 1e-3, 1e-4] for all methods, and the (recommendation) embedding size within the range of [64, 128, 256]. Regarding weight decay, we set it to 1e-3 for all LLM-based methods, while we tune it in the range of [1e-2, 1e-3, . . . , 1e-7] for all other smaller models. For SASRec, we establish the maximum length of historical interaction sequences in accordance with the average user interaction count in the training data, as specified in the original paper. We adopt TALLRec's [12] practice of setting the maximum sequence length to 10 for all other methods. For DIN and CTRL (DIN), we conduct additional tuning for the dropout ratio within the range of [0.2, 0.5, 0.8]. We also adjust their hidden layer size, which varies between [200×80×1] and [256×128 × 64×1], corresponding to the two settings described in the CTRL and DIN papers. Regarding other specific parameters of the baseline models, we adhere to the configurations outlined in their original papers. For CoLLM, we set the hidden layer size of the MLP in the CIE module as ten times larger than the input size. For the LoRA module, we follow the same configuration as in the TALLRec

paper, setting r, $alpha$, $dropout$, and $target\_modules$, to 8, 16, 0.05, and "[q_proj, v_proj]", respectively.

B. Performance Comparison (RQ1)

In this section, we study the recommendation performance of CoLLM over all users as well as in different subgroups.

1) Overall Performance: The overall performance comparison between CoLLM and baselines is summarized in Table II (where the CoLLM-DIN results on ML-1 M are obtained by tuning the whole CIE). From the table, we have the following observations:

- When compared to baselines, the best version of CoLLM outperforms them in both metrics on the two datasets. The results confirm the excellence of our approach.
- In comparison between the best LLMRec baseline (TALL-Rec) and collaborative models, it outperforms MF, Light-GCN, and SASRec but falls short of beating DIN. However, after introducing collaborative information into LLMRec using CoLLM, LLM consistently achieves performance improvements (except for UAUC/NDCG for SASRec on Amazon-Book) and surpasses the corresponding collaborative baselines. This demonstrates the necessity of incorporating collaborative information.
- When focusing on LLMRec methods, the ICL method consistently produces the weakest results, in line with previous findings [12]. This underscores LLM's inherent limitation in recommendation and emphasizes the importance of tuning LLM for recommendation tasks. Interestingly, when we scrutinize Prompt4NR (Vicuna), it not only fine-tunes LLM itself but also incorporates some adaptable prompts, bearing resemblances to CoLLM. Nevertheless, it not only falls short of matching CoLLM's performance but even lags behind TALLRec, which exclusively fine-tunes LLM. This suggests that the improvements in CoLLM stem from its collaborative information modeling mechanism rather than only relying on adaptively updatable prompts.
- Regarding the CTRL approach, which integrates LM and collaborative models, it exhibits the capability to enhance

[4]For instance, BIGRec's highest AUC on ML-1 M is only 0.56, whereas TALLRec surpasses 0.70.

TABLE III
ENSEMBLE RESULTS ON THE AUC METRIC

Methods	Single			Ensemble	
	MF	TALLRec	CoLLM-MF	MF+TALLRec	MF+CoLLM-MF
ML-1M	0.6482	0.7097	<u>0.7295</u>	0.7239	**0.7364**
Amazon-book	0.7134	0.7375	<u>0.8109</u>	0.7782	**0.8112**

one metric while adversely affecting another across both datasets. This implies its capability to effectively harness both the LM's strengths and collaborative information for enhanced recommendations is limited. The operation of CTRL involves initially distilling LM information into DIN and subsequently fine-tuning DIN to incorporate collaborative information. This operation faces an inherent challenge in terms of the forgetting problem, resulting in the loss of language model information. Additionally, it continues to depend on DIN for recommendations, lacking the utilization of the LM model's inherent capabilities, such as reasoning. Our CoLLM aligns collaborative information with LLMs while still relying on the LLM for predictions, effectively mitigating these limitations and consistently improving performance.

- When implementing CoLLM's CIE module with various collaborative models, it consistently yields improvements compared to both the respective collaborative method baselines and LLMRec baselines in almost all cases. This showcases CoLLM's flexibility in incorporating different collaborative modeling mechanisms. Furthermore, it's worth highlighting that CoLLM's performance is roughly positively correlated with the performance of the corresponding collaborative model. This suggests that introducing better collaborative modeling mechanisms could contribute to enhancing the performance of CoLLM.

Ensemble: We have not included the ensemble method as our baseline above, as it could also be applied to our approach. Here, we conduct a detailed study. To do so, we employ ensemble averaging [57] to combine the MF and TALLRec models, and then compare the results with our CoLLM-MF approach. Furthermore, we investigate whether ensembling CoLLM-MF with MF can achieve further improvements through ensemble averaging. The performance of these methodologies is summarized in Table III. The table indicates that combining MF and TALLRec via ensemble averaging consistently yields inferior results compared to CoLLM-MF. However, applying ensemble averaging to CoLLM-MF still leads to some marginal improvements. These findings underscore the superiority of CoLLM's mechanism for integrating collaborative information with LLMs, surpassing mere ensemble techniques and facilitating the full utilization of LLMs' capabilities.

2) Performance in Warm and Cold Scenarios: Previous research has demonstrated that LLMRec excels in cold-start scenarios [12], while collaborative information is advantageous for modeling user interests when rich data is available [22], [58]. CoLLM seeks to incorporate collaborative information into LLMRec, aiming to make it perform well in both warm and cold scenarios. To assess the success of this goal, we conduct a detailed examination of the methods' performance in warm

and cold scenarios. In particular, we divide the testing set into warm and cold subsets: the warm subset comprises interactions between users and items that have appeared at least three times in the training dataset, while the cold subset includes the remaining interactions. Notably, our cold-start scenario is not strictly cold, as it allows users/items to have a few interactions. Without losing generality, we primarily compare MF, TALLRec, and CoLLM in terms of their performance using these two subsets. Our findings are presented in Fig. 3, from which we make three observations:

- In the warm scenario, across two datasets, TALLRec exhibits a lower AUC score compared to MF, and MF in turn is outperformed by CoLLM. In terms of UAUC, MF falls short of TALLRec, which lags behind CoLLM. These results suggest that, at the very least, in the overall evaluation (AUC) context, existing LLMRec (TALLRec) has shortcomings in warm scenarios. However, the introduction of collaborative information can lead to improvements in warm scenarios.
- In the cold scenario, both TALLRec and CoLLM clearly outperform MF. This highlights the advantages of LLMRec methods in cold scenarios, while collaborative methods lack the proficiency to handle cold scenarios. Meanwhile, CoLLM broadly maintains comparability with TALLRec, performing better in AUC and slightly worse in UAUC. These imply that CoLLM can still effectively leverage the strengths of LLMRec in cold scenarios.

Overall, CoLLM has demonstrated remarkable improvements over TALLRec in warm scenarios while maintaining its proficiency in cold scenarios. This underscores CoLLM's successful integration of collaborative information to achieve the goal of enabling LLMRec to perform effectively in both cold and warm scenarios.

C. In-Depth Analysis (RQ2)

In this subsection, we conduct experiments to study the influence of different design choices on the CoLLM.

1) The Effect of CIE Module: We begin by investigating the impact of model architecture designs. The central element of our designs is the introduction of the CIE module to extract collaborative information for LLMs. To assess its influence on CoLLM's performance, we compare CoLLM-MF with two variants: 1) the variant that directly omits the CIE module (referred to as "w/o CIE"); and 2) the variant that excludes the CIE module but instead directly introduces tokens and token embeddings to represent users and items in the LLM (referred to as "w/ UI-token"). The first variant is equal to TALLRec. The second variant follows the straightforward approach for
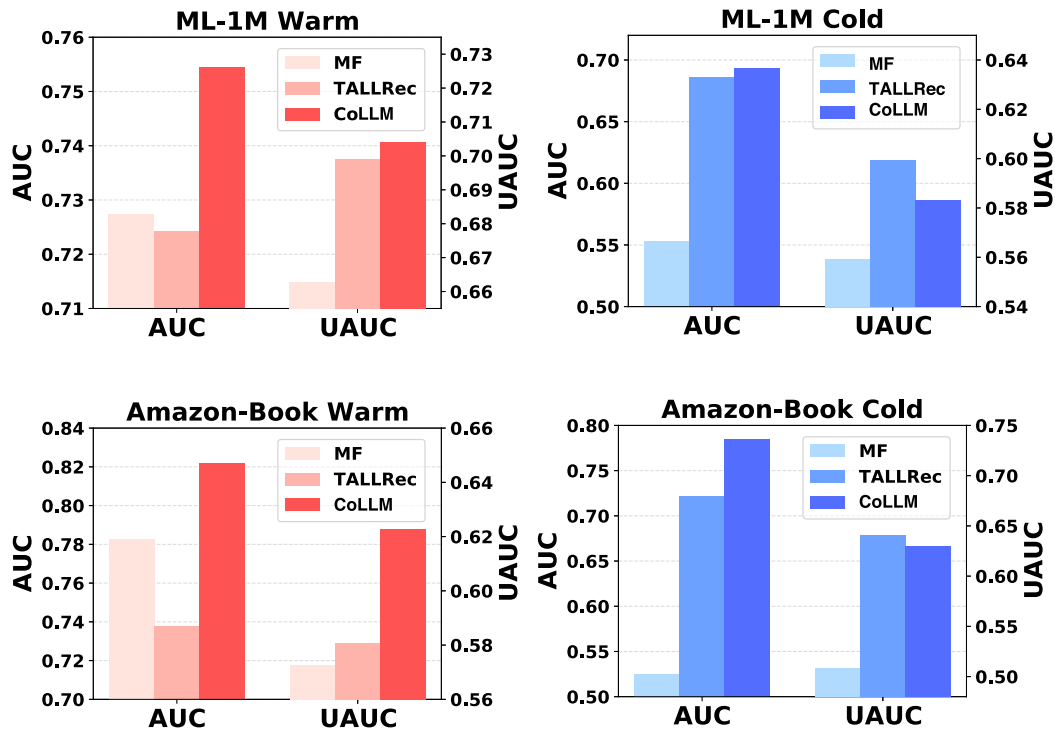
Fig. 3. Performance comparison in warm (left) and cold (right) scenarios on ML-1 M and Amazon-Book.

TABLE IV
RESULTS OF THE ABLATION STUDIES OVER CoLLM WITH RESPECT TO THE CIE MODULE

Dataset	ML-1M		Amazon-Book	
Methods	AUC	UAUC	AUC	UAUC
CoLLM-MF	0.7295	0.6875	0.8133	0.6314
w/o CIE	0.7097	0.6818	0.7375	0.5983
w/ UI-token	0.7214	0.6563	0.7273	0.5956

TABLE V
OVERALL PERFORMANCE OF CoLLM WITH DIFFERENT TUNING STRATEGIES

Dataset	ML-1M		Amazon-Book	
Tuning Methods	AUC	UAUC	AUC	UAUC
Default	0.7295	0.6875	0.8109	0.6225
T1	0.7360	0.6946	0.8154	0.6139
T2	0.7418	0.6906	0.8288	0.6352
T3	0.7131	0.6661	0.8104	0.5753

modeling collaborative information described in Section I. The comparison results are summarized in Table IV.

According to the figure, the "w/o CIE" variant falls short of the original CoLLM. This result underscores the core role of the CIE module in CoLLM to enhance the performance. The "w/ UI-token" variant also yields inferior performance compared to CoLLM and even performs worse than the variant lacking collaborative information modeling (i.e., "w/o CIE"). This observation confirms that directly introducing tokens for users and items in the LLM cannot effectively capture collaborative information for it. The rationale behind this could be that incorporating tokens (along with token embeddings) for encoding collaborative information increases tokenization redundancy within the LLM and subsequently diminishes the model's compression efficiency, leading to a reduction in predictive capabilities, as discussed in [26]. In contrast, our method employs a traditional collaborative model for encoding, effectively maintaining the modeled collaborative information in a low-rank state to reduce redundancy.

2) The Influence of Tuning Choices: We now delve into the impact of training choices on CoLLM's performance. Our default approach involves a two-step tuning strategy. Initially, we exclusively take textual information to train the LoRA module for recommendation task learning. Subsequently, we tune the mapping layer of the CIE module while keeping the collaborative model fixed as pre-trained (i.e., tuning $\Omega = \phi$ while retaining $\psi = \hat{\psi}$ after step 1). In this subsection, we further explore the following tuning strategies:

- T1, aligning with our default two-step tuning but tunes the entire CIE model ($\Omega = \{\phi, \psi\}$) with the collaborative model (ψ) initialized to the pre-trained one ($\hat{\psi}$) in the second step.
- T2, following the default two-step tuning approach but tunes the entire CIE model ($\Omega = \{\phi, \psi\}$) from scratch in the second step.
- T3, employing a one-step tuning approach, directly tuning the LoRA module and the CIE's mapping layer simultaneously while fixing the pre-trained collaborative model.

We compare these methods in terms of their overall performance, as shown in Table V, as well as their performance in warm and cold scenarios, as indicated in Fig. 4. Please note that in the figure, we have omitted the results on the Amazon-book, as they exhibit similar phenomena to those on the ML-1 M, to save space.

Based on the figure and table, we observe that within our two-step update framework, the additional tuning of the col-
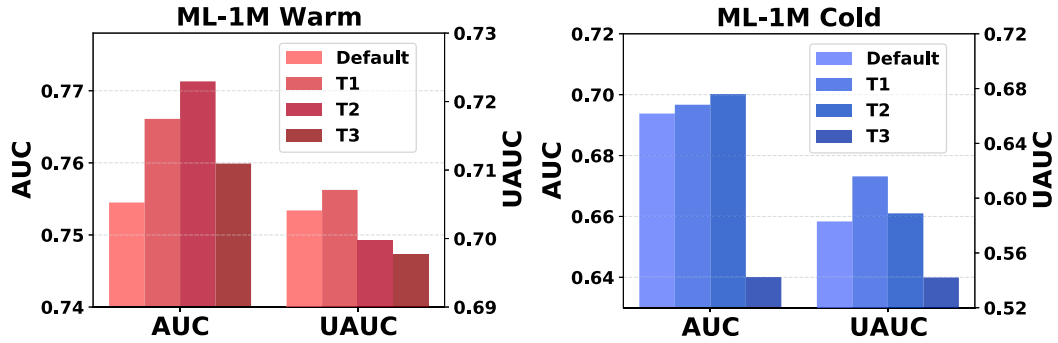
Fig. 4.   Performance of different tuning strategies for CoLLM on ML-1 M in warm and cold scenarios.

TABLE VI
TRAINING AND TOTAL INFERENCE TIME COMPARISON: TALLREC VS.
COLLM-MF

	Train Time		Inference Time	
Dataset	ML-1M	Book	ML-1M	Book
TALLRec	32min	354min	72s	360s
CoLLM-MF	36min	418min	82s	398s
Δ	13%	18%	14%	11%

"Δ" indicates the relative cost improvement of CoLLM-MF over
TALLRec. "book" is short for "amazon-book"

TABLE VII
PERFORMANCE COMPARISON ON QWEN2-1.5B BACKBONE ACROSS ML-1 M
AND AMAZON-BOOK DATASETS

Dataset	ML-1M		Amazon-Book	
Metric	AUC	UAUC	AUC	UAUC
MF	0.6482	0.6361	0.7134	0.5565
TALLRec	0.7027	0.6638	0.7256	0.5830
CoLLM-MF	0.7354	0.6950	0.8068	0.6147

laborative model in CIE (i.e., T1 and T2) can yield additional performance improvements in most cases, as expected. The additional tuning allows the captured collaborative information to be seamlessly integrated by LLMs. However, it does introduce extra computational overhead and slower convergence rates, e.g., requiring at least five times the training effort in ML-1 M. As a result, we opt for more efficient approaches. Furthermore, when comparing the single-step tuning method, T3, with the other two-step methods, we notice that it usually exhibits relatively inferior results, particularly in cold scenarios, where its performance decline is quite noticeable. This underscores the significance of the first step in our two-step training, which uses text-only data to learn recommendation tasks, to ensure recommendation performance in cold start scenarios.

3) Efficiency: Efficiency challenges pose a significant impediment to the application of the LLMRec. We next investigate how our design, which incorporates collaborative information, influences the training and inference efficiency of LLMRec. In terms of training, the primary additional cost in our approach arises from the training of the CIE module in our two-step tuning. Fortunately, the pre-training of the collaborative module (i.e., $f_{\psi=\hat{\psi}}$) in the CIE alleviates the need for extensive additional training in CoLLM when compared to the representative LLM-Rec method TALLRec. Table VI illustrates that, under identical resource conditions,[5] the training time of CoLLM increases by only approximately 15.5%, averaged across the two datasets, compared to the baseline TALLRec.

Concerning inference, the cost associated with the CIE module is anticipated to be negligible due to its significantly smaller

scale compared to the LLM. The primary additional cost in our method arises from the extra tokens required to describe collaborative information in our prompt template (as indicated in the underlined part) in Section IV-A1. These additional tokens constitute only a small proportion of the total prompts. Consequently, the supplementary inference cost is expected to be relatively modest. As demonstrated in Table VI, CoLLM incurs only a 12.5% increase in total inference cost on average across the two datasets.

4) Method Generalization: This section investigates whether our method can effectively apply to other datasets and LLM backbones.

Other LLM Backbone: To further validate the effectiveness of our approach on different LLM backbones, we implement CoLLM-MF and the top-performing LLMRec baseline, TALL-Rec, using the Qwen2-1.5B [59] backbone to compare their performance. The results, summarized in Table VII, show that CoLLM consistently outperforms both TALLRec and MF. This demonstrates the general applicability of our method across different LLM backbones.

Other Dataset: To further validate the effectiveness of our approach across different datasets, we have included two additional datasets: Video Games and CDs & Vinyl from the Amazon dataset. We take CoLLM-MF as the study example and compare it with the most related baselines MF and TALLRec. The results demonstrate that our method could still effectively enhance LLMRec by incorporating collaborative information. Specifically, on the Video Games dataset, the AUC values for MF, TALLRec, and CoLLM-MF are 0.6161, 0.7356, and 0.7440, respectively. Similarly, on the CDs & Vinyl dataset, the AUC values are 0.6957, 0.6607, and 0.7237, respectively. The consistent superior performance of the proposed method across different datasets indicates its general effectiveness.

[5]Training utilizes two Nvidia A100 80G GPUs, while testing is performed on a single GPU of the same type.

VI. CONCLUSION

In this study, we underscore the significance of collaborative information modeling in enhancing recommendation performance for LLMRec, particularly in warm scenarios. We introduce CoLLM, a novel approach tailored to incorporating collaborative information for LLMRec. By externalizing traditional collaborative models for LLMs, CoLLM not only ensures effective collaborative information modeling but also provides flexibility in adjusting the modeling mechanism. Extensive experimental results illustrate the effectiveness and adaptability of CoLLM, successfully enabling LLM to excel in both warm and cold recommendation scenarios. Currently, our experiments have been exclusively conducted on Vicuna-7B. In the future, we will explore other LLMs. Moreover, considering the evolving nature of collaborative information in the actual world, we intend to investigate CoLLM's incremental learning capabilities.

REFERENCES

[1] L. Ouyang et al., "Training language models to follow instructions with human feedback," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 27730–27744.

[2] T. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.

[3] H. Touvron et al., "LLaMA: Open and efficient foundation language models," 2023, *arXiv:2302.13971*.

[4] W. X. Zhao et al., "A survey of large language models," 2023, *arXiv:2303.18223*.

[5] J. Li, D. Li, S. Savarese, and S. C. H. Hoi, "BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 19730–19742.

[6] K. Singhal et al., "Large language models encode clinical knowledge," *Nature*, vol. 620, no. 7972, pp. 172–180, 2023.

[7] I. Singh et al., "ProgPrompt: Generating situated robot task plans using large language models," in *Proc. 2023 IEEE Int. Conf. Robot. Automat.*, 2023, pp. 11523–11530.

[8] L. Yang, H. Chen, Z. Li, X. Ding, and X. Wu, "Give us the facts: Enhancing large language models with knowledge graphs for fact-aware language modeling," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3091–3110, Jul. 2024.

[9] S. Sanner, K. Balog, F. Radlinski, B. Wedin, and L. Dixon, "Large language models are competitive near cold-start recommenders for language-and item-based preferences," in *Proc. 17th ACM Conf. Recommender Syst.*, 2023, pp. 890–896.

[10] L. Wu et al., "A survey on large language models for recommendation," *World Wide Web*, vol. 27, no. 60, 2024.

[11] Z. Zhao et al., "Recommender systems in the era of large language models (LLMs)," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 6889–6907, Nov. 2024.

[12] K. Bao, J. Zhang, Y. Zhang, W. Wang, F. Feng, and X. He, "TALLRec: An effective and efficient tuning framework to align large language model with recommendation," in *Proc. 17th ACM Conf. Recommender Syst.*, 2023, pp. 1007–1014.

[13] J. Zhang, K. Bao, Y. Zhang, W. Wang, F. Feng, and X. He, "Is chatgpt fair for recommendation? evaluating fairness in large language model recommendation," in *Proc. 17th ACM Conf. Recommender Syst.*, 2023, pp. 993–999.

[14] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[15] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proc. 25th Int. Conf. World Wide Web*, 2016, pp. 507–517.

[16] Y. Zhang et al., "Language models as recommender systems: Evaluations and limitations," in *Proc. I (Still) Can't Believe Not Better! NeurIPS Workshop*, 2021.

[17] S. Dai et al., "Uncovering chatgpt's capabilities in recommender systems," in *Proc. 17th ACM Conf. Recommender Syst.*, 2023, pp. 1126–1132.

[18] Y. Gao, T. Sheng, Y. Xiang, Y. Xiong, H. Wang, and J. Zhang, "Chat-REC: Towards interactive and explainable LLMs-augmented recommender system," 2023, *arXiv:2303.14524*.

[19] W.-C. Kang et al., "Do LLMs understand user preferences? evaluating LLMs on user rating prediction," 2023, *arXiv:2305.06474*.

[20] K. Bao et al., "A bi-step grounding paradigm for large language models in recommendation systems," *ACM Trans. Recomm. Syst.*, Feb. 2025, doi: 10.1145/3716393.

[21] J. Zhang, R. Xie, Y. Hou, W. X. Zhao, L. Lin, and J.-R. Wen, "Recommendation as instruction following: A large language model empowered recommendation approach," *ACM Trans. Inf. Syst.*, Dec. 2024, doi: 10.1145/3708882.

[22] J. Lin et al., "ReLLa: Retrieval-enhanced large language models for lifelong sequential behavior comprehension in recommendation," in *Proc. ACM Web Conf.*, 2024, pp. 3497–3508.

[23] X. Li, B. Chen, L. Hou, and R. Tang, "CTRL: Connect tabular and language model for CTR prediction," *ACM Trans. Recomm. Syst.*, Feb. 2025, doi: 10.1145/3713080.

[24] L. Wu, X. He, X. Wang, K. Zhang, and M. Wang, "A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 5, pp. 4425–4445, May 2023.

[25] X. Luo, Y. Zhou, Z. Liu, and M. Zhou, "Fast and accurate non-negative latent factor analysis of high-dimensional and sparse matrices in recommender systems," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3897–3911, Apr. 2023.

[26] G. Delétang et al., "Language modeling is compression," in *Proc. 12th Int. Conf. Learn. Representations*, 2024.

[27] J. Xu, H. Zhou, C. Gan, Z. Zheng, and L. Li, "Vocabulary learning via optimal transport for neural machine translation," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 7361–7373.

[28] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. 43 rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 639–648.

[29] J. Lin et al., "How can recommender systems benefit from large language models: A survey," *ACM Trans. Inf. Syst.*, vol. 43, no. 2, Mar. 2025, Art. no. 28, doi: 10.1145/3678004.

[30] Q. Ai et al., "Information retrieval meets large language models: A strategic report from chinese ir community," *AI Open*, vol. 4, pp. 80–90, 2023.

[31] J. Liu, C. Liu, R. Lv, K. Zhou, and Y. Zhang, "Is chatgpt a good recommender? a preliminary study," 2023, *arXiv:2304.10149*.

[32] Y. Feng et al., "A large language model enhanced conversational recommender system," 2023, *arXiv:2308.06212*.

[33] X. Huang, J. Lian, J. Lei, J. Yao, D. Lian, and X. Xie, "Recommender AI agent: Integrating large language models for interactive recommendations," 2023, *arXiv:2308.16505*.

[34] J. Wei et al., "Larger language models do in-context learning differently," 2023, *arXiv:2303.03846*.

[35] Y. Zhu, L. Wu, Q. Guo, L. Hong, and J. Li, "Collaborative large language model for recommender systems," in *Proc. ACM Web Conf.*, 2024, pp. 3162–3172.

[36] B. Zheng et al., "Adapting large language models by integrating collaborative semantics for recommendation," in *Proc. IEEE 40th Int. Conf. Data Eng.*, 2024, pp. 1435–1448.

[37] A. Muhamed et al., "CTR-BERT: Cost-effective knowledge distillation for billion-parameter teacher models," in *Proc. NeurIPS Efficient Natural Lang. Speech Process. Workshop*, 2021.

[38] S. Wu, H. Fei, L. Qu, W. Ji, and T.-S. Chua, "Next-GPT: Any-to-any multimodal LLM, 2023," in *Proc. 41st Int. Conf. Mach. Learn.*, 2024.

[39] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, "MiniGPT-4: Enhancing vision-language understanding with advanced large language models," in *Proc. 12th Int. Conf. Learn. Representations*, 2024.

[40] D. Zhang et al., "SpeechGPT: Empowering large language models with intrinsic cross-modal conversational abilities," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2023, pp. 15757–15773.

[41] D. Driess et al., "PaLM-E: An embodied multimodal language model," in *Proc. 40th Int. Conf. Mach. Learn.*, 2023, pp. 8469–8488.

[42] W.-L. Chiang et al., "Vicuna: An open-source chatbot impressing GPT-4 with 90% chatgpt quality," 2023. [Online]. Available: https://lmsys.org/blog/2023--03-30-vicuna/

[43] E. J. Hu et al., "Lora: Low-rank adaptation of large language models," in *Proc. Int. Conf. Learn. Representations*, 2022.

[44] I. Gim, G. Chen, S.-S. Lee, N. Sarda, A. Khandelwal, and L. Zhong, "Prompt cache: Modular attention reuse for low-latency inference," in *Proc. 7th Annu. Conf. Mach. Learn. Syst.*, 2024.

[45] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interactive Intell. Syst.*, vol. 5, no. 4, pp. 1–19, 2015.

[46] W. Wang, Y. Xu, F. Feng, X. Lin, X. He, and T.-S. Chua, "Diffusion recommender model," in *Proc. 46th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2023, pp. 832–841.

[47] Y. Ji, A. Sun, J. Zhang, and C. Li, "A critical study on data leakage in recommender system offline evaluation," *ACM Trans. Inf. Syst.*, vol. 41, no. 3, pp. 75:1–75:27, 2023.

[48] Y. Zhang et al., "How to retrain recommender system? a sequential meta-learning method," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 1479–1488.

[49] Y. Liu et al., "Concept-aware denoising graph neural network for micro-video recommendation," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, 2021, pp. 1099–1108.

[50] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, 2002.

[51] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *Proc. IEEE Int. Conf. Data Mining*, 2018, pp. 197–206.

[52] G. Zhou et al., "Deep interest network for click-through rate prediction," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1059–1068.

[53] Z. Zhang and B. Wang, "Prompt learning for news recommendation," in *Proc. 46th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2023, pp. 227–237.

[54] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol.*, 2019, pp. 4171–4186.

[55] S. Geng, S. Liu, Z. Fu, Y. Ge, and Y. Zhang, "Recommendation as language processing (RLP): A unified pretrain, personalized prompt & predict paradigm (P5)," in *Proc. 16th ACM Conf. Recommender Syst.*, 2022, pp. 299–315.

[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.

[57] T. G. Dietterich, "Ensemble methods in machine learning," in *Proc. Int. Workshop Mult. Classifier Syst.*, Springer, 2000, pp. 1–15.

[58] Y. Xu, L. Zhu, Z. Cheng, J. Li, Z. Zhang, and H. Zhang, "Multi-modal discrete collaborative filtering for efficient cold-start recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 1, pp. 741–755, Jan. 2023.

[59] A. Yang et al., "Qwen2 technical report," 2024, *arXiv:2407.10671*.
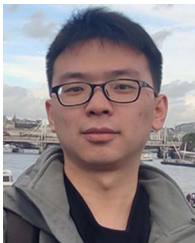
Yang Zhang received the PhD from the University of Science and Technology of China (USTC). He is a research fellow with the National University of Singapore. His research interest lies in the recommender system. He has authored over ten publications featured in top conferences and journals like SIGIR and TKDE, with one winning the Best Paper Honorable Mention with SIGIR 2021. He has served as the program committee member, and reviewer for top conferences/journals, including KDD, SIGIR, TKDE, TOIS, etc.

Fuli Feng is a professor with the University of Science and Technology of China (USTC). His research interests include information retrieval and data mining. He has about 100 publications that appeared in top conferences, such as SIGIR, WWW, and journals including TKDE and TOIS. He received the Best Paper Honourable Mention in SIGIR 2021 and the Best Poster Award in WWW 2018. Moreover, He has served as the Associate Editor for ACM TORS, the SPC/PC-member for top-tier conferences including SIGIR, WWW, SIGKDD, NeurIPS, ICLR, ICML, ACL, and the invited reviewer for prestigious journals, such as *ACM Transactions on Information Systems*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, and *IEEE Transactions on Neural Networks and Learning Systems*.

Jizhi Zhang is currently working toward the PhD degree with the University of Science and Technology of China (USTC), supervised by Prof. Fuli Feng and Prof. Xiangnan He. His research interest lies in the recommender system and LLMs. He has several publications in top conferences such as SIGIR, RecSys, and ACL. He also serves as a reviewer for academic journals including TOIS and TORS.

Keqin Bao is currently working toward the PhD degree with the University of Science and Technology of China (USTC), supervised by Prof. Fuli Feng and Prof. Xiangnan He. His research interest lies in the recommender system and LLMs. He has several publications in top conferences, such as RecSys, EMNLP and WWW. He has served as the PC member and reviewer for the top conferences and journals including TOIS and RecSys.

Qifan Wang received the PhD degree in computer science from Purdue University, in 2015. He is a research scientist with Meta AI, leading a team building innovative Deep Learning and Natural Language Processing models for Recommendation System. His research interests include deep learning, natural language processing, information retrieval, data mining, and computer vision. He has co-authored more than 100 publications in top-tier conferences and journals, including NeurIPS, ICLR, ICML, ACL, CVPR, SIGKDD, WWW, SIGIR, TPAMI, TKDE, TOIS, etc. He also serves as area chair, program committee member, editorial board member, and reviewer for academic conferences and journals.

Xiangnan He (Senior Member, IEEE) is a professor with the University of Science and Technology of China (USTC). His research interests span recommender system, data mining, and multi-media analytics. He has more than 100 publications that have appeared in top conferences, such as SIGIR, WWW, and KDD, and journals including *IEEE Transactions on Knowledge and Data Engineering* and *ACM Transactions on Information Systems*. His work on recommender systems has received the Best Paper Award Honorable Mention in SIGIR 2023/2021/2016 and WWW 2018. He has served as the associate editor for journals including *ACM Transactions on Information Systems*, *IEEE Transactions on Knowledge and Data Engineering*, etc. Moreover, he has served as SPC/PC member for several top conferences including SIGIR, WWW, KDD, MM, WSDM, ICML etc., and the regular reviewer for journals including TKDE, TOIS, etc and (senior) PC member for conferences including SIGIR, WWW, KDD, MM, etc.