Large Language Models meet Collaborative Filtering: An Efficient All-round LLM-based Recommender System

Sein Kim*
rlatpdlsgns@kaist.ac.kr
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea

Hongseok Kang*
ghdtjr0311@kaist.ac.kr
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea

Seungyoon Choi
csyoon08@kaist.ac.kr
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea

Donghyun Kim
amandus.kim@navercorp.com
NAVER Corporation
Seongnam, Republic of Korea

Minchul Yang
minchul.yang@navercorp.com
NAVER Corporation
Seongnam, Republic of Korea

Chanyoung Park[†]
cy.park@kaist.ac.kr
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea

ABSTRACT

Collaborative filtering recommender systems (CF-RecSys) have shown successive results in enhancing the user experience on social media and e-commerce platforms. However, as CF-RecSys struggles under cold scenarios with sparse user-item interactions, recent strategies have focused on leveraging modality information of user/items (e.g., text or images) based on pre-trained modality encoders and Large Language Models (LLMs). Despite their effectiveness under cold scenarios, we observe that they underperform simple traditional collaborative filtering models under warm scenarios due to the lack of collaborative knowledge. In this work, we propose an efficient **A**ll-round **LLM**-based **Rec**ommender system, called A-LLMRec, that excels not only in the cold scenario but also in the warm scenario. Our main idea is to enable an LLM to directly leverage the collaborative knowledge contained in a pre-trained state-of-the-art CF-RecSys so that the emergent ability of the LLM as well as the high-quality user/item embeddings that are already trained by the state-of-the-art CF-RecSys can be jointly exploited. This approach yields two advantages: (1) model-agnostic, allowing for integration with various existing CF-RecSys, and (2) efficiency, eliminating the extensive fine-tuning typically required for LLM-based recommenders. Our extensive experiments on various real-world datasets demonstrate the superiority of A-LLMRec in various scenarios, including cold/warm, few-shot, cold user, and cross-domain scenarios. Beyond the recommendation task, we also show the potential of A-LLMRec in generating natural language outputs based on the understanding of the collaborative knowledge by performing a favorite genre prediction task. Our code is available at https://github.com/ghdtjr/A-LLMRec.

---

*Both authors contributed equally to this research.
[†]Corresponding author.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Recommender System, Large Language Models, Collaborative Filtering

1 INTRODUCTION

With the recent exponential growth in the number of users and items, collaborative filtering models [14, 15, 20, 40] encounter the long-standing cold-start problem [1, 43, 52], stemming from the inherent sparsity of user-item interaction data. In other words, for users/items with few interactions, it becomes challenging to construct collaborative knowledge with other similar users/items, leading to suboptimal recommendation performance, especially in the cold-start scenarios. To overcome this issue, recent studies have focused on leveraging modality information of users/items (e.g., user demographics, item titles, descriptions, or images) to enhance recommendation performance under cold-start scenarios. Specifically, MoRec [51] utilizes pre-trained modality encoders (e.g., BERT [9] or Vision-Transformer [10]) to project raw modality features of items (e.g., item texts or images), thereby replacing the item embeddings typically used in collaborative filtering recommendation models. Similarly, CTRL [25] considers tabular data and its textual representation as two different modalities and uses them to pre-train collaborative filtering recommendation models through a contrastive learning objective, which is then fine-tuned for specific recommendation tasks.

---

[1]An item is categorized as 'warm' if it falls within the top 35% of interactions, and if it falls within the bottom 35%, it is classified as a 'cold' item.
[2]After training each model using all the available data in the training set, we separately evaluate on cold and warm items in the test set.
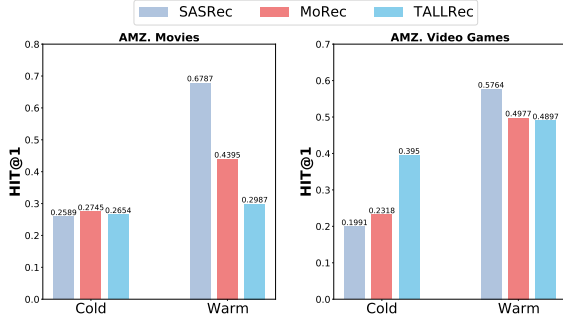
Figure 1: Comparisons between collaborative filtering model (SASRec), modality-aware model (i.e., MoRec), and LLM-based model (i.e., TALLRec) under the cold/warm[1] scenarios on Amazon Movies/Video Games dataset (Hit@1)[2].

Despite the effectiveness of modality-aware recommender systems in cold scenarios, the recent emergence of Large Language Models (LLMs), known for their rich pre-trained knowledge and advanced language understanding capabilities, has attracted significant interest in the recommendation domain to effectively extract and integrate modality information [37, 48]. Early studies on LLM-based recommendation [12, 16, 44] have employed OpenAI-GPT with *In-context Learning* [4]. This approach adapts to new tasks or information based on the context provided within the input prompt and demonstrates the potential of LLMs as a recommender system. Moreover, to bridge the gap between the training tasks of LLMs and recommendation tasks, TALLRec [2] fine-tunes LLMs with recommendation data using LoRA [18]. This approach has empirically demonstrated that, in cold scenarios and cross-domain scenarios, fine-tuned LLMs outperform traditional collaborative filtering models.

Although modality-aware and LLM-based recommender systems have proven effective in cold scenarios with limited user-item interactions, we argue that these methods suffer from the lack of collaborative knowledge due to their heavy reliance on textual information [51]. Consequently, *when abundant user-item interactions are available (i.e., warm scenario), modality-aware and LLM-based recommenders are rather inferior to simple traditional collaborative filtering models*. As shown in Figure 1, while the modality-aware recommender (i.e., MoRec) and the LLM-based recommender (i.e., TALLRec) significantly outperform the traditional collaborative filtering model (i.e., SASRec [20]) in the cold scenario, they are outperformed by the traditional collaborative filtering model in the warm scenario. This is mainly because the textual information becomes less important in the warm scenario, where ID-based collaborative filtering models excel at modeling popular items [6, 51]. However, while excelling in the cold scenario is crucial, the majority of user interactions and the revenue are predominantly generated from already existing and active items (i.e., warm items) in real-world application of recommendation systems, which contribute up to 90% of interactions in offline-industrial data [8, 49]. Furthermore, as demonstrated by DCBT [49], modeling both warm and cold items is essential for improving overall user engagement, which is evidenced by A/B testing with real-world industrial data. This implies that the warm scenario should not be overlooked.

In this paper, we propose an efficient all-round LLM-based recommender system, called A-LLMRec (**A**ll-round **LLM**-based **Rec**ommender system), that excels not only in the cold scenario but also in the warm scenario (hence, all-round recommender system). Our main idea is to enable an LLM to directly leverage the collaborative knowledge contained in a pre-trained state-of-the-art collaborative filtering recommender system (CF-RecSys) so that the emergent ability [45] of the LLM, as well as the high-quality user/item embeddings that are already trained by the state-of-the-art CF-RecSys, can be jointly exploited. More precisely, we devise an alignment network that aligns the item embeddings of the CF-RecSys with the token space of the LLM, aiming at transferring the collaborative knowledge learned from a pre-trained CF-RecSys to the LLM enabling it to understand and utilize the collaborative knowledge for the downstream recommendation task.

The key innovation of A-LLMRec is that it *requires the fine-tuning of neither the CF-RecSys nor the LLM*, and that the alignment network is the only neural network that is trained in A-LLMRec, which comes with the following two crucial advantages:

(1) **(Model-agnostic)** A-LLMRec allows any existing CF-RecSys to be integrated, which implies that services using their own recommender models can readily utilize the power of the LLM. Besides, any updates of the recommender models can be easily reflected by simply replacing the old models, which makes the model practical in reality.

(2) **(Efficiency)** A-LLMRec is efficient in that the alignment network is the only trainable neural network, while TALLRec [2] requires the fine-tuning of the LLM with LoRA [18]. As a result, A-LLMRec trains approximately **2.53** times and inferences **1.71** times faster than TALLRec, while also outperforming both TALLRec and CF-RecSys in both cold and warm scenarios.

Our extensive experiments on various real-world datasets demonstrate the superiority of A-LLMRec, revealing that aligning high-quality user/item embeddings with the token space of the LLM is the key for solving not only cold/warm scenarios but also few-shot, cold user, and cross-domain scenarios. Lastly, beyond the recommendation task, we perform a language generation task, i.e., favorite genre prediction, to demonstrate that A-LLMRec can generate natural language outputs based on the understanding of users and items through the aligned collaborative knowledge from CF-RecSys. Our main contributions are summarized as follows:

- We present an LLM-based recommender system, called A-LLMRec, that directly leverages the collaborative knowledge contained in a pre-trained state-of-the-art recommender system.
- A-LLMRec requires the fine-tuning of neither the CF-RecSys nor the LLM, while only requiring an alignment network to be trained to bridge between them.
- Our extensive experiments demonstrate that A-LLMRec outperforms not only the conventional CF-RecSys in the warm scenario but also the LLMs in the cold scenario.

2 RELATED WORK

2.1 Collaborative Filtering

Collaborative Filtering (CF) is the cornerstone of recommendation systems, fundamentally relying on leveraging users' historical

preferences to inform future suggestions. The key idea is to rely on similar users/items for recommendations. The emergence of matrix factorization marked a significant advancement in CF, as evidenced by numerous studies [19, 22, 38], demonstrating its superiority in capturing the latent factors underlying user preferences. This evolution continued with the introduction of Probabilistic Matrix Factorization (PMF) [5, 33] and Singular Value Decomposition (SVD) [30, 54], which integrate probabilistic and decomposition techniques to further refine the predictive capabilities of CF models. AutoRec [39] and Neural Matrix Factorization (NMF) [15] utilized deep learning to enhance CF by capturing complex user-item interaction patterns. Recently, [7, 21, 34, 36] proposed modeling collaborative filtering based on sequential interaction history. Caser [41] and NextItNet [50] utilize Convolutional Neural Networks (CNNs) [23] to capture the local sequence information, treating an item sequence as images. While these methods effectively capture user preferences using interaction history, including user and item IDs, they overlook the potential of the modality information of the user/item, which could enhance model performance and offer a deeper analysis of user behaviors.

2.2 Modality-aware Recommender Systems

Modality-aware recommenders utilize modality information such as item titles, descriptions, or images to enhance the recommendation performance mainly under cold scenarios. Initially, CNNs were used to extract visual features, modeling human visual preferences based on Mahalanobis distance [31]. With advancements in pre-trained modality encoders like BERT [9, 27, 29, 47, 51] and ResNet/Vision-Transformer [10, 11], modality-aware recommender systems have accelerated research by utilizing modality knowledge on recommendation tasks. For example, NOVA [27] and DMRL [28] proposed non-invasive fusion and disentangled fusion of modality, respectively, by carefully integrating pure item embeddings and text-integrated item embeddings using the attention mechanism. MoRec [51] leverages modality encoders to project raw modality features, thereby replacing item embeddings used in collaborative filtering models. As for the pre-training based models, Liu et al. [29] constructs user-user and item-item co-interaction graphs to extract collaborative knowledge, then integrates with user/item text information through attention mechanism in an auto-regressive manner, and CTRL [25] pre-trains the collaborative filtering models using paired tabular data and textual data through a contrastive learning objective, subsequently fine-tuning them for recommendation tasks. Most recently, RECFORMER [24] proposed to model user preferences and item features as language representations based on the Transformer architecture by formulating the sequential recommendation task as the next item sentence prediction task, where the item key-value attributes are flattened into a sentence.

2.3 LLM-based Recommender Systems

Recently, research on LLMs has gained prominence in the field of modality-aware recommendation systems, with LLM-based recommendations emerging as a significant area of focus. The pre-trained knowledge and the reasoning power of LLMs based on the advanced comprehension of language are shown to be effective for recommendation tasks, and many approaches have been proposed leveraging

LLM as a recommender system. More precisely, [12, 16, 44] utilize LLMs with *In-context Learning* [4], adapting to new tasks or information based on the context provided within the input prompt. For example, Sanner et al. [37] employs In-context Learning for recommendation tasks, exploring various prompting styles such as completion, instructions, and few-shot prompts based on item texts and user descriptions. Gao et al. [12] assigns the role of a recommender expert to rank items that meet users' needs through prompting and conducts zero-shot recommendations. These studies empirically demonstrated the potential of LLMs using its rich item information and natural language understanding in the recommendation domain. However, these approaches often underperform traditional recommendation models [20, 40], due to the gap between the natural language downstream tasks used for training LLMs and the recommendation task [2]. To bridge this gap, TALL-Rec [2] employs the *Parameter Efficient Fine-Tuning* (PEFT) method, also known as LoRA [18]. This methodology enables TALLRec to demonstrate enhanced efficacy, surpassing traditional collaborative filtering recommendation models, particularly in mitigating the challenges posed by the cold start dilemma and in navigating the complexities of cross-domain recommendation scenarios. However, it is important to note that since TALLRec simply converts the conventional recommendation task into an instruction text and uses it for fine-tuning, it still fails to explicitly capture the collaborative knowledge that is crucial in warm scenarios.

3 PROBLEM FORMULATION

In this section, we introduce a formal definition of the problem including the notations and the task description.

Notations. Let $\mathcal{D}$ denote the historical user-item interaction dataset $(\mathcal{U}, \mathcal{I}, \mathcal{T}, \mathcal{S}) \in \mathcal{D}$, where $\mathcal{U}, \mathcal{I}, \mathcal{T}$, and $\mathcal{S}$ denote the set of users, items, item titles/descriptions, and item sequences, respectively. $\mathcal{S}^u = (i_1^u, i_2^u, \cdots, i_k^u, \cdots i_{|\mathcal{S}^u|}^u) \in \mathcal{S}$ is a sequence of item interactions of a user $u \in \mathcal{U}$, where i_k^u denotes the k-th interaction of user u, and this corresponds to the index of the interacted item in the item set $\mathcal{I}$. Moreover, each item $i \in \mathcal{I}$ is associated with title and description text $(t^i, d^i) \in \mathcal{T}$.

Task: Sequential Recommendation. The goal of sequential recommendation is to predict the next item to be interacted with by a user based on the user's historical interaction sequence. Given a set of user historical interaction sequences $\mathcal{S} = \left\{ \mathcal{S}^1, \mathcal{S}^2, \cdots, \mathcal{S}^{|\mathcal{U}|} \right\}$, where $\mathcal{S}^u$ denotes the sequence of user u, the subset $\mathcal{S}_{1:k}^u \subseteq \mathcal{S}^u$ represents the sequence of user u from the first to the k-th item denoted as $\mathcal{S}_{1:k}^u = (i_1^u, i_2^u, \cdots, i_k^u)$. Given an item embedding matrix $\mathbf{E} \in \mathbb{R}^{|I| \times d}$, the embedding matrix of items in $\mathcal{S}_{1:k}^u$ is denoted by $\mathbf{E}_{1:k}^u = (\mathbf{E}_{i_1^u}, \mathbf{E}_{i_2^u}, ..., \mathbf{E}_{i_k^u}) \in \mathbb{R}^{k \times d}$, where $\mathbf{E}_{i_j^u}$ denotes the i_j^u-th row of $\mathbf{E}$. This sequence embedding matrix is fed into a collaborative filtering recommender (e.g., SASRec [20]) to learn and predict the next item in the user behavior sequence $\mathcal{S}_{1:k}^u$ as follows:

$$\max_{\Theta} \prod_{u \in \mathcal{U}} \prod_{k=1}^{|\mathcal{S}^u|-1} p(i_{k+1}^u | \mathcal{S}_{1:k}^u; \Theta) \tag{1}$$

where $p(i_{k+1}^u | \mathcal{S}_{1:k}^u; \Theta)$ represents the probability of the $(k+1)$-th interaction of user u conditioned on the user's historical interaction sequence $\mathcal{S}_{1:k}^u$, and Θ denotes the set of learnable parameters of the
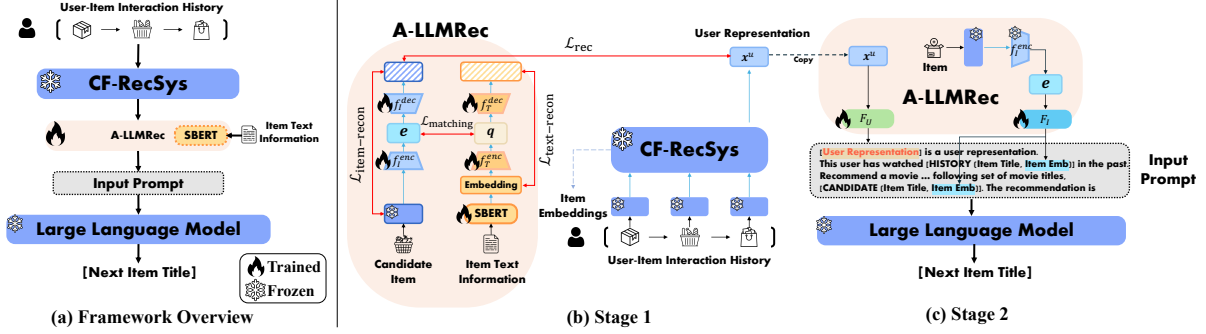
Figure 2: (a) is the overview of A-LLMRec. (b) and (c) are the detailed architecture of Stage 1 and Stage 2, respectively.

collaborative filtering recommender (CF-RecSys). By optimizing Θ to maximize Equation 1, the model can obtain the probability of the next items for user u, over all possible items.

It is important to note that although we mainly focus on the sequential recommendation task in this work, A-LLMRec can also be readily applied to non-sequential recommendation tasks by simply replacing the backbone CF-RecSys, e.g., from SASRec [20] (sequential) to NCF [15] (non-sequential), which will be demonstrated in the experiments (Section 5.4.3).

4 PROPOSED METHOD: A-LLMREC

In this section, we propose A-LLMRec, a novel LLM-based recommender framework that aligns a frozen pre-trained collaborative filtering recommender (CF-RecSys) with a frozen LLM aiming to enhance the recommendation performance not only in the cold scenario but also in the warm scenario. To bridge the modality gap, A-LLMRec aligns collaborative knowledge of the CF-RecSys with the token space of the LLM. Our approach involves two pre-training stages: (1) Aligning collaborative and textual knowledge with a frozen CF-RecSys (Section 4.1), and (2) Recommendation stage with a frozen LLM (Section 4.2) in which the joint collaborative and textual knowledge is projected onto the LLM.

4.1 Alignment between Collaborative and Textual Knowledge (Stage-1)

In this section, we introduce how to align the item embeddings from a frozen CF-RecSys with their associated text information to capture both collaborative and textual knowledge. We employ a pre-trained Sentence-BERT (SBERT) [35] model, which is fine-tuned during training, to extract text embeddings from textual information associated with items[3]. Then, we introduce two encoders, i.e., item encoder f_I^{enc} and text encoder f_T^{enc}, each containing a 1-layer Multi-Layer Perceptron (MLP), to align the item embeddings from a frozen CF-RecSys with the text embeddings from SBERT. Given an item i, the item encoder $f_I^{enc} : \mathbb{R}^d \to \mathbb{R}^{d'}$ encodes an item embedding $\mathbf{E}_i \in \mathbb{R}^d$ into a latent item embedding $\mathbf{e}_i \in \mathbb{R}^{d'}$, i.e., $\mathbf{e}_i = f_I^{enc}(\mathbf{E}_i)$, while the text encoder $f_T^{enc} : \mathbb{R}^{768} \to \mathbb{R}^{d'}$ encodes a text embedding $\mathbf{Q}_i \in \mathbb{R}^{768}$ from SBERT, whose output dimension size is 768, into a latent text embedding $\mathbf{q}_i \in \mathbb{R}^{d'}$, i.e., $\mathbf{q}_i = f_T^{enc}(\mathbf{Q}_i)$. Then, we perform latent space matching between item embeddings

[3]Although using a larger language model, such as OPT [53] and LLaMA [42], would further enhance the quality of the text embeddings, we adopt SBERT for efficiency.

and text embeddings as follows:

$$\mathcal{L}_{\text{matching}} = \mathop{\mathbb{E}}_{\mathcal{S}^u \in \mathcal{S}} \left[\mathop{\mathbb{E}}_{i \in \mathcal{S}^u} \left[MSE(\mathbf{e}_i, \mathbf{q}_i) \right] \right]$$
$$= \mathop{\mathbb{E}}_{\mathcal{S}^u \in \mathcal{S}} \left[\mathop{\mathbb{E}}_{i \in \mathcal{S}^u} \left[MSE(f_I^{enc}(\mathbf{E}_i), f_T^{enc}(\mathbf{Q}_i)) \right] \right] \tag{2}$$

where $\mathbf{Q}_i = SBERT(\text{"Title} : t^i, Description : d^i\text{"})$ denotes the encoded representation of item text (i.e., item title and description) by SBERT, and *MSE* is the mean squared error loss. That is, we match the item embeddings from a frozen CF-RecSys and the text embeddings from SBERT in the latent space of the encoders, so as to align the semantics of items and their associated texts for later use in the LLM.

4.1.1 Avoiding Over-smoothed Representation. On the other hand, simply optimizing the latent space matching loss defined in Equation 2 would result in over-smoothed representations, i.e., the encoders would be trained to produce similar outputs (i.e., $\mathbf{e}_i \approx \mathbf{q}_i$) to minimize $\mathcal{L}_{\text{matching}}$. In an extreme case, the output of the encoders would be collapsed to a trivial representation by assigning their weights to all zeros. Hence, to prevent this issue and preserve the original information of the item and its associated text embedding, we add a decoder to each of the encoders and introduce reconstruction losses as follows:

$$\mathcal{L}_{\text{item-recon}} = \mathop{\mathbb{E}}_{\mathcal{S}^u \in \mathcal{S}} \left[\mathop{\mathbb{E}}_{i \in \mathcal{S}^u} \left[MSE(\mathbf{E}_i, f_I^{dec}(f_I^{enc}((\mathbf{E}_i)))) \right] \right] \tag{3}$$

$$\mathcal{L}_{\text{text-recon}} = \mathop{\mathbb{E}}_{\mathcal{S}^u \in \mathcal{S}} \left[\mathop{\mathbb{E}}_{i \in \mathcal{S}^u} \left[MSE(\mathbf{Q}_i, f_T^{dec}(f_T^{enc}((\mathbf{Q}_i)))) \right] \right] \tag{4}$$

where f_I^{dec} and f_T^{dec} are the decoders added to the encoders f_I^{enc} and f_T^{enc}, respectively. In Section 5.3.1, we empirically demonstrate the benefit of introducing the reconstruction losses.

4.1.2 Recommendation Loss. Besides aligning the collaborative knowledge from the user-item interactions with the textual knowledge from the associated text information, we introduce a recommendation loss to explicitly incorporate the collaborative knowledge, while informing the model about the recommendation task. Specifically, the recommendation loss is defined as follows [20]:

$$\mathcal{L}_{\text{rec}} = - \sum_{\mathcal{S}^u \in \mathcal{S}} \left[log(\sigma(s(\mathbf{x}^u_{|\mathcal{S}^u|-1}, f_I^{dec}(f_I^{enc}(\mathbf{E}_{i^u_{|\mathcal{S}^u|}}))))) \right.$$
$$\left. + log(1 - \sigma(s(\mathbf{x}^u_{|\mathcal{S}^u|-1}, f_I^{dec}(f_I^{enc}(\mathbf{E}_{i^{u,-}_{|\mathcal{S}^u|}}))))) \right] \tag{5}$$

where $\mathbf{x}^u_{|\mathcal{S}^u|-1} = \text{CF-RecSys}(\mathcal{S}^u_{1:|\mathcal{S}^u|-1}) \in \mathbb{R}^d$ is the user representation extracted from the collaborative filtering recommender

system, i.e., CF-RecSys, obtained after the user u has interacted with the last item in the sequence $\mathcal{S}^u_{1:|\mathcal{S}^u|-1}$, and $\mathbf{E}_{i^{u,-}_{|\mathcal{S}^u|}} \in \mathbb{R}^d$ is the embedding of a negative item of $i^u_{|\mathcal{S}^u|}$, i.e., $i^{u,-}_{|\mathcal{S}^u|}$, and $s(\mathbf{a}, \mathbf{b})$ is a dot product between $\mathbf{a}$ and $\mathbf{b}$.

4.1.3 Final Loss of Stage-1. Finally, the final objective of Stage-1, i.e., $\mathcal{L}_{\text{stage-1}}$, is the sum of the matching loss defined in Equation 2, reconstruction losses defined in Equation 3 and 4, and recommendation loss in Equation 5:

$$\mathcal{L}_{\text{stage-1}} = \mathcal{L}_{\text{matching}} + \alpha \mathcal{L}_{\text{item-recon}} + \beta \mathcal{L}_{\text{text-recon}} + \mathcal{L}_{\text{rec}} \quad (6)$$

where α and β are the coefficients that control the importance of each term. Note that for efficiency in training, we only considered the last item in $\mathcal{S}^u$ for each user u to minimize $\mathcal{L}_{\text{stage-1}}$. However, considering all items in the sequence further enhances the recommendation performance, which will be shown in Section 5.4.2.

4.1.4 Joint Collaborative-Text Embedding. Having trained the autoencoder based on Equation 6, we consider $\mathbf{e}_i = f^{enc}_I(\mathbf{E}_i)$ as the joint collaborative-text embedding (shortly joint embedding) of item i, which will be passed to the LLM as input. The joint embedding introduces the collaborative and textual knowledge to LLMs, which will be described in Section 4.2.

It is important to note that when encountering new items that have not been seen during the training of the collaborative filtering recommender, we can instead rely on the text encoder f^{enc}_T to extract the joint collaborative-text embedding, i.e., $\mathbf{q}_i = f^{enc}_T(\mathbf{Q}_i)$. Since the two encoders f^{enc}_I and f^{enc}_T are jointly trained to match their latent spaces, we expect the joint embedding $\mathbf{q}_i$ to not only capture the textual knowledge but also to implicitly capture the collaborative knowledge. In summary, we use $\mathbf{e}_i = f^{enc}_I(\mathbf{E}_i)$ as the joint collaborative-text embedding by default, but we use $\mathbf{q}_i = f^{enc}_T(\mathbf{Q}_i)$ when item i lacks interactions, i.e., cold item, few-shot, and cross-domain scenarios, which will be demonstrated in the experiments in Section 5.2.2, Section 5.2.4, and Section 5.2.5, respectively.

4.2 Alignment between Joint Collaborative-Text Embedding and LLM (Stage-2)

Recall that in Stage-1 we obtained the joint collaborative-text embeddings by aligning the collaborative knowledge with item textual information. Our goal in Stage-2 is to align these joint embeddings with the token space of the LLM (Section 4.2.1), and design a prompt that allows the LLM to solve the recommendation task by leveraging the learned collaborative knowledge (Section 4.2.2). Figure 2 shows the overall architecture of Stage-2. Note that the component trained in Stage-1, which is also utilized in Stage-2, i.e., f^{enc}_I, is frozen in Stage-2.

4.2.1 Projecting collaborative knowledge onto the token space of LLM. We first project the user representations $\mathbf{x}^u \in \mathbb{R}^d$ and the joint collaborative-text embeddings $\mathbf{e}_i \in \mathbb{R}^{d'}$ obtained from Stage-1 onto the token space of LLM, i.e., $\mathbb{R}^{d^{\text{token}}}$. By doing so, we allow the LLM to take them as inputs. More precisely, we introduce two 2-layer MLPs, i.e., $F_U : \mathbb{R}^d \rightarrow \mathbb{R}^{d^{\text{token}}}$ and $F_I : \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d^{\text{token}}}$, to project the user representations and the joint collaborative-text embeddings to the token space of LLM, respectively, as follows:

$$\mathbf{O}_u = F_U(\mathbf{x}^u), \; \mathbf{O}_i = F_I(\mathbf{e}_i) \quad (7)$$
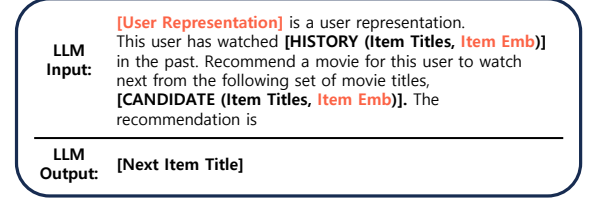

Figure 3: An example prompt of A-LLMRec designed for the Amazon Movies dataset. For other datasets, we keep the same format but adjust the verbs and nouns to fit the context (e.g., 'watched' → 'bought', 'movie' → 'item').

where $\mathbf{O}_u \in \mathbb{R}^{d^{\text{token}}}$ and $\mathbf{O}_i \in \mathbb{R}^{d^{\text{token}}}$ are the projected embeddings of the representation of user u and the joint collaborative-text embedding of item i, and they can now be used as inputs to LLM prompts, which allow the LLM to perform recommendation without any fine-tuning.

4.2.2 Prompt Design for Integrating Collaborative Knowledge. Prompt engineering helps in understanding the capabilities and limitations of LLMs, enabling them to perform complex tasks such as question answering and arithmetic reasoning [4, 46]. Recent studies on LLM-based recommender systems have shown that carefully crafted prompts enhance the performance of LLMs [2, 16, 37]. However, as existing LLM-based recommender systems focus on cold scenarios with few user-item interactions, their prompts mainly consider ways to incorporate modality information (e.g., item description text), while overlooking the collaborative knowledge. To this end, we introduce a novel approach to prompt design for LLM-based recommender system, which combines collaborative knowledge with recommendation instructions (See Figure 3). This is done by directly incorporating user representations $\mathbf{O}_u$ and joint collaborative-text embeddings $\mathbf{O}_i$ into the textual prompts in the token embedding space. In other words, as $\mathbf{O}_u$ and $\mathbf{O}_i$ have been projected into the LLM token space, they can be considered as ordinary tokens used by the LLM and readily incorporated within a prompt. To facilitate the understanding of the LLM regarding the given user, which is crucial for personalized recommendation, we place the projected user representation $\mathbf{O}_u$ at the beginning of the prompt to provide the LLM with the information about users, which is analogous to soft prompts [26]. Moreover, we add the projected joint embedding of an item $\mathbf{O}_i$ next to its title. This structured prompt then serves as an input to the LLM, with the expected output being recommendations tailored to the user. The learning objective of Stage-2 is given as follows:

$$\max_\theta \sum_{\mathcal{S}^u \in \mathcal{S}} \sum_{k=1}^{|y^u|} log(P_{\theta,\Theta}(y^u_k | p^u, y^u_{<k})) \quad (8)$$

where θ denotes the learnable parameters of F_U and F_I, Θ is the frozen parameters of LLM, p^u and y^u are the input prompt and the next item title of user u, respectively. y^u_k is the k-th token of y^u and $y^u_{<k}$ represents the tokens before y^u_k. Note that we only use the last item of each user sequence to train Equation 8 for efficiency.

Table 1: Overall model performance (Hit@1) over various datasets. The best performance is denoted in bold.

	Collaborative filtering				Modality-aware			LLM-based			
	NCF	NextItNet	GRU4Rec	SASRec	MoRec	CTRL	RECFORMER	LLM-Only	TALLRec	MLP-LLM	A-LLMRec
Movies and TV	0.4273	0.5855	0.5215	0.6154	0.4130	0.3467	0.4865	0.0121	0.2345	0.5838	**0.6237**
Video Games	0.3159	0.4305	0.4026	**0.5402**	0.4894	0.2354	0.4925	0.0168	0.4403	0.4788	0.5282
Beauty	0.2957	0.4231	0.4131	0.5298	0.4997	0.3963	0.4878	0.0120	0.5542	0.5548	**0.5809**
Toys	0.1849	0.1415	0.1673	0.2359	0.1728	0.1344	0.2871	0.0141	0.0710	0.3225	**0.3336**

Table 2: Statistics of the dataset after preprocessing. Avg. Len denotes the average sequence length of users.

Datasets	#Users	#Items	#Interactions.	Avg. Len
Movies and TV	297,498	59,944	3,409,147	11.46
Video Games	64,073	33,614	598,509	8.88
Beauty	9,930	6,141	63,953	6.44
Toys	30,831	61,081	282,213	9.15

Table 3: Hyperparameter specifications of A-LLMRec

	Learning rate stage 1	Learning rate stage 2	embedding dim (CF-RecSys) d	embedding dim $(f_I^{enc}, f_T^{enc})\, d'$	alpha	beta
Movies and TV	0.0001	0.0001	50	128	0.5	0.5
Video Games	0.0001	0.0001	50	128	0.5	0.5
Beauty	0.0001	0.0001	50	128	0.5	0.2
Toys	0.0001	0.0001	50	128	0.5	0.2

5 EXPERIMENTS

5.1 Experimental Setup

Datasets. For comprehensive evaluations, we used four datasets from Amazon datasets [13, 32], i.e., Movies and TV, Video Games, Beauty, and Toys, which consist of comprehensive textual information including "title" and "description." Note that we deliberately selected datasets with varying statistics in terms of number of users and items to conduct an extensive analysis of the models. The statistics for each dataset after preprocessing are presented in Table 2 and we describe details regarding data preprocessing as follows:

- **Movies and TV** To evaluate the models on a large scale, we select about 300K users and 60K items. Following existing studies [20, 51], we removed users and items with fewer than 5 interactions.
- **Video Games** To evaluate the models on moderate-scale data, which is smaller than the Movies and TV dataset, we select about 64K users and 33K items, removing users and items with fewer than 5 interactions, as in the Movies and TV dataset.
- **Beauty** To compose a small and cold dataset, we select about 9K users and 6K items, removing users and items with fewer than 4 interactions. To retain some information from user-item feedback, we categorized user ratings by treating items above 3 as positive and all others including non-interacted items as negative.
- **Toys** For the evaluation of the models where the number of items is larger than number of users, unlike other datasets, we select about 3K users and 6K items, with the number of items being twice as large as the number of users, and remove users and items with fewer than 4 interactions. Similar to the Beauty dataset, to preserve some information from user-item feedback, we categorize positive and negative items with the criterion of rating 3.

Baselines. We compare A-LLMRec with the following baselines that can be categorized into three types: collaborative filtering recommender systems (NCF [15], NextItNet [50], GRU4Rec [17] and SASRec [20]), modality-aware recommender systems (MoRec [51], CTRL [25], and RECFORMER [24]), and LLM-based recommender

systems (LLM-Only, TALLRec [2] and MLP-LLM). For more detail regarding the baselines, please refer to Appendix A

Evaluation Setting. We divide user sequences into training, validation, and test sets. For each user sequence, the most recently interacted item, denoted as $i_{|S^u|}^u$, is used as the test set, while the second most recent user interaction item, $i_{|S^u|-1}^u$, is used as the validation set. The remaining sequence of items is used as the training set. To evaluate the performance of sequential recommendation models, we add 19 randomly selected non-interacted items to the test set, so that the test set of each user contains 1 positive item and 19 negative items. For quantitative comparison, we employ a widely used metric, Hit Ratio at 1 (Hit@1) for all experiments.

Implementation Details. Although A-LLMRec is model-agnostic, in this work, we adopt OPT-6.7B [53] as the backbone LLM and SASRec [20] as the pre-trained CF-RecSys. For fair comparisons, we also used OPT-6.7B as the backbone LLM for other LLM-based models (i.e., LLM-Only, TALLRec [2] and MLP-LLM). Moreover, we use SASRec as the CF-RecSys in other modality-aware models (i.e., MoRec [51] and CTRL [25]), and fix the dimension of item and model embeddings to 50 for all the methods and datasets. For RECFORMER [24], we follow the paper and employ Longformer [3] as the backbone network. We set the batch size to 128 for all collaborative filtering-based and modality-aware models. Moreover, the batch size is set to 32 for Stage-1 of A-LLMRec, and 4 for MLP-LLM, TALLRec, and Stage-2 of A-LLMRec. We trained Stage-1 of A-LLMRec for 10 epochs, and Stage-2 of A-LLMRec for 5 epochs, and TALLRec is trained for a maximum of 5 epochs. We use the Adam optimizer to train the models in all datasets. For hyperparameters, we tune the model in certain ranges as follows: learning rate η_1, η_2 in $\{0.01, 0.001, 0.0005, 0.0001\}$ for the training stage each, coefficient α, β in $\{0.1, 0.2, 0.5, 0.75, 1.0\}$ for each, we report the best-performing hyper-parameters for each dataset in Table 3. We use four NVIDIA GeForce A6000 48GB for the Movies and TV dataset to train LLM-based models, and one NVIDIA GeForce A6000 48GB for other datasets including LLM-based and other models.

5.2 Performance Comparison

For comprehensive evaluations of A-LLMRec, we perform evaluations under various scenarios, i.e., general scenario (Sec. 5.2.1),

Table 4: Results (Hit@1) on cold/warm item scenario. A-LLMRec (SBERT) is a variant of A-LLMRec that uses q instead of e for inference.

	Movies and TV		Video Games		Beauty	
	Cold	Warm	Cold	Warm	Cold	Warm
SASRec	0.2589	0.6787	0.1991	0.5764	0.1190	0.6312
MoRec	0.2745	0.4395	0.2318	0.4977	0.2145	0.5425
CTRL	0.1517	0.3840	0.2074	0.2513	0.1855	0.4711
RECFORMER	0.3796	0.5449	0.3039	0.5377	0.3387	0.5133
TALLRec	0.2654	0.2987	0.3950	0.4897	0.5462	0.6124
A-LLMRec	0.5714	**0.6880**	0.4263	**0.5970**	**0.5605**	**0.6414**
A-LLMRec (SBERT)	**0.5772**	0.6802	**0.4359**	0.5792	0.5591	0.6405

cold/warm item scenario (Sec. 5.2.2), cold user scenario (Sec. 5.2.3), few-shot training scenario (Sec. 5.2.4), cross-domain scenario (Sec. 5.2.5).

5.2.1 Overall Performance. The results of the recommendation task on four datasets are given in Table 1. We have the following observations: **1)** A-LLMRec outperforms other LLM-based recommender systems that do not consider the collaborative knowledge from user-item interactions (i.e., LLM-Only and TALLRec), implying that the collaborative knowledge is crucial for improving the performance of recommendation in general. **2)** We observe that MLP-LLM, which replaces the alignment module of A-LLMRec with a simple MLP, underperforms A-LLMRec. This implies that bridging between CF-RecSys and LLM is a challenging problem and that our proposed two-stage alignment module is beneficial. **3)** 'LLM-Only' performs the worst among the LLM-based models, implying that naively adopting an LLM based on a prompt designed for the recommendation task is not sufficient. Note that the prompt used by 'LLM-Only' is exactly the same as the prompt shown in Figure 3 without user representation and item embeddings. This again demonstrates the importance of incorporating collaborative knowledge into the LLM for improving the recommendation performance. **4)** While TALLRec fine-tunes the LLM for the recommendation task, it underperforms a collaborative filtering model, SASRec. This highlights that the text information alone may not generate sufficient knowledge for capturing collaborative knowledge effectively even with fine-tuning the LLM. This again demonstrates the superiority of our alignment module. **5)** Although the modality-aware models (MoRec and CTRL) use SASRec as the backbone CF-RecSys, they underperform SASRec. Moreover, RECFORMER struggles to outperform SASRec despite using Longformer for item text attributes, due to the emphasis on textual information in similarity matching between user and item sentences. This shows that the modality knowledge might hinder the learning of collaborative knowledge, leading to performance degradation.

5.2.2 Cold/Warm Item Scenarios. This section evaluates the models under cold/warm item scenarios. Items are labeled as 'warm' if they belong to the top 35% of interactions, while those in the bottom 35% are labeled as 'cold' items. After training each model using all the available data in the training set, we separately evaluate cold and warm items in the test set (Table 4). We make the following observations: **1)** A-LLMRec outperforms all other baselines across both scenarios, which demonstrates that our alignment network indeed allows the LLM to understand and utilize the collaborative knowledge. **2)** On the other hand, TALLRec outperforms SASRec

Table 5: Results (Hit@1) on cold user scenario.

	Movies and TV	Video Games	Beauty
SASRec	0.2589	0.4048	0.4459
MoRec	0.3918	0.3572	0.4815
CTRL	0.2273	0.1737	0.3902
RECFORMER	0.4481	0.3989	0.4644
TALLRec	0.2143	0.3895	0.5202
MLP-LLM	0.4909	0.3960	0.5276
A-LLMRec	**0.5272**	**0.4160**	**0.5337**

Table 6: Results (Hit@1) on the few-shot training scenario on various datasets (K: num. users in the training set).

	K	SASRec	MoRec	TALLRec	A-LLMRec	A-LLMRec (SBERT)
Movies and TV	256	0.2111	0.2208	0.1846	0.2880	**0.2963**
	128	0.1537	0.1677	0.1654	0.2518	**0.2722**
Video Games	256	0.1396	0.1420	0.2321	0.2495	**0.2607**
	128	0.1089	0.1157	0.1154	0.1608	**0.1839**
Beauty	256	0.2243	0.2937	0.3127	0.3467	**0.3605**
	128	0.1813	0.2554	0.2762	0.3099	**0.3486**

only under cold scenario, whereas SASRec outperforms TALLRec only under warm scenario. This demonstrates the importance of capturing both the collaborative knowledge and the text information to excel in both cold/warm scenarios. **3)** A-LLMRec (SBERT) outperforms A-LLMRec under the cold item scenario, while A-LLMRec generally outperforms A-LLMRec (SBERT) under the warm item scenario. As discussed in Section 4.1.4, this implies that the joint collaborative-text embedding obtained from the text encoder given the text information (i.e., $\mathbf{q_i} = f_T^{enc}(\mathbf{Q}_i)$) is more useful than that obtained from the item encoder given the item embedding (i.e., $\mathbf{e_i} = f_I^{enc}(\mathbf{E}_i)$).

5.2.3 Cold User Scenarios. Besides evaluations under the cold item scenario, we additionally conduct evaluations under the cold user scenario (Table 5). To simulate the cold user scenario, we sample users who have interacted with exactly three items, where the last item in the sequence serves as the test set. Then, we use the models trained on the entire set of users except for the sampled users to perform inference on the sampled users. We observe that A-LLMRec consistently outperforms other models in the cold user scenario, while SASRec struggles to perform well, especially on a large dataset, i.e., Movies and TV, due to the lack of collaborative knowledge from users. Moreover, LLM-based models demonstrate superior performance in handling cold users as text information becomes useful under cold scenarios.

5.2.4 Few-shot Training Scenario. To investigate the impact of unseen/new items on recommendation models, we conduct experiments on a few-shot training scenario where the number of users in the training set is extremely limited to only K users, i.e., K-shot (Table 6). Under this scenario, we expect the models to encounter a large amount of unseen/new items at the inference stage, which would make it hard to provide accurate recommendations. We have the following observations: **1)** A-LLMRec outperforms all other baselines under the few-shot scenario. Despite being trained with extremely small amount of users, A-LLMRec relies on CF-RecSys to capture the collaborative knowledge, which is combined with the textual knowledge of items, leading to superior performance in few-shot learning. **2)** A-LLMRec (SBERT) outperforms A-LLMRec,

implying again that using the text encoder to extract the joint text-collaborative knowledge is useful when items lack interactions. **3)** Under the few-shot scenario, LLM-based models outperform the CF-Resys, i.e., SASRec, due to the textual understanding of LLM, which helps extract information from the text of the unseen item, while CF-RecSys suffers from the lack of collaborative knowledge regarding unseen/new items.

Table 7: Results (Hit@1) on a cross-domain scenario (i.e., Pretrained: Movies and TV, Evaluation: Video Games).

	SASRec	MoRec	RECFORMER	TALLRec	A-LLMRec	A-LLMRec (SBERT)
Movies and TV → Video Games	0.0506	0.0624	0.0847	0.0785	0.0901	**0.1203**

5.2.5 Cross-domain Scenario. To further investigate the generalization ability of A-LLMRec, we evaluate the models on the cross-domain scenario, where the models are evaluated on datasets that have not been used for training (Table 7). Specifically, we pre-train the models on the Movies and TV dataset and perform evaluations on the Video Games dataset. We have the following observations: **1)** A-LLMRec outperforms all the baselines in the cross-domain scenario, and A-LLMRec (SBERT) particularly performs well. This is again attributed to the text encoder that becomes useful when collaborative information is lacking. **2)** SASRec underperforms modality-aware models and LLM-based models, indicating that using textual knowledge is crucial for the cross-domain scenario due to the lack of collaborative information.

5.3 Ablation Studies

In this section, we show ablation studies for our model. We mainly analyze the effect of each component in A-LLMRec regarding Stage-1 (Section 5.3.1) and Stage-2 (Section 5.3.2).

Table 8: Ablation studies on Stage-1 of A-LLMRec (Hit@1).

Ablation	Movies and TV	Beauty	Toys
A-LLMRec	**0.6237**	**0.5809**	**0.3336**
w/o $\mathcal{L}_{\text{matching}}$	0.5838	0.5548	0.3225
w/o $\mathcal{L}_{\text{item-recon}}$&$\mathcal{L}_{\text{text-recon}}$	0.5482	0.5327	0.3204
w/o $\mathcal{L}_{\text{rec}}$	0.6130	0.5523	0.1541
Freeze SBERT	0.6173	0.5565	0.1720

5.3.1 Effect of Components in Stage-1. This section presents the experimental results showing the benefit of each component during the Stage-1. Across all datasets, the exclusion of any loss resulted in decreased performance. We make the following observations: **1)** Removing $\mathcal{L}_{\text{matching}}$ from in Equation 2 results in a significant performance decline across all datasets. This implies that the alignment between the item and the text information is effective and that the LLM can comprehend item textual information in joint collaborative-text embeddings to enhance recommendation capabilities. **2)** Removing $\mathcal{L}_{\text{item-recon}}$ and $\mathcal{L}_{\text{text-recon}}$ leads to performance drop, owing to the risk of over-smoothed representations (i.e., $\mathbf{e} \approx \mathbf{q}$), as discussed in Section 4.1.1. **3)** We observe that removing $\mathcal{L}_{rec}$ leads to performance drop. Since $\mathcal{L}_{rec}$ is introduced to explicitly incorporate the collaborative knowledge while informing the model about the recommendation task, the performance drop

indicates the reduction of collaborative knowledge between items and users, which is crucial for recommendation tasks. **4)** Lastly, we kept SBERT frozen while training A-LLMRec. We observe that freezing SBERT leads to poor performance across all datasets. This implies that fine-tuning SBERT facilitates the text embeddings to adapt to the recommendation task.

Table 9: Ablation study on Stage-2 of A-LLMRec (Hit@1).

Row	Ablation	Movies and TV	Video Games	Beauty	Toys
(1)	A-LLMRec	**0.6237**	**0.5282**	**0.5809**	**0.3336**
(2)	A-LLMRec w/o user representation	0.5925	0.5121	0.5547	0.3217
(3)	A-LLMRec w/o joint embedding	0.1224	0.4773	0.5213	0.2831
(4)	A-LLMRec with random joint embedding	0.1200	0.4729	0.5427	0.0776

5.3.2 Effect of the Alignment method in Stage-2. Recall that a user representation and item embeddings are injected to the LLM prompt as shown in Figure 3. In this section, we verify the benefit of injecting them into the prompt (rows (2-4) in Table 9). We have the following observations: Across all datasets, **1)** the absences of either the user representation (row (2)) or the joint embedding (row (3)) from the prompt led to a reduction in performance. Notably, the exclusion of the joint embedding results in a more substantial decrease, underscoring its significant role in transferring collaborative knowledge. Moreover, as joint embeddings also capture the textual information about items, their exclusion is particularly detrimental. **2)** When we replace the joint embedding with a randomly initialized embedding (row (4)), which means A-LLMRec is trained with item embeddings without collaborative knowledge, we observe performance degradation across all datasets. This indicates the importance of leveraging the collaborative knowledge for recommendation.

5.4 Model Analysis

5.4.1 Train/Inference Speed. Recall that A-LLMRec requires the fine-tuning of neither the CF-RecSys nor the LLM. Specifically, A-LLMRec efficient in that the alignment network is the only trainable neural network, while TALLRec [2] requires the fine-tuning of the LLM with LoRA. In this section, we compare the training and the inference time of A-LLMRec and TALLRec. As for the training time, we measured the total time spent until the end of training, and as for the inference time, we measured the time spent per mini-batch. Table 10 shows that A-LLMRec exhibits significantly faster training and inference time compared with TALLRec. Notably, a more substantial improvement is observed in training time, since A-LLMRec does not require the LLM to be fine-tuned unlike TALLRec, which demonstrates the applicability of LLM in large-scale recommendation datasets. Moreover, the faster inference time demonstrates the practicality of A-LLMRec in real-world scenarios, especially in the context of real-time recommendation services where inference time is critically important.

5.4.2 Training with all items in each sequence. Recall that for efficiency in training, we used only the last item of each user sequence when optimizing the final loss in Stage-1 (Equation 6) and Stage-2 (Equation 8) of A-LLMRec. In this section, we report the recommendation performance in terms of Hit@1 and train/inference speed when using all items in each user sequence for optimization (see

Table 10: Train/Inference time comparison (Beauty dataset).

	Train time (min)	Inference time (sec/batch)	Hit@1
TALLRec	588.58	3.36	0.5542
A-LLMRec	**232.5**	**1.98**	0.5809
A-LLMRec$_{all}$	643.33	**1.98**	**0.6002**

Table 11: Results showing A-LLMRec is model-agnostic.

Model	Beauty	Toys
SASRec	0.5298	0.2359
A-LLMRec (SASRec)	**0.5809**	**0.3336**
NextItNet	0.4231	0.1415
A-LLMRec (NextItNet)	0.5642	0.3203
GRU4Rec	0.4131	0.1673
A-LLMRec (GRU4Rec)	0.5542	0.3089
NCF	0.2957	0.1849
A-LLMRec (NCF)	0.5431	0.3263

A-LLMRec$_{all}$ in Table 10). We observe that as expected the recommendation performance is further improved when using all items in each user sequence. However, considering that the training time also increased approximately 3 times, the improvement seems marginal. It is important to note that since vanilla A-LLMRec is trained based on only the last item in each user sequence, there is a large amount of unseen/new items that appear in the test set[4]. However, valilla A-LLMRec still showed comparable performance with A-LLMRec$_{all}$, implying the generalization ability of A-LLMRec.

5.4.3 A-LLMRec is Model-Agnostic. Although A-LLMRec adopts SASRec as the backbone CF-RecSys, it can be replaced with any existing collaborative filtering recommender systems, thanks to the model-agnostic property. Hence, we adopt three other collaborative filtering recommender systems including two sequential recommenders (i.e., NextItNet and GRU4Rec), and one non-sequential recommender (i.e., NCF) to A-LLMRec. We make the following observations from Table 11. **1)** Adopting the SASRec backbone performs the best, which is expected since SASRec outperforms other CF-RecSys in their vanilla versions. This implies that transferring high-quality collaborative knowledge can enhance the performance of A-LLMRec. **2)** Adopting A-LLMRec to any backbone improves the performance of the vanilla model. This implies that if the SOTA model changes in the future, our framework has the potential to further improve performance by replacing the existing CF-RecSys in the framework. **3)** We observe that while the performance difference between SASRec and NCF is nearly double when they operate as standalone CF-RecSys, the integration with A-LLMRec, which leverages the modality of item text information and the intensive capabilities of LLM, reduces this performance gap.

5.4.4 Beyond Recommendation: Language Generation Task (Favorite genre prediction). To validate whether A-LLMRec can generate natural language outputs based on the understanding of users and items through the aligned collaborative knowledge from CF-RecSys, we conduct a favorite genre prediction task (Figure 4). That is, given the same prompt format, we ask the LLM-based models (i.e., A-LLMRec and LLM-Only) using the same backbone LLM, which is OPT-6.7B, to predict the movie genres that a given user would enjoy

---

[4]About 13% of items are unseen during training in the Beauty dataset.
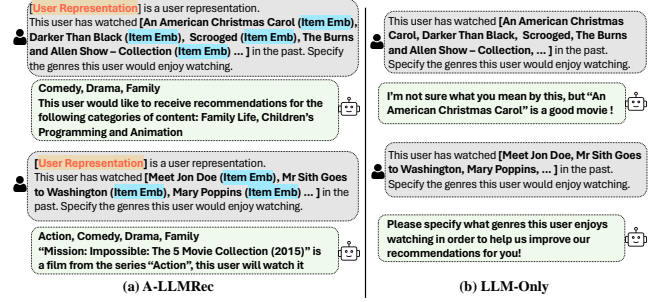
(a) A-LLMRec

(b) LLM-Only

Figure 4: A-LLMRec v.s. LLM-Only on the favorite genre prediction task (Movies and TV dataset used).

watching. The only difference in the prompt is that while LLM-only is only given titles of movies watched by the user in the past, A-LLMRec is given the user representation and item embeddings along with the movie titles. In Figure 4, we observe that A-LLMRec indeed generates proper answers, while LLM-Only fails to do so. We attribute this to the fact that the item embeddings of the CF-RecSys are well aligned with the token space of the LLM, which enables the LLM to understand and utilize collaborative knowledge. Note that although we also experimented with TALLRec, we were not able to obtain valid outputs. We conjecture that since the LLM in TALLRec is fine-tuned via an instruction-tuning process that makes the model provide responses as part of the recommendation task, generating valid natural language outputs has become a non-trivial task. Please refer to Appendix B for the results of TALLRec.

6 CONCLUSION

In this paper, we propose a novel LLM-based recommender system, named A-LLMRec. The main idea is to enable LLMs to utilize the collaborative knowledge from pre-trained CF-RecSys. By doing so, A-LLMRec outperforms existing CF-RecSys, modality-aware recommender systems, and LLM-based recommenders under various scenarios including cold/warm items, cold user, few-shot, and cross-domain scenarios. Moreover, we also demonstrate that the two advantages originated from fine-tuning neither pre-trained CF-RecSys nor LLMs, i,e, Model-agnostic and efficiency. Lastly, we show the potential of A-LLMRec in generating natural language tasks based on the understanding of collaborative knowledge from CF-RecSys. For future work, we plan to further enhance the ability of the LLM in A-LLMRec based on advanced prompt engineering such as chain-of-thought prompting [46].

Ethics Statement To the best of our knowledge, this paper aligns with the KDD Code of Ethics without any ethical concerns. The datasets and codes employed in our research are publicly available.

ACKNOWLEDGMENTS

REFERENCES

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling Popularity Bias in Learning-to-Rank Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems* (Como, Italy) *(RecSys '17)*. Association for Computing Machinery, New York, NY, USA, 42–46. https://doi.org/10.1145/3109859.3109912

[2] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. *arXiv preprint arXiv:2305.00447* (2023).

[3] Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* (2020).

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[5] Allison JB Chaney, David M Blei, and Tina Eliassi-Rad. 2015. A probabilistic model for using social networks in personalized item recommendation. In *Proceedings of the 9th ACM Conference on Recommender Systems*. 43–50.

[6] Jiawei Chen, Hande Dong, Yang Qiu, Xiangnan He, Xin Xin, Liang Chen, Guli Lin, and Keping Yang. 2021. AutoDebias: Learning to Debias for Recommendation *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 21–30. https://doi.org/10.1145/3404835.3462919

[7] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Where you like to go next: successive point-of-interest recommendation. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence* (Beijing, China) *(IJCAI '13)*. AAAI Press, 2605–2611.

[8] Robert G. Cooper and Scott J. Edgett. 2012. Best Practices in the Idea-to-Launch Process and Its Governance. *Research Technology Management* 55, 2 (2012), 43–54. https://www.jstor.org/stable/26586220

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

[11] Xiaoyu Du, Zike Wu, Fuli Feng, Xiangnan He, and Jinhui Tang. 2022. Invariant Representation Learning for Multimedia Recommendation. In *Proceedings of the 30th ACM International Conference on Multimedia* (<conf-loc>, <city>Lisboa</city>, <country>Portugal</country>, </conf-loc>) *(MM '22)*. Association for Computing Machinery, New York, NY, USA, 619–628. https://doi.org/10.1145/3503161.3548405

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

[13] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *Proceedings of the 25th International Conference on World Wide Web* (Montréal, Québec, Canada) *(WWW '16)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 507–517. https://doi.org/10.1145/2872427.2883037

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

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[16] Zhankui He, Zhouhang Xie, Rahul Jha, Harald Steck, Dawen Liang, Yesu Feng, Bodhisattwa Prasad Majumder, Nathan Kallus, and Julian McAuley. 2023. Large language models as zero-shot conversational recommenders. In *Proceedings of the 32nd ACM international conference on information and knowledge management*. 720–730.

[17] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[18] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

[19] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*. Ieee, 263–272.

[20] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[21] Sein Kim, Namkyeong Lee, Donghyun Kim, Minchul Yang, and Chanyoung Park. 2023. Task Relation-aware Continual User Representation Learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)*. Association for Computing Machinery, New York, NY, USA, 1107–1119. https://doi.org/10.1145/3580305.3599516

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

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc.

[24] Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. 2023. Text Is All You Need: Learning Language Representations for Sequential Recommendation *(KDD '23)*. Association for Computing Machinery, New York, NY, USA, 1258–1267. https://doi.org/10.1145/3580305.3599519

[25] Xiangyang Li, Bo Chen, Lu Hou, and Ruiming Tang. 2023. CTRL: Connect Tabular and Language Model for CTR Prediction. *arXiv preprint arXiv:2306.02841* (2023).

[26] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

[27] Chang Liu, Xiaoguang Li, Guohao Cai, Zhenhua Dong, Hong Zhu, and Lifeng Shang. 2021. Noninvasive self-attention for side information fusion in sequential recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4249–4256.

[28] Fan Liu, Huilin Chen, Zhiyong Cheng, Anan Liu, Liqiang Nie, and Mohan Kankanhalli. 2022. Disentangled multimodal representation learning for recommendation. *IEEE Transactions on Multimedia* (2022).

[29] Zhuang Liu, Yunpu Ma, Matthias Schubert, Yuanxin Ouyang, and Zhang Xiong. 2022. Multi-Modal Contrastive Pre-training for Recommendation. In *Proceedings of the 2022 International Conference on Multimedia Retrieval* (Newark, NJ, USA) *(ICMR '22)*. Association for Computing Machinery, New York, NY, USA, 99–108. https://doi.org/10.1145/3512527.3531378

[30] Chih-Chao Ma. 2008. A guide to singular value decomposition for collaborative filtering. *Computer (Long Beach, CA)* 2008 (2008), 1–14.

[31] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes *(SIGIR '15)*. Association for Computing Machinery, New York, NY, USA, 43–52. https://doi.org/10.1145/2766462.2767755

[32] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 43–52.

[33] Andriy Mnih and Russ R Salakhutdinov. 2007. Probabilistic matrix factorization. *Advances in neural information processing systems* 20 (2007).

[34] Yunhak Oh, Sukwon Yun, Dongmin Hyun, Sein Kim, and Chanyoung Park. 2023. MUSE: Music Recommender System with Shuffle Play Recommendation Enhancement. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23)*. Association for Computing Machinery, New York, NY, USA, 1928–1938. https://doi.org/10.1145/3583780.3614976

[35] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).

[36] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web* (Raleigh, North Carolina, USA) *(WWW '10)*. Association for Computing Machinery, New York, NY, USA, 811–820. https://doi.org/10.1145/1772690.1772773

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

[38] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[39] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*. 111–112.

[40] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[41] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 565–573.

[42] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[43] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing Cold Start in Recommender Systems. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/dbd22ba3bd0df8f385bdac3e9f8be207-Paper.pdf

[44] Lei Wang and Ee-Peng Lim. 2023. Zero-Shot Next-Item Recommendation using Large Pretrained Language Models. *arXiv preprint arXiv:2304.03153* (2023).

[45] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682* (2022).

[46] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

[47] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, Richang Hong, and Tat-Seng Chua. 2019. MMGCN: Multi-modal Graph Convolution Network for Personalized Recommendation of Micro-video *(MM '19)*. Association for Computing Machinery, New York, NY, USA, 1437–1445. https://doi.org/10.1145/3343031.3351034

[48] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2023. A Survey on Large Language Models for Recommendation. *arXiv preprint arXiv:2305.19860* (2023).

[49] Jieyu Yang, Liang Zhang, Yong He, Ke Ding, Zhaoxin Huan, Xiaolu Zhang, and Linjian Mo. 2023. DCBT: A Simple But Effective Way for Unified Warm and Cold Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 3369–3373. https://doi.org/10.1145/3539618.3591856

[50] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the twelfth ACM international conference on web search and data mining*. 582–590.

[51] Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. 2023. Where to Go Next for Recommender Systems? ID- vs. Modality-based Recommender Models Revisited *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 2639–2649. https://doi.org/10.1145/3539618.3591932

[52] Sukwon Yun, Kibum Kim, Kanghoon Yoon, and Chanyoung Park. 2022. Lte4g: Long-tail experts for graph neural networks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2434–2443.

[53] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).

[54] Xun Zhou, Jing He, Guangyan Huang, and Yanchun Zhang. 2015. SVD-based incremental approaches for recommender systems. *J. Comput. System Sci.* 81, 4 (2015), 717–733.
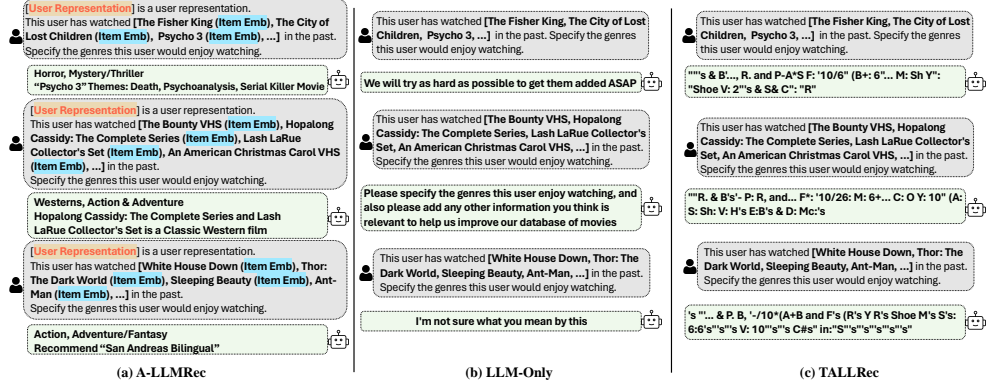
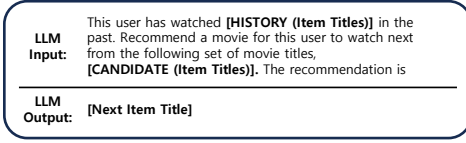**Figure 5: A-LLMRec, LLM-Only, and TALLRec on the favorite genre prediction task (Movies and TV dataset used).**

Figure 6: An example prompt designed for the Amazon Movies dataset used by LLM-based models, i.e., TALLRec and LLM-Only models.

Table 12: Source code links of the baseline methods.

Methods	Source code
SASRec	https://github.com/pmixer/SASRec.pytorch
NextItNet	https://github.com/syiswell/NextItNet-Pytorch
GRU4Rec	https://github.com/hungpthanh/GRU4REC-pytorch
RECFORMER	https://github.com/AaronHeee/RecFormer
TALLRec	https://github.com/SAI990323/TALLRec
A-LLMRec	https://github.com/ghdtjr/A-LLMRec

A BASELINES

(1) Collaborative filtering recommender systems
- **NCF** [15] combines neural networks (MLP) to capture the collaborative information. Note that NCF is a two-tower model comprised of separate components for the user and item embedding matrix.
- **NextItNet** [50] proposes a temporal convolutional network that utilizes 1D-dilated convolutional layers and residual connections to capture the long-term dependencies inherent in interaction sequence.
- **GRU4Rec** [17] adopts RNNs to model user behavior sequences for session-based recommendations.
- **SASRec** [20] is our main baseline, a state-of-the-art collaborative filtering recommender system (CF-RecSys) that adopts a self-attention encoding method to model user preferences from user behavior sequences.

(2) Modality-aware recommender systems
- **MoRec** [51] employs a pre-trained SBERT to utilize the text information of items to generate the initial embeddings for items that will be used in collaborative filtering models. We utilize SASRec as the backbone model of MoRec.
- **CTRL** [25] employs a two-stage learning process: the first stage involves contrastive learning on textual information of items to initialize the backbone model, and the second stage, fine-tunes the model on recommendation tasks. We use SASRec as the backbone model of CTRL.

- **RECFORMER** [24] models user preferences and item features using the Transformer architecture, transforming sequential recommendation into a task of predicting the next item as if predicting the next sentence, by converting item attributes into a sentence format.

(3) LLM-based recommender systems
- **LLM-Only** utilizes an open-source LLM model OPT [53] with prompts related to recommendation tasks as shown in Figure 6. In our experiments, we adopt the 6.7B size version of OPT for all LLM-based recommendations.
- **TALLRec** [2] is our main baseline, which learns the recommendation task based on prompts consisting solely of text and fine-tunes the LLMs using the LoRA. Their approach involves providing user interaction history and one target item and determining whether a user will prefer this target item. This simpler task necessitates only a brief prompt for the LLMs. In contrast, our recommendation task requires a more extensive prompt. Even though this adjustment results in a smaller batch size, the same as A-LLMRec, for training TALLRec. We use the prompt shown in Figure 6.
- **MLP-LLM** is an additionally designed LLM-based recommendation model for analysis. Compared with A-LLMRec, this model directly connects the user and item embeddings from frozen CF-RecSys and LLM using only MLP layers, instead of the auto-encoders in A-LLMRec that involve various techniques to align the collaborative knowledge of CF-RecSys with the LLM. Note that we use the prompt shown in Figure 3.

B LANGUAGE GENERATION TASK

In Figure 5, we present additional favorite genre prediction task results for experiment in shown in Section 5.4.4. As mentioned in Section 5.4.4, TALLRec could not generate valid natural language outputs due to the fine-tuning via instruction tuning process, which makes the LLM of TALLRec being able to answer only with some particular prompts used in instruction tuning process. The additional results indicate that A-LLMRec can generate the favorite genres for the users based on the understanding of the aligned user representation and item embeddings while LLM-only fails to do so.

C REPRODUCIBILITY

For implementing the baseline, we followed the official codes published by authors as detailed in Table 12. Refer to our source code and instructions to run code for reproducing the results reported in the experiments.