Knowledge Adaptation from Large Language Model to Recommendation for Practical Industrial Application

Jian Jia*
jian.jia@outlook.com
Kuaishou Technology
Beijing, China

Yipei Wang*
220213711@seu.edu.cn
Southeast University
Nanjing, China

Yan Li
yan.li@cripac.ia.ac.cn
Kuaishou Technology
Beijing, China

Honggang Chen
yuanhong.chg@gmail.com
Kuaishou Technology
Beijing, China

Xuehan Bai
baixuehanmiao@gmail.com
Kuaishou Technology
Beijing, China

Zhaocheng Liu
lio.h.zen@gmail.com
Kuaishou Technology
Beijing, China

Jian Liang
liangjianzb12@gmail.com
Kuaishou Technology
Beijing, China

Quan Chen
myctllmail@163.com
Kuaishou Technology
Beijing, China

Han Li
lee.han06@gmail.com
Kuaishou Technology
Beijing, China

Peng Jiang
jp2006@139.com
Kuaishou Technology
Beijing, China

Kun Gai
gai.kun@qq.com
Unaffliate
Beijing, China

ABSTRACT

Contemporary recommender systems predominantly rely on collaborative filtering techniques, employing ID-embedding to capture latent associations among users and items. However, this approach overlooks the wealth of semantic information embedded within textual descriptions of items, leading to suboptimal performance in cold-start scenarios and long-tail user recommendations. Leveraging the capabilities of Large Language Models (LLMs) pretrained on massive text corpus presents a promising avenue for enhancing recommender systems by integrating open-world domain knowledge. In this paper, we propose an Llm-driven knowlEdge Adaptive RecommeNdation (LEARN) framework that synergizes open-world knowledge with collaborative knowledge. We address computational complexity concerns by utilizing pretrained LLMs as item encoders and freezing LLM parameters to avoid catastrophic forgetting and preserve open-world knowledge. To bridge the gap between the open-world and collaborative domains, we design a twin-tower structure supervised by the recommendation task and tailored for practical industrial application. Through offline experiments on the large-scale industrial dataset and online experiments on A/B tests, we demonstrate the efficacy of our approach.

*Both authors contributed equally to this research.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Large-Language Model, Sequential Recommendation, Recommendation

ACM Reference Format:
Jian Jia, Yipei Wang, Yan Li, Honggang Chen, Xuehan Bai, Zhaocheng Liu, Jian Liang, Quan Chen, Han Li, Peng Jiang, and Kun Gai. 2018. Knowledge Adaptation from Large Language Model to Recommendation for Practical Industrial Application. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Recommendation systems (RS) have become indispensable tools in various online platforms, facilitating personalized recommendations to users amidst the overwhelming abundance of content. Contemporary RS represents user and item features by distinct ID-embedding, which is learned based on the interactions between users and items and is effective in capturing latent associations among users and items. However, ID-embedding based RS overlooks the wealth of semantic information contained in textual descriptions of items. Consequently, the acquired recommendation model struggles to generalize to unseen data, resulting in suboptimal performance, particularly in cold-start scenarios and long-tail user recommendations. Thus, to provide incremental information to the current RS, how to explore the content descriptions of items attracts attention from academics and industry.

With the attention towards the power demonstrated by LLM in natural language processing (NLP) [1, 5, 34], *e.g.*, rich open-world

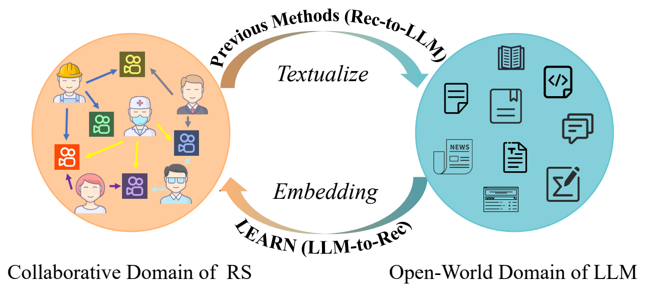

Figure 1: Illustration of proposed LEARN framework and previous "Rec-to-LLM" methods. Previous methods textualize the recommendation data to natural language conversations, which are the input of LLM to get the text predictions. In contrast, our method LEARN transforms the text information of items into embedding, which is projected into the collaborative domain to achieve alignment with industry recommendation tasks.

domain knowledge or great logical reasoning ability, etc., it shows a novel path for the development of recommender systems, that is to get recommender systems to embrace LLM and leverage the open-world domain knowledge of LLM [9, 23]. In this work, we aim to explore the learning of user and item embeddings for recommender systems from LLM-based content representations. Owing to pretraining on extensive general textual corpora, content embeddings derived from Large Language Models (LLMs) by comprehending and reasoning about textual descriptions are categorized within the general knowledge domain. In contrast, ID embeddings learned from RS fall within the collaborative knowledge domain. Therefore, bridging the gap between the general and recommendation-specific domains is crucial. This enables the leveraging of LLM's open-world knowledge to provide a substantial informational increment to existing RS, enhancing their effectiveness and generalization.

Prior research on integrating LLMs with RS can be broadly categorized into two approaches. The first approach freezes the LLM parameters and adapts the recommendation domain data into conversational formats. These methods leverage the LLM's comprehension and reasoning capabilities, transforming user-item interaction data into textual sentences. This data is then processed by the LLM using various prompts to perform tasks like generating top-K recommendations, sequence recommendations, and explanations for recommendations [11, 18, 25, 27, 31, 44]. The second approach fine-tunes the LLM on specially prepared textual datasets from the recommendation domain [2, 3, 22, 24, 41]. By utilizing prompt design, these methods enable the LLM to learn latent relationships between users and items through the next token prediction task.

Both discussed methods convert user-item interactions from the recommendation domain into the textual data format of the open-world domain, as shown in Fig. 1. These methods utilize the LLM to deliver personalized recommendations, aligning the input organization and target tasks with those of the LLM's pretraining stage. We define this process as "Rec-to-LLM adaptation", a form of domain adaptation from the recommendation domain (target domain) to the open-world domain of LLMs (source domain). However, our empirical investigations reveal that the "Rec-to-LLM" adaptation fails to yield practical benefits in real-world industry applications. This inefficacy can be attributed to the inherent shortcomings of this approach, which include the following key aspects:

1) **Inference LLM or finetuning LLM with the user history interactions is impractical in industry scenarios**. Handling a user interaction sequence that comprises hundreds or thousands of items annually poses significant computational challenges. Given the input constraints of LLMs limited to 2K or 4K tokens [1, 34], accommodating such extensive data within a single input is impractical. Moreover, the computational complexity of LLMs increases quadratically with the length of the input sequence. Therefore, handling the global history interactions of users with LLM poses significant computational burdens.

2) **Finetuning LLM for recommendation systems often leads to catastrophic forgetting and suboptimal performance.** Despite computational feasibility, finetuning Large Language Models (LLMs) with full parameters on extensive recommendation data typically results in catastrophic forgetting of open-world knowledge, leading to suboptimal outcomes. Even partial parameter tuning like LoRA [19] often fails to achieve satisfactory performance. Two primary issues contribute to these challenges. One is the domain gap. There is a profound gap between the collaborative knowledge required in the recommendation domain and the general open-world knowledge held by LLMs, as shown in Fig. 1. Research in continual learning indicates that when pretrained LLMs are adapted for specific tasks, they frequently suffer significant performance degradation in their original knowledge domains [32]. This results in severe catastrophic forgetting during extensive finetuning with full parameters. The other is the misalignment of training objectives. LLMs are generally pretrained to focus on the next token prediction, a task that develops a broad comprehension of general knowledge from large text corpora. In contrast, finetuning for recommendation systems emphasizes retrieval-oriented tasks that heavily depend on collaborative signals from user-item interactions. Both issues hamper our efforts to harness the LLM's open-world knowledge to augment the existing RS with valuable incremental information.

To overcome the noted limitations, we introduce the Llm-driven knowlEdge Adaptive RecommeNdation (LEARN) approach, designed to synergize the open-world knowledge of LLMs with the collaborative knowledge of recommendation systems. Contrary to previous methods following "Rec-to-LLM" adaptation, our approach adapts knowledge from LLM to recommendation (LLM-to-Rec). We employ the LLM as a content extractor, with the recommendation task serving as the training target. This strategy not only facilitates a seamless transition from the open-world domain of the LLM to the collaborative domain of RS but also ensures a better alignment with the practical needs of industrial online RS. The distinct advantages of our method compared to previous "Rec-to-LLM" approaches are depicted in Fig. 1.

Specifically, the proposed LEARN framework adopts a twin-tower structure: user and item tower. Both towers consist of the Content-Embedding Generation (CEG) and Preference CompreHension (PCH) modules. To address the computational challenges associated with processing extensive user history interaction, the CEG module employs the pretrained LLM (Baichuan2-7B [40]) as an item encoder rather than as a user preference encoder to reduce computational overhead. This LLM extracts content embeddings from item descriptions, including title, category, caption, price, and attributes. To prevent catastrophic forgetting of open-world knowledge, we freeze the LLM during the training stage. Furthermore, to bridge the domain gap between open-world and collaborative knowledge as shown in Fig 1, we designed the PCH module and adopted the self-supervised training target of the recommendation task to guide model optimization. The PCH module utilizes a sequence of content embeddings from items that a user has interacted with as input and generates a user embedding as output for RS. Self-supervised contrastive learning is adopted as the training target to effectively enhance the discrimination of the model in distinguishing between preferred and non-preferred items.

To validate the effectiveness of the proposed method, we build a large-scale recommendation dataset collected from the real recommendation scenario. Our data contains over 31 million items interacted with by 12 million users over a 10-month period. We deploy our method on the real recommendation system and validate the efficacy of our approach in A/B experiments, which shows substantial improvements. We also achieve state-of-the-art (SOTA) performance on the public large-scale dataset Amazon Reviews (Books), surpassing the HSTU [43] approach proposed by Meta.

We conclude our contributions as follows:

- We propose the Llm-driven knowlEdge Adaptive RecommeNdation (LEARN) framework to efficiently aggregate the open-world knowledge encapsulated within LLMs into RS.
- We propose the CEG and PCH modules to solve the catastrophic forgetting of open-world knowledge and bridge the domain gap between open-world and collaborative knowledge.
- We verify the effectiveness of our method by achieving SOTA performance on the large-scale practical industrial scenario and the public recommendation dataset. To the best of our knowledge, we are pioneers in successfully deploying LLM in industrial recommendation scenarios and achieving profitability.

2 RELATE WORK

2.1 Content-based Recommendation.

Traditional RS predominantly rely on ID-based embeddings, which frequently suffer from limited generalizability. To address this, extensive research has focused on deepening the understanding of user and item content to bring incremental information and enhance the generalization capabilities for online RS. Wu *et al.* developed the large-scale MIND text dataset [38] specifically for news recommendation, advancing research into the impact of text content understanding on recommendation systems. Following this, various studies have leveraged the BERT [7] model to improve content understanding. Examples include UNBERT [45] and PLM-NR [37], which fully fine-tune BERT. Additionally, TBIN [4] utilizes pretrained BERT to encode textual descriptions within user behavior data. Beyond text, MoRec [41] and MISSRec [35] incorporate visual content from item images using ResNet [14] and ViT [8] for sequential modeling. With the recognized scaling laws and emergent behaviors of LLMs, LLM2BERT4Rec [13] substitutes BERT4REC's item ID embedding with LLM embeddings processed through PCA for dimension reduction.

2.2 LLM-based Recommendation.

Due to the powerful capabilities demonstrated by Large Language Models (LLMs) in understanding textual content and reasoning with common sense, an increasing number of studies are exploring the integration of LLMs into recommender systems (RS) [10, 23]. Prior research can be categorized into two types, and both types adapt the recommendation domain data into conversational formats. The first type freezes the LLM parameters and takes the LLM as a recommender (LLM-as-Rec). Some works [11, 18, 25, 39, 44, 46] proposed the task-specific prompt to construct recommendation dialogues and used ChatGPT to generate the candidates. Sanner *et al.* [31] adopted the PaLM [6] as a cold-start recommender. RLM-Rec [30] utilized ChatGPT to generate user/item profiles. RecMind [36] proposed an LLM-powered autonomous agent for various recommendation tasks. The second type fine-tunes the LLM on specially prepared textual datasets from the recommendation domain. GPT4Rec [21] combined the GPT-2 with BM25 search engine to predict multiple candidatea and retrieval top-k items for each candidate. LlamaRec [42] took the item title as the textual data and optimized the LLM by the ranking scores. TALLRec [3] proposed a two-stage tuning framework and finetunes the LLM with LoRA [19] for few-shot recommendation. LLaRA [22] combined the LLM prompt with ID embeddings to align the LLM and the well-established sequential recommenders. ReLLa [24] proposed a retrieval-enhanced instruction tuning method and finetuned the Vicuna [5] on a mixed dataset.

3 METHOD

In this section, we first verify the discrimination of item content embedding generated by LLM and confirm the feasibility of content embedding applied to recommendation system (RS). Then, we introduce the proposed Llm-driven knowlEdge Adaptive RecommeNdation (LEARN) framework following a twin-tower structure.

3.1 Content Embedding of LLM

Integrating content-based embeddings into existing RS is a key focus in both academic and industrial research, given the advanced content comprehension and reasoning capabilities of LLMs. By processing textual inputs with carefully constructed prompts, LLMs can summarize user interests [16], extract product information keywords [33], and identify related products [21]. However, this textual information must be transformed into embeddings to be utilized in industrial RS. We propose using the output embeddings from LLM as representations of input textual information, rather than using output words predicted by LLM. Given the limited research on the effectiveness of LLM-generated embeddings, particularly with models primarily using decoder architectures, we devise an item-to-item (I2I) retrieval experiment. This experiment involves a gallery set of 1 million items to assess the utility of LLM embeddings.

We first collect the content description of items from the online platform. An item is characterized by up to 6 aspects: title, category, brand, price, keywords, and attribute. A simple prompt is constructed to concatenate the 6 types of information into one content-descriptive sentence, as shown in Fig. 4. Then, the content sentence is tokenized into discrete indices and sent to LLM (Baichuan2-7B [1]). The output embeddings of token indices are

	Query	Top 1	Top 2	Top 3
Title:	Envy Versatile Five-Petal Flower Earrings 357	Envy New Style Versatile Elegant Earrings 242	Envy Simple Double-Wear Fringe Earrings 351 Earrings	Envy Snowflake Ear Thread Earrings 203 Earrings
Category:	Earrings	Earrings	Earrings	Earrings
Brand:	Envy	Envy	Envy	Envy
Price:	9.9 yuan	6.9 yuan	5.99 yuan	9.8 yuan
Keywords:	Earrings, Five-Petal Flower Earrings	Earrings	Fringe Earrings, Earrings	Earrings, Ear Thread, Earrings
Attribute:	Returns for Damaged Packages	Returns for Damaged Packages	Returns for Damaged Packages	Returns for Damaged Packages

(a) Retrieval results of an "Earring".

	Query	Top 1	Top 2	Top 3
Title:	Children's Digital Building Blocks (1-5 Years Early Education and Intelligence)	Children's Educational Toy Intelligence Digital Building Blocks Assembly Puzzle Baby Brain Development Boy Girl Early Education	Infant Toddler Educational Developmental Puzzle Multifunctional Early Learning Building Blocks 1-2-3 Years Old Boys Girls Baby Toys	Educational Shape and Number Learning Box
Category:	Building Block Puzzle	Building Block Puzzle	Building Block Puzzle	Other Early Education
Brand:	——	Wooden Ball	Wooden Ball	——
Price:	39.0 yuan	27.9 yuan	19.9 yuan	16.9 yuan
Keywords:	Digital Building Blocks, Building Blocks	Toy, Children's Toy	Baby Toys, Building Blocks	Number Learning Box, Learning Box
Attribute:	Returns for Damaged Packages , Flash Sale/Welfare Purchase, Pre-sale, Buy Now Pay Later	Returns for Damaged Packages, Buy Now Pay Later	Returns for Damaged Packages, Buy Now Pay Later	Returns for Damaged Packages, Buy Now Pay Later

(b) Retrieval results of a "Building Block Puzzle".

Figure 2: Results of ANN retrieval based on LLM content embedding. The query and retrieved top 3 items are listed. Results show that the LLM embeddings are discriminative.

average-pooled into one content embedding. We believe that this embedding encapsulates the understanding and reasoning of LLM regarding the textual information about item content. Finally, we normalize the generated content embedding for an approximate nearest neighbor (ANN) retrieval. The results in Figure 2 indicate that the LLM-generated embedding is discriminative and semantically aligned with the item's content. While we also test using the last token embedding as the content representation, average pooling yields superior retrieval performance and is therefore adopted as our default implementation.

3.2 Model Architecture

Our goal is to learn a recommendation model that understands the user's interests from the history interaction and predicts the next item the user is interested in. We construct history interactions U^{hist} and target interactions U^{tar} in chronological order. The user's chronological sequence of interactions is divided into two parts according to a certain timestamp, the first part as the history interaction sequence and the second part as the target sequence. The history and target interactions of the i-user are denoted as $U_i^{hist} = \{Item_{i,1}, Item_{i,2}, \ldots, Item_{i,H}\}$ and $U_i^{tar} = \{Item_{i,H+1}, Item_{i,H+2}, \ldots, Item_{i,H+T}\}$. H and T denote the length of history and target interactions, respectively. The subscripts of user index are ignored in the following sections for simplicity.

3.2.1 *User Tower.*

The user tower consists of a Content-Embedding Generation (CEG) module and a Preference CompreHension (PCH) Module, as shown in Fig. 4(a) and Fig. 4(b).

Our CEG module employs a pretrained LLM as the backbone network where the parameters are frozen during training. We discard the linear classifier layer (lm_head) and instead use the hidden states from the final decoder layer to generate token embeddings. The LLM processes textual descriptions by tokenizing the input sentences and producing token embeddings. The textual description for each item is detailed in Fig. 4 and referred to as $Item^T$. The prompt is designed concisely to verify the effects of the content information. An average pooling layer is applied to aggregate all token embeddings into the content embedding emb^c for each item, as shown in Fig. 4(a). All items share the same CEG module.

The Preference CompreHension (PCH) module generates user embeddings that reflect user preferences based on the content embeddings of each item. For user history interaction sequence U^{hist}, we convert each item's textual description into a content embedding E^c, forming an embedding sequence $\hat{U}^{hist} = E^c_0, E^c_1, \ldots, E^c_H$. The PCH module utilizes this sequence as input, starting with a content projection layer for dimension transformation. Subsequently, a 12-layer transformer, similar to the BERT-base [7] model, serves as the backbone network. This transformer is specifically designed to learn implicit item relationships and model user preferences. Unlike bidirectional attention in BERT [7], which considers all items regardless of their sequence, our module employs the causal attention mechanism. This allows the PCH to focus attention on past items in the sequence, aligning with the chronological nature of user preferences. The output embeddings from our transformer are further processed through linear layers to produce user embeddings $E^{user} \in R^{64}$, which are directly utilized in our online recommendation system (detailed in Sec.4.4). These user embeddings are optimized towards the recommendation task, as outlined in Sec.3.2.3.

3.2.2 *Item Tower.*

The item tower processes textual descriptions of item content and outputs item embedding specifically tailored for the recommendation domain. As depicted in Fig. 3, we propose three model variants of the item tower.

Variant 1. As illustrated in Fig. 3(a), the item tower adopts the same architecture and model weights as the user tower. However, instead of processing user history interactions U^{hist}, the item tower utilizes user target interactions U^{tar} as inputs. This approach enhances the alignment between user and item embeddings by employing the same causal attention mechanism used in the user tower, effectively improving the relevance of recommendations. Due to the superior performance in Tab. 4, Variant 1 is adopted as the default setting.

Variant 2. Contrasting with the causal attention mechanism used in Variant 1, this variant, depicted in Fig. 3(b), employs a self-attention mechanism where each item attends to its own content exclusively. This approach treats the content information of each item as an independent input. Although Variant 2 also achieves the projection from the content domain to the collaborative domain, its performance is not as well as Variant 1, which utilizes the causal attention mechanism.

Variant 3. Both Variant 1 and Variant 2 are designed to project the content embeddings from the open-world domain of LLMs into

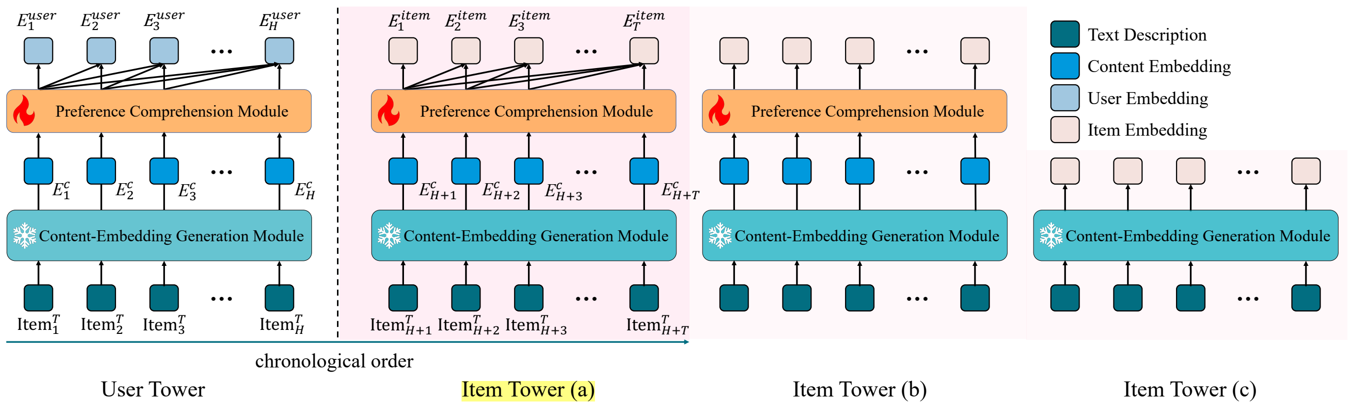
E_1^{user} E_2^{user} E_3^{user} E_H^{user}
Preference Comprehension Module
E_1^c E_2^c E_3^c E_H^c
Content-Embedding Generation Module
Item_1^T Item_2^T Item_3^T Item_H^T
E_1^{item} E_2^{item} E_3^{item} E_T^{item}
E_{H+1}^c E_{H+2}^c E_{H+3}^c E_{H+T}^c
Item_{H+1}^T Item_{H+2}^T Item_{H+3}^T Item_{H+T}^T
chronological order
User Tower
Item Tower (a)
Item Tower (b)
Item Tower (c)
Text Description
Content Embedding
User Embedding
Item Embedding

Figure 3: Illustration of our Llm-driven knowlEdge Adaptive RecommeNdation (LEARN) framework. The LEARN framework employs a twin-tower architecture comprising a user tower and an item tower. The user tower processes history interactions to generate user embeddings E^{user}, while the item tower handles target interactions to produce item embeddings E^{item}. User Tower and Item Tower (a) employ the causal attention mechanism, focusing solely on past items to model sequential dependencies. Item Tower (b) utilizes a self-attention mechanism that concentrates exclusively on the item itself. Item Tower (c) operates without the PCH module, directly using the content embedding as the item embedding.

the collaborative domain of the RS. In contrast, Variant 3 takes a distinct approach by directly using the content embedding E^c, as the item embedding E^{item}, for the recommendation task, bypassing the alignment process.

In the training stage, Variant 1 takes the user target interaction sequence as input, while Variant 2 and Variant 3 take an individual item independently. In the inference stage, all variants only take text descriptions of one item as the input and output item embedding independently.

3.2.3 Training Target of Recommendation Domain. To bridge the gap between the content embedding in the open-world domain of LLM and user (item) embedding in the collaborative domain of RS, we adopt the recommendation task as the training target.

In the online RS, the ranking model computes similarities between the user embedding and embeddings of all items in the gallery. The top K items with the highest similarity scores are identified as those of interest to the user. To align with the goals of the online RS, we utilize the self-supervised contrastive learning mechanism to model user preferences. This approach is designed to maximize the similarity between the user embedding and the item embeddings of interest, while minimizing the similarities from other items.

We adopt the InfoNCE loss [28] to guide the LEARN framework training. The user embedding, E^{user}, is derived from embedding sequence $\hat{U}^{hist}$ of user history interactions through the user tower, whereas the item embedding, E^{item}, is obtained from user target interactions through the item tower. Positive sample pairs consist of the user embedding and the item embedding from the same user's target interactions, whereas negative pairs comprise the user embedding and item embeddings from different users. The loss function is formulated as follows:

$$\mathcal{L} = -\sum_{i=1}^{N_u}\sum_{j=1}^{H}\sum_{k=1}^{T} \log \frac{e^{s(E_{i,j}^{user}, E_{i,k}^{item})}}{e^{s(E_{i,j}^{user}, E_{i,k}^{item})} + \sum_{z\neq i}\sum_k e^{s(E_{i,j}^{user}, E_{z,k}^{item})}}, \tag{1}$$

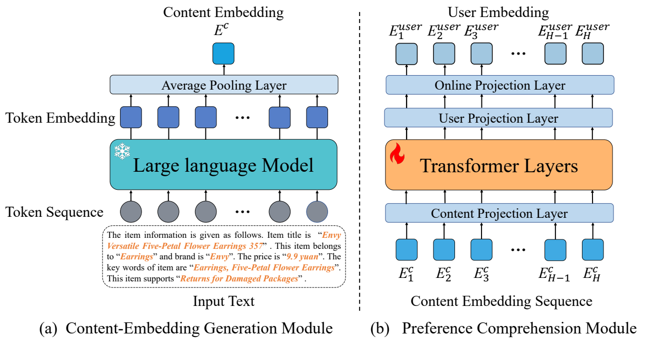
Content Embedding
E^c
Average Pooling Layer
Token Embedding
Large language Model
Token Sequence
The item information is given as follows. Item title is "Envy Versatile Five-Petal Flower Earrings 357" . This item belongs to "Earrings" and brand is "Envy". The price is "9.9 yuan". The key words of item are "Earrings, Five-Petal Flower Earrings". This item supports "Returns for Damaged Packages" .
Input Text
(a) Content-Embedding Generation Module
User Embedding
E_1^{user} E_2^{user} E_3^{user} E_{H-1}^{user} E_H^{user}
Online Projection Layer
User Projection Layer
Transformer Layers
Content Projection Layer
E_1^c E_2^c E_3^c E_{H-1}^c E_H^c
Content Embedding Sequence
(b) Preference Comprehension Module

Figure 4: Illustration of the Content-Embedding Generation (CEG) module and Preference CompreHension (PCH) module. The CEG module utilizes a pretrained Large Language Model (LLM) as the backbone network to generate content embeddings from item text descriptions. Subsequently, the Preference CompreHension (PCH) module takes these content embeddings and projects them from the open-world domain into the collaborative domain embeddings used in the online Recommender System (RS).

where $E_{i,j}^{user}$ denotes the j-th embedding of the i-th user generated from the user history interaction of length H and $E_{i,k}^{item}$ denotes the k-th embedding of the i-th user from the target interaction of length T. N_u represents the total number of users in the training set. As depicted in Fig. 3, given the interaction lengths H and T, we generate H user embeddings and T item embeddings, respectively.

For an individual user, we construct $N_{hist} \times N_{tar}$ positive pairs and $N_{hist} \times (bs-1) \times N_{tar}$ negative pairs, where bs is the batch size. The loss function, Eq. 1, is applied for each positive pair as the dense all-action loss [29].

4 EXPERIMENTS

In this section, we first present a large-scale offline dataset collected from the real industry scenario. Then, we conduct various experiments on the proposed dataset to explore ways to effectively incorporate LLM into the recommender system (RS). Finally, we introduce how to deploy our method on the online RS and show the online revenue achieved by our method.

4.1 Offline Dataset

All users and items are collected from the e-commerce platform of the KuaiShou application. The span of data collection is set to be 10 months, from June 2022 to April 2023. After filtering out invalid and outlier users, we collected a total of 12 million users. Items in the user interaction sequence are sampled quarterly by behavior type (item buy, item search, item view, item click), resulting in the collection of 31 million items. The sample size of each behavior is based on its corresponding 80th percentile. We collect the content information of items from six perspectives, including title, category, brand, price, keywords, and attributes. The attributes refer to the service types supported by the items, such as "huabei installment plan", "return for damaged packages ", and "buy now pay later". In the 10-month user interaction sequence, the first 9 months are designated as the historical interaction U^{hist}, while the last month is designated as the target interaction U^{tar}. We divided the dataset into training and test sets. The training set includes 12 million users, encompassing 25 million items from user history interactions and 12 million items from user target interactions. The test set comprises 240K users with 5 million items from user history interactions, along with a gallery set containing 200K items.

4.2 Offline Experimental Settings

4.2.1 Implementation Details. We adopt the Baichuan2-7B [1] as the LLM due to its robust capabilities in understanding Chinese text. The parameters are frozen during the training stage. Adam optimizer is utilized, and the batch size is set to 30. We adopt the cosine learning rate scheduler with a learning rate of 5e-5, and the warm-up parameter is set to 0.1. The model is trained for a total of 10 epochs. All experiments are run on 8 NVIDIA V100 GPUs. Following previous works [17, 26], we use the hit rate (H@50, H@100) and recall (R@50, R@100) for performance evaluation. The length of user history and target interaction are set to 80 and 40 as default settings.

4.3 Offline Results

To explore the way to incorporate the LLM with the recommender system and demonstrate the superiority of our framework, we conduct experiments by answering research questions as follows.

- **RQ1:** Does LLM embedding offer advantages over traditional ID embedding and other content embedding?
- **RQ2:** Does the proposed LEARN framework offer advantages over previous "Rec-to-LLM" methods?
- **RQ3:** Does the proposed LEARN framework offer advantages over SOTA models proposed for industry scenarios?
- **RQ4:** Are all the modules within our proposed LEARN framework indispensable?

*4.3.1 Comparison of embedding types (**RQ1**).* To validate the effectiveness of content embeddings as input, we replaced content embeddings with ID embeddings like the traditional recommendation system. As shown in Tab. 1, the content embeddings based on LLM achieved a significant performance improvement, particularly in the H@100 metric, increasing from 0.0370 to 0.0701, representing an enhancement of 89.46%. To further verify the representation of the content embedding, we adopt [CLS] token of BERT [7] to replace the LLM embedding. Compared to content embeddings generated by BERT, the embeddings generated by LLM improve the performance from 0.0576 to 0.0701. Content embeddings generated by LLM contain a greater amount of information and demonstrate stronger capabilities in expressing textual information about items. We attribute this phenomenon to the extensive text corpus utilized during the LLM pre-training stage, enhancing its feature representation capability.

Table 1: Performance comparison of input embedding in the LEARN framework on the offline dataset

Input Embedding	H@50	R@50	H@100	R@100
ID	0.0244	0.0533	0.0370	0.0769
BERT	0.0357	0.0552	0.0576	0.0843
LLM (Ours)	**0.0440**	**0.0610**	**0.0701**	**0.0905**

*4.3.2 Comparison with previous "Rec-to-LLM" method (**RQ2**).* To confirm the gap between the collaborative domain of recommender and the open-world domain of LLM, we propose two baseline approaches: one with frozen LLM parameters and the other with finetuning all parameters of LLM.

The first baseline (Baseline-v1) utilizes pretrained LLM with frozen parameters. Texts of all items from user history interactions are concatenated into a text sequence, serving as the input to LLM. We perform average pooling on the output of the token embedding by LLM to obtain the user embedding. For item embedding, the texts of each item are individually fed into LLM. We evaluate performance using the same user-to-item retrieval setting as in previous experiments. The second baseline (Baseline-v2) adopts the same LLM but with all learnable parameters. This baseline adopts a similar approach as previous "Rec-to-LLM" methods [3, 11, 18, 22, 24, 25, 30], which transforms the recommendation data into natural language conversation and is optimized by the next token prediction. We designed a prompt to amalgamate the textual information of history items that interacted with the user into a conversational format. We take the text information of one item in the user target interaction as the training targets.

Experiment results are reported in Tab.2. We observed that baseline v1 performed the worst. This can be attributed to the fact that the pre-training data for LLM is sourced from open-world domain text. By directly applying Baseline-v1, which utilizes the output features of LLM with frozen parameters as user embeddings, these embeddings lack collaborative knowledge of the recommendation domain, rendering them unsuitable for recommendation tasks. Despite the improvement compared to Baseline-v1 on H@100, the performance of Baseline-v2 remains far from our proposed LEARN

Table 2: The quantitative metrics results of LLM-based generative recommendation model.

Model	LLM Weights	H@50	R@50	H@100	R@100
Baseline-v1	frozen	0.0069	0.0154	0.0101	0.0210
Baseline-v2	learnable	0.0134	0.0208	0.0180	0.0262
LEARN (Ours)	frozen	**0.0440**	**0.0610**	**0.0701**	**0.0905**

method after finetuning LLM on conversational text data. We believe that the domain gap between collaborative and open-world knowledge leads to catastrophic forgetting of open-world knowledge when finetuning LLM on large-scale textualized recommendation data. Particularly, our training data is collected from real-world recommendation scenarios in the industry, where a significant amount of noise exists in recommendation data, exacerbating the catastrophic forgetting phenomenon. This is why the performance of baseline v2 is unsatisfactory.

The proposed Baseline-v2 follows a similar pattern to previous works in integrating LLM into recommendation systems. However, we observed that our conclusions differ from those of previous methods. This can be attributed to several reasons. Firstly, previous methods are validated on public datasets such as MovieLens [12] and Beauty [15], which are relatively small in scale and consist of clean, curated data that may not reflect real-world industrial scenarios. For instance, the MovieLens-1M dataset contains only 6K users and 4K movies [24], while the Amazon Beauty dataset contains only 22K users and 12K items [13, 21]. In contrast, our dataset comprises 31 million items and 12 million users, reflecting a larger and more realistic industrial-scale scenario. Secondly, due to the small scale of the datasets and the cleanliness of the data, finetuning LLM does not lead to catastrophic forgetting. The world knowledge retained in LLM, along with its understanding and reasoning abilities regarding text content, can provide incremental information for traditional ID-based recommendation models. Therefore, sending textualized recommendation data into LLM and obtaining text descriptions of user preferences can enhance the performance of "Rec-to-LLM" methods [3, 11, 18, 22, 24, 25, 30, 44].

*4.3.3 Comparison with SOTA models proposed for industry scenarios (**RQ3**).* Considering that our method is designed for practical industrial applications, we conduct comparisons with previous SOTA methods aimed at industrial use. To closely approximate real-world industrial scenarios, we chose the Amazon Book Reviews 2014 [1], which comprises 22M reviews of 8M users about 2M books, as the evaluation dataset. To make a fair comparison, we follow the same data processing and evaluation settings as the HSTU [43]. As shown in Fig. 3, compared to the SASRec [20], we achieve significant performance improvement in all six metrics. Compared to the latest method HSTU [43], we achieve a 2.92% and 9.72% improvement on H@50 and H@200, a 10.38% and 6.08% improvement on N@50 and N@200, while slight low performance is achieved on H@10 and N@10. The reasons why our LEARN and HSTU [43] perform differently in the top 10, top 50, and top 200 metrics are as follows. Models with ID embedding typically capture specific identity information of users or items closely tied to their historical interactions. This method focuses on utilizing historical interaction data, which may yield better predictive outcomes for items that are frequently interacted with, popular, or often engaged by specific user groups. Consequently, for smaller recommendation candidates (such as the top 10), models with ID embedding can more accurately identify items with high relevance. Models with content embeddings focus on the textual descriptions of items. This approach enables the model to understand and recommend items that are content-similar yet may not have been frequently interacted with. This method is particularly effective in larger recommendation lists (such as the top 50 or top 100), as it allows for the exploration of a broader range of items, including those that may receive less user attention but have high content relevance. Compared to SASRec [20] and HSTU [43] based on ID embeddings, our proposed LEARN shows a significant performance improvement, indicating the effectiveness of our method and the information richness of the content representations generated by LLM.

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

Table 3: Comparison with SOTA methods. Hit rate (H) and NDCG (N) are adopted as the metrics. We report the performance improvement compared with SASRec.

Method	H@10	H@50	H@200
SASRec [20]	0.0306	0.0754	0.1431
HSTU [43]	**0.0416** (+35.95%)	0.0957 (+26.92%)	0.1735 (+21.24%)
LEARN (Ours)	0.0407 (+33.01%)	**0.0979**(+29.84%)	**0.1874** (+30.96%)
–	N@10	N@50	N@200
SASRec [20]	0.0164	0.0260	0.0362
HSTU [43]	**0.0227** (+38.41%)	0.0344 (+32.31%)	0.0461 (+27.35%)
LEARN (Ours)	0.0224 (+36.59%)	**0.0371** (+42.69%)	**0.0483** (+33.43%)

Table 4: Ablation studies of item tower in the LEARN framework on the offline dataset. Item tower v1 is adopted as the default settings.

Item Tower	H@50	R@50	H@100	R@100
Variant 3	NaN	NaN	NaN	NaN
Variant 2	0.0313	0.0488	0.0505	0.0675
Variant 1 (Ours)	**0.0440**	**0.0610**	**0.0701**	**0.0905**

*4.3.4 Ablation studies of modules within the LEARN framework (**RQ4**).* We design three variants to verify the necessity of our item tower structure. Experimental results are presented in Tab. 4. Variant 3, which directly utilizes content embeddings from LLMs as item embeddings, suffers from model non-convergence. This is attributed to significant architectural differences between the user and item towers, as well as the domain gap between the LLM-generated content embeddings and the item embeddings required for recommendation tasks. In contrast, while Variant 2 with the self-attention mechanism performs less effectively, Variant 1, which incorporates the causal attention mechanism, notably enhances the item embedding representation by integrating features from previous item embeddings. Consequently, we have selected Variant 1, depicted in Fig. 3(a), as the standard architecture for our LEARN framework.

Table 5: Ablation studies of the PCH module.

Backbone	Parameter Initialization	Training mode	Trainable parameters	H@100	R@100
LLM	Pre-train	LoRA	134M	0.0376	0.0560
			286M	0.0504	0.0709
			572M	0.0513	0.0720
Transformer	Random Init.	Finetuning	100M	**0.0701**	**0.0905**

Given extensive world knowledge and superior abilities of LLM in text comprehension and common-sense reasoning, we attempted to replace the transformer layers in the PCH module in Fig. 3 with LLM to further explore its utilization. We finetune the LLM with LoRA [19] and make different LoRA [19] settings to change the amount of trainable parameters. The experiment results are shown in Tab. 5. As the number of trainable parameters increased from 134M to 572M, the performance of finetuning LLM using LoRA improved from 0.0376 to 0.0513, representing a 36.4 % enhancement. However, this performance still exhibits a significant gap compared to using transformer layers as the backbone, which is trained from scratch. The analysis of this discrepancy is as follows. LoRA [19] introduces additional trainable parameters to generate new features, which are combined with the features obtained from the original parameters to serve as the final feature for the LoRA finetuning model. Consequently, the model's output feature is a mixture of the original features trained in the open-world domain and the LoRA features trained in the recommendation domain. Additionally, the original features dominate because the LLM has more frozen parameters compared to the trainable LoRA parameters. However, a substantial domain gap exists between the open-world knowledge embedded in LLM's original pretrained parameters and the collaborative knowledge in LoRA parameters. We conclude this mixed feature is inferior for recommendation tasks.

4.4 Ranking A/B Experiments

We test the LEARN framework in A/B experiments of the ranking model, and deploy our method in the short video feed advertising scenario of KuaiShou App since Jan. 2024.

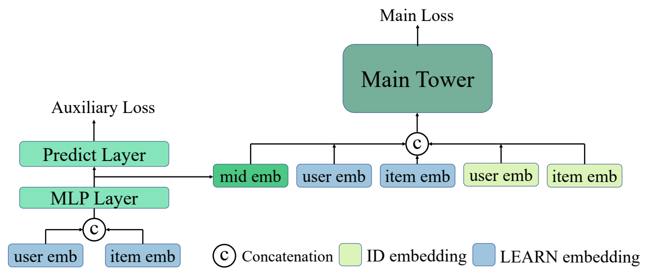

Figure 5: Illustration of Online model structure. The LEARN embedding denotes the user and item embedding generated by the proposed LEARN framework. The ID embedding denotes the conventional sparse embedding.

4.4.1 Model Structure. To enhance the integration of LLM-based user and item embeddings generated by the LEARN framework with the ID embeddings used in online RS, we have integrated an auxiliary network into the existing online model architecture. This addition aims to merge these diverse embedding types more effectively, enhancing the overall synergy of the system. As illustrated in Fig. 5, the auxiliary network employs a CVR loss to guide the aggregation of user and item embeddings produced by LEARN approach. These embeddings are then amalgamated with ID embeddings and a fusion embedding – created by the MLP layers of the auxiliary network – as inputs to the main tower of the online RS.

4.4.2 AUC Evaluation. We conduct offline AUC evaluations using a billion-scale dataset from the KuaiShou e-commerce platform. The training dataset, sampled on December 7, 2023, includes interactions involving 300 million users and 10 million e-commerce products, while the testing data is collected the following day. Our evaluation utilizes two metrics: UAUC and WUAUC, where UAUC represents the average AUC for each user with both positive and negative samples, and WUAUC is the weighted average of UAUC based on the sample count per user. We follow the common practice in the industry to adopt UAUC and WUAUC instead of AUC because these metrics can better evaluate the ranking performance for each user and then can better reflect the user experience. Specifically, UAUC further reflects the performances on long-tail users since it uses uniform weights.

Table 6: The AUC results on KuaiShou App data.

Method	UAUC	WUAUC
Baseline	0.6885	0.7002
LEARN (Ours)	0.6969 (+0.84pp)	0.7078 (+0.76pp)

As depicted in Tab. 6, compared to the baseline model, our method achieves improvements of 0.84 percentage points (pp) and 0.76pp in UAUC and WUAUC, respectively. We hypothesize that the observed improvements in UAUC and WUAUC can be attributed to the superior generalization capabilities of the LEARN framework, which effectively captures the interests of long-tail users. To validate this hypothesis, we have implemented a more comprehensive experimental analysis through online A/B testing.

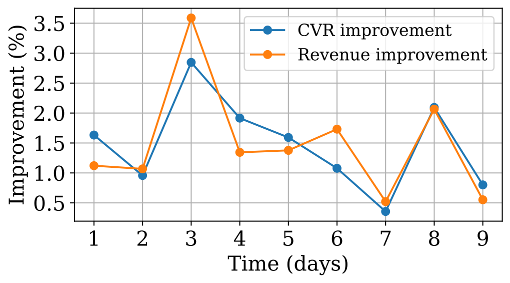

Figure 6: CVR and Revenue improvements during A/B testing. Compared to the baseline method, our method achieves a steady and significant increase in Revenue and CVR.

4.4.3 Online Revenue Improvement. We conduct an online A/B test on the KuaiShou App, a popular short-video streaming platform with daily active users (DAU) exceeding 300 million.

We allocate 20% of the platform's traffic to our proposed and baseline models. The experiments are deployed in a real-time system over a span of 9 days, and the outcomes are depicted in Fig. 6. We observe a steady and significant increase in Revenue and CVR. Given that the revenue for online recommendation models is measured in tens of millions, an improvement of just 2% is quite significant.

Table 7: A/B test results for different user and item types. "Proportion" indicates the percentage of users (items) within each category, relative to the total number of users (items).

Level	Type	Proportion	Revenue	AUC
User	cold-start	7.16%	+1.56%	+0.17pp
	long-tail	27.54%	+5.79%	+0.68pp
	others	65.30%	+0.32%	+0.021pp
Item	cold-start	3.15%	+8.77%	+0.29pp
	long-tail	26.47%	+4.63%	+0.21pp
	others	70.38%	+0.35%	+0.01pp

As illustrated in Tab. 7, we categorize users based on their interaction data over the past six months, including product clicks, impressions, and purchase frequencies. Specifically, users with no recorded interactions are identified as cold-start users, those with 1 to 35 interactions are classified as long-tail users, and those with more than 35 interactions are grouped as others. Our results indicate significant performance enhancements by our proposed LEARN, especially among the cold-start and long-tail user segments.

We also categorize products into three tiers based on their purchase history: Cold-start products, which have no prior purchases; Long-tail products, purchased no more than 30 times in the past week; Other products, purchased more than 30 times during the same timeframe. As depicted in Tab. 7, our proposed method significantly improves revenue and AUC performance for cold-start and long-tail products. These enhancements are attributed to the robust representations LEARN generates for products with sparse purchase histories.

5 CONCLUSION

In this work, we explore the integration of Large Language Models (LLMs) with recommendation systems and develop an approach that successfully combines LLMs with real-world industrial recommendation scenarios, resulting in significant business benefits. We introduce the LEARN framework, which follows a twin-tower structure, and design the CEG module based on LLMs to extract item content embedding from the text description. Additionally, we develop a transformer-based PCH module and adopt recommendation tasks as the training target, facilitating the domain adaptation from open-world knowledge of LLMs to the collaborative knowledge of RS. Our method's effectiveness is demonstrated through substantial results on the large-scale real-world recommendation dataset and public academic dataset.

REFERENCES

[1] Baichuan. 2023. Baichuan 2: Open Large-scale Language Models. *arXiv preprint arXiv:2309.10305* (2023). https://arxiv.org/abs/2309.10305

[2] Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yancheng Luo, Fuli Feng, Xiangnaan He, and Qi Tian. 2023. A bi-step grounding paradigm for large language models in recommendation systems. *arXiv preprint arXiv:2308.08434* (2023).

[3] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems.* 1007–1014.

[4] Shuwei Chen, Xiang Li, Jian Dong, Jin Zhang, Yongkang Wang, and Xingxing Wang. 2023. TBIN: Modeling Long Textual Behavior Data for CTR Prediction. *arXiv preprint arXiv:2308.08483* (2023).

[5] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)* 2, 3 (2023), 6.

[6] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research* 24, 240 (2023), 1–113.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[9] Wenqi Fan, Zihuai Zhao, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Jiliang Tang, and Qing Li. 2023. Recommender systems in the era of large language models (llms). *arXiv preprint arXiv:2307.02046* (2023).

[10] Wenqi Fan, Zihuai Zhao, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Jiliang Tang, and Qing Li. 2023. Recommender systems in the era of large language models (llms). *arXiv preprint arXiv:2307.02046* (2023).

[11] 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).

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

[13] Jesse Harte, Wouter Zorgdrager, Panos Louridas, Asterios Katsifodimos, Dietmar Jannach, and Marios Fragkoulis. 2023. Leveraging large language models for sequential recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems.* 1096–1102.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 770–778.

[15] 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.* 507–517.

[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] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. 2022. Towards universal sequence representation learning for recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 585–593.

[18] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2024. Large language models are zero-shot rankers for recommender systems. In *European Conference on Information Retrieval.* Springer, 364–381.

[19] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

[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] Jinming Li, Wentao Zhang, Tian Wang, Guanglei Xiong, Alan Lu, and Gerard Medioni. 2023. GPT4Rec: A generative framework for personalized recommendation and user interests interpretation. *arXiv preprint arXiv:2304.03879* (2023).

[22] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2023. Llara: Aligning large language models with sequential recommenders. *arXiv preprint arXiv:2312.02445* (2023).

[23] Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, et al. 2023. How can recommender systems benefit from large language models: A survey. *arXiv preprint arXiv:2306.05817* (2023).

[24] Jianghao Lin, Rong Shan, Chenxu Zhu, Kounianhua Du, Bo Chen, Shigang Quan, Ruiming Tang, Yong Yu, and Weinan Zhang. 2023. Rella: Retrieval-enhanced large language models for lifelong sequential behavior comprehension in recommendation. *arXiv preprint arXiv:2308.11131* (2023).

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

[26] Qiang Liu, Zhaocheng Liu, Zhenxi Zhu, Shu Wu, and Liang Wang. 2023. Deep Stable Multi-Interest Learning for Out-of-distribution Sequential Recommendation. *arXiv preprint arXiv:2304.05615* (2023).

[27] Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, and Jiebo Luo. 2023. Llmrec: Personalized recommendation via prompting large language models. *arXiv preprint arXiv:2307.15780* (2023).

[28] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[29] Nikil Pancha, Andrew Zhai, Jure Leskovec, and Charles Rosenberg. 2022. PinnerFormer: Sequence Modeling for User Representation at Pinterest. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 3702–3712.

[30] Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2023. Representation learning with large language models for recommendation. *arXiv preprint arXiv:2310.15950* (2023).

[31] 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.

[32] Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, and Hao Wang. 2024. Continual Learning of Large Language Models: A Comprehensive Survey. *arXiv preprint arXiv:2404.16789* (2024).

[33] Juntao Tan, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Zelong Li, and Yongfeng Zhang. 2024. Towards LLM-RecSys Alignment with Textual ID Learning. *arXiv preprint arXiv:2403.19021* (2024).

[34] 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).

[35] Jinpeng Wang, Ziyun Zeng, Yunxiao Wang, Yuting Wang, Xingyu Lu, Tianxiang Li, Jun Yuan, Rui Zhang, Hai-Tao Zheng, and Shu-Tao Xia. 2023. Missrec: Pre-training and transferring multi-modal interest-aware sequence representation for recommendation. In *Proceedings of the 31st ACM International Conference on Multimedia.* 6548–6557.

[36] Yancheng Wang, Ziyan Jiang, Zheng Chen, Fan Yang, Yingxue Zhou, Eunah Cho, Xing Fan, Xiaojiang Huang, Yanbin Lu, and Yingzhen Yang. 2023. Recmind: Large language model powered agent for recommendation. *arXiv preprint arXiv:2308.14296* (2023).

[37] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021. Empowering news recommendation with pre-trained language models. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval.* 1652–1656.

[38] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. 2020. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th annual meeting of the association for computational linguistics.* 3597–3606.

[39] Yunjia Xi, Weiwen Liu, Jianghao Lin, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, Rui Zhang, and Yong Yu. 2023. Towards open-world recommendation with knowledge augmentation from large language models. *arXiv preprint arXiv:2306.10933* (2023).

[40] Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305* (2023).

[41] 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. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 2639–2649.

[42] Zhenrui Yue, Sara Rabhi, Gabriel de Souza Pereira Moreira, Dong Wang, and Even Oldridge. 2023. LlamaRec: Two-stage recommendation using large language models for ranking. *arXiv preprint arXiv:2311.02089* (2023).

[43] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, et al. 2024. Actions Speak Louder than Words: Trillion-Parameter Sequential Transducers for Generative Recommendations. *arXiv preprint arXiv:2402.17152* (2024).

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

Recommender Systems. 993–999.

[45] Qi Zhang, Jingjie Li, Qinglin Jia, Chuyuan Wang, Jieming Zhu, Zhaowei Wang, and Xiuqiang He. 2021. UNBERT: User-News Matching BERT for News Recommendation.. In *IJCAI*, Vol. 21. 3356–3362.

[46] Yabin Zhang, Wenhui Yu, Erhan Zhang, Xu Chen, Lantao Hu, Peng Jiang, and Kun Gai. 2024. RecGPT: Generative Personalized Prompts for Sequential Recommendation via ChatGPT Training Paradigm. *arXiv preprint arXiv:2404.08675* (2024).