

# LEADRE: Multi-Faceted Knowledge Enhanced LLM Empowered Display Advertisement Recommender System

Fengxin Li\*  
Renmin University of China  
lifengxin@ruc.edu.cn

Yi Li  
Yue Liu  
Chao Zhou  
Yuan Wang  
sincereli@tencent.com  
herculesliu@tencent.com  
derekczhou@tencent.com  
leoyuanwang@tencent.com  
Tencent Inc.

Xiaoxiang Deng  
Wei Xue  
Dapeng Liu  
reeseedeng@tencent.com  
weixue@tencent.com  
rocliu@tencent.com  
Tencent Inc.

Lei Xiao  
Haijie Gu  
Jie Jiang  
shawnxiao@tencent.com  
jerrickgu@tencent.com  
zeus@tencent.com  
Tencent Inc.

Hongyan Liu†  
hyliu@tsinghua.edu.cn  
Tsinghua University

Biao Qin†  
Jun He†  
qinbiao@ruc.edu.cn  
hejun@ruc.edu.cn  
Renmin University of China

## ABSTRACT

Display advertising plays a crucial role in benefiting advertisers, publishers, and users. Traditional display advertising systems employ a multi-stage architecture comprising retrieval, coarse ranking, ranking, and re-ranking. However, conventional retrieval methods primarily rely on ID-based learning-to-rank mechanisms, often underutilizing the content information of ads, like ads' title, and description. This limitation reduces the ability to generate diverse and relevant recommendation lists.

To address this challenge, we propose leveraging the extensive world knowledge of large language models (LLMs). However, effectively integrating LLMs into advertising systems presents three key challenges: (i) *How to accurately capture user interests*, (ii) *How to bridge the knowledge gap between LLMs and advertising systems*, and (iii) *How to efficiently deploy LLMs at scale*. To overcome these challenges, we introduce **LEADRE**—the **LLM Empowered Display ADvertisement REcommender** system. LEADRE consists of three core components. The **Intent-Aware Prompt Engineering** module introduces multi-faceted knowledge and constructs intent-aware *<Prompt, Response>* pairs, fine-tuning LLMs to generate ads tailored to users' personal interests. The **Advertising-Specific Knowledge Alignment** module incorporates auxiliary fine-tuning tasks and Direct Preference Optimization (DPO) to align LLMs with advertising semantics and business objectives. The **Latency-Aware Model Deployment** module integrates a hybrid service framework that balances latency-tolerant and latency-sensitive service, ensuring seamless online deployment.

Extensive offline experiments validate the effectiveness of LEADRE, demonstrating significant improvements across multiple evaluation metrics. Furthermore, online A/B tests reveal a **1.57%** and **1.17%**

increase in Gross Merchandise Value (GMV) for serviced users on WeChat Channels and Moments, respectively. LEADRE has been successfully deployed on both platforms, handling tens of billions of requests daily.

## PVLDB Reference Format:

Fengxin Li, Yi Li, Yue Liu, Chao Zhou, Yuan Wang, Xiaoxiang Deng, Wei Xue, Dapeng Liu, Lei Xiao, Haijie Gu, Jie Jiang, Hongyan Liu, Biao Qin, and Jun He. LEADRE: Multi-Faceted Knowledge Enhanced LLM Empowered Display Advertisement Recommender System. PVLDB, 18(12): 4763 - 4776, 2025.

doi:10.14778/3750601.3750602

## 1 INTRODUCTION

Online display advertising plays a crucial role in facilitating targeted content delivery and meeting users' personal interest, benefiting advertisers, publishers, and users [40, 41]. Traditional display advertising systems employ a multi-stage architecture, including retrieval, coarse ranking, ranking, and re-ranking. The retrieval stage, which initiates the process, is critical for identifying user interests and mitigating the "information cocoon" effect by providing a diverse set of ad options [5, 25].

Conventional retrieval methods primarily rely on ID-based learning-to-rank mechanisms to learn collaborative semantics for efficient ad filtering. However, these methods often underutilize ad content information, such as the title and description of the ads, which limits

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit <https://creativecommons.org/licenses/by-nc-nd/4.0/> to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing [info@vldb.org](mailto:info@vldb.org). Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.

Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097.  
doi:10.14778/3750601.3750602

\*Work was done while Fengxin Li was intern at Tencent.

†Corresponding authors.

in generating diverse recommendations, particularly in scenarios with sparse user behaviors and long-tail ads [10, 48].

Recently, large language models (LLMs) have demonstrated remarkable capabilities in understanding, generalization, and reasoning by leveraging vast amounts of general knowledge [27, 31, 62]. Researchers have explored incorporating LLMs into recommender systems [37, 47] to enhance retrieval performance. However, applying LLMs in industrial-scale display advertising presents several critical challenges:

**(1) Capturing user interests in scenarios with implicit intents and sparse behaviors:** While LLMs exhibit strong capabilities in understanding user intent, display advertising often lacks explicit queries, making it difficult to infer user intent. Additionally, user interactions in the ad domain are typically sparse, necessitating effective utilization of limited data and supplementary behaviors to enhance commercial intent modeling. **(2) Bridging the knowledge gap between LLMs and advertising systems:** Although LLMs excel in generating natural language responses based on general knowledge, display advertising requires the generation of relevant ads from a predefined inventory. It is essential to bridge the gap between the general knowledge of LLMs and the specific requirements of advertising systems. Furthermore, the generated ads must align with business objectives, requiring tailored fine-tuning strategies to optimize LLMs for advertising-specific goals. **(3) Efficiently deploying LLMs in large-scale advertising systems serving billions of requests per day:** Integrating LLMs into an online advertising system imposes substantial computational demands, potentially conflicting with cost efficiency requirements. Thus, optimizing LLM deployment is essential to balance computational costs while maintaining scalability and real-time responsiveness.

To address these challenges, we propose **LEADRE**—a multifaceted knowledge enhanced LLM Empowered display **AD**vertisement **RE**commender system. LEADRE consists of three core components:

**(1) Intent-Aware Prompt Engineering:** This module designs intent-aware *<Prompt, Response>* pairs that fine-tune LLMs to generate ads tailored to users' interests. To mitigate data sparsity, user behaviors from related content domains, such as micro-videos and news, are integrated. Furthermore, commercial intent is modeled by incorporating both **long-term interests** (from user profiles and historical behaviors) and **short-term interests** (from recent interactions) into the prompt. **(2) Advertising-Specific Knowledge Alignment:** To bridge the semantic gap between natural language and advertising data, this module introduces auxiliary fine-tuning tasks. Additionally, Direct Preference Optimization (DPO) is applied to balance user intent with business objectives, ensuring that the generated ads have high commercial value. **(3) Latency-Aware Model Deployment:** LEADRE is deployed using a hybrid architecture that integrates both latency-tolerant and latency-sensitive services. To further enhance computational efficiency, we optimize deployment using **TensorRT LLM Acceleration**.

We implement LEADRE using Hunyuan with 1B parameters and evaluate its performance through extensive offline and online experiments in Tencent's display advertising system. The offline results validate the contributions of the individual modules, demonstrating their effectiveness. In online A/B tests, LEADRE achieved a **1.57%** increase in Gross Merchandise Value (GMV) on Tencent

WeChat Channels and a **1.17%** increase on Tencent WeChat Moments. Currently, LEADRE is deployed on both platforms, serving billions of users and processing tens of billions of requests per day.

The retrieved ads are further incorporated into the ranking phase as additional features. Specifically, on the user side, retrieved ads extend user interest representations, while on the item side, match scores between retrieved ads and target ads serve as new item-level features. These enhancements contribute to an additional **1.43%** improvement in GMV on Tencent WeChat Channels.

The contributions of this work can be summarized as follows:

- To the best of our knowledge, this is the first study to deploy LLMs in an online display advertising system. We introduce LEADRE, a novel LLM-based generative retrieval framework, and deploy it through a hybrid architecture that integrates both latency-tolerant and latency-sensitive service.
- LEADRE integrates Intent-Aware Prompt Engineering and Advertising-Specific Knowledge Alignment to ensure the generated ads are accurate, diverse, commercially valuable, and aligned with user interests.
- Extensive offline and online experiments demonstrate the effectiveness of our approach. We observe substantial improvements in offline metrics such as HitRatio, as well as in online metrics like GMV.

## 2 RELATED WORK

### 2.1 Sequential Recommendation

Sequential recommendation leverages the user behavior sequence on items (ads) to predict the next item that user is likely to click or convert on [54, 66]. Early approaches in sequential recommendation primarily focused on ID-based methods. These methods typically assign a unique ID to each item and employ sequential deep learning models, such as RNNs [53], CNNs [55], and Transformers [21, 42], to learn sequence representations for next-item prediction. However, these approaches often struggle with issues related to cold start and data sparsity. To address these challenges, subsequent research has incorporated additional feature information, including categorical features, numerical features, and graph structure features [4, 23, 30, 59]. Some studies have also explored the integration of multi-modal information, such as item text descriptions and images [26, 29, 52]. By leveraging this supplementary information, sequential recommendation methods can effectively mitigate cold start and data sparsity problems.

Moreover, some researchers have begun to adopt large language models (LLMs) to comprehend item features and infer user preferences, enabling direct recommendations without relying on traditional sequential recommendation methods [1, 2, 14, 47]. In this paper, we introduce LLMs into a large-scale display advertising system to tackle challenges encountered in industrial scenarios, such as implicit user intent and high computational costs.

### 2.2 Large Language Models-based Recommender

Large Language Models (LLMs) have demonstrated remarkable capabilities in both understanding and generation tasks [18, 57],

making them widely applicable across various domains, including document summarization [22, 64], conversational agents [15, 45], code completion [13, 63], and others [60, 67]. Recently, researchers have begun exploring the application of LLMs in recommender systems [50]. These efforts can be broadly categorized into two roles for LLMs: *feature encoder* and *ranker*.

As feature encoders, LLMs leverage their deep understanding of textual content to generate rich representations of users and items. Typically, user or item features are transformed into structured textual descriptions using pre-designed templates, which are then processed by LLMs to obtain feature embeddings [1, 39, 47]. These embeddings can subsequently be utilized for downstream tasks such as candidate retrieval or ranking, improving personalization and recommendation quality.

As rankers, LLMs leverage their generative capabilities to predict user preferences based on historical behavior sequences. In this paradigm, user behavior sequences are represented as textual sequences, and LLMs are trained to predict the next likely item (or user action) based on context [14, 61]. To adapt LLMs effectively for ranking tasks, fine-tuning on domain-specific recommendation datasets is typically required, ensuring the models acquire relevant domain knowledge and effectively capture user preferences [2, 14, 28].

Despite these advancements, the direct application of LLMs in industrial display advertising remains challenging due to several factors, including implicit user intent, high computational costs, and the need for real-time inference. In this work, we present the first application of LLM-based generative retrieval in an industrial display advertising system. To address these challenges, we introduce intent-aware prompt engineering and ad-specific knowledge alignment, enabling LLMs to generate diverse, business-aligned, and user-tailored ad recommendations. Furthermore, we mitigate computational overhead by deploying LLMs in a hybrid architecture that integrates latency-tolerant and latency-sensitive services. This approach ensures that the benefits of LLM-based retrieval can be realized in a scalable and cost-efficient manner.

## 2.3 Retrieval in Advertising Systems

Display advertising systems typically follow a multi-stage architecture composed of multiple stages, including retrieval, coarse-ranking, ranking, re-ranking, and others [8, 12]. The retrieval stage, positioned at the top of the funnel, plays a critical role in identifying user interests and generating a pool of candidate ads for further processing [20, 51]. Most advertising systems adopt ID-based learning to ranking mechanisms to recall all ads that users may find appealing. These methods often rely on a two-tower model, where one tower encodes user features and the other encodes item (ad) features [3, 19].

Display advertising systems typically follow a multi-stage architecture consisting of retrieval, coarse ranking, ranking, re-ranking, and others [8, 12]. The retrieval stage, positioned at the entry of the process, is crucial for identifying user interests and generating an initial pool of candidate ads for subsequent ranking stages [20, 51]. Most advertising systems employ ID-based learning-to-rank mechanisms to retrieve relevant ads for users. These methods often utilize a two-tower model, where one tower encodes user features and the

other encodes item (ad) features [3, 19]. While efficient, ID-based retrieval approaches exhibit several limitations: (1) Information Cocoon Effect: The reliance on past interactions can reinforce existing preferences, leading to a lack of novelty in recommendations. (2) Limited Content Awareness: ID-based approaches often fail to leverage rich ad content, missing potential signals for better personalization. (3) Reduced Diversity: The retrieved candidate set may be dominated by frequently interacted items, limiting exposure to new and diverse ads.

To address these challenges, we propose incorporating LLMs into the retrieval process. Unlike traditional ID-based methods, LLMs can analyze contextual information and generate a more diverse set of candidate ads by expanding retrieval beyond strict collaborative filtering signals. By leveraging their generative capabilities and semantic understanding, LLMs enhance retrieval diversity, improve content-awareness, and ultimately increase user engagement and advertising effectiveness.

## 3 PRELIMINARY

### 3.1 Problem Definition

Let  $\mathcal{U}$  and  $\mathcal{A}$  denote the sets of users and advertisements (ads), respectively, within the target advertising system. Given the limited user interactions with ads, we introduce a content domain to enrich user behavior data. Let  $\mathcal{C}$  denote the set of content items in the system. The sequence of user behaviors is represented as  $\mathcal{S}_u = [i_u^1, i_u^2, \dots, i_u^L]$  in chronological order, where  $u \in \mathcal{U}$ ,  $i_u \in \mathcal{A} \cup \mathcal{C}$ , and  $L = |\mathcal{S}_u|$  is the length of the user's behavior sequence. Given this user behavior sequence, the goal of generative retrieval is to generate the next relevant ad,  $a_u^{L+1} \in \mathcal{A}$ , optimizing both user engagement and business objectives.

### 3.2 Ads Indexing

Traditional advertising systems typically index ads incrementally and learn dense embeddings for each ad [4, 49]. However, this indexing lacks semantic information, creating a gap between the advertising system and human understanding. To bridge this gap, we introduce **Semantic IDs (S-IDs)** based on ad features, inspired by previous works on item indexing [11, 37, 65].

To learn S-IDs, we first define a **feature mapping rule** in the form of a template string to generate text description of each ad, as follows:

The name of the ad is **<ads\_name>**; The product type is **<ads\_type>**; The first-level category is **<first\_cat>**; The second-level category is **<second\_cate>**; The attributes include: **<basic\_att>**.

For each ad  $a \in \mathcal{A}$ , all its features are filled in above template string to generate a textual description  $t_a$ . For example:

The name of the ad is **SAIC Volkswagen-New Energy-ID.3**; The product type is **Automobile Products**; The first-level category is **Automobile**; The second-level category is **SAIC Volkswagen-New Energy**; The attributes include: **automobile brand\_Volkswagen, automobile series\_Volkswagen ID.3**.

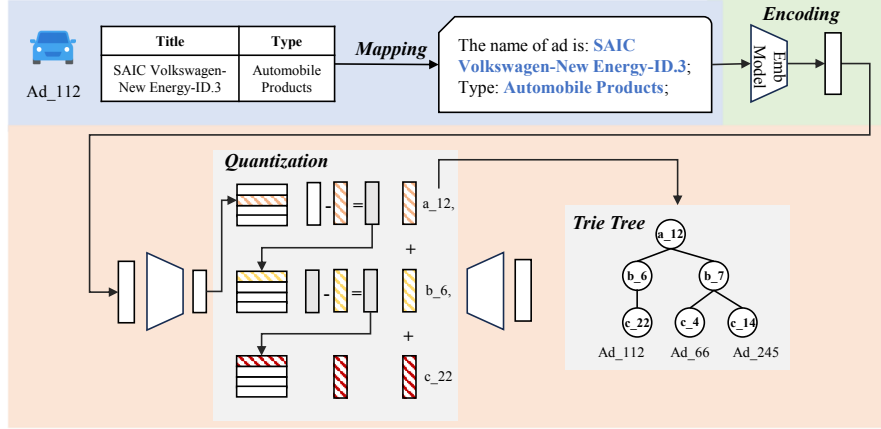


Figure 1: Overall Framework of Ads indexing in LEADRE.

After obtaining the textual description  $t_a$ , we employ a pre-trained language model (e.g. Hunyuan [43], E5 [46]) to **encode**  $t_a$ , bridging the gap between the advertising system and human understandable text. This step can be formulated as

$$\mathbf{x}_a = f_{PLM}(t_a; \theta_{PLM}) \quad (1)$$

where  $f_{PLM}$  represents the pre-trained language model parameterized by  $\theta_{PLM}$ , and  $\mathbf{x}_a \in \mathbb{R}^{d_{PLM}}$  is the textual embedding of ad  $a$ , and  $d_{PLM}$  denotes the output dimension of the pre-trained language model.

After that, we employ a Residual-Quantized Variational AutoEncoder (RQ-VAE) [24] to generate concise semantic IDs for the ads. The RQ-VAE takes the ad's embedding  $\mathbf{x}_a$  as input and uses an **encoder-quantization-decoder** mechanism to generate a list of discrete semantic tokens. In the encoding phase, the textual embedding  $\mathbf{x}_a$  is encoded by an encoder,  $f_{En}(\cdot; \theta_{En})$ , to obtain a hidden embedding, formulated as  $\hat{\mathbf{z}}_a = f_{En}(\mathbf{x}_a; \theta_{En})$ , where  $\hat{\mathbf{z}}_a \in \mathbb{R}^{d_{RQ}}$  denotes the hidden embedding, and  $d_{RQ}$  is the hidden embedding's dimension. During the quantization phase, we apply  $M$  layers of residual vector quantization. For each layer  $l$ , we maintain a codebook  $C^l = \{\mathbf{e}_k^l\}_{k=1}^K$ , where  $\mathbf{e}_k^l \in \mathbb{R}^{d_{RQ}}$  and  $K$  denotes the number of codes in the codebook. The quantization process is defined as follows:

$$c_a^l = \text{argmin}_k \|\mathbf{r}_a^l - \mathbf{e}_k^l\|_2^2 \quad (2)$$

$$\mathbf{r}_a^{l+1} = \mathbf{r}_a^l - \mathbf{e}_{c_a^l}^l \quad (3)$$

$$\mathbf{z}_a = \sum_{l=1}^M \mathbf{e}_{c_a^l}^l \quad (4)$$

where  $\mathbf{r}_a^l \in \mathbb{R}^{d_{RQ}}$  is the  $l$ -th layer residual embedding of ad  $a$ ,  $\mathbf{r}_a^1 = \hat{\mathbf{z}}_a$ ,  $c_a^l$  is the index from the  $l$ -th codebook for ad  $a$ , and  $\mathbf{z}_a \in \mathbb{R}^{d_{RQ}}$  is the final quantized embedding of ad  $a$ . In the decoding phase, the quantized embedding  $\mathbf{z}_a$  is used to reconstruct the original textual embedding  $\mathbf{x}_a$  through a decoder  $f_{De}(\cdot; \theta_{De})$ , formulated as  $\hat{\mathbf{x}}_a = f_{De}(\mathbf{z}_a; \theta_{De})$

To train the encoder  $\theta_{En}$ , decoder  $\theta_{De}$ , and codebooks  $\{C^l\}_{l=1}^M$ , we introduce two loss components: the reconstruction loss and the

#### Algorithm 1 Trie-Tree Construction

**Require:** Set of all ads' S-IDs  $C_A = \{c_1, c_2, \dots, c_a, \dots, c_{|\mathcal{A}|}\}$ , where each

S-IDs sequence  $c_a = [c_a^1, c_a^2, \dots, c_a^l, \dots, c_a^{(M+1)}]$ .

**Ensure:** Trie-Tree  $T$  representing all ads in  $\mathcal{A}$ .

- 1: **Initialize:** Create a head node  $h$  for the trie-tree.
- 2: **for** each S-IDs sequence  $c_a \in C_A$  **do**
- 3:   Set current node  $cur \leftarrow h$
- 4:   **for** each S-ID  $c_a^l$  in  $c_a$  **do**
- 5:     **if**  $c_a^l$  is not a child of  $c$  **then**
- 6:       Create a new node  $n$
- 7:       Add  $n$  as a child of  $cur$  with label  $c_a^l$
- 8:     **end if**
- 9:     Set  $c \leftarrow$  child of  $cur$  labeled by  $c_a^l$
- 10:   **end for**
- 11:   Mark node  $cur$  as the end of ad  $a$
- 12: **end for**
- 13: **return** Trie  $T$

quantization loss. The overall loss function for RQ-VAE is given by:

$$\mathcal{L}_{recons} = \sum_a \|\mathbf{x}_a - \hat{\mathbf{x}}_a\|_2^2 \quad (5)$$

$$\mathcal{L}_{quant} = \sum_a \sum_{l=1}^M \left( \|\text{sg}[\mathbf{r}_a^l] - \mathbf{e}_{c_a^l}^l\|_2^2 + \beta_{quant} \|\mathbf{r}_a^l - \text{sg}[\mathbf{e}_{c_a^l}^l]\|_2^2 \right) \quad (6)$$

where  $\text{sg}[\cdot]$  is the stop-gradient operator, and  $\beta_{quant}$  is a weight factor for the loss term.

This process ensures that ads are represented by discrete S-IDs, which allows the LLMs to generate valid ads from the predefined ads set during constrained decoding. In the ad indexing process, each ad  $a$  is represented by a list of semantic tokens, denoted as  $\hat{c}_a = [c_a^1, c_a^2, \dots, c_a^M]$ . To address potential collisions where different ads map to the same S-IDs, we introduce an additional code to distinguish them [37]. Consequently, the S-IDs for each ad are updated to  $c_a = [c_a^1, c_a^2, \dots, c_a^L, c_a^{(M+1)}]$ , where  $c_a^{(M+1)}$  serves as the disambiguation code for ads with identical S-ID lists. To distinguish between different levels of semantic tokens, we use prefixes such as  $\langle a\_ , b\_ , c\_ , \dots \rangle$  for various S-ID levels.

---

**Algorithm 2** Constrained Decoding Using Trie-Tree

---

**Require:** Trie-tree  $T$  representing all ads, Beam search size  $B$ , S-IDs list length  $(M + 1)$ , Current token probability distribution  $P_{LLM}(c^l | c^{1:l-1}; \theta_{LLM})$  from the LLM.

**Ensure:** Generated ad set of ads  $\mathcal{B}$ .

```
1: Initialize generated ad set  $\mathcal{B} \leftarrow \{[h] \times B\}$ , where  $h$  is the head node of the trie-tree.
2: for  $l = 1$  to  $M + 1$  do
3:   Set current layer  $cur\_layer$  as the  $l$ -th layer of trie-tree  $T$ .
4:   Initialize new generated set  $\mathcal{B}' \leftarrow \emptyset$ .
5:   for each candidate list  $c \in \mathcal{B}$  do
6:     Retrieve valid next S-IDs from the children of node  $c^{l-1}$  in  $cur\_layer$ :  $C_{valid} \leftarrow \text{children of}(c^{l-1})$ .
7:     for each S-ID  $c^l \in C_{valid}$  do
8:       Obtain the score for S-ID  $c^l$  from the LLM:
           $P_{valid}(c^l) \leftarrow P_{LLM}(c^l | c^{1:l-1}; \theta_{LLM})$ .
9:       Append  $c^l$  to the candidate list:  $c \leftarrow c \cup c^l$ .
10:      Add the updated list  $(c, P_{valid}(c^l))$  to new set  $\mathcal{B}'$ .
11:    end for
12:  end for
13:  Update the generated ad set:  $\mathcal{B} \leftarrow \text{select\_top\_B}(\mathcal{B}', B)$ .
14: end for
15: return Generated ad set of ads  $\mathcal{B}$ .
```

---

### 3.3 Trie-Tree Construction

We organize all ads using a trie-tree, where each path from the root to a leaf represents the S-IDs of an ad. This structure enables efficient ad organization and supports fast retrieval and prefix-based searches. The **Trie-Tree Construction Algorithm** (Algorithm 1) builds the trie by sequentially adding each S-ID from an ad as a child node, creating new nodes as needed, and marking the end of each ad's S-ID sequence.

**Example:** Suppose we have the following ads, each represented by a sequence of S-IDs: Ad\_66:  $[a_{12}, b_7, c_4]$ ; Ad\_245:  $[a_{12}, b_7, c_{14}]$ ; Ad\_112:  $[a_{12}, b_6, c_{22}]$ . We construct the trie-tree as follows:

- (1) **Initialize** the trie with an empty root node.
- (2) **Insert Ad\_66:** Add  $a_{12}$  as a child of root, then  $b_7$  under  $a_{12}$ , and  $c_4$  under  $b_7$  (mark as ad end).
- (3) **Insert Ad\_245:**  $a_{12}$  and  $b_7$  already exist; add  $c_{14}$  under  $b_7$  (mark as ad end).
- (4) **Insert Ad\_112:**  $a_{12}$  exists; add  $b_6$  under  $a_{12}$ , then  $c_{22}$  under  $b_6$  (mark as ad end).

The resulting trie-tree (Figure 1) allows for efficient ad retrieval and prefix-based searches, since ads sharing S-ID prefixes follow the same path from the root.

### 3.4 Constrained Decoding

Unlike conventional LLM-based chatbots, generative retrieval in advertising requires generating a list of valid ads from a predefined set, ensuring both diversity and validity. To achieve this, we use constrained decoding with a trie-tree [16, 17, 32]. This method ensures that generated ads are both likely according to the LLM and valid within the ad set. The **Constrained Decoding Algorithm** (Algorithm 2) expands candidate S-ID lists layer by layer through the trie-tree, selecting the top  $B$  candidates at each step based on their scores, until all layers or the maximum length is reached.

**Example:** Suppose trie-tree is built from the following S-ID sequences: Ad\_66:  $[a_{12}, b_7, c_4]$ ; Ad\_245:  $[a_{12}, b_7, c_{14}]$ ; Ad\_112:  $[a_{12}, b_6, c_{22}]$ . Using constrained decoding with a beam size  $B = 2$ :

- (1) **Initialization:** Start with an empty beam set at the root.
- (2) **First Layer:** Valid first S-ID from the trie-tree is  $a_{12}$  with  $P(a_{12}) = 0.6$ . Beam:  $[a_{12}]$  (score 0.6).
- (3) **Second Layer:** For  $[a_{12}]$ , valid next S-IDs are  $b_7$  and  $b_6$  with  $P(b_7) = 0.5$ ,  $P(b_6) = 0.4$ :
  - $[a_{12}, b_7]$  (score  $0.6 \times 0.5 = 0.3$ )
  - $[a_{12}, b_6]$  (score  $0.6 \times 0.4 = 0.24$ )Keep both as  $B = 2$ .
- (4) **Third Layer:**
  - For  $[a_{12}, b_6]$ : valid next S-ID is  $c_{22}$  with  $P(c_{22}) = 0.8$ , yielding  $[a_{12}, b_6, c_{22}]$  (score  $0.24 \times 0.8 = 0.192$ ).
  - For  $[a_{12}, b_7]$ : valid next S-IDs are  $c_4$  and  $c_{14}$  with  $P(c_4) = 0.8$ ,  $P(c_{14}) = 0.4$ :
    - $[a_{12}, b_7, c_4]$  (score  $0.3 \times 0.8 = 0.24$ )
    - $[a_{12}, b_7, c_{14}]$  (score  $0.3 \times 0.4 = 0.12$ )After scoring, keep the top  $B = 2$ :  $[a_{12}, b_7, c_4]$  and  $[a_{12}, b_6, c_{22}]$ .

The algorithm returns the final beam set of S-ID sequences, ensuring that the generated ads are valid according to the trie-tree and have the highest predicted probabilities.

## 4 METHOD

In this section, we present a detailed explanation of multi-faceted knowledge enhanced LLM Empowered display **AD**vertisement **RE**commender system (LEADRE), which integrates large language models (LLMs) into our display advertising system.

### 4.1 Overview

LEADRE effectively aligns user intent with business objectives, improving the generation of relevant and high-value ads in advertising environments. Its framework consists of three novel modules:

(1) **Intent-Aware Prompt Engineering:** This module designs intent-aware  $\langle \text{Prompt}, \text{Response} \rangle$  pairs that incorporate user profiles and behavior sequences, serving as a fine-tuning corpus for LLMs to learn user interests effectively.

(2) **Advertising-Specific Knowledge Alignment:** This module integrates auxiliary fine-tuning tasks and Direct Preference Optimization (DPO) to bridge the knowledge gap between LLMs and the advertising system, enhancing ad generation quality.

(3) **Latency-Aware Model Deployment:** This module combines both latency-tolerant and latency-sensitive service systems to ensure scalable and real-time performance during LLM deployment.

### 4.2 Intent-Aware Prompt Engineering

To accurately capture user intent and predict the next relevant ad, the LLM should be fine-tuned with proper corpus first. We build textual  $\langle \text{Prompt}, \text{Response} \rangle$  pairs with user behavior sequences and ad descriptions as the corpus. Unlike search advertising [37], where user queries explicitly express intent and could be used as the prompt, display advertising operates in a low-intent environment where explicit queries are absent and user behavior is often sparse. As a result, the prompt must effectively leverage available

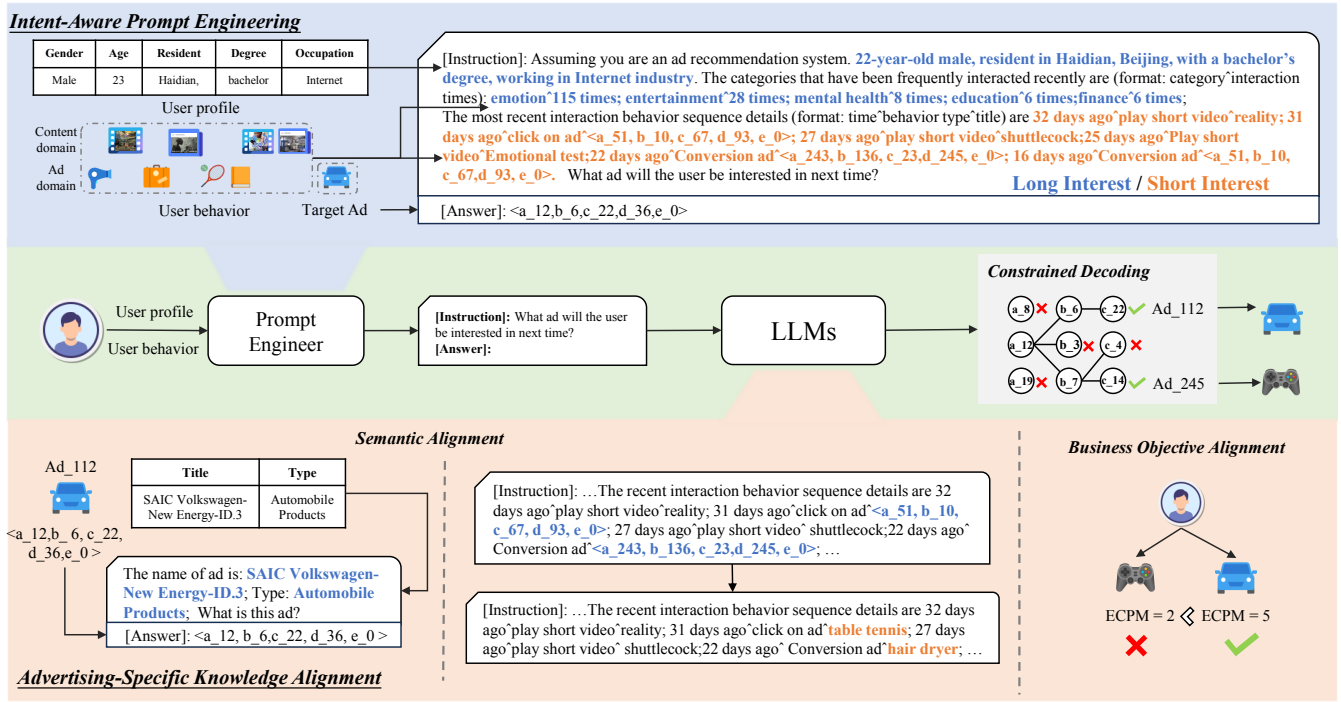


Figure 2: Overall Framework of Ads Constrained Generation Module and LLM Fine-tuning Module.

user data within the advertising system and supplement it with user data outside of the advertising system, such as user behaviors in the content domain, to adequately capture commercial intent. Considering the affordable prompt length is quite limited, user data should be encoded concisely enough. Below, we detail the prompt engineering process.

**4.2.1 Prompt Components.** To enable LLMs to capture user commercial intent effectively, we incorporate both **long-term interests** (derived from user profiles and historical behaviors) and **short-term interests** (based on recent behaviors) into prompt design. Short-term interests model dynamic, immediate preferences. For example, a user planning to purchase a phone would likely click phone or phone accessory ads within a brief period. However, such interests are inherently noisy and transient (e.g., after purchasing the phone, the user would cease clicking similar ads). Conversely, long-term interests capture stable, enduring preferences. For example, a student user would consistently engage with educational coaching ads over extended periods. Specifically, The prompt is assembled with the following key components:

(1) **Task Instruction:** This part describes the task for the LLM to perform. Furthermore, it provides a learnable token for all fine-tuning tasks, similar to Soft Prompt formats [33, 44], helping the LLM recognize instructions related to ad generation and retrieval. The template of the task description is shown below:

The following is an instruction describing a task. Please give a response to complete this request appropriately.  
[Instruction]:<Learnable Token>Assuming you are an ad recommender system, <Prompt>, what ad will the user be interested in next time?

(2) **User Profiles:** This part provides demographical information about the user, such as age, gender, and region. The template is shown below:

<age><sex>, resident in <residence>, with a <education\_level> degree, working in <occupation>, with a <consumption\_level>

(3) **User Interest Summary:** This part summarizes the user's long-term positive and negative feedbacks on commercial interest categories and standard product units (SPUs) across the ad and content domains. The template is shown below:

The categories that have been frequently interacted recently are (format: category`interaction times): <category\_1>`<count\_1> times; <category\_2>`<count\_2> times; <category\_3>`<count\_3> times;

(4) **User Ad Domain Behavior Sequence:** This part expresses the user's short-term behaviors in the ad domain. To filter the noise signals in the sequence, only positive user actions like clicks and conversions are considered. As discussed in Section 3.1, each ad is represented by a list of semantic tokens denoted as Semantic-IDs



(S-IDs). To distinguish different level semantic tokens, we use  $\langle a\_ , b\_ , c\_ , \dots \rangle$  as prefix of different S-IDs level.

(5) **User Content Domain Behavior Sequence:** This part captures the user's short-term behaviors in the content domain. Each interaction is described in terms of the commercial category and SPU. Similarly, only positive actions like watching full videos and searching are considered. The ad and content domain behaviors are merged and presented in chronological order. The template is shown below:

*The most recent interaction behavior sequence details (format: time^behavior type^title) are  $\langle \text{time\_1} \rangle$  days ago^ $\langle \text{type\_1} \rangle$ ^  $\langle \text{title\_1} \rangle$ / $\langle \text{SIDs\_1} \rangle$ ;  $\langle \text{time\_2} \rangle$  days ago^ $\langle \text{type\_2} \rangle$ ^ $\langle \text{title\_2} \rangle$ / $\langle \text{SIDs\_2} \rangle$ ;  $\langle \text{time\_3} \rangle$  days ago ^  $\langle \text{type\_3} \rangle$ ^  $\langle \text{title\_3} \rangle$ / $\langle \text{SIDs\_3} \rangle$*

By concatenating the previous components, we obtain a complete prompt. An example prompt is shown below:

*The following is an instruction describing a task. Please give a response to complete this request appropriately. 22-year-old male, resident in Haidian, Beijing, with a bachelor's degree, working in Internet industry, with a medium consumption level. The categories that have been frequently interacted recently are (format: category^interaction times): emotion^115 times; entertainment^28 times; mental health^8 times; education^6 times; finance^6 times; The most recent interaction behavior sequence details (format: time^behavior type^title) are 32 days ago^play short video^reality; 31 days ago^click on ad^ $\langle a\_51, b\_10, c\_67, d\_93, e\_0 \rangle$ ; 27 days ago^play short video^shuttlecock; 25 days ago^Play short video^Emotional/psychological age test; 22 days ago^Conversion ad^ $\langle a\_243, b\_136, c\_23, d\_245, e\_0 \rangle$ ; 19 days ago^Click ad^ $\langle a\_164, b\_243, c\_38, d\_88, e\_0 \rangle$ ; 16 days ago^Conversion ad^ $\langle a\_51, b\_10, c\_67, d\_93, e\_0 \rangle$ . what ad will the user be interested in next time?*

For the Response, the last click or conversion ad in the user behavior sequence is used as a supervisory signal for fine-tuning the LLM. A sample response is shown below:

*[Response]:  $\langle a\_122, b\_28, c\_35, d\_15, e\_0 \rangle$*

**4.2.2 Prompt Augmentation.** We develop a series of prompt augmentation strategies to enhance the fine-tuning process. These strategies are outlined below:

**Multiple Prompt Templates:** We introduce a variety of prompt templates to map a single user behavior sequence into multiple prompts. By altering the arrangement of sequence components and varying the instruction descriptions, these templates facilitate a broader understanding of user intentions. This diversification allows the model to capture different aspects of user intent, enriching the generation process.

**User Profile Reordering:** To prevent LLMs from simply memorizing specific token sequences or orders, we propose a user profile reordering method. It reshuffles user profile descriptions, encouraging the model to focus on the semantic meaning rather than the fixed order of inputs.

**Ad Positive Interaction Reuse:** Inspired by auto-regressive mechanisms, We augment the original user behavior sequence into multiple prompt samples by reusing positive interaction, enhancing the model's ability to generalize from limited interaction data. For example, Assuming that user behavior sequence is denoted by  $\mathcal{S}_u = [c_u^1, a_u^2, a_u^3, c_u^4, a_u^5]$ , where  $c_u^1, c_u^4 \in \mathcal{C}$  denote content domain items and  $a_u^2, a_u^3, a_u^5 \in \mathcal{A}$  denote ads. We augment the original sequence into prompt samples:  $\langle [c_u^1, a_u^2, a_u^3, c_u^4], a_u^5 \rangle, \langle [c_u^1, a_u^2], a_u^3 \rangle, \langle [c_u^1, a_u^2] \rangle$ .

### 4.3 Advertising-Specific Knowledge Alignment

The advertising system poses a great gap with LLMs, which hurts the ad generation ability of LLMs. To bridge this gap, we conduct semantic alignment by auxiliary tuning tasks to align the ad's Semantic-IDs (S-IDs) with the LLM, and business objectiveness alignment by Direct Preference Optimization (DPO) [35, 58] to encourage the generation of ads with higher Effective Cost Per Mille (ECPM).

**4.3.1 Semantic Alignment.** Since LLM lacks prior knowledge of ads and their S-IDs, direct training on the main task encourages rote memorization rather than genuine understanding. This likes teaching calculus to students without foundational mathematics, which often results in solution memorization instead of conceptual comprehension. To address this, we introduce auxiliary tasks to guide LLMs learn the basic understand of advertising system and S-IDs of ads[65]. The auxiliary tasks consist of two components: an explicit alignment task and an implicit alignment task.

**The explicit alignment task** enables LLM to acquire S-ID knowledge by predicting S-IDs from detailed ad descriptions. This task directly aligns the language model with the ad system by making the LLMs understand how textual descriptions map to the corresponding ad S-IDs. A sample *prompt-response* pair for this task is shown below:

*[Prompt]: Given the ad's detailed description "The name of the ad is SAIC Volkswagen-New Energy-ID.3; The product type is Automobile Products; The first-level category is Automobile; The second-level category is SAIC Volkswagen-New Energy; The attributes include: automobile brand\_Volkswagen, automobile series\_Volkswagen ID.3.", what is the corresponding ad?*

*[Response]:  $\langle a\_12, b\_22, c\_50, d\_25, e\_0 \rangle$*

In this example, LLM is provided with a detailed textual description of an ad, including the product's name, type, brand, and category. The model's task is to map this description to the corresponding S-IDs, which are internal identifiers used by the ad system. By adopting this task, LLM will have a better understanding about the S-IDs and ads' feature, which plays a basic ability of generating ads.

**The implicit alignment task** develops advertising knowledge within user interaction contexts. This task builds on the Next Ad Generation mechanism described in the main task but replaces the S-IDs in the user's ad interaction history with the ad descriptions. Instead of predicting the next ad based on a sequence of S-IDs (as in the main task), the model now predicts the next ad using textual

descriptions of the ads. A sample *<Prompt, Response>* pair for this task is shown below:

[Prompt]: The following is an instruction describing a task. Please give a response to complete this request appropriately. ... The most recent interaction behavior sequence details (format: time`behavior type`title) are **32 days ago`play short video`reality**; **31 days ago`click on ad`Bali 7-Day Tour**; **27 days ago`play short video`shuttlecock**; **25 days ago`Play short video`Emotional/psychological age test**; **22 days ago`Conversion ad`psychological test**; **19 days ago`Click ad`Educational Counseling**; **16 days ago`Conversion ad`Tarot reading**. What ad will the user be interested in next time?

[Response]: **<a\_122, b\_28, c\_35, d\_15, e\_0>**

This subtle modification enhances the LLMs' ability to associate ad descriptions with user behaviors, further aligning the model with the advertising system in a more implicit manner.

**Task Order:** The tasks are conducted in the following order: explicit alignment task  $\rightarrow$  implicit alignment task  $\rightarrow$  main task. It is essential to prioritize alignment tasks, both explicit and implicit, before addressing the main task. These alignment tasks, particularly the explicit alignment task, serve as crucial preparatory steps that equip the LLMs to understand the mapping between textual ad descriptions and the S-IDs used by the ad system. By completing these tasks first, the model gains a deeper comprehension of the structural and semantic aspects of the ad domain, ensuring that it is properly aligned with the system's internal representations. This alignment process is not merely a supplementary step but a foundational one, directly influencing the model's effectiveness in the primary task. In essence, the explicit and implicit alignment tasks establish the groundwork necessary for the model's success in predicting the next ad, making them indispensable for optimal performance in the main task.

**4.3.2 Business Objectiveness Alignment.** Since LLMs inherently lack business context, they may treat all advertisements as having equivalent value. However, in advertising systems, each ad possesses distinct business value that directly impacts system revenue. When tuned exclusively using prior tasks, the generated ads often demonstrate limited commercial value, contributing minimally to both cost and Gross Merchandise Volume (GMV). To address this limitation, we employ **Direct Preference Optimization (DPO)** to encourage generating ads with higher business value.

Since LLMs inherently lack business context, they may treat all advertisements as having equivalent value. However, in advertising systems, each ad possesses distinct business value that directly impacts system revenue. When tuned exclusively using prior tasks, the generated ads often demonstrate limited commercial value, contributing minimally to both cost and Gross Merchandise Volume (GMV). To address this limitation, we employ **Direct Preference Optimization (DPO)** to encourage generating ads with higher business value.

As defined in Section 3.1, for a specific user  $u$ , we aim to predict the next interacted  $a_u^{L+1}$  based on user behavior sequence  $\mathcal{S}_u$ . When considering two equally appealing ads  $a_1$  and  $a_2$  where  $a_1$

holds greater business value, recommending  $a_1$  is preferable due to its higher revenue potential. We therefore implement DPO to increase the probability of generating high-value ads ( $a_h$ ) while reducing probability for lower-value ads ( $a_l$ ). DPO preference pairs  $\langle u, a_h, a_l \rangle$  are constructed using the Expected Cost Per Mille (ECPM) metric from the ranking phase.

Specifically, for a given user  $u$ , we select two potential ads,  $a_1$  and  $a_2$ , that are likely to result in a click or conversion. We then calculate the ECPM for each ad. The ad with the higher ECPM is considered the high-value ad  $a_h$ , while the ad with the lower ECPM is considered the low-value ad  $a_l$ . The training loss associated with DPO can be defined as:

$$\mathcal{L}_{DPO} = -\mathbb{E}_{(u, a_h, a_l) \sim \mathcal{D}} \left[ \log \sigma \left( \beta \frac{\pi_{\theta}(a_h|u)}{\pi_{ref}(a_h|u)} - \beta \frac{\pi_{\theta}(a_l|u)}{\pi_{ref}(a_l|u)} \right) \right]$$

where  $\mathcal{D}$  is the set of collected DPO training triplets,  $\pi_{\theta}$  is the LLM tuned by DPO loss,  $\pi_{ref}$  is the LLM tuned by primary and auxiliary tasks,  $\sigma(\cdot)$  is the sigmoid function, and  $\beta$  is a hyper-parameter. This loss function aims to maximize the likelihood of generating high-value ads while minimizing the likelihood of low-value ads, thereby aligning the LLM's output more closely with business objectives.

## 4.4 Latency-Aware Model Deployment

In this section, we detail the deployment framework and engineering techniques for LLM inference used to ensure efficient and responsive ad generation.

**4.4.1 Deployment Framework.** Typical LLMs exhibit high latency, which is inadequate for real-time display advertising services. To address this, we implement a hybrid system by integrating latency tolerant service module and latency sensitive service module [5, 25]. The deployment consists of the following components:

### (1) Inference:

**Latency Sensitive Service Module :** This component is activated when a user makes a request. A Mixer receives the user request and queries the User Server for the ad retrieval list. The User Server retrieves the pre-computed retrieval list from the user feature database and sends it to the Mixer and further responds to the user. After that, the user behaviors update and ad feature update will be stored in the user feature database and ad feature database.

**Latency Tolerant Service Module:** This module adopts nearline computing for LLM inference and is triggered after a user request. This component receives the fine-tuned LLM and the Trie-Tree from the offline stage for inference. It inquiries users' recent behavior to construct prompt samples and generates a list of ads for the users. The generated ad list is stored in the user feature database and made available for online service.

Even for nearline inference, GPU resources are relatively scarce and cannot be allocated to every request. To address this, we introduce the **Adaptive Resource Distribution Strategy**. This strategy categorizes users into 25 user groups according to Average Revenue Per User (ARPU) value, prioritizing GPU inference for high-value users.

**(2) Training:** The training stage is responsible for LLM fine-tuning, and it is updated daily. The process begins with engineering samples from the user feature database to extract ad features to construct *<Prompt, Response>* pairs for LLM fine-tuning. The *<Prompt,*



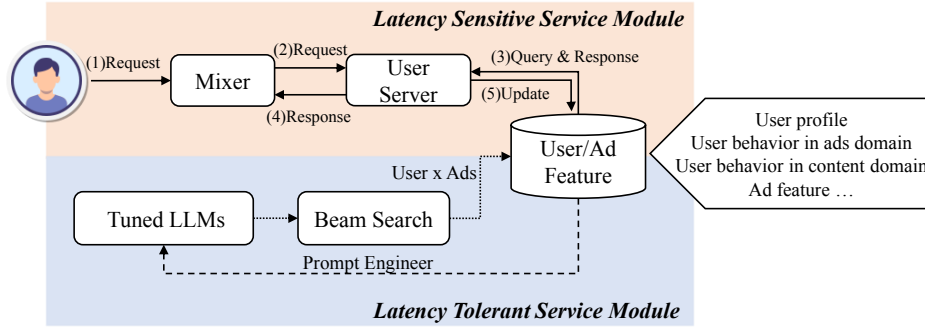


Figure 3: Framework of Latency-Aware Model Deployment.

*Response*> pairs are utilized to fine-tune the LLM, enabling it to generate ads effectively. Once the training is complete, the fine-tuned LLM is deployed to the latency tolerant service module.

**4.4.2 TensorRT LLM Acceleration.** To further enhance the computational efficiency of nearline LLM inference, we employ TensorRT<sup>1</sup> for inference acceleration. This approach leverages several optimization techniques to significantly reduce latency and improve performance, particularly in advertising scenarios:

#### 1) TensorRT-LLM Kernel Optimization

**Softmax Kernel Optimization:** The softmax kernel, which accounts for over 80% of decoding time, is optimized by using 'float4' for vectorized memory access, reducing bandwidth usage and doubling kernel performance. This reduces end-to-end time by 5%. Additionally, for the prefill step, we introduced a 'ComputeMode' to calculate softmax only for beam 0, improving prefill performance by 3-4 times.

**Finalize Kernel Optimization:** In beam search, the finalize kernel recursively obtains token IDs. Initially, it used a single thread, underutilizing GPU resources. By assigning a block per beam width, performance improved by 50 times, reducing latency by 10%.

**2) TensorRT-LLM Quantization:** LLM inference generally involves two stages: prefill and generate. Prefill is typically the computational bottleneck, while the generate stage is more memory-bound. However, in advertising scenarios with a large beam width and batching functionality supported by the inference engine, the generate stage also becomes a computational bottleneck. Therefore, we adopt quantization schemes that focus on activation values, such as smooth quantization ('w8a8c8') and FP8 quantization ('w8a8c8'), to mitigate these bottlenecks.

**Smooth Quantization ('w8a8c8')**: We apply smooth quantization (int8 precision for weights and activations), doubling the computational power for operations like matrix multiplication. This results in a 50% performance improvement in search advertising.

**FP8 Quantization ('w8a8c8')**: FP8 quantization offers a simpler alternative to smooth quantization and achieves similar performance on H20 GPUs without sacrificing precision.

**3) Proxy Load Balancing for Multi-GPU Utilization:** Previous optimizations, such as proxy global load balancing, mainly focused on improving the performance of individual GPUs. To enhance the

Table 1: Comparison of performances under different components of prompt. "w.o. content" variant removes the content domain behaviors from the prompt, retaining only the ad domain behaviors, and "w.o. summary" removes the user interest summary from the prompt.

Variants	HR@1	HR@4	HR@8	NDCG@4	NDCG@8
LEADRE	<b>0.0764</b>	<b>0.1567</b>	<b>0.2021</b>	<b>0.1199</b>	<b>0.1360</b>
w.o. content	0.0660	0.1343	0.1773	0.1029	0.1181
	-13.60%	-14.28%	-12.26%	-14.19%	-13.15%
w.o. summary	0.0760	0.1543	0.2043	0.1181	0.1358
	-0.62%	-1.54%	+1.08%	-1.52%	-0.18%

utilization of multiple GPUs, we optimized the scheduling strategy to better distribute workloads across the GPU cluster. In our system, multiple proxies interact with multiple TensorRT-LLM instances. Without coordination, multiple proxies may send requests to the same TensorRT-LLM instance, leading to uneven load distribution.

TensorRT-LLM is highly sensitive to queries per second (QPS). If one instance processes even a couple more requests than others, it can become the bottleneck for the entire cluster. To mitigate this, we introduced Redis-based global counters to evenly distribute requests across all TensorRT-LLM instances, ensuring balanced load distribution. As a result, the TensorRT-LLM cluster now achieves over 90% of its theoretical maximum load capacity.

## 5 EXPERIMENT

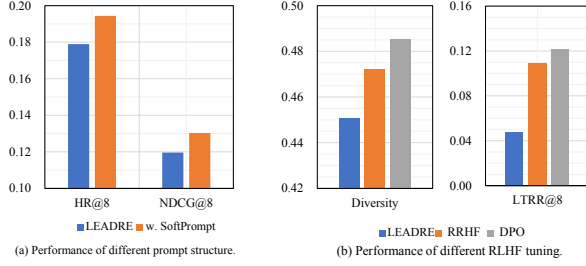
In this section, we verify the effectiveness of LEADRE by conducting extensive offline and online experiments.

### 5.1 Experiment Setup

**5.1.1 Dataset.** We conducted all experiments using industrial-scale datasets on display advertising of Tencent WeChat Channels, as public datasets were deemed unsuitable due to gaps in applicability to our serving system and significant discrepancies with our internal models. The experiment process is suitable for all industrial display advertising systems.

We created user behavior sequences in chronological order and applied the "leave-one-out" strategy for dataset splitting [4, 14, 49]. Specifically, the last ad interaction in each user's sequence was designated as the test set, while the remaining interactions were

<sup>1</sup><https://github.com/NVIDIA/TensorRT-LLM/tree/release/0.5.0>



**Figure 4: Ablation Studies Results on (a) Soft Prompt and (b) DPO tuning.**

**Table 2: Comparison of performances under different tuning tasks and order. "Main" denotes the main task (Next Ad Generation), "EX" denotes the explicit alignment task, "IM" denotes the implicit alignment task, and "mix" denotes mix the tuning < Prompt, Response > pairs and random sample pairs to tune the LLM.**

Tasks and Order	HR@1	HR@4	HR@8	NDCG@4	NDCG@8
EX → IM → Main	<b>0.0780</b>	<b>0.1610</b>	<b>0.2062</b>	<b>0.1232</b>	<b>0.1391</b>
EX → Main	0.0767 -1.71%	0.1566 -2.74%	0.1998 -3.09%	0.1201 -2.50%	0.1354 -2.67%
Main+IM+EX mix	0.0745 -4.60%	0.1523 -5.43%	0.2009 -2.55%	0.1165 -5.39%	0.1337 -3.90%
Main+EX mix	0.0766 -1.91%	0.1516 -5.84%	0.1948 -5.49%	0.1174 -4.64%	0.1327 -4.59%
Main only	0.0707 -9.42%	0.1444 -10.32%	0.1879 -8.87%	0.1107 -10.16%	0.1260 -9.42%

used for training. The maximum user behavior window was set to 90 days and the maximum prompt token length was set to 2096, covering user behavior in both the ads and content domains.

**5.1.2 Evaluation Metric.** We used the Hit Ratio (HR@K) and Normalized Discounted Cumulative Gain (NDCG@K) as the primary metrics to evaluate the offline predictive performance of the models. Hit Ratio measures whether the model successfully retrieves the clicked or converted ads for users. NDCG accounts for both the relevance of items and their positions in the ranked list.

**5.1.3 Implementation Details.** The LEADRE is implemented on the Hunyuan with 1B parameters [43]. Training was conducted on 16 A100Pro GPUs, while inference was performed on hundreds of L40S GPUs. For the Ads indexing, we employed the Hunyuan to encode ad features. The number of residual quantization steps was set to 4, with each layer containing 1024 code vectors, each having a dimension of 8. The length of the S-ID sequence was set to 5. For LLM fine-tuning, we employed the AdamW optimizer, setting the learning rate to 6e-5, weight decay to 0.9, and a minimum learning rate of 6e-6. By leveraging data parallelism and gradient accumulation, the batch size was set to 64. To avoid overfitting, training was performed for 3 epochs.

## 5.2 Offline Performance

**5.2.1 Effectiveness of Prompt Components.** We perform ablation studies on the different components of the prompt in the main task (Section 4.2) to assess their individual contributions.

**Content Domain Behaviors and Interest Summary:** To evaluate the impact of content domain behaviors and user interest summary, we design two variants: "w.o. content" and "w.o. summary". The "w.o. content" variant removes the content domain behaviors from the prompt, retaining only the ad domain behaviors, while "w.o. summary" removes the user interest summary from the prompt. As shown in Table 1, both variants lead to a performance drop across all metrics. This demonstrates the effectiveness of including both content domain behaviors and the user interest summary in capturing user preferences and intent.

**Soft Prompt:** To investigate the effectiveness of adding an instruction-based task description, we introduce a learnable token at the beginning of the prompt, referred to as "w. SoftPrompt". The results, depicted in Figure 4(a), show a performance improvement after incorporating the learnable token, confirming the positive impact of the SoftPrompt on the task.

**5.2.2 Effectiveness of Tuning Tasks and Their Order.** To study the effect of various tuning tasks and their order, we design several tuning strategies: "EX → IM → Main", "EX → Main", "Main+IM+EX mix", "Main+EX mix", and "Main only". The "EX → IM → Main" strategy refers to tuning the LLM sequentially on the explicit alignment task, implicit alignment task, and then the main task. The "EX → Main" strategy skips the implicit task and directly tunes on the explicit task followed by the main task. "Main+IM+EX mix" and "Main+EX mix" represent mixed-tuning strategies where pairs are randomly sampled and tuned together. "Main only" refers to tuning solely on the main task, without alignment tasks.

Table 2 presents the tuning results, leading to the following observations: (1) The explicit and implicit alignment tasks enhance the performance of the main task by bridging the gap between the language model and the ad system. (2) The order of tuning matters, as the fixed-order strategies outperform the mixed strategies. This suggests that the alignment tasks provide a foundational understanding of the ad system, which better prepares the LLM for the main task.

**5.2.3 Effectiveness of DPO.** After tuning the LLM on the auxiliary and main tasks, we apply Reinforcement Learning from Human Feedback (RLHF) to align the language model with business objectives, using both RRHF [58] and DPO [35] techniques. To evaluate the alignment with business values, we use two metrics: Diversity score and LTRR@K.

The Diversity score is calculated based on TopK List Concentration and TopK List Abundance. Concentration is the mean proportion of the most frequent category across all users, while Abundance refers to the mean number of categories present in the top-K list for all users. The Diversity score combines both metrics, where a higher score indicates a more diverse top-K list with lower Concentration and higher Abundance. LTRR@K is computed by Recall@K based on Learning-to-Rank (LTR) labels provided by other retrieval strategies, with higher LTRR@K indicating better alignment with business objectives.

**Table 3: Comparison of different emb models.**

Emb model	Accuracy			
	Top1	Top10	Top50	Top100
Hunyuan Embedding[43]	<b>98.03%</b>	96.46%	86.61%	65.67%
E5-large-instruct [46]	97.21%	<b>96.48%</b>	<b>87.42%</b>	<b>66.38%</b>
bge-m3[6]	95.00%	91.07%	78.76%	59.64%
Bert-Chinese[9]	97.24%	<u>92.65%</u>	74.12%	50.20%
Electra_Chinese[7]	94.60%	85.98%	63.09%	43.62%
Sentence-Bert/L12[38]	95.40%	90.11%	70.15%	48.44%
Sentence-Bert/L6[38]	81.97%	72.65%	53.61%	37.97%
XLNet Chinese[56]	80.31%	63.32%	38.27%	26.14%
CLIP-32[34]	80.32%	67.91%	45.90%	29.87%
CLIP-16[34]	81.57%	67.40%	45.20%	30.33%
T5[36]	67.10%	51.22%	33.90%	23.89%

**Table 4: Performance comparison of different emb models.**

S-IDs	HR@4	HR@8	LTRR@4	LTRR@8
Hunyuan S-IDs	0.1070	0.2140	0.0294	0.0443
E5 S-IDs	0.1052	0.2140	0.0256	0.0407

The RLHF results, shown in Figure 4, lead to the following conclusions: (1) RLHF improves the alignment of the language model with business values. (2) DPO outperforms RRHF, likely due to the constraints imposed by the tuned model during the alignment process.

**5.2.4 Effectiveness of Feature Embedding.** To assess the effectiveness of the ads’ feature embedding used in ads indexing, we compare the performance of different embedding models (emb. models). We first sampled 100 ads from each of the 48 ad categories and retrieved the Top-K nearest ads for each sampled ad based on embedding similarity. The accuracy is defined as the proportion of Top-K retrieved ads that belong to the same category as the original ad. The performance comparison is presented in Table 3. Our observations indicate that the E5 model demonstrates the best performance on this task, while the Hunyuan embedding achieves competitive results.

Subsequently, we trained the VQ-VAE using both the E5 and Hunyuan embeddings to infer the ads’ S-IDs. We then constructed a trie tree and fine-tuned the LLM using these S-IDs. The performance of the tuned LLM is shown in Table 4. The results indicate that Hunyuan S-IDs exhibit competitive performance in terms of HR@K and outperform in LTRR@K, attributed to their enhanced capability in understanding the ad system.

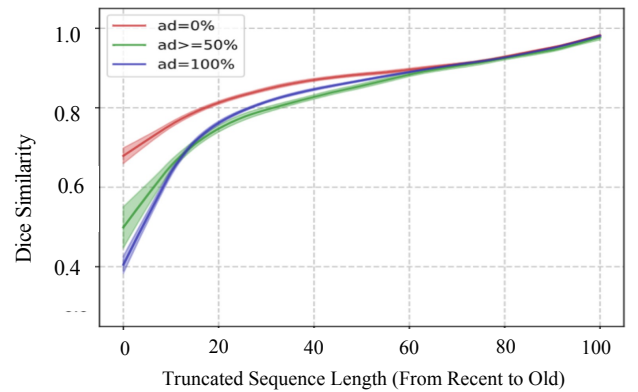
**5.2.5 Sensitivity and Effectiveness of S-IDs.** To evaluate the sensitivity of the ads’ Semantic IDs (S-IDs), we varied the codebook size and quantity, assessing the trained ads indexing using three metrics: Collision Rate↓, Max Collision Number↓, and Average Codebook Usage Rate↑. As shown in Table 5, increasing the codebook size and quantity reduces ad collisions, indicating that different ads are assigned distinct S-IDs. Notably, the codebook usage rate remains at 100%, signifying no occurrence of "codebook collapse."

**Table 5: Performance comparison of different codebook size and quantity.**

Quantity	Size	Collision Rate↓	Max collision Number↓	Average Usage Rate↑
3	1024	4.79%	44	1.0
	512	7.95%	46	1.0
	256	16.38%	67	1.0
4	1024	1.74%	25	1.0
	256	3.34%	25	1.0

**Table 6: Performance comparison of different S-Ids variants.**

Variants	HR@1	HR@4	HR@8	NDCG@4	NDCG@8
S-IDs	<b>0.0800</b>	<b>0.1526</b>	<b>0.1975</b>	<b>0.1194</b>	<b>0.1353</b>
Original Text	0.0306	0.0551	0.0679	0.0439	0.0487
	-61.75%	-63.89%	-65.62%	-63.23%	-64.01%
Compressed Text	0.0572	0.1102	0.1317	0.0861	0.0967
	-28.50%	-27.79%	-33.32%	-27.89%	-28.53%



(c) LLM User Action Response w.r.t Sequence Length.

**Figure 5: LLM User Action Response w.r.t Sequence Length.**

Additionally, to assess the effectiveness of the S-IDs, we removed them and generated both the original and compressed text of ads using the LLM. The performance results are presented in Table 6. The findings indicate a significant performance drop when generating either the original or compressed text, highlighting the importance of S-IDs. Furthermore, the compressed text outperformed better than the original, suggesting that the original text contains redundancy and noise.

**5.2.6 Effectiveness of User Behavior.** To assess how user behavior contributes to LEADRE’s ad generation process, we conducted a user study to gain a deeper understanding of the generation mechanism.

To investigate the impact of different positions within user behavior sequences, we conducted the following experiment: For a user  $u$  with a behavior sequence  $\mathcal{S}_{u,1} = [i_u^1, i_u^2, \dots, i_u^L]$ , we sequentially truncated the sequence by removing earlier interactions, forming sequences such as  $\mathcal{S}_{u,2} = [i_u^2, i_u^3, \dots, i_u^L]$  and  $\mathcal{S}_{u,3} = [i_u^3, i_u^4, \dots, i_u^L]$ .

LEADRE then made predictions using each truncated sequence, generating top-K lists denoted by  $\mathcal{R}_{u,1}$ ,  $\mathcal{R}_{u,2}$ , and so on. To quantify the similarity between these lists, we used the Dice similarity coefficient:

$$\text{Dice}(\mathcal{R}_{u,1}, \mathcal{R}_{u,l}) = \frac{2|\mathcal{R}_{u,1} \cap \mathcal{R}_{u,l}|}{|\mathcal{R}_{u,1}| + |\mathcal{R}_{u,l}|} \quad (7)$$

A higher Dice similarity score indicates that the retrieved list generated from the truncated sequence is more similar to that of the complete sequence. This, in turn, suggests that the removed interaction is of lesser importance in influencing the retrieval outcome. We computed the Dice similarity for all users, grouped by the partitioning of their ad behaviors, and plotted the results as a function of truncated sequence length (Figure 5). The following key observations were made: (1) Longer behavior sequences capture more user preference information. (2) LEADRE places more emphasis on the earlier parts of the sequence, particularly the first 20 behaviors, and less emphasis on more recent behaviors. (3) Ad domain behaviors have a stronger influence on the generated outcomes compared to content domain behaviors, and they encourage LEADRE to focus more on the beginning of the sequence.

**5.2.7 Summary.** In this section, we evaluated the impact of various components, including prompt engineering, tuning tasks, and S-IDs. Our findings are summarized as follows: (1) The S-IDs and the main tuning task are fundamental for ad generation; removing either results in the largest performance drops. (2) Auxiliary tasks, such as explicit and implicit alignment tasks and Direct Preference Optimization (DPO), contribute less than the S-IDs and main task, but still provide a crucial 9% performance increase. (3) User behaviors from the content and advertising domains are the most significant contributors to ad generation compared to other prompt components.

### 5.3 Online Performance

We conducted a 20% traffic A/B test on both the Tencent WeChat Channels and Tencent WeChat Moments display advertising systems to evaluate the effectiveness of LEADRE. LEADRE acts as a complementary retrieval sub-branch. Over several weeks' A/B test on WeChat Channels, LEADRE demonstrated a 1.57% increase in Gross Merchandise Value (GMV) for serviced users. Similarly, several weeks' A/B test on WeChat Moments showed a 1.17% increase in GMV for serviced users. These results highlight LEADRE's capability to generate accurate, diverse, and high-value ads that contribute positively to business outcomes.

LEADRE operates as a retrieval sub-branch, collaborating with other LTR branches to complete the retrieval process. To highlight the competition between branches, we introduce the consumption. Consumption is the proportion of exposed ads exclusively retrieved by a specific branch relative to all exposed ads. Online A/B test shows that LEADRE achieves a consumption ratio of 7.33% (Wechat Channels). This demonstrates its ability to retrieve ads missed by other branches, directly contributing to the increase in GMV.

To effectively leverage the capability of LLMs in breaking the "information cocoon" effect, the retrieved ads are further incorporated into the ranking phase as new features. Specifically, on the user side, the retrieved ads are used as additional features to enhance user profiling, while on the item side, the match scores

between the retrieved ads and the target ad are introduced as new item-level features. These new features contribute to an extra 1.43% improvement in GMV on Tencent WeChat Channels.

Currently, LEADRE has been deployed as a complementary retrieval sub-branch on both Tencent WeChat Channels and Moments, serving billions of users and processing tens of billions of requests each day, highlighting its scalability and practical impact within a large-scale advertising system.

## 6 CONCLUSION

In this work, we present the first industrial application of generative retrieval in display advertising. To address the challenges of "How to capture user commercial intent", "How to bridge the gap between LLMs and ad-Specific knowledge", and "How to efficiently deploy LLMs", we introduce a novel LLM-based framework called Multi-Faceted Knowledge Enhanced LLM Empowered Display **AD**vertisement **RE**commender system (LEADRE). LEADRE comprises three core modules: the Intent-Aware Prompt Engineering, the Advertising-Specific Knowledge Alignment, and the Latency-Aware Model Deployment. To evaluate the effectiveness of LEADRE, we conducted extensive experiments, including both offline and online evaluations. The results demonstrate its ability to generate more accurate and diverse ads. Moreover, online A/B test revealed that LEADRE resulted in a 1.57% increase on Tencent WeChat Channels, as well as a 1.17% increase on Tencent WeChat Moments in GMV for serviced users.

Looking ahead, our future work will explore the following directions: (1) Next-N Generation for LLMs: Next-N ad prediction provides a deeper understanding of user intent and offers a more accurate and diverse retrieval list. However, LEADRE currently supports only the generation of a single ad due to its reliance on a statistical trie-tree and fixed tuning tasks. Future efforts will focus on designing appropriate trie-tree structures and constructing tuning tasks for the next-N generation. (2) Development of More Reasonable S-IDs: Current results indicate that the RQ-VAE tends to allocate most information to the first codebook, while the subsequent codebooks retain limited information. Future research will aim to design an information-controllable quantization model to construct more effective S-IDs.

## ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China (62436010, 62172421) and the Tencent Innovation Fund. Biao Qin is the first corresponding author.

## REFERENCES

- [1] Keqin Bao, Jizhi Zhang, Xinyu Lin, Yang Zhang, Wenjie Wang, and Fuli Feng. 2024. Large Language Models for Recommendation: Past, Present, and Future. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2993–2996.
- [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. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1007–1014.
- [3] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable multi-interest framework for recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2942–2951.
- [4] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2021. Sequential recommendation with graph neural networks.

- In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 378–387.
- [5] Jun Chen, Cheng Chen, Huayue Zhang, and Qing Tan. 2022. A Unified Framework for Campaign Performance Forecasting in Online Display Advertising. *arXiv preprint arXiv:2202.11877* (2022).
  - [6] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. *arXiv:2402.03216*
  - [7] K Clark. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555* (2020).
  - [8] Zhihua Cui, Xianghua Xu, XUE Fei, Xingjuan Cai, Yang Cao, Wensheng Zhang, and Jinjun Chen. 2020. Personalized recommendation system based on collaborative filtering for IoT scenarios. *IEEE Transactions on Services Computing* 13, 4 (2020), 685–695.
  - [9] Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
  - [10] Manqing Dong, Feng Yuan, Lina Yao, Xiwei Xu, and Liming Zhu. 2020. Mamo: Memory-augmented meta-optimization for cold-start recommendation. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 688–697.
  - [11] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*. 299–315.
  - [12] Zhabiz Gharibshah, Xingquan Zhu, Arthur Hainline, and Michael Conway. 2020. Deep learning for user interest and response prediction in online display advertising. *Data Science and Engineering* 5, 1 (2020), 12–26.
  - [13] Daya Guo, Canwen Xu, Nan Duan, Jian Yin, and Julian McAuley. 2023. Longcoder: A long-range pre-trained language model for code completion. In *International Conference on Machine Learning*. PMLR, 12098–12107.
  - [14] 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.
  - [15] 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*.
  - [16] Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. *arXiv preprint arXiv:1704.07138* (2017).
  - [17] J Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. 2019. Improved lexically constrained decoding for translation and monolingual rewriting. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 839–850.
  - [18] Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuxin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *arXiv preprint arXiv:2210.11610* (2022).
  - [19] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2553–2561.
  - [20] Houye Ji, Ye Tang, Zhaoxin Chen, Lixi Deng, Jun Hu, and Lei Su. 2024. Neural Graph Matching for Video Retrieval in Large-Scale Video-driven E-commerce. *arXiv preprint arXiv:2408.00346* (2024).
  - [21] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.
  - [22] Huan Yee Koh, Jiaxin Ju, Ming Liu, and Shirui Pan. 2022. An empirical survey on long document summarization: Datasets, models, and metrics. *ACM computing surveys* 55, 8 (2022), 1–35.
  - [23] Srikanth Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1269–1278.
  - [24] Doyup Lee, Chihyeon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. 2022. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
  - [25] Jin Li, Jie Liu, Shangzhou Li, Yao Xu, Ran Cao, Qi Li, Biye Jiang, Guan Wang, Han Zhu, Kun Gai, et al. 2021. Truncation-Free Matching System for Display Advertising at Alibaba. *arXiv preprint arXiv:2102.09283* (2021).
  - [26] Jiacheng Li, Ming Wang, Jin Li, Jimiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. 2023. Text is all you need: Learning language representations for sequential recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1258–1267.
  - [27] Kunze Li and Yu Zhang. 2024. Planning First, Question Second: An LLM-Guided Method for Controllable Question Generation. In *Findings of the Association for Computational Linguistics ACL 2024*. 4715–4729.
  - [28] Xinyu Lin, Wenjie Wang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. 2024. Bridging items and language: A transition paradigm for large language model-based recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1816–1826.
  - [29] 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.
  - [30] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. 2016. Context-aware sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1053–1058.
  - [31] Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2024. Using an llm to help with code understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.
  - [32] Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. *arXiv preprint arXiv:1804.06609* (2018).
  - [33] Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. *arXiv preprint arXiv:2104.06599* (2021).
  - [34] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.
  - [35] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems* 36 (2024).
  - [36] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21, 140 (2020), 1–67. <http://jmlr.org/papers/v21/20-074.html>
  - [37] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. 2024. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems* 36 (2024).
  - [38] N Reimers. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *arXiv preprint arXiv:1908.10084* (2019).
  - [39] Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Representation learning with large language models for recommendation. In *Proceedings of the ACM on Web Conference 2024*.
  - [40] Vinay Singh, Brijesh Nanavati, Arpan Kumar Kar, and Agam Gupta. 2023. How to maximize clicks for display advertisement in digital marketing? A reinforcement learning approach. *Information Systems Frontiers* 25, 4 (2023), 1621–1638.
  - [41] Derun Song, Enneng Yang, Guibing Guo, Li Shen, Linying Jiang, and Xingwei Wang. 2024. Multi-scenario and multi-task aware feature interaction for recommendation system. *ACM Transactions on Knowledge Discovery from Data* 18, 6 (2024), 1–20.
  - [42] 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.
  - [43] Xingwu Sun, Yanfeng Chen, Yiqing Huang, Ruobing Xie, Jiaqi Zhu, Kai Zhang, Shuaipeng Li, Zhen Yang, Jonny Han, Xiaobo Shu, et al. 2024. Hunyuan-Large: An Open-Source MoE Model with 52 Billion Activated Parameters by Tencent. *arXiv:2411.02265*
  - [44] Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2021. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904* (2021).
  - [45] Bryan Wang, Gang Li, and Yang Li. 2023. Enabling conversational interaction with mobile ui using large language models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–17.
  - [46] Liang Wang, Nan Yang, Xiaolong Huang, Bingxin Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533* (2022).
  - [47] Wei Wei, Xubin Ren, Jiabin Tang, Qingyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 806–815.
  - [48] Yinwei Wei, Xiang Wang, Qi Li, Liqiang Nie, Yan Li, Xuanping Li, and Tat-Seng Chua. 2021. Contrastive learning for cold-start recommendation. In *Proceedings of the 29th ACM International Conference on Multimedia*. 5382–5390.
  - [49] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT: Sequential recommendation via personalized transformer. In *Proceedings of the 14th ACM conference on recommender systems*. 328–337.
  - [50] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2024. A survey on large language models for recommendation. *World Wide Web* 27, 5 (2024), 60.



- [51] Ruobing Xie, Qi Liu, Liangdong Wang, Shukai Liu, Bo Zhang, and Leyu Lin. 2022. Contrastive cross-domain recommendation in matching. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*.
- [52] Yueqi Xie, Peilin Zhou, and Sunghun Kim. 2022. Decoupled side information fusion for sequential recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 1611–1621.
- [53] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Jiajie Xu, Victor S Sheng S. Sheng, Zhiming Cui, Xiaofang Zhou, and Hui Xiong. 2019. Recurrent convolutional neural network for sequential recommendation. In *The world wide web conference*. 3398–3404.
- [54] Mingming Xu, Fangai Liu, and Weizhi Xu. 2019. A survey on sequential recommendation. In *2019 6th international conference on information science and control engineering (ICISCE)*. IEEE, 106–111.
- [55] An Yan, Shuo Cheng, Wang-Cheng Kang, Mengting Wan, and Julian McAuley. 2019. CosRec: 2D convolutional neural networks for sequential recommendation. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 2173–2176.
- [56] Zhilin Yang. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237* (2019).
- [57] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing* (2024), 100211.
- [58] Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. 2024. RRHF: Rank responses to align language models with human feedback. *Advances in Neural Information Processing Systems* 36 (2024).
- [59] Xu Yuan, Dongsheng Duan, Lingling Tong, Lei Shi, and Cheng Zhang. 2021. Icair: Item categorical attribute integrated sequential recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 1687–1691.
- [60] ChengXiang Zhai. 2024. Large language models and future of information retrieval: Opportunities and challenges. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- [61] 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.
- [62] Jing Zhang, Hui Gao, Peng Zhang, Boda Feng, Wenmin Deng, and Yuxian Hou. 2024. LA-UCL: LLM-augmented unsupervised contrastive learning framework for few-shot text classification. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*. 10198–10207.
- [63] Mingxuan Zhang, Bo Yuan, Hanzhe Li, and Kangming Xu. 2024. LLM-Cloud Complete: Leveraging cloud computing for efficient large language model-based code completion. *Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023* 5, 1 (2024), 295–326.
- [64] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. 2024. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics* 12 (2024), 39–57.
- [65] Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. 2024. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 1435–1448.
- [66] Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Filter-enhanced MLP is all you need for sequential recommendation. In *Proceedings of the ACM web conference 2022*. 2388–2399.
- [67] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107* (2023).