

Unlocking Scaling Law in Industrial Recommendation Systems with a Three-step Paradigm based Large User Model

Bencheng Yan*, Shilei Liu*, Zhiyuan Zeng*, Zihao Wang, Yizhen Zhang, Yujin Yuan, Langming Liu, Jiaqi Liu, Di Wang, Wenbo Su, Wang Pengjie, Jian Xu and Bo Zheng[†]

Alibaba Group

China

{bencheng.ybc, liushilei.lsl, zengzhiyuan.zzy, wzh454725, zhangyizhen.zyz, yujin.yyj, liulangming.llm, ljq414468, zhemu.wd, vincent.swb, pengjie.wpj, xiyu.xj, bozheng}@alibaba-inc.com

Abstract

Recent advancements in autoregressive Large Language Models (LLMs) have achieved significant milestones, largely attributed to their scalability, often referred to as the "scaling law". Inspired by these achievements, there has been a growing interest in adapting LLMs for Recommendation Systems (RecSys) by reformulating RecSys tasks into generative problems. However, these End-to-End Generative Recommendation (E2E-GR) methods tend to prioritize idealized goals, often at the expense of the practical advantages offered by traditional Deep Learning based Recommendation Models (DLRMs) in terms of features, architecture, and practices. This disparity between idealized goals and practical needs introduces several challenges and limitations, locking the scaling law in industrial RecSys. In this paper, we introduce a large user model (LUM) that addresses these limitations through a three-step paradigm, designed to meet the stringent requirements of industrial settings while unlocking the potential for scalable recommendations. Our extensive experimental evaluations demonstrate that LUM outperforms both state-of-the-art DLRMs and E2E-GR approaches. Notably, LUM exhibits excellent scalability, with performance improvements observed as the model scales up to 7 billion parameters. Additionally, we have successfully deployed LUM in an industrial application, where it achieved significant gains in an A/B test, further validating its effectiveness and practicality.

ACM Reference Format:

Bencheng Yan*, Shilei Liu*, Zhiyuan Zeng*, Zihao Wang, Yizhen Zhang, Yujin Yuan, Langming Liu, Jiaqi Liu, Di Wang, Wenbo Su, Wang Pengjie, Jian Xu and Bo Zheng[†]. 2025. Unlocking Scaling Law in Industrial Recommendation Systems with a Three-step Paradigm based Large User Model. In . ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 Introduction

In recent years, autoregressive large language models (LLMs) have achieved significant breakthroughs, primarily due to their adherence to the principle of scalability, often referred to as the "scaling

* These authors contributed equally to this work and are co-first authors.

[†] Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, Washington, DC, USA

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

law". This law posits that there is a predictable and continuous improvement in model performance as the computational resources allocated to the model are increased[17, 18].

In parallel, the field of recommendation systems (RecSys) has been actively investigating the potential for leveraging similar scalability principles. Early research has highlighted that traditional deep learning-based recommendation models (DLRMs), widely used in industry, do not exhibit the same level of scalability observed in LLMs[23]. This discrepancy can be attributed to the fundamental differences between generative models and discriminative models. Specifically, generative models require a more substantial capacity to effectively capture the complex joint probability distribution $p(x, y)$ of the data, whereas discriminative models, such as traditional DLRMs, focus on modeling the simpler conditional probability $p(y|x)$. As a result, the benefits of increased computational resources are less pronounced in discriminative models [5, 13, 23].

Consequently, several studies have endeavored to mimic the generative models approach by reformulating the retrieval and ranking tasks within RecSys as generative tasks. This is achieved through utilizing User Behavior Sequences (UBS) as the corpus and training a transformer architecture on a "next-item prediction" task in an end-to-end fashion [6, 23]. However, these end-to-end Generative Recommendation methods (E2E-GRs) tend to overemphasize idealized goals, thereby neglecting the inherent strengths in features, architecture, and practices of DLRMs. This gap between idealized goals and practical implementation in the modeling paradigm has subsequently given rise to a series of challenges and limitations.

(1) Inconsistency Between Generative Training and Discriminative Application: Despite their proficiency in capturing the intricate patterns and distributions inherent in data, E2E-GRs often exhibit limitations when applied to specific discriminative tasks, such as click-through rate prediction where both calibration ability and ranking ability are paramount [20]. This disparity stems from the intrinsic focus of generative models on the process of data generation rather than on the precise predictive outcomes. As a result, although generative models are adept at capturing inherent distributions of data, they may not consistently satisfy the rigorous demands of discriminative applications, which require high levels of accuracy and specificity [2, 14, 21].

(2) Efficiency Challenges: The demand for high efficiency in continuous streaming training, coupled with the stringent low-latency requirements for online real-time inference (beyond the throughput considerations mentioned in [23]), presents significant challenges for the direct implementation of E2E-GRs in industrial settings.

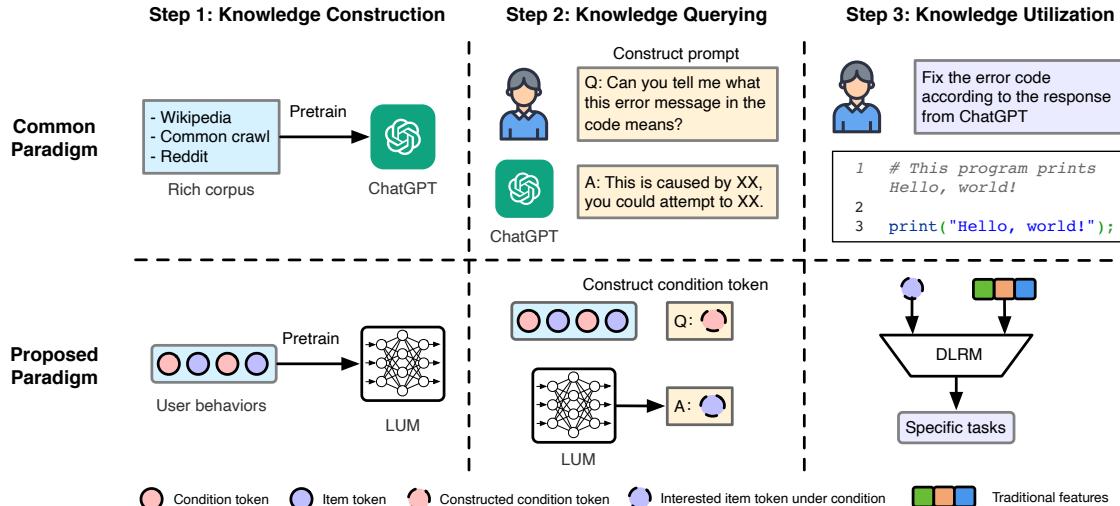


Figure 1: Intuitive insight from the common paradigm in using LLM to the proposed multi-step, generative-to-discriminative paradigm.

Even when deployment is feasible, these efficiency constraints continue to impede the scalability of these E2E models (Section 5.4). **(3) Lack of Flexibility:** E2E-GRs exhibit a notable limitation in their ability to adapt to the dynamic and evolving nature of business requirements. Specifically, the incorporation of new types of behavioral data, such as refund behaviors or behaviors associated with new scenarios, presents a significant challenge. The inherent structure of E2E-GRs necessitates that any modification to the input schema, including the addition of new elements, triggers a requirement for retraining the entire model. This rigidity not only complicates the integration of new features but also imposes substantial constraints on the system's adaptability. The need to retrain the model from scratch upon any schema alteration is both time-consuming and resource-intensive, thereby diminishing the practicality of E2E-GRs in industrial settings where rapid and flexible responses to changing conditions are often critical.

(4) Limited Compatibility: E2E-GRs are constructed using raw UBS with a transformer architecture. This approach inherently limits their compatibility with pre-existing industrial knowledge, such as explicit feature engineering and the parameter inheritance from DLRMs. Consequently, a significant performance gap often emerges between the developmental stages of these E2E-GRs and their real-world deployment in industrial settings (Section 5.3). This gap is particularly pronounced in applications where the online model has been refined over several years or even decades. This discrepancy underscores the need for more robust integration strategies to bridge the gap between theoretical advancements and practical applications.

In this paper, we rethink the critical question: **How can we effectively harness generative models to unlock scaling law in industrial settings?** This question compels us to revisit the common paradigm in using LLMs (Figure 1). Specifically, LLMs are initially trained in a generative manner on vast datasets, enabling them to assimilate a broad spectrum of knowledge. End-users then interact with these models through diverse prompts, querying them across a wide array of topics. Ultimately, the responses

generated by LLMs are utilized by end-users to make informed decisions across multiple domains. This **multi-step, generative-to-discriminative** nature of LLMs provides a foundational framework for leveraging generative models in practical applications. Drawing upon this insight, we propose a three-step paradigm for training a Large User Model (LUM) tailored for industrial use. These steps are outlined as follows (see Figure 1): (1) **Step 1: Knowledge Construction.** A LUM is introduced, utilizing a transformer architecture and pre-trained through generative learning. This model captures user interests and the collaborative relationships among items, thereby characterizing a comprehensive knowledge base. (2) **Step 2: Knowledge Querying.** In this phase, LUM is queried with predefined questions pertaining to user-specific information, facilitating the extraction of relevant insights. Intuitively, this process can be fundamentally conceptualized as a form of "prompt engineering" which is specifically designed to elicit an extensive knowledge. (3) **Step 3: Knowledge Utilization.** The outputs from LUM, obtained in Step 2, serve as supplementary features. These are integrated into traditional DLRMs to enhance their predictive accuracy and decision-making capabilities.

Generally speaking, we can benefit the three-step paradigm from (1) In the first step, the generative learning of LUM enables the exploration of scaling laws, which are critical for enhancing model performance. (2) The decoupled designed of our paradigm eliminates the constraints associated with continue streaming training or serving. This separation facilitates the implementation of caching strategies for the LUM, thereby mitigating efficiency limitations. (3) The third step ensures that DLRMs can meet the requirements for real-time learning, flexibility, and compatibility. This is achieved by integrating the previously learned LUM with DLRMs, thereby enhancing their adaptability to dynamic environments and ensuring seamless integration with existing systems.

The remaining challenge is to effectively transfer the learned data joint distribution $p(x, y)$ from the first step to downstream discriminative tasks, thereby addressing the aforementioned limitation. Ideally, the user knowledge encapsulated within LUM should

exhibit a substantial correlation with the discriminative tasks at hand. To achieve this, we introduce a novel tokenization strategy for UBS, wherein each item is expanded into two distinct tokens: a condition token and an item token (Figure 2 and Section 4.1.1). Subsequently, we redefine the autoregressive learning process of UBS from "*next-item prediction*" to "*next-condition-item prediction*". This reformulation enables us to seamlessly trigger the relevant knowledge from the LUM into the discriminative tasks by specifying various conditions during the second step of the process. Finally, we have applied this approach to both offline datasets and online industrial applications, achieving significant improvements (Section 5.2 and 5.6). Furthermore, the LUM demonstrates scaling properties similar to those observed in LLMs, allowing it to be successfully scaled up to 7 billion parameters while maintaining consistent performance enhancements. These findings underscore the robustness and adaptability of the LUM in diverse industrial application scenarios.

In summary, our contribution are summarized as follows:

- We introduce a pioneering three-step paradigm specifically tailored for industrial applications.
- We propose a large user model (LUM) that incorporates a "next-condition-item prediction" task, to bridge the gap between generative pre-training and discriminative applications.
- Comprehensive empirical evaluations demonstrate that the proposed three-step paradigm based LUM significantly enhances the performance of downstream tasks in both public and industrial settings. Notably, our experiments reveal a clear scaling law governing the performance of LUM, which underscores the importance of scaling up the model size for optimal results. Additionally, we have successfully deploy it on our industrial applications and achieve significant gains in an A/B test. This observation not only validates the effectiveness of our approach but also serves as a catalyst for further exploration and advancement in this research direction.

2 Related works

Deep Learning based Recommendation Models. Traditional DLRMs typically employ a deep neural network and can be broadly categorized into two primary classes: (1) Retrieval-Oriented Models: These models primarily focus on retrieving the most relevant items for users. They often utilize a two-tower architecture, such as EDB [9], or sequential models, like SASRec [10], BERT4Rec [19], and GRU4Rec[8] to effectively capture the correlations between users and items. (2) Click-Through Rate (CTR) Prediction Models: This category of models is designed to predict the click-through rate. Many of these models adopt an Embedding + Multi-Layer Perceptron (MLP) architecture. Some works, such as DeepFM [7] and xDeepFM [11], aim to build more complex interaction signals, while others, including DIN [26], DIEN [25], SIM [16] and TWIN [3], explore the significance of UBS. Despite the significant improvements achieved by these methods in RecSys, they generally fail to scale with increasing computational resources and do not fully leverage the advancements in large foundation models.

Generative Recommendation Models. To investigate the scaling laws, several studies have been conducted to emulate LLM architectures and construct autoregressive transformers through next-item prediction in an end-to-end manner [6, 23]. However, as discussed in Section 1, these approaches often make strong and

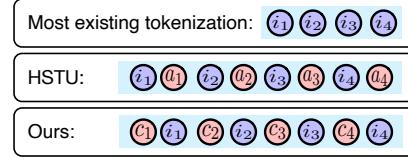


Figure 2: The comparison of different tokenizations

idealized assumptions that overlook the advantages of traditional DLRMs in terms of features, architecture, and practices. This results in a series of issues and limitations when these models are applied to industrial settings. In this paper, we propose a three-step paradigm with LUM to address these limitations. It is worth noting that there are also works that attempt to leverage the open-world knowledge in LLMs to build content-based recommendation models via end-to-end (E2E) or multi-step training [1, 4, 12, 22]. These efforts, however, fall outside the scope of our current study. Our focus is on developing a scalable model that captures collaborative filtering signals rather than content signals.

3 Preliminary

3.1 Traditional DLRMs

In the domain of RecSys, two primary tasks are identified: retrieval and ranking. For the purpose of this discussion, we will focus on a search scenario, though it is important to note that the paradigm proposed herein is equally applicable to other industrial applications. Given a user $u \in \mathcal{U}$, an item $i \in \mathcal{I}$, a search term $s \in \mathcal{S}$:

Retrieval task: This task is aimed at identifying a subset of candidate items from the corpus that align with the user's u 's preferences, as influenced by the query s . A conventional approach to the retrieval task employs a two-tower architecture [9], comprising a user-query tower $UEnc$ and an item tower $IEncl$. These towers may consist of any suitable neural network structures, such as MLPs. The user-query tower encodes the user u and the search term s into a unified embedding $e_{us}^r = UEnc(us)$, while the item tower generates an embedding for the item i denoted as $e_i^r = IEncl(i)$. Subsequently, contrastive learning techniques are utilized to refine the representations produced by these two towers.

Ranking Task: In contrast, the ranking task focuses on forecasting the likelihood of a user u clicking on an item i in response to a specific query s . This is mathematically formalized as: $\hat{y} = f(u, i, s)$ where \hat{y} represents the predicted CTR and f refers to the Embedding+MLP architecture.

3.2 E2E-GRs via Next-item Prediction

In the context of E2E-GRs, given a UBS for user u represented as $B_u = \{i_1, i_2, \dots, i_L\}$ where L denotes the length of the behavior sequence, the next-item prediction framework posits that the probability of an item i_k appearing next is conditionally dependent on the preceding items $\{i_1, i_2, \dots, i_{k-1}\}$. Consequently, the likelihood of the entire UBS B_u can be mathematically expressed as: $p(i_1, i_2, \dots, i_L) = \prod_{l=1}^L p(i_l | i_1, i_2, \dots, i_{l-1})$. The objective of the autoregressive learning process within E2E-GRs is to optimize the distribution $p_\theta(i_1, i_2, \dots, i_L)$, a method commonly referred to as "next-item prediction".

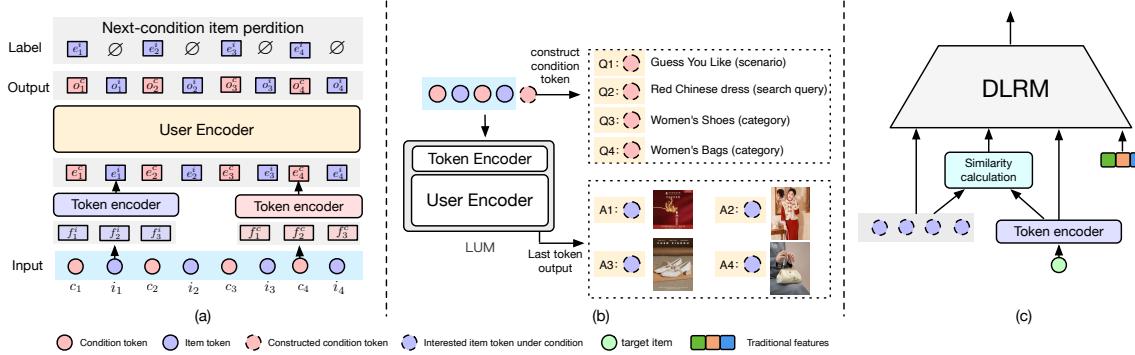


Figure 3: (a) The architecture of LUM. (b) An example of query knowledge from pre-trained LUM. (c) An example of utilizing knowledge in DLRMs.

4 Method

4.1 Step 1: Knowledge Construction via Pre-training LUM

4.1.1 Tokenization. In this study, we propose a novel approach to reformulating autoregressive modeling in UBS by transitioning from a traditional "next-item prediction" framework to a more sophisticated "next-condition-item prediction" paradigm. Specifically, each item i within the sequence B_u is decomposed into two distinct tokens: a condition token and an item token. Consequently, the UBS can be represented as a sequence of alternating tokens: $\{c_1, i_1, c_2, i_2, \dots, c_l, i_l\}$, where c_k denotes the condition token associated with the item i_k (as illustrated in Figure 2). To elucidate, consider a practical scenario where a user's behavior encompasses interactions with items across multiple scenarios, such as recommendation and search scenarios. In this context, the condition token c_i can be defined as a scenario token, thereby capturing the specific environment in which the item i was encountered. This approach facilitates a nuanced understanding of user preferences and behaviors across different scenarios. It is important to note that while other methods, such as HSTU [23], have introduced additional action tokens, their methodology differs fundamentally from our proposed framework. In HSTU, each item is extended as <item, action> where the action a_k is associated with the preceding item i_k , rather than the subsequent item (Figure 2). As a result, HSTU's strategy is limited in its ability to predict the next item based on varying conditions and fails to adequately capture user preferences across different aspects.

4.1.2 Architecture. The overarching framework of LUM is depicted in Figure 3 (a), presenting a hierarchical structure that encompasses both a Token Encoder and a User Encoder.

Token Encoder. The input tokens for LUM exhibit heterogeneity, characterized by two primary categories: condition token c and item token i . Moreover, individual tokens may carry diverse attribute features, such as ID, statistical, and content features associated with item tokens. The token encoder is devised to integrate these heterogeneous inputs into a unified token embedding. This process involves initially concatenating the various features or embeddings of each token, followed by a projection layer to consolidate and transform these features into a common representational space. Mathematically, this transformation is formalized as: $e^t = \text{proj}^t(\text{concat}(f_1^t; f_2^t; f_3^t; \dots); t \in \{i, c\})$ where f_k^t refers to

the features of item token i or condition token c , and e^t denotes the resultant token embedding. In this study, a linear projection is employed for proj^t unless otherwise specified.

User Encoder: The user encoder is structured to capture user preferences and the collaborative information among items. Specifically, the sequence of input tokens $\{c_1, i_1, c_2, i_2, \dots, c_L, i_L\}$ is represented as $\{e_1^c, e_1^i, e_2^c, e_2^i, \dots, e_L^c, e_L^i\}$ through the token encoder. Subsequently, as illustrated in Figure 3 (a), the user encoder utilizes a conventional autoregressive transformer architecture to process these embeddings. The final output of the user encoder is denoted as o_k^c , encapsulating the integrated information from the input sequence.

4.1.3 Next-condition-item Prediction. Unlike next-item prediction, which focuses on predicting the subsequent item directly, next-condition-item prediction is concerned with predicting the next item given a specific condition. This approach necessitates the application of an autoregressive loss solely on the output of the condition token to infer the next item. Consequently, the autoregressive likelihood for next-condition-item prediction can be formulated as follows:

$$p(c_1, i_1, c_2, i_2, \dots, c_L, i_L) = \prod_{l=1}^L p(i_l | c_1, i_1, c_2, i_2, \dots, i_{l-1}, c_l) \quad (1)$$

Furthermore, to enhance the optimization of $p_\theta(c_1, i_1, c_2, i_2, \dots, c_L, i_L)$ in practical industrial applications, we employ the InfoNCE loss function [15] and introduce a packing strategy.

InfoNCE Loss. In industrial applications, the vocabulary size of items can scale to billions, rendering the direct computation of generative probabilities over the entire set of items impractical. To address this challenge, we employ the InfoNCE loss for predicting the next conditional items. The InfoNCE loss can be mathematically formulated as follows:

$$\text{Loss} = - \sum_{l=1}^L \log \left(\frac{\exp(\text{sim}(o_{l-1}^c, e_l^i)))}{\exp(\text{sim}(o_{l-1}^c, e_l^i)) + \sum_{k=1}^K \exp(\text{sim}(o_{l-1}^c, e_k^i)))} \right) \quad (2)$$

where sim is the similarity function. For each item l , the other items within the same batch serve as negative samples and K is the number of negative items. e_k^i is the embedding of k -th negative items.

Packing. In practical applications, the lengths of UBSs exhibit significant variability among users. Indeed, the majority of UBS lengths

are substantially shorter than the predefined maximum length. Processing each UBS individually in such scenarios is computationally inefficient. Drawing inspiration from the packing strategies employed in the GPT series [17, 18, 24], we adopt a similar approach by grouping multiple UBSs into a single sequence, thereby maximizing the utilization of the available sequence length.

4.2 Step 2: Knowledge Querying with Given Conditions

In the Step 1, we construct the joint probability distribution denoted as $p(c_1, i_1, c_2, i_2, \dots, c_L, i_L)$. The next step involves extracting related knowledge from this established probability. Specifically, the tokenization methodology introduced in Section 4.1.1 facilitates the knowledge querying under various conditions. Given a query condition c_q , the conditional probability $p(i_q|c_1, i_1, c_2, i_2, \dots, c_L, i_L, c_q)$ can be computed to ascertain the likelihood of a user's interest in item i_q (see Figure 3 (b)). It is noteworthy that this approach to trigger conditional knowledge establishes a bridge between generative models and discriminative tasks, thereby enhancing effectiveness in industrial applications. The following examples illustrate the application of different condition tokens:

Example 1. When the condition token corresponds to a scenario token, the model can infer user interests across different scenarios.

Example 2. If the condition token represents a search query token within search scenarios, the model can deduce user interests based on varying search queries.

Example 3. In cases where the condition token signifies a category token, the model can ascertain user interests in different categories.

Furthermore, the model's flexibility allows for the simultaneous consideration of multiple types of conditions by incorporating additional condition features into the set $\{f'_1; f'_2; f'_3; \dots\}$. Empirical evidence demonstrates that integrating diverse conditions significantly enhances performance (see Section 5.3). In essence, this process can be conceptualized as a form of "**prompt engineering**" designed to elicit a wide range of knowledge.

Group query for efficiency. Moreover, given that a single user may respond multiple queries, each pertaining to the same UBS, processing these queries in isolation can lead to significant inefficiencies. This issue is exacerbated in practical scenarios where the number of users can easily scale to billions, leading to an extensive number of <user, query> pairs requiring inference. To mitigate this challenge, we introduce a novel group query strategy aimed at enhancing computational efficiency. As illustrated in Figure 4, all queries are concatenated into a single sequence, represented as $p(i_{q_1}, i_{q_2}, \dots | c_1, i_1, c_2, i_2, \dots, c_L, i_L, c_{q_1}, c_{q_2}, \dots)$. To ensure that the inference process remains coherent and accurate, we apply a masking mechanism to prevent attentional interactions between different query condition c_{q_j} . This approach allows the common prefix $\{c_1, i_1, c_2, i_2, \dots, c_L, i_L\}$ of various queries to be computed only once, while simultaneously querying the items i_{q_j} under different conditions. Empirical evaluations demonstrate that, with the implementation of the group query strategy, the inference process can be significantly accelerated by 78% (see Section 5.3).

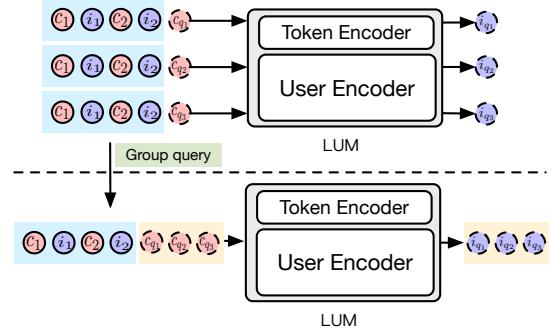


Figure 4: An example of group query.

4.3 Step 3: Knowledge Utilization in DLRMs

After Step 2, we acquire a set of N next-condition items $\{i_{q_1}, i_{q_2}, \dots, i_{q_N}\}$, with their corresponding outputs represented as $o_{q_n}^i$. Additionally, each item $i \in \mathcal{I}$ is encoded through the token encoder (Section 4.1.2), yielding an embedding denoted as e_i^i .

To enhance existing DLRMs, we propose two strategies (see Figure 3 (c)): (1) Direct Feature Incorporation: We integrate the outputs $o_{q_n}^i$ and the embedding e_i^i of target item i as fixed additional features into the DLRMs. This approach leverages the rich representations derived from the next-condition items directly. (2) Interest Matching via Similarity Measurement: We assess the alignment between the target item i and user interests by computing the similarity $sim(o_{q_n}^i, e_i^i)$ (as defined in Equ 2). This similarity score quantifies how well the target item matches the context provided by the next-condition items.

Formally, for retrieval tasks, the two-tower model can be reformulated as: $e_{us}^r = UEnc(us, \{o_{q_1}^i, o_{q_2}^i, \dots, o_{q_N}^i\})$ and $e_i^r = IEnc(i, e_i^i)$. For ranking tasks, the ranking model can be rewritten as: $\hat{y} = f(u, i, s, \{o_{q_n}^i, sim(o_{q_n}^i, e_i^i)\}_{n=1, \dots, N}, e_i^i)$. This framework effectively integrates contextual information and item embeddings to improve both retrieval and ranking performance in recommendation systems.

4.4 Discussion

The proposed three-step paradigm based on LUM addresses the four limitations mentioned in Section 1 as follows:

- **Addressing Limitations 1:** To address the first limitation, we design a generative-to-discriminative process. In Step 1, the joint distribution $p(c_1, i_1, c_2, i_2, \dots, c_L, i_L)$ is constructed through generative learning. Subsequently, the design of 'next-condition-item prediction' enables the triggering of relevant knowledge for RecSys tasks. Finally, discriminative learning in Step 3 is performed to meet the requirements of discriminative applications.

- **Addressing Limitations 2:** The model training and serving in Step 3 naturally support the efficiency requirements in industrial applications, as the backbone of the model follows the DLRM-style architecture. Additionally, the computational costs associated with Steps 1 and 2 are minimal due to the decomposed design, which allows these processes to be pre-computed and their results to be pre-stored.

- **Addressing Limitations 3:** The dynamic and evolving nature of business requirements can be naturally addressed in Step 3, similar to traditional DLRMs. Moreover, these dynamic requirements can be unified as various conditions, which aligns with our design.

Table 1: The statistic of datasets.

	#Interaction	#User	#Item
Amazon Books	914,014	694,897	686,623
MovieLen 1M (ML-1M)	9,810,868	6,040	3,883
MovieLen 20M (ML-20M)	17,253,665	138,493	27,278
Industrial Dataset	4 billion	0.1 billion	0.1 billion

for condition tokens. This means that regardless of changes in requirements, they can be consistently characterized as <condition token, item token>, enabling continuous training of LUM to adapt to new requirements.

- **Addressing Limitations 4:** In an industrial setting, the DLRM backbone in Step 3 can be set as the online model. This setup allows for the easy utilization of pre-existing industrial knowledge. Furthermore, the model can continuously benefit from advancements in DLRM technology.

5 Experiments

5.1 Experimental setting

Datasets. In this study, we utilize three public datasets and one industrial dataset to evaluate the performance of our proposed method, LUM. The public datasets include two benchmark datasets: MovieLens, which comprises two subsets (1M and 20M), and Amazon Books [23, 25]. The industrial dataset is sourced from the Taobao e-commerce platform. The statistical information for these datasets is summarized Table 1.

Baselines. To comprehensively evaluate the performance of LUM, we compare it against a variety of state-of-the-art models. For the traditional retrieval model, we use the two-tower architecture-based EDB [9]. For traditional ranking models, we consider DIN [26], DIEN [25], SIM [16], and TWIN [3]. Additionally, we compare LUM with E2E-GRs, specifically HSTU [23]. We also include the traditional sequential recommendation model SASRec [10], which adopts a transformer architecture to model UBS, as a baseline.

Training Details. To ensure a fair comparison, we adhere to the following training configurations. By default, we maintain a similar configuration for transformer-style models, including LUM, HSTU, and SASRec, to ensure comparable model sizes. The configurations for other DLRMs are set according to the recommendations provided in their respective original papers. All models are trained from scratch using the same set of features. The sequence length is set to 256 for public datasets and 4096 for the industrial dataset. For LUM, the backbone of DLRMs in step 3 is configured as SIM for ranking tasks and EDB for retrieval tasks in public datasets. For the industrial dataset, the backbone is set as the online model currently deployed in production.

5.2 Performance on Recommendation tasks

5.2.1 Performance on public datasets. We first conduct experiments on public datasets to evaluate the performance of LUM. The performance metric reported is AUC¹. The results are summarized in Table 2. From the results, we observe that LUM achieves significant improvements across all datasets. This indicates that the

Table 2: Performance on Public Datasets

Model	ML-1M	ML-20M	Amazon Books
SASRec	0.7295	0.7166	0.6699
HSTU	0.7533	0.7463	0.6712
DIN	0.7455	0.7299	0.6139
DIEN	0.7527	0.7319	0.6130
SIM	0.7579	0.7341	0.6551
TWIN	0.7539	0.7331	0.6538
LUM	0.7615	0.7483	0.6727

Table 3: Overall performance in industrial settings. Imp. denotes the improvements relative to the best baseline.

Model	Ranking	Retrieval	
	AUC	R@10	R@50
SASRec	0.7322	0.2560	0.4740
DIN	0.7336	-	-
HSTU	0.7334	0.2594	0.4781
Online Model	0.7338	0.2482	0.4651
LUM	0.7514	0.2727	0.4915
Imp.	+0.0176	+0.0133	+0.0134

three-step paradigm based LUM effectively captures a wide range of user interests and enhances the predictive accuracy of DLRMs.

5.2.2 Performance in industrial setting. In this section, we compare the performance of LUM against both DLRMs and E2E-GRs in an industrial setting. For the baseline, we use the online model in our applications, which follows an Embedding+MLP architecture for ranking and a two-tower architecture for retrieval. We report AUC for ranking and Recall@K (R@K) for retrieval. For LUM, the backbone DLRM in Step 3 is also set to the online model. Additionally, we include traditional state-of-the-art DLRMs (SASRec [10] and DIN [26]) and an E2E-GR (HSTU [23]) as baselines for comparison. To ensure a fair comparison, all models use the same set of features. We set the maximum sequence length to 4096 for all models and train them from scratch. The results are summarized in Table 3. LUM achieves significant improvements over the best baseline, with a +0.0176 increase in AUC, a +0.0133 increase in R@10, and a +0.0134 increase in R@50. The substantial improvements can be attributed primarily to the generative-to-discriminative design of our proposed paradigm.

5.3 Effectiveness Evaluation

In this section, we examine the advantages of the proposed three-step paradigm based LUM.

Impact on various DLRMs. Owing to the decomposed design of our proposed paradigm, LUM can be universally integrated into any DLRM during Step 3. To evaluate the effectiveness of LUM, we examine its impact on various DLRMs by comparing the performance of the original DLRMs (denoted as Base) and DLRMs augmented with LUM (denoted as Base+LUM) on the ranking task. The results are summarized in Table 4. The results demonstrate

¹Note 0.001 absolute AUC gain is regarded as significant for the ranking task[26]

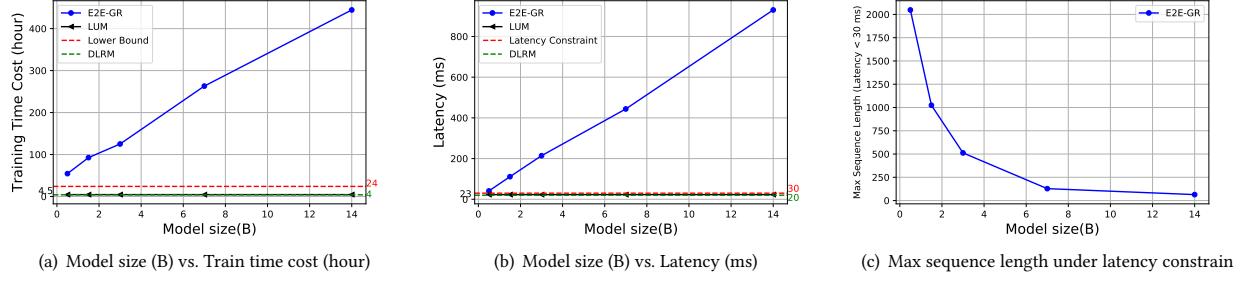


Figure 5: The Results of Efficiency Evaluation.

Table 4: The performance of LUM on various DLRMs. "Ave" denotes the average results across all cases, and "Imp." indicates the improvement of "Base+LUM" compared to "Base".

	DIN	DIEN	SIM	TWIN	Ave
ML-1M	0.7455	0.7527	0.7579	0.7539	0.7525
	0.7472	0.7604	0.7615	0.7675	0.7592
	0.0016	0.0078	0.0036	0.0136	0.0067
ML-20M	0.7299	0.7319	0.7341	0.7331	0.7323
	0.7413	0.7361	0.7483	0.7422	0.7420
	0.0114	0.0042	0.0142	0.0090	0.0097
Amazon Book	0.6139	0.6130	0.6551	0.6538	0.6340
	0.6261	0.6194	0.6727	0.6591	0.6443
	+0.0122	+0.0063	+0.0176	+0.0053	+0.0103

Table 5: Results of effectiveness evaluation.

Model	Training Mode	AUC
E2E-GR	scratch	0.7334
DLRM	scratch	0.7338
LUM	scratch	0.7514
DLRM (feature)	warming up	0.7541
DLRM (param)	warming up	0.7525
DLRM (param+feature)	warming up	0.7777
LUM (feature)	warming up	0.7659
LUM (param)	warming up	0.7620
LUM (param+feature)	warming up	0.7794
LUM (w/o condition token)	scratch	0.7416
LUM (multi-conditions)	scratch	0.7545
LUM (direct feature)	scratch	0.7402
LUM (direct feature+interest matching)	scratch	0.7514

that, with the assistance of LUM, all methods achieve significant improvements ranging from +0.0053 to +0.0176. These findings highlight the versatility and effectiveness of LUM in improving the predictive accuracy of various DLRMs.

Impact of warming up setting in industrial applications. In industrial applications, online models typically incorporate complex feature engineering and are continually trained on billions of data per day. Existing E2E-GRs often ignore the knowledge derived from these online models, which can significantly challenge their performance, especially in applications where the online model has been developed over several years or even decades. LUM, however,

offers excellent compatibility, allowing it to be trained in a warming up setting. In this section, we investigate the impact of this compatibility by implementing various versions of LUM, and comparing them with an E2E-GR (HSTU) and a DLRM (online model). We use AUC on an industrial dataset as the performance metric.

The results are shown in Table 5 (Note "scratch" refers to models trained from scratch with the same set of features. "feature" indicates the use of complex feature engineering. "param" refers to initializing model parameters from the continually daily trained online model). From Table 5, we can draw the following conclusions: (1) Compared to LUM, the models under warming up settings, including LUM (feature), LUM (param), and LUM (feature+param), achieve noticeable gains in AUC, ranging from +0.0106 to +0.028. This highlights the significant benefit of leveraging existing knowledge from the online model. (2) While HSTU outperforms the DLRM in scratch setting, it still exhibits a large performance gap (0.7334 vs. 0.7777) compared to the DLRM (param+feature). This underscores the importance of being compatible with the online model. The limited compatibility of E2E-GRs may hinder their practical deployment in real-world applications.

Impact of the proposed tokenization. We evaluate the effectiveness of the proposed tokenization. Specifically, we develop a LUM (w/o condition token), and report AUC in Table 5. The results show that compared with LUM (w/o condition token), LUM achieves better performance due to a better understanding of UBS via given conditions. Furthermore, we also evaluate the effect of using multiply conditions (Section 4.2) including scenario condition and search term condition, denoted as LUM (multi-conditions). From Table 5, we can observe adding more conditions can further improve the performance, which shows the potential in terms of performance.

Impact of knowledge utilization. We evaluate the different strategies for utilizing knowledge in Step 3. The results are presented in Table 5. Here, "direct feature" and "interest matching" refer to different strategies of knowledge utilization, as detailed in Section 4.3. Both LUM (direct feature) and LUM (direct feature + interest matching) achieve significant improvements over DLRM, demonstrating the effectiveness of the proposed strategies.

Impact of packing and group query. Packing and group query are designed to accelerate the processes in Step 1 and Step 2, respectively. Table 6 demonstrates the efficiency gains of these strategies in an industrial setting, where Step 1 and Step 2 can be sped up by 82% and 78%, respectively.

Table 6: The impact of group query and packing. Imp. refers to the relative improvements.

Phase	Time cost (hour)	Imp.
LUM (w/o packing)	Step 1	-
LUM (w/ packing)	Step 1	+82%
LUM (w/o group query)	Step 2	-
LUM (w/ group query)	Step 2	+78%

5.4 Efficiency Evaluation

Training Efficiency. Figure 5 (a) illustrates the training time costs of different models on one day's data in an industrial setting. For E2E-GR, we followed HSTU's [23] optimization and trained it from impression-level to user-level. For DLRM, we used the online model in our application as the baseline. For LUM, the backbone DLRM is also set as the online model. Since Steps 1 and 2 in LUM can be pre-processed (4.4), their time costs are excluded when training the downstream model in Step 3. The sequence length is set to 4096, and the model sizes of LUM and E2E-GR range from 0.5 billion to 14 billion parameters. All models are trained on 128 GPUs. Additionally, the lower bound time cost is set to 24 hours, as continuous training in practice requires models to process daily data within this timeframe. Overall, LUM achieves similar training time costs to DLRM and remains relatively insensitive to model size due to its decomposed three-step paradigm. This characteristic unlocks the potential for scaling laws during training in industrial settings. In contrast, E2E-GR is 12× to 98× slower than LUM. None of the E2E-GR models (across different sizes) meet the training cost requirement of completing within 24 hours. To match the throughput of LUM, E2E-GR would require 12× to 98× GPUs, and to meet the lower bound requirement, it would need 2× to 18× GPUs.

Serving Efficiency. Figures 5 (b) and (c) illustrate the latency of different models during online serving. For E2E-GR, we followed HSTU's M-FALCON implementation [23]². DLRM is set as the online model for both the baseline and the backbone ranking model of LUM. The latency of LUM during Step 3 is evaluated, as Step 1 and 2 can be pre-computed (Section 4.4). In Figure 5 (b), the sequence length is set to 4096, and the model sizes of LUM and E2E-GR range from 0.5 billion to 14 billion parameters. The latency constraint is set to less than 30 milliseconds (ms), and the number of ranking candidates is approximately 100 in our case. Overall, the latency of LUM is independent of the model size, allowing us to scale up LUM without violating the latency constraint. On the contrary, E2E-GR fails to provide timely responses (less than 30 ms) even for a small model (0.5 billion parameters). To further investigate, we attempted to reduce the sequence length of E2E-GR to meet the latency constraint (Figure 5 (c)). Disappointingly, the maximum sequence length that meets the latency constraint (<30 ms) is only 64 when using a 14 billion parameter model, which is far from the typical setting (64 times smaller in our case). These results demonstrate that, while E2E-GR may exhibit scaling laws in offline

²Note that, unlike LLMs where the trained parameters are seldom updated during serving, in industrial settings, the parameters of the ranking model are updated in real-time, meaning results or intermediate results cannot be shared across requests. Therefore, the cache strategy in M-FALCON is not applicable here.

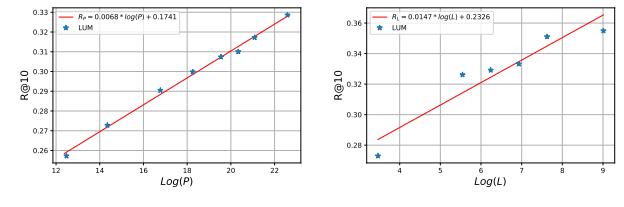


Figure 6: Scaling law for LUM.

performance, the practical application of large-scale models is severely limited in industrial settings due to latency constraints. In contrast, LUM's decoupled architecture ensures consistent latency performance, making it a more feasible and scalable solution for real-time industrial applications.

5.5 Scaling Law for LUM

Following the protocols established in [17, 18], we examine whether our LUM model complies to similar scaling law. Specifically, we focus on evaluating the scaling law with respect to model parameters and sequence length. For model parameters, to evaluate the impact of model parameters, we maintain a constant sequence length of 4096 and train models across a range of sizes, from 19 million to 7 billion parameters. For sequence length, to evaluate the impact of sequence length, we train models with varying sequence lengths, from 256 to 8192, under a fixed parameter size of 300 million. The results are plotted in Figure 6, where we observe a clear power-law scaling trend, consistent with previous findings [17, 18, 23]. The power-law scaling laws can be expressed as:

$$R_P = 0.0068 \cdot \log(P) + 0.1741 \quad (3)$$

$$R_L = 0.0147 \cdot \log(L) + 0.2326 \quad (4)$$

where R_P and R_L refer to the R@10 metric for different model sizes and sequence lengths, respectively. P denotes the model size, and L denotes the sequence length. These results confirm the strong scalability of LUM, demonstrating that increasing the model size and sequence length can continuously improve the model's performance. This finding underscores the potential of LUM to achieve higher performance as it scales, making it a promising approach for large-scale industrial applications.

5.6 Online Results

To evaluate the effectiveness of LUM in an industrial setting, we implemented it in the sponsored search system of Taobao, the largest e-commerce platform in China. As discussed in Section 4.4, our evaluation process involves several key steps: (1) Offline, we first pretrained LUM and pre-triggered the necessary knowledge under various conditions. (2) The responses generated by LUM in Step 2 are stored, allowing them to be directly utilized during online serving. This pre-computation helps meet the strict latency constraints required in real-time industrial applications. (3) Finally, we conducted online A/B experiments to test LUM in the ranking task. Key performance metrics, CTR and RPM (Revenue Per Mile), demonstrated a significant improvement of 2.9% and 1.2% respectively. These findings highlight the practical benefits of LUM,

demonstrating its ability to improve user engagement and business outcomes in large-scale e-commerce platforms.

6 Conclusion

In conclusion, LUM effectively unlocks scaling laws in industrial recommendation systems through a three-step paradigm. The decomposed design and next-condition-item prediction ensure that LUM can be efficiently scaled and deployed, leading to significant performance improvements in real-world applications. Our experimental and deployment results demonstrate the robustness and practicality of LUM, making it valuable for enhancing user engagement and business outcomes in large-scale e-commerce platforms.

References

- [1] 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.
- [2] JM Bernardo, MJ Bayarri, JO Berger, AP Dawid, D Heckerman, AFM Smith, and M West. 2007. Generative or discriminative? getting the best of both worlds. *Bayesian statistics 8*, 3 (2007), 3–24.
- [3] Jianxin Chang, Chenbin Zhang, Zhiyi Fu, Xiaoxue Zang, Lin Guan, Jing Lu, Yiqun Hui, Dewei Leng, Yanan Niu, Yang Song, et al. 2023. TWIN: Two-stage interest network for lifelong user behavior modeling in CTR prediction at kuaishou. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3785–3794.
- [4] Junyi Chen, Lu Chi, Bingyue Peng, and Zehuan Yuan. 2024. Hllm: Enhancing sequential recommendations via hierarchical large language models for item and user modeling. *arXiv preprint arXiv:2409.12740* (2024).
- [5] Yijie Ding, Yupeng Hou, Jiacheng Li, and Julian McAuley. 2024. Inductive Generative Recommendation via Retrieval-based Speculation. *arXiv preprint arXiv:2410.02939* (2024).
- [6] 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.
- [7] Huirong Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).
- [8] B Hidasi. 2015. Session-based Recommendations with Recurrent Neural Networks. *arXiv preprint arXiv:1511.06939* (2015).
- [9] 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.
- [10] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.
- [11] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1754–1763.
- [12] Jianghao Lin, Bo Chen, Hangyu Wang, Yunjia Xi, Yanru Qu, Xinyi Dai, Kangning Zhang, Ruiming Tang, Yong Yu, and Weinan Zhang. 2024. ClickPrompt: CTR Models are Strong Prompt Generators for Adapting Language Models to CTR Prediction. In *Proceedings of the ACM on Web Conference 2024*. 3319–3330.
- [13] Zihan Liu, Yupeng Hou, and Julian McAuley. 2024. Multi-Behavior Generative Recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 1575–1585.
- [14] Andrew Ng and Michael Jordan. 2001. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems* 14 (2001).
- [15] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).
- [16] Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. 2020. Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2685–2692.
- [17] Alec Radford. 2018. Improving language understanding by generative pre-training. (2018).
- [18] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.
- [19] 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.
- [20] Le Yan, Zhen Qin, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2022. Scale Calibration of Deep Ranking Models. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, Aidong Zhang and Huzeifa Rangwala (Eds.). ACM, 4300–4309. <https://doi.org/10.1145/3534678.3539072>
- [21] Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898* (2017).
- [22] Xiaohan Yu, Li Zhang, Xin Zhao, Yue Wang, and Zhongrui Ma. 2024. RA-Rec: An Efficient ID Representation Alignment Framework for LLM-based Recommendation. *arXiv preprint arXiv:2402.04527* (2024).
- [23] 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).
- [24] Yu Zhao, Yuanbin Qu, Konrad Staniszewski, Szymon Tworkowski, Wei Liu, Piotr Miłos, Yuxiang Wu, and Pasquale Minervini. 2024. Analysing The Impact of Sequence Composition on Language Model Pre-Training. *arXiv preprint arXiv:2402.13991* (2024).
- [25] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5941–5948.
- [26] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1059–1068.