Towards Large-scale Generative Ranking

Yanhua Huang, Yuqi Chen, Xiong Cao, Rui Yang, Mingliang Qi, Yinghao Zhu
Qingchang Han, Yaowei Liu, Zhaoyu Liu, Xuefeng Yao, Yuting Jia, Leilei Ma, Yinqi Zhang
Taoyu Zhu, Liujie Zhang, Lei Chen, Weihang Chen, Min Zhu, Ruiwen Xu, Lei Zhang
{yanhuahuang,ruiwenxu}@xiaohongshu.com
Xiaohongshu Inc.
Shanghai, China

Abstract

Generative recommendation has recently emerged as a promising paradigm in information retrieval. However, generative ranking systems are still understudied, particularly with respect to their effectiveness and feasibility in large-scale industrial settings. This paper investigates this topic at the ranking stage of Xiaohongshu's Explore Feed, a recommender system that serves hundreds of millions of users. Specifically, we first examine how generative ranking outperforms current industrial recommenders. Through theoretical and empirical analyses, we find that the primary improvement in effectiveness stems from the generative architecture, rather than the training paradigm. To facilitate efficient deployment of generative ranking, we introduce GenRank, a novel generative architecture for ranking. We validate the effectiveness and efficiency of our solution through online A/B experiments. The results show that GenRank achieves significant improvements in user satisfaction with nearly equivalent computational resources compared to the existing production system.

Keywords

Information Retrieval; Generative Recommendation; Large-scale Ranking;

ACM Reference Format:

Yanhua Huang, Yuqi Chen, Xiong Cao, Rui Yang, Mingliang Qi, Yinghao Zhu, Qingchang Han, Yaowei Liu, Zhaoyu Liu, Xuefeng Yao, Yuting Jia, Leilei Ma, Yinqi Zhang, Taoyu Zhu, Liujie Zhang, Lei Chen, Weihang Chen, Min Zhu, Ruiwen Xu, Lei Zhang . 2025. Towards Large-scale Generative Ranking. In *Proceedings of Second Workshop on Generative AI for Recommender Systems and Personalization (KDD '25).* ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Recommender systems are essential components of social media platforms, enabling users to browse and engage with personalized item suggestions [3, 4, 7, 23]. To balance efficiency and effectiveness, industrial recommender systems typically employ a cascade pipeline [7, 23], consisting of four stages, as shown in Figure 1 (right). The retrieval stage initially selects tens of thousands of candidates from billions of items. This is followed by the pre-ranking stage, which performs coarse matching to narrow the candidate set to hundreds. The ranking stage then makes precise predictions for each candidate. Finally, the policy stage re-ranks dozens of candidates based on sequential information and commercial considerations to produce the final recommendation.

In modern recommender systems, the ranking stage generally follows the MLP & Embedding paradigm [4], where sequential modeling has achieved notable success in capturing user interests [8, 10, 16, 29]. The advent of generative recommendation has further enhanced the capabilities of sequential approaches. Unlike traditional approaches, generative recommendations formulate the recommendation problem as a sequence generation task [13, 23], directly predicting target behaviors from historical user actions. Rajput et al. [13] propose achieving generative retrieval by quantizing items with hierarchical semantic IDs. Yang et al. [22] further introduce a coarse-to-fine generation process to address information loss caused by quantization. Despite their novelty, generative recommenders for ranking tasks remain understudied, especially in large-scale industrial contexts.

This paper investigates generative ranking systems in large-scale industrial scenarios. In particular, we first analyze potential sources of effectiveness in generative recommendations and then conduct experiments to validate our hypotheses using existing generative recommenders [23]. Experimental results demonstrate that the generative architecture is critical to achieving strong performance. However, current generative architectures tend to be inefficient, particularly in large-scale settings. To address this issue, we propose a novel architecture, GenRank, to meet the requirements of large-scale training and inference. Online A/B experiments on the Explore Feed in Xiaohongshu[1] (Figure 1 (left)), a recommender system serving hundreds of millions of users, demonstrate the effectiveness and efficiency of our proposed solution.

The main contributions of this paper are summarized as follows:

1. We identify and analyze the sources of effectiveness in generative recommendation, highlighting the critical role of generative architecture in overall performance.
2. We propose an efficient generative architecture, specifically designed for industrial scenarios, which includes an action-oriented sequence organization approach and novel strategies for position and time biases.
3. We conduct large-scale online A/B testing to demonstrate the effectiveness and feasibility of generative ranking in industrial recommender systems.

KDD '25, Toronto, Canada

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

[1] Also known as RedNote: https://www.xiaohongshu.com/explore.

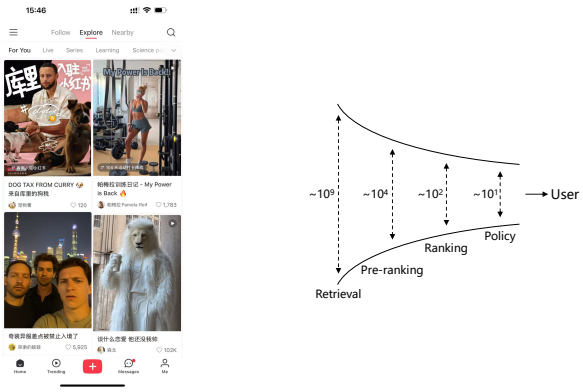

Figure 1: Left: A screenshot of Xiaohongshu's Explore Feed product. Right: Illustration of cascade pipeline for industrial recommender systems, where each stage needs to process a large number of items.

2 Problem Setup

This paper studies generative recommendations at the ranking stage. Here, the recommender system is required to make predictions for a set of predefined tasks, such as predicting the click-through probability or the expected duration a user spends when presented with a candidate item. To construct our dataset for offline experiments, we collect hundreds of billions of item exposure logs from Xiaohongshu's Explore Feed over 15 days. There are three types of input features:

- Categorical features: user ID, item ID, user historical behaviors, hashtags, etc.
- Numerical features: user age, item publish time, number of author fans, etc.
- Frozen embeddings: multi-modal item embeddings, graph-based author embeddings, etc.

Following prior work [4, 26], numerical features are discretized into categorical features using predefined boundaries, while categorical features are transformed into dense embeddings via embedding tables. The frozen embeddings provided by pre-trained models are treated as side information, offering prior knowledge relevant to the features they are associated with. We use the area under the ROC curve (AUC) as the offline evaluation metric. Notably, an absolute increase of 0.0010 in AUC for the main tasks is considered significant in our settings, as it typically yields a 0.5% improvement in topline metrics for hundreds of millions of users online.

3 Source of Effectiveness in Generative Recommendation

There have been numerous works on generative recommendation [13, 14, 22, 23]. However, the effectiveness of generative ranking, particularly in large-scale industrial settings, has been underexplored. To better understand the factors contributing to the effectiveness of generative ranking, we conduct experiments from two perspectives:

1. The generative recommendation paradigm diverges from conventional approaches through distinct underlying mechanisms. We are especially interested in identifying mechanisms where minor modifications cause major performance drops, as these may be critical to the success of generative ranking methods.
2. The current ranking paradigm integrates several well-established modules, such as SIM [10] and content embeddings [7, 27]. We examine key modules that show marked performance differences in generative settings, providing valuable insights for future studies.

Specifically, we choose HSTU [23] as the baseline model to present our findings from the above perspectives. By default, the number of blocks is 3, the number of attention heads is 8, and the hidden dimension is 768. Each user sequence has a maximum length of 480, including both historical behaviors and candidate items. We use the mixed precision training strategy on NVIDIA H20 GPUs.

3.1 Key Mechanisms in Generative Paradigm

In contrast to conventional paradigms that learn sophisticated feature interactions from historical behaviors, generative recommendation reformulates the ranking as a sequential transduction task [23]. In this context, the generative ranking differs significantly in two ways: the manner of sequential interactions and the organization of training samples.

The manner of sequential interactions in generative ranking is auto-regressive. Note that HSTU computes the loss only at positions corresponding to candidate items, as shown in Figure 2(a). This approach can be regarded as supervised fine-tuning, where the user information and candidate items serve as the input prompt. One reason modern LLMs employ the auto-regressive manner during supervised fine-tuning is to retain abilities acquired during pre-training. However, generative ranking does not involve a pre-training stage. This raises the question: Is the auto-regressive manner truly necessary for generative ranking?

To investigate this question, we conduct two sets of experiments. In the first set, we compute the loss at positions corresponding to historical behaviors. We observe a decrease in AUC of more than 0.0100, even when only a small number of historical positions are included. We attribute this to the one-epoch issue described in Zhang et al. [28], in which the model learns incorrect patterns from sparse features. In the second set, we replace the causal mask with a fully visible mask at historical positions. This modification is analogous to the T5 model [12], in which the attention mask maximizes feature interaction across the prompt. However, this change causes an AUC drop of more than 0.0015, and the decline grows more significant with larger model sizes. These results support the conclusion that the auto-regressive manner is critical to the effectiveness of generative ranking.

The organization of training samples in conventional paradigms is typically point-wise; that is, each training sample corresponds to an item exposure log. In contrast, generative ranking groups the temporally adjacent behaviors of a user into a single training sample. We hypothesize two possible benefits of this organization. First, since two exposure logs from the same request overlap significantly in features (particularly user features), processing them in the same

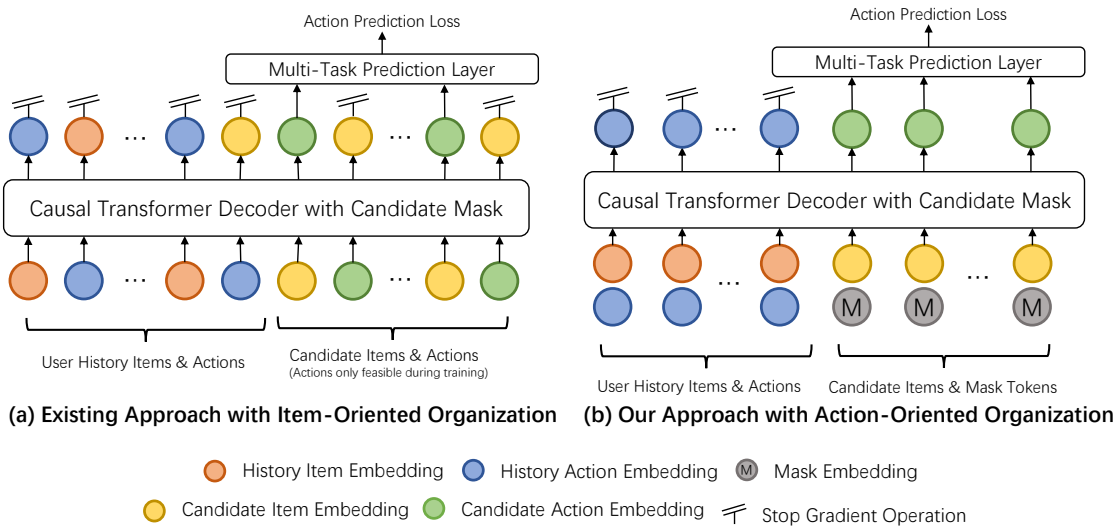

Figure 2: Model architecture of GenRank. Compared to existing approaches, e.g., HSTU [23], which adopt an item-oriented organization, our solution adopts an action-oriented organization.

batch improves gradient estimation stability. Second, we consider a practical viewpoint. In large-scale online distributed training, the order in which samples are processed does not strictly follow the actual temporal order, potentially causing information leakage. Under such conditions, the model can deduce the user preference for an item from historical behavior features prior to observing its exposure log during training. The organization in generative ranking helps mitigate the risk of training later-occurring samples before earlier ones.

However, our empirical results do not strongly support these two hypotheses. Specifically, we train the generative recommender with grouped training samples but in a point-wise order to simulate conventional training. This approach results in only a slight decrease in AUC. Therefore, we conclude that the effectiveness of generative recommendations primarily stems from the architecture, rather than the way training samples are organized.

3.2 Module Performance Comparison across Paradigms

To compare the influence of modules between the two paradigms, we perform experiments measuring the performance gains achieved by various modules. In particular, we select four important modules that are commonly employed in industrial ranking systems: SIM [10] for sequential modeling, PPNet [3] for personalized representation learning, content embeddings [7, 25, 27] for prior knowledge, and PLE [17] for multi-task learning. The results show that SIM, PPNet, and PLE achieve comparable improvements in both paradigms, suggesting that the generative paradigm is compatible with these modules. Furthermore, we observe that content embeddings yield over twice the AUC improvement under the generative paradigm. We attribute this enhancement to the architectural consistency between the generative training of content embeddings and their application in downstream tasks, allowing optimal utilization of their capabilities.

We also study the impact of feature engineering, which is critical for the performance of industrial recommendations [6]. HSTU [23] proposes to remove these features because generative recommenders can express statistical patterns sufficiently. Our experiments show that while the majority of features provide negligible benefits to generative architectures, certain real-time statistical features, especially window-based ones, remain remarkably effective at enhancing performance. We posit that these features offer direct signals to the model, allowing the generative architecture to learn complex patterns. Notably, the significant computational overhead associated with feature engineering limits ranking models' ability to process large candidate sets in real time. Generative architectures address this limitation through their minimal feature engineering requirements, thereby enhancing inference scalability. Furthermore, the KV cache mechanism enables generative architectures to scale more efficiently with increasing candidate set sizes [23]. We envision that with continued reductions in computational overhead, generative architectures could potentially unify ranking and pre-ranking stages in future systems.

4 Efficient Generative Ranking in Industrial Scenarios

The previous section highlights the importance of architecture in generative ranking. Not only is it crucial for performance, but it also impacts the overall design of future recommendation systems. This section introduces a novel generative architecture, GenRank, to enable efficient training and inference on large-scale ranking tasks. GenRank differs from existing work in two ways: item-action organization (Section 4.1) and position & time biases (Section 4.2).

Table 1 summarizes our empirical results in training performance. We use HSTU [23] as the baseline method. Transitioning to

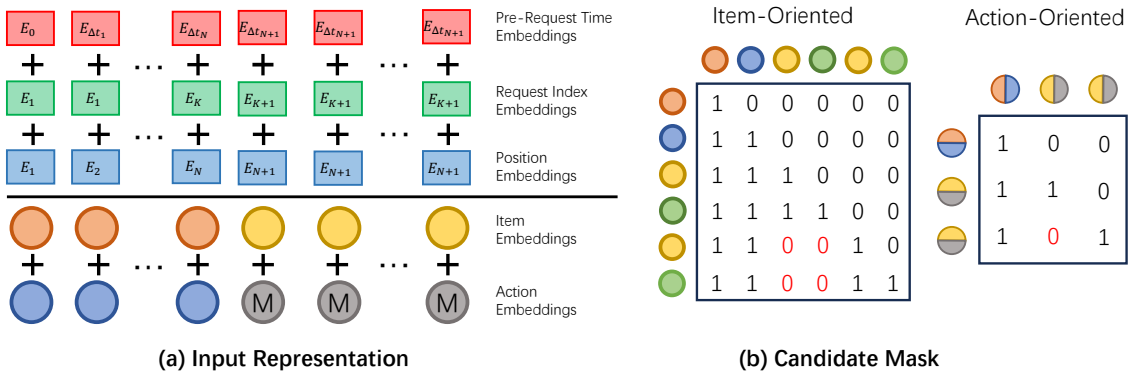

Figure 3: Illustration of input representation and candidate mask. (a) The input representation of GenRank contains five kinds of embeddings. (b) Candidate Mask: We depict the mask structure for a user's sequence with one historical behavior and two candidate items.

Table 1: Ablation study of GenRank.

Variant	Speed-Up	AUC Diff
Baseline (HSTU)	/	/
+ Action-oriented Organization	+78.7%	-0.0003
+ Proposed Position & Time Biases	+25.0%	+0.0009
+ All (GenRank)	+94.8%	+0.0006

the action-oriented organization yields a speed-up of 78.7%, while adopting the proposed position & time biases yields a speed-up of 25.0%. In general, GenRank achieves a total speed-up of 94.8% during training, with a slight improvement in AUC on the test set.

4.1 Item-Action Organization

Conventional sequential recommendation methods typically construct models by treating individual items as fundamental units [2, 8, 15, 16, 29], an organizational framework we term the item-oriented architecture. To adapt these approaches to meet the action-aware formulation for ranking tasks, HSTU [23] treats action tokens as an additional modality in the sequence. As illustrated in Figure 2(a), it interleaves items and actions in a single sequence, thus enabling the model to predict either an item or an action based on the contextualized sequence. Although such approaches can support retrieval and ranking tasks within a unified framework, they introduce substantial overhead for ranking because the sequence length is doubled.

To address this limitation, we propose a new perspective: **we treat items as positional information and focus on iteratively predicting the actions associated with each item**, which we refer to as the action-oriented organization. In this paradigm, actions become the fundamental units in sequence generation, while items serve as contextual signals to guide the generative process, as depicted in Figure 2(b). This approach focuses on action prediction and offers significant advantages in terms of efficiency. This design halves the attention mechanism's input sequence length, cutting attention costs by 75% and linear projection costs by 50%.

Formally, we consider a list of N user tokens $x_1, x_2, ..., x_N$, ordered chronologically, where $x_i \in \mathcal{X}$ (the set of items). For each item x_i, there is an associated action $a_i \in \mathcal{A}$ (the set of actions), which occurs at timestamp t_i. Thus, the sequence of actions is $a_1, a_2, ..., a_N$ and the corresponding timestamps are $t_1, t_2, ..., t_N$. In our setup, the model learns to approximate the distribution $p(a_k|x_1, a_1, ..., x_k)$. To implement an action-oriented generative ranking, each input token combines both item and action embeddings, as shown in Figure 3(a). For each position in the user's history sequence, the token embedding is obtained by the sum of the item embedding and the action embedding, i.e., $e_i = \varphi(x_i) + \phi(a_i)$, where $\varphi(\cdot)$ and $\phi(\cdot)$ denote the item and action embedding modules, respectively. Our task is to predict the user's action on the next candidate item. To achieve this, the token embedding of a candidate item is given by $e_j = \varphi(x_j) + M$, where M is a mask action embedding. Note that, to prevent information leakage between candidates, a candidate mask is applied, as illustrated on the right side of Figure 3(b).

4.2 Position & Time Biases

HSTU [23] utilizes a learnable relative attention bias to encode position and time information. Although this design is critical for performance [5], it introduces a computational bottleneck: the I/O operations for attention biases scale quadratically with sequence length, incurring significant overhead as the context window grows. This inefficiency motivates us to design new position & time biases that significantly reduce system cost. Specifically, we present a comprehensive design for position and time embeddings, requiring only linear I/O operations, including:

- Position Embeddings: A learnable positional embedding is used to record the index of items in the user sequence, denoted as $E_{pe,i} = \Omega_{pe}(i)$. To ensure consistency between training and inference, candidate items within the same request share the same position.
- Request Index Embeddings: In practice, a user can interact with multiple items within a single request. We treat all

Table 2: Online A/B test result in the Explore Feed scenario.

	Time Spent	Reads	Engagements	LT7
Improvement	+0.3345%	+0.6325%	+1.2474%	+0.1481%

items belonging to the same request as a group and define the request index embedding $E_{ri,i}$ as $\Omega_{ri}(|\{t_1, ..., t_i\}|)$, where $|\cdot|$ represents the cardinality.

- Pre-Request Time Embeddings: This embedding captures the bucketed time difference between each item and the time of the previous request, reflecting the user's activity level. Specifically, it is defined as $E_{rt,i} = \Omega_{rt}(\text{bucket}(t_i - \max_{t_j<t_i} t_j))$.

The above design introduces minimal training overhead while preserving positional and temporal information. Finally, the input representation fed into the subsequent network is:

$$e_i^{(p,t)} = \varphi(x_i) + \phi(a_i) + E_{pe,i} + E_{ri,i} + E_{rt,i} \ . \tag{1}$$

Additionally, a critical limitation of the above position and time embeddings is the lack of interaction between time and position information. To address this, we propose to employ a parameter-free bias, ALiBi [11], as the relative position & time biases in attention mechanisms. ALiBi has two main advantages. It penalizes attention scores between distant query-key pairs, with the penalty increasing as the distance between a key action token and a query action token grows. We believe this design is more aligned with the pattern of user interest modeling. Moreover, ALiBi is parameter-free, i.e., it does not require $O(N^2)$ memory access overhead or gradient backpropagation. By fusing ALiBi into the flash attention [21], we incur only minimal computational cost.

5 Related Work

5.1 Generative Recommendation

Generative recommendation has emerged as a promising paradigm in information retrieval [1, 13, 19, 20, 22]. Unlike traditional recommendation approaches, generative recommendation aims to directly generate recommendations from a user's historical behaviors by formulating the recommendation as a sequence generation task. TIGER [13] is the first generative retrieval framework. It first acquires hierarchical IDs of items by quantizing their semantic embeddings, and then trains a sequence-to-sequence model that predicts the semantic ID of the next item. ColaRec [19] and LETTER [18] study the problem of enhancing collaborative signals in quantization to integrate content knowledge and collaborative interaction. COBRA [22] addresses the information loss from the quantization by a coarse-to-fine generation mechanism, enabling more expressive generative modeling. Despite their advancements, the effectiveness and feasibility of generative ranking are still underexplored in large-scale scenarios in the real world. HSTU [23] is the first study to investigate the generative ranking task. It introduces an interleaved organization to predict the action by treating user actions as a new modality. In contrast, GenRank views items as positional indicators and reformulates recommendation as an action-oriented generation problem. Furthermore, we systematically analyze the efficacy drivers in generative recommendation, yielding crucial insights for both understanding generative ranking paradigms and informing future architectural designs.

5.2 Scaling Law in Recommendation System

Scaling laws, well-established in natural language processing and computer vision [9, 24], describe predictable relationships between model performance and factors such as model size, dataset size, and computational resources. In the domain of recommendation systems, similar scaling behaviors have been observed and validated across various stages of the pipeline, including retrieval [1, 5, 23] and ranking [23, 26]. Among recent advances, HSTU [23] emerges as a promising approach for generative recommendation. However, deploying such models in large-scale real-world scenarios necessitates careful consideration of efficiency. In this paper, we introduce an efficient generative architecture for ranking tasks while maintaining a comparable overhead to that of the current industrial recommender.

6 Online Experiments

To validate the effectiveness and feasibility of generative ranking in product scenarios, we conducted online experiments in Xiaohongshu's Explore Feed. All models traced back more than three months of data and were trained in an online manner. For the control group, we randomly selected 10% of users of Xiaohongshu and applied the production ranking model. For the treatment group, we applied GenRank to a randomly selected 10% of users. Each group contains tens of millions of users, and there is no overlap between groups.

From the offline metrics, the improvements in both AUC and GAUC [3, 29] for primary tasks exceed 0.0020, while the improvements for other tasks range between 0.0005 and 0.0015. From the online metrics, we select four metrics to measure the online performance: time spent, the number of reads, the number of engagements, and lifetime over 7 days (LT7). The online A/B test result averaged over a 15-day experimental period is shown in Table 2, where GenRank outperforms the production ranking in all metrics. Specifically, we observe that the improvement of GenRank on cold-start items is particularly significant. We believe this improvement stems from GenRank's enhanced ability to leverage the world knowledge from content embeddings.

In terms of overhead, GenRank and the production ranking model require a comparable amount of overall resources. Specifically, GenRank incurs higher training costs but lower inference and storage costs. Furthermore, GenRank demonstrates a significant improvement in P99 response time, outperforming the production ranking model by over 25%. This highlights the potential for further optimization in test-time scaling.

7 Conclusion

In this paper, we investigate the effectiveness and feasibility of generative ranking in large-scale industrial settings. Through both theoretical analysis and empirical results, we find that the generative architecture is the primary source of effectiveness in generative recommendation. We also introduce a novel generative architecture, named GenRank, which treats items as positional information and focuses on iteratively predicting user behaviors to address the inefficiencies present in existing approaches. Extensive large-scale offline and online experiments demonstrate the effectiveness and efficiency of our proposed solution.

References

[1] Hongru Cai, Yongqi Li, Ruifeng Yuan, Wenjie Wang, Zhen Zhang, Wenjie Li, and Tat-Seng Chua. 2025. Exploring Training and Inference Scaling Laws in Generative Retrieval. *arXiv preprint arXiv:2503.18941* (2025).

[2] 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.

[3] Jianxin Chang, Chenbin Zhang, Yiqun Hui, Dewei Leng, Yanan Niu, Yang Song, and Kun Gai. 2023. Pepnet: Parameter and embedding personalized network for infusing with personalized prior information. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3795–3804.

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[5] Wei Guo, Hao Wang, Luankang Zhang, Jin Yao Chin, Zhongzhou Liu, Kai Cheng, Qiushi Pan, Yi Quan Lee, Wanqi Xue, Tingjia Shen, et al. 2024. Scaling New Frontiers: Insights into Large Recommendation Models. *arXiv preprint arXiv:2412.00714* (2024).

[6] Yanhua Huang, Hangyu Wang, Yiyun Miao, Ruiwen Xu, Lei Zhang, and Weinan Zhang. 2022. Neural statistics for click-through rate prediction. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1849–1853.

[7] Yanhua Huang, Weikun Wang, Lei Zhang, and Ruiwen Xu. 2021. Sliding spectrum decomposition for diversified recommendation. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 3041–3049.

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

[9] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).

[10] 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.

[11] Ofir Press, Noah A Smith, and Mike Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409* (2021).

[12] 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.

[13] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. 2023. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems* 36 (2023), 10299–10315.

[14] Yuxin Ren, Qiya Yang, Yichun Wu, Wei Xu, Yalong Wang, and Zhiqiang Zhang. 2024. Non-autoregressive generative models for reranking recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5625–5634.

[15] Zihua Si, Lin Guan, ZhongXiang Sun, Xiaoxue Zang, Jing Lu, Yiqun Hui, Xingchao Cao, Zeyu Yang, Yichen Zheng, Dewei Leng, et al. 2024. Twin v2: Scaling ultra-long user behavior sequence modeling for enhanced ctr prediction at kuaishou. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 4890–4897.

[16] 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.

[17] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Proceedings of the 14th ACM conference on recommender systems*. 269–278.

[18] Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. 2024. Learnable item tokenization for generative recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 2400–2409.

[19] Yidan Wang, Zhaochun Ren, Weiwei Sun, Jiyuan Yang, Zhixiang Liang, Xin Chen, Ruobing Xie, Su Yan, Xu Zhang, Pengjie Ren, et al. 2024. Content-Based Collaborative Generation for Recommender Systems. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 2420–2430.

[20] Ye Wang, Jiahao Xun, Minjie Hong, Jieming Zhu, Tao Jin, Wang Lin, Haoyuan Li, Linjun Li, Yan Xia, Zhou Zhao, et al. 2024. EAGER: Two-Stream Generative Recommender with Behavior-Semantic Collaboration. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3245–3254.

[21] Rengan Xu, Junjie Yang, Yifan Xu, Hong Li, Xing Liu, Devashish Shankar, Haoci Zhang, Meng Liu, Boyang Li, Yuxi Hu, et al. 2024. Enhancing Performance and Scalability of Large-Scale Recommendation Systems with Jagged Flash Attention. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 778–780.

[22] Yuhao Yang, Zhi Ji, Zhaopeng Li, Yi Li, Zhonglin Mo, Yue Ding, Kai Chen, Zijian Zhang, Jie Li, Shuanglong Li, et al. 2025. Sparse Meets Dense: Unified Generative Recommendations with Cascaded Sparse-Dense Representations. *arXiv preprint arXiv:2503.02453* (2025).

[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] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. 2022. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12104–12113.

[25] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*. 11975–11986.

[26] Buyun Zhang, Liang Luo, Yuxin Chen, Jade Nie, Xi Liu, Daifeng Guo, Yanli Zhao, Shen Li, Yuchen Hao, Yantao Yao, et al. 2024. Wukong: Towards a scaling law for large-scale recommendation. *arXiv preprint arXiv:2403.02545* (2024).

[27] Chao Zhang, Haoxin Zhang, Shiwei Wu, Di Wu, Tong Xu, Xiangyu Zhao, Yan Gao, Yao Hu, and Enhong Chen. 2024. Notellm-2: Multimodal large representation models for recommendation. *arXiv preprint arXiv:2405.16789* (2024).

[28] Zhao-Yu Zhang, Xiang-Rong Sheng, Yujing Zhang, Biye Jiang, Shuguang Han, Hongbo Deng, and Bo Zheng. 2022. Towards understanding the overfitting phenomenon of deep click-through rate models. In *Proceedings of the 31st ACM international conference on information & knowledge management*. 2671–2680.

[29] 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.