

FuXi- β : Towards a Lightweight and Fast Large-Scale Generative Recommendation Model

Yufei Ye
aboluo2003@mail.ustc.edu.cn
University of Science and Technology
of China
Hefei, China

Wei Guo
guowei67@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Hao Wang^{*}
wanghao3@ustc.edu.cn
University of Science and Technology
of China
Hefei, China

Hong Zhu
zhuhong8@huawei.com
Consumer Business Group, Huawei
Shenzhen, China

Yuyang Ye
yeyuyang@mail.ustc.edu.cn
University of Science and Technology
of China
Hefei, China

Yong Liu^{*}
liu.yong6@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Huifeng Guo
huifeng.guo@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Ruiming Tang
tangruiming@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Defu Lian^{*}
Enhong Chen^{*}
liandefu@ustc.edu.cn
cheneh@ustc.edu.cn
University of Science and Technology
of China
Hefei, China

Abstract

Recent discoveries of scaling laws in autoregressive generative recommendation models present the possibility of developing larger and more versatile recommendation systems. However, larger systems also imply increased response latency and higher training costs. To accelerate training and inference, we investigated the recent generative recommendation models HSTU and *FuXi- α* , identifying two efficiency bottlenecks: the indexing operations in relative temporal attention bias and the computation of the query-key attention map. Additionally, we observed that relative attention bias in self-attention mechanisms can also serve as attention maps. Previous works like Synthesizer have shown that alternative forms of attention maps can achieve similar performance, naturally raising the question of whether some attention maps are redundant. Through empirical experiments, we discovered that using the query-key attention map might degrade the model's performance in recommendation tasks. To address these bottlenecks, we propose a new framework applicable to Transformer-like recommendation models. On one hand, we introduce Functional Relative Attention Bias, which avoids the time-consuming operations of the original relative attention bias, thereby accelerating the process. On the other hand, we remove the query-key attention map from the original self-attention layer and design a new Attention-Free Token Mixer module. Furthermore, by applying this framework to *FuXi- α* , we introduce a new model, *FuXi- β* . Experiments across multiple datasets demonstrate that *FuXi- β* outperforms previous state-of-the-art models and achieves significant acceleration compared to *FuXi- α* , while also adhering to the scaling law. Notably, *FuXi- β* shows an improvement of 27% to 47% in the NDCG@10 metric on large-scale industrial datasets compared to *FuXi- α* . Our code is available in a public repository: <https://github.com/USTC-StarTeam/FuXi-beta>

1 INTRODUCTION

Recommender systems play a crucial role in modern information society. An effective recommender system not only provides users with higher quality and more personalized content in a world overloaded with information, but also creates more commercial value for enterprises. In recommender systems, increasing the model size has always faced various challenges [3, 14, 15, 64, 69, 78]. However, the recently demonstrated scaling law in autoregressive generative recommendation models [48, 57, 73, 75] offers the possibility of creating large-scale recommender systems with better performance. Although larger models improve performance, they also introduce several cost and efficiency issues: (1) Training and serving a large-scale recommendation model requires substantial GPU computational resources. If the computing cost significantly exceeds the commercial value it can generate, then the model loses its value. (2) Most recommendation applications have stringent latency constraints (usually less than 100ms) [34, 54, 55, 59, 63, 73, 79], which poses significant challenges for the deployment of large-scale recommendation models.

Recent autoregressive generative recommendation models, such as HSTU [73] and *FuXi- α* [68], are based on the Transformer [56] architecture. To address the efficiency challenges posed by increasing model sizes, we revisited recent explorations aimed at optimizing the efficiency of Transformer-like structures. Notable strategies include low-precision quantization [9, 12, 38], distillation of smaller models using larger teacher models [23, 35], and pruning of model weights [16, 37, 50]. Additionally, there are numerous methods for directly designing more efficient models, such as simplifying the model [19], accelerating through sparse attention weights [4, 72], utilizing kernel methods for attention acceleration [6], employing

mixture of experts models [7], and leveraging RNN characteristics for acceleration [40, 51]. However, the training processes for pruning and distillation are relatively complex, as there are additional steps conducted after the base model training, and thus, they cannot directly expedite the training process. Moreover, quantized models necessitate hardware support, and deploying them on different GPUs may result in model failure. Some quantization methods [9, 12] still do not accelerate the training process. Although numerous efficient Transformer-like structures exist, they are tailored for NLP tasks and may not be suitable for recommendation tasks. Currently, the structural design for efficient generative recommendation models remains underdeveloped.

To design a more efficient structure for recommendation tasks, we analyzed the execution times of various operators during training *FuXi- α* [68] and HSTU [73]. Among the top three operators with the highest execution times, the one with the highest execution time is associated with computing the query-key attention map. The second and third highest execution times both pertain to the indexing operator in the Relative Attention Bias (RAB) module for timestamps. On the one hand, the index operators used in RAB require noncontiguous memory access, which are not hardware-friendly, and the number of these operations is related to the square of the sequence length. These factors result in the actual runtime of the RAB being slower than the matrix multiplication, even though the latter may have a higher computational complexity. On the other hand, calculating the query-key attention map aims to facilitate feature interaction among items, but this process incurs significant computational and memory costs. Previous research suggests that the attention map used in self-attention does not necessarily need to be generated in this manner. For example, using a fixed attention map [43] or a learnable constant attention map [52] can still achieve comparable results. In *FuXi- α* and HSTU, aside from the query-key attention map, the bias term produced by RAB can itself be regarded as an attention map. This naturally raises the question: Are some of these attention maps redundant? To answer this question, we conducted empirical experiments on multiple datasets and found that removing the temporal or spatial attention maps results in varying degrees of performance degradation. However, removing the query-key attention map, conversely, improved performance. This suggests that the query-key attention map might be redundant in recommendation tasks.

To address the aforementioned issues, we propose a new framework that can be utilized in various Transformer-like models. On the one hand, we introduced the Functional Relative Attention Bias (FRAB) module, which addresses the inefficiency of RAB by avoiding various hardware-unfriendly operations inherent in RAB. On the other hand, we removed the query-key attention map from the original self-attention layer and designed a new module called Attention-Free Token Mixer (AFTM). Subsequently, we apply this framework to the prior state-of-the-art generative recommendation model and introduce a new model, *FuXi- β* . And *FuXi- β* achieves performance comparable to the prior state-of-the-art models on public datasets while significantly reducing training time. On large-scale industrial datasets, it achieves both effective acceleration and notable performance improvements. Additionally, we have validated

on industrial datasets that our model exhibits the properties of scaling laws, demonstrating its potential for application in large-scale recommendation systems. Our contributions are as follows:

- We propose a new framework that can simplify Transformer-like generative recommendation models and introduce a new efficient and lightweight generative recommendation model, *FuXi- β* .
- We empirically find that the query-key attention map can have a negative impact on recommendation tasks. Additionally, we designed a new module called Attention-Free Token Mixer.
- We propose a novel and efficient method for modeling relative temporal information, termed Functional Relative Attention Bias.
- Experiments on four real-world datasets demonstrate that *FuXi- β* outperforms several state-of-the-art models in performance and efficiency. Furthermore, its scaling law properties suggest potential for large-scale recommendation systems.

2 RELATED WORK

2.1 Positional and Temporal Embedding Methods

Due to the inability of self-attention to capture the sequential order, it is necessary to incorporate positional encodings as an additional component. Positional encodings can be categorized into absolute and relative positional encodings. Absolute positional encodings involve directly adding a position embedding to each token's hidden representation. Common approaches include Sinusoidal positional encodings [56] and learnable positional encodings [13]. In contrast, relative positional encodings consider the relative distances between tokens when computing attention weights. In 2018, Shaw et al. [45] introduced the concept of relative positional encodings for the first time. Subsequent works [8, 20, 27, 77] have proposed various forms of relative position encoding by considering the interaction terms between positions and tokens, as well as between positions themselves. T5 [42] incorporates a learnable bias based solely on relative position information directly into the attention weights. Huang et al. [25] propose more sophisticated relative positional encodings, asserting that existing methods do not fully leverage the potential of relative positions.

In the domain of sequential recommendation, it is crucial to consider both positional and temporal information [17, 18, 32, 58, 73]. TiSASRec [32] and STAN [36] model time intervals by segmenting them into discrete bins, each associated with an embedding vector added to the key vector. MEANTIME [5] applies various functions to the relative time differences to obtain temporal embeddings, which are then incorporated into the key vector. HSTU [73] employs a logarithmic bucketing approach for time intervals, followed by a T5-style relative position encoding to integrate temporal information.

In our approach to modeling time intervals, we adopt the T5-style relative positional encoding but replace bucketing and learnable vector parameters with learnable functions, thereby enhancing computational efficiency.

2.2 Efficient Transformer-based Models

Transformer-based models have been widely applied across various domains [10, 26, 53, 56, 60, 65, 67, 71]. However, the quadratic time and space complexity of self-attention with respect to sequence

length imposes significant limitations. Consequently, numerous algorithmic strategies have been proposed to reduce these complexities. Techniques such as model pruning [16, 37, 50], which removes less important weights to reduce parameter count and improve computational efficiency, and model quantization [9, 12], which compresses models by converting high-precision floating-point parameters to lower-precision floats or integers, are commonly used. Knowledge distillation [23, 35, 70, 76] further accelerates models by training a smaller student model under the guidance of a larger teacher model, effectively transferring learned knowledge.

Additionally, designing more efficient model architectures is also an option to address these challenges [19, 24, 51, 61, 62]. ALBERT [30] reduces parameters through low-rank factorization of embedding layers and cross-layer parameter sharing. Approaches such as MQA [46] and GQA [2] decrease memory usage by reducing the number of query and value heads. MLA [33] builds on GQA by incorporating additional projections to enhance performance while further compressing the KV cache. However, these efforts can only accelerate the inference process. He et al. [19] simplified the Transformer architecture by removing components such as the value projection matrix and normalization layers. FLASH [24] achieves simplification by integrating Self-attention layers with Gated Linear Units (GLU), while reducing the complexity and memory overhead of self-attention through chunking and linear attention mechanisms. RWKV [40, 41] and RetNet [51] modified the attention mechanism to achieve the low-cost inference characteristic of RNNs, while simultaneously maintaining the capability for parallel training.

3 PROBLEM STATEMENT

In the paradigm of generative recommendation, the primary objective is to predict the next item a user is likely to interact with, based on their historical interaction sequence. Besides the item and position information, we need to pay additional attention to the timestamp of each interaction. Formally, consider a set of users $\mathcal{U} = \{u_1, u_2, \dots, u_{|\mathcal{U}|}\}$ and a set of items $\mathcal{I} = \{i_1, i_2, \dots, i_{|\mathcal{I}|}\}$. For each user $u \in \mathcal{U}$, we define an interaction sequence

$$S_u = \left[\left(i_1^{(u)}, t_1^{(u)}, 1 \right), \left(i_2^{(u)}, t_2^{(u)}, 2 \right), \dots, \left(i_{n_u}^{(u)}, t_{n_u}^{(u)}, n_u \right) \right] \quad (1)$$

which is a chronologically ordered list of items, their corresponding timestamps and positions.

The task is to predict the next item $i_{n_u+1}^{(u)}$ that user u will interact with, based on their interaction sequence S_u . This prediction can be formulated as estimating the probability distribution over the item set \mathcal{I} for the next interaction, conditioned on the historical interactions: $P(i_{n_u+1}^{(u)} = i \mid S_u)$ for all $i \in \mathcal{I}$. During training, our objective is to predict the subsequent item $i_{j+1}^{(u)}$ for every prefix j of the given sequence S_u . For an input sequence S_u , the desired output sequence is $[i_2^{(u)}, i_3^{(u)}, \dots, i_{n_u+1}^{(u)}]$ [26].

4 PRELIMINARY

4.1 Decoder-only Transformer

The standard Transformer [56] comprises an encoder and a decoder. The encoder employs a bidirectional attention mechanism, allowing each token to attend to all other tokens. In contrast, the decoder

uses a unidirectional attention mechanism, where each token can only attend to its preceding tokens. Consequently, decoders are often used independently for various autoregressive tasks [1, 21].

A Transformer Decoder is constructed by stacking multiple Decoder Blocks, each primarily comprising one multi-head attention layer and one feed-forward neural network. Let the input to a Decoder Block be $X \in \mathbb{R}^{n \times d}$, where n is the sequence length and d is the embedding dimension. The i -th row of X represents the latent vector of the i -th token. To ensure training stability, pre-norm is typically employed in the decoder [66]. Denote the multi-head attention as f_{MHSA} and the feedforward neural network as f_{FFN} . Thus, the final output of the decoder, Y , is given by:

$$M = X + f_{\text{MHSA}}(\text{norm}(X)) \quad (2)$$

$$Y = M + f_{\text{FFN}}(\text{norm}(M)) \quad (3)$$

4.1.1 Multi-Head Self-Attention The multi-head self-attention consists of h heads, each with a size of $d_h = d/h$. In each head, query, key, and value vectors are computed through projection. The attention score between any two tokens is calculated by the dot product of a token's query vector and another's key vector, followed by normalization and multiplication with the value vector to yield the output of this head, h_i :

$$h_i = \text{mask} \left(\varphi((XW_q^{(i)})(XW_k^{(i)})^T) \right) (XW_v^{(i)}) \quad (4)$$

where $W_q^{(i)}, W_k^{(i)}, W_v^{(i)} \in \mathbb{R}^{d \times d_h}$ are learnable parameters, and φ is the normalization function, typically softmax in NLP, while SiLU is found to be more effective in recommendation tasks [73]. The function $\text{mask}(X)$ applies a causal mask to the matrix X by setting its upper triangular part (excluding the diagonal) to zero. The final output of the multi-head self-attention is:

$$f_{\text{MHSA}}(X) = \text{concat}(h_1, \dots, h_h)W_o \quad (5)$$

where $W_o \in \mathbb{R}^{d \times d}$ is a learnable parameter.

4.1.2 Feed-Forward Neural Network In a feed-forward neural network, the input vector is first projected into a higher-dimensional space and then passed through a nonlinear activation function. This process can be represented mathematically as follows:

$$f_{\text{FFN}}(X) = \phi(XW_1)W_2 \quad (6)$$

where $W_1 \in \mathbb{R}^{d \times d_{\text{FFN}}}$, $W_2 \in \mathbb{R}^{d_{\text{FFN}} \times d}$ are learnable parameters, and ϕ denotes a non-linear activation function.

4.2 Relative Attention Bias

To address the inability to perceive the position of tokens when calculating attention weights, absolute or relative positional encodings are incorporated to add positional information. Relative Attention Bias (RAB), proposed in T5 [42], is a method of relative positional encoding that adds a trainable bias term directly to the attention map, denoted as $B = (b_{i,j})_{n \times n} \in \mathbb{R}^{n \times n}$:

$$h_i = \text{mask}(\varphi((XW_q^{(i)})(XW_k^{(i)})^T + B))(XW_v^{(i)}) \quad (7)$$

The bias term is calculated using $b_{i,j} = \beta_{f(i-j)}$, where $\beta \in \mathbb{R}^{d_{\text{rab}}}$ represents learnable parameters, and f is a bucketing function that

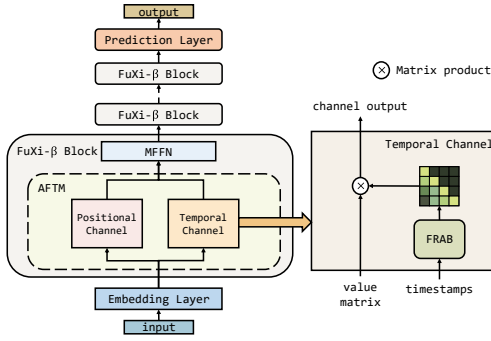


Figure 1: The overview of the proposed model FuXi- β .

maps the relative position to one value of β . Similarly, we can express relative temporal information as:

$$b_{i,j}^t = \beta_{f_t(t_i - t_j)}^t \quad (8)$$

where, t_i denotes the timestamp of the i -th item, and f_t represents the bucketing function for relative time. In HSTU [73], the attention map is computed as follows:

$$A = (XW_q)(XW_k)^T + B + B^t \quad (9)$$

In *FuXi- α* [68], the relative time and position bias terms, B and B^t , are directly applied to individual heads as the attention map. In subsequent sections, we will refer to this method as the "bucketed relative attention bias" for differentiation purposes.

5 METHODOLOGY

We introduce our newly proposed framework based on the current state-of-the-art generative recommendation model, *FuXi- α* , and we refer to the new model as *FuXi- β* . Although we present this framework based on *FuXi- α* , it can be applied to other Transformer-like generative recommendation models. *FuXi- β* is also a Transformer-like model, mainly consisting of L layers of stacked *FuXi- β* Blocks, as shown in Figure 1. In the following subsections, we will introduce *FuXi- β* in the following order: (1) an embedding layer that provides item embeddings and absolute positional information; (2) a new and fast module in the framework to model relative temporal information: the FRAB module; (3) a method within the framework to simplify the self-attention layer; (4) a discussion of other details within the *FuXi- β* Block; (5) prediction and training loss.

5.1 Embedding Layer

To address the variability in user interaction sequence lengths, we standardize these sequences into a fixed length n through truncation or padding with a special "padding item" prior to the embedding layer. In the embedding layer, each unique item $i \in \mathcal{I}$ is mapped to a d -dimensional vector using a learnable embedding matrix $E \in \mathbb{R}^{|\mathcal{I}| \times d}$. Additionally, we incorporate absolute positional information into each token in the embedding layer by using learnable positional encodings [13]. Let the positional embedding vector for the i -th position be p_i , and the embedding of the i -th item be e_i . For a user u with sequence $S_u = [i_1^{(u)}, \dots, i_{n_u}^{(u)}]$, the

output of the embedding layer is expressed as

$$S_u = [e_1^{(u)} + p_1, \dots, e_{n_u}^{(u)} + p_{n_u}, 0, \dots, 0] \quad (10)$$

where 0 denotes the positions that are padded.

5.2 Functional Relative Attention Bias

The bucketed relative attention bias used for modeling relative time was inefficient due to numerous time-consuming non-continuous indexing operations. To address this inefficiency, the Functional Relative Attention Bias is proposed, which only utilizes special function and arithmetic operations to model relative temporal information. Let the timestamp of the i -th item be denoted as t_i . The attention bias for modeling the relative temporal information are computed via the following equation:

$$b_{i,j}^t = f(t_j - t_i) \quad (11)$$

where f is a monotonically function of the following form:

$$f(x) = a(1+x)^{-b} \quad (12)$$

here, a and b are learnable parameters. Because the given sequence is arranged in a chronological order, we assume x is non-negative. To prevent numerical issues with identical timestamps, we add 1 to x before applying the power function.

5.3 Attention-Free Token Mixer

In the self-attention layer, beyond deriving the attention map from the query-key multiplication, the bias term from the relative attention bias module can also act as an attention map. Our ablation experiments, detailed in Section 6.4.3, showed that the query-key derived attention map is unnecessary for feature interaction among items. Consequently, we removed the query and key matrices in a Transformer-like architecture and utilized the attention map from the relative attention bias module to facilitate inter-item feature interaction. We refer to this new inter-item interaction layer as the Attention-free Token Mixer (AFTM). Let $X \in \mathbb{R}^{n \times d}$ denote the latent vector representation of the input items, and let $t \in \mathbb{R}^n$ represent the sequence of timestamps corresponding to these items. Initially, the matrices U and V are computed through projection as follows:

$$U = \phi(XW_u), V = \phi(XW_v) \quad (13)$$

where ϕ represents the SiLU activation function, and $W_u \in \mathbb{R}^{d \times 2d}$ and $W_v \in \mathbb{R}^{d \times 2d}$ are learnable parameters. Subsequently, the Relative Attention Bias (RAB) is used to transform positional information into an attention map B , and the Functional Relative Attention Bias (FRAB) is used to convert temporal information into an attention map B^t . The output of the AFTM is given by:

$$f_{\text{AFTM}}(X, t) = U \odot \text{concat}(BV, B^tV) \quad (14)$$

5.4 FuXi- β Block

Each *FuXi- β* Block consists of an Attention-Free Token Mixer (AFTM) and an Multistage FFN (MFFN) utilizing SwiGLU [47] as the activation function, and employs RMSNorm [74] as the normalization function. The MFFN is a variant of FFN introduced in *FuXi- α* [68]. The specific computational process is as follows:

$$M = f_{\text{AFTM}}(\text{RMSNorm}(X), t) \quad (15)$$

$$f_{\text{block}}(X, t) = f_{\text{MFFN}}(M, X) \quad (16)$$

5.5 Prediction Layer & Optimization objective

Following the passage through L layers of $FuXi-\beta$ blocks, each position has garnered ample information related to the items that previously interacted. We implement a multiplication with the transpose of the input embedding matrix, which is subsequently subjected to a softmax function, thereby generating a probability distribution across the predicted items. This transformation can be mathematically articulated as follows:

$$P\left(i_j^{(u)} = i \mid (i_1^{(u)}, t_1^{(u)}, 1), \dots, (i_{j-1}^{(u)}, t_{j-1}^{(u)}, j-1)\right) = \text{softmax}\left(\mathbf{x}^B \mathbf{E}^T\right)_i \quad (17)$$

To expedite the training process, we incorporate the sampled softmax loss utilizing N randomly selected negative samples [28].

5.6 Complexity Analysis

In this section, we theoretically analyze the complexity variations of several Transformer-like models, namely LLaMa [11], HSTU [73], and $FuXi-\alpha$ [68], before and after applying our framework. We assume that in the self-attention layer, the condition $d_h \times h = d$ holds, where d_h represents the projection dimension of each attention head, and h denotes the number of attention heads. Definitions of other variables can be found in the previous sections. Since the discontinuous index operations involved in the bucketed relative attention bias modules are significantly more time-consuming than arithmetic operations, like multiplication and addition, we list them separately in our analysis. The computed complexities for different models are presented in Table 1.

- LLaMa’s self-attention requires one computation of QKV projections and output projection, with a computational cost of $4nd^2$, and one computation of attention weights, followed by multiplication with the V matrix, which incurs an additional cost of $2n^2d$. HSTU incurs an extra computational cost of nd^2 for calculating the U matrix. In $FuXi-\alpha$, the dimension of the concatenated vector of all the channels is three times the embedding dimension. Therefore, this adds an extra $4nd^2$ to the complexity compared to HSTU, along with an additional $2n^2d$ due to the calculation of weights from two additional attention maps.
- In LLaMa- β and HSTU- β , there is no need to compute queries and keys, nor their dot products, thereby reducing the complexity by $2nd^2 + n^2d$ compared to their respective base models. In $FuXi-\beta$, the dimension of the concatenated vector is reduced, resulting in a computational reduction of $4nd^2 + 2n^2d$ compared to $FuXi-\alpha$.
- In both HSTU and $FuXi-\alpha$, the application of FRAB can reduce time consumption by avoiding discontinuous index operations. However, LLaMa uses RoPE [49] to utilize positional information, and replacing it with FRAB would increase computational complexity. Nonetheless, this substitution addresses the issue that RoPE encoding cannot simultaneously utilize both temporal and positional information.

In a word, our method can reduce approximately half of the computational load in the self-attention layer and replace the high time-consuming indexing operations in the bucketed relative attention bias with more efficient calculations.

6 EXPERIMENTS

6.1 Experiment Setup

6.1.1 Datasets We conducted experiments on two public datasets (MovieLens-1M and MovieLens-20M) and two private large-scale industrial datasets (Daily Recommendations and All Scenarios). The details of these datasets are outlined below:

- **MovieLens**¹. MovieLens is a popular benchmark dataset for movie recommendation. The dataset comprises various subsets of different sizes. In our work, we use the MovieLens-1M and MovieLens-20M subsets for our experiment.
- **Daily Recommendations**. In the daily recommendation scenario, 30 exclusive songs are selected for each user every day. The displayed list is updated in a daily manner. We sampled a portion of the data to form an offline dataset, which includes tens of millions of users and billions of interactions.
- **All Scenarios**. The all scenarios dataset includes user behaviors from all music recommendation scenarios, with a more complex and challenging data distribution, which can better reflect the effectiveness and robustness of the model. We also sampled tens of millions of users and billions of interactions to build an offline dataset.

The statistical information about the datasets is presented in Table 2. For the public datasets, we employed the same preprocessing techniques and the train/validation/test set partitioning method as used in HSTU [73] and $FuXi-\alpha$ [68]. For the industrial datasets, we applied similar preprocessing methods.

6.1.2 Baselines We conducted a validation of $FuXi-\beta$ in two aspects: performance and efficiency. For performance, we employed traditional models (BPRMF [44], GRU4Rec [22], and NARM [31]) as well as autoregressive generative models (SASRec [26], LLaMa [11], HSTU [73], and $FuXi-\alpha$ [68]) as baselines. Regarding efficiency, we also introduced some work on efficient transformers in NLP, including FLASH [24], and RetNet [51].

6.1.3 Evaluation Metrics We employ three widely used metrics (NDCG@K, HR@K, and MRR) to evaluate the recall quality of the model. Higher values of these metrics indicate better performance. We rank all items based on the model’s results and report the outcomes for $K = 10$ and $K = 50$ by default. Additionally, we measure the efficiency of the model using the total training time; a shorter training time implies a faster model.

6.1.4 Parameter Settings We implemented our model, $FuXi-\beta$, using PyTorch [39], and employed the Accelerate library [29] to achieve multi-NPU and multi-machine parallel training. In public datasets, we adopted the same parameters as HSTU [73], except for the number of layers. For models with a Feedforward Neural Network (FFN), the hidden layer width is the same as the embedding dimension in MovieLens-1M, while it is four times the embedding dimension in MovieLens-20M. In experiments on public datasets, we set the number of layers for the base model to 2, whereas the Large model refers to the corresponding model with 8 layers. For industrial datasets, we set the embedding size to 256 and the number of layers to 4 for all models to maintain consistency with the online version.

¹<https://grouplens.org/datasets/movielens/>

Table 1: Time complexity of different models

Model	Computational Complexity	Position and Time Embedding Complexity
LLaMa	$O(4nd^2 + 2n^2d + 3nd_{FFN}d)$	$O(n)$ arithmetic operations
LLaMa-β	$O(2nd^2 + n^2d + 3nd_{FFN}d)$	$O(n^2)$ special functions and arithmetic operations
HSTU	$O(5nd^2 + 2n^2d)$	$O(n^2)$ special function computation, arithmetic, and discontinuous indexing operations.
HSTU-β	$O(3nd^2 + n^2d)$	$O(n^2)$ special function computation and arithmetic operations
FuXi-α	$O(9nd^2 + 4n^2d + 3nd_{FFN}d)$	$O(n^2)$ special function computation, arithmetic, and discontinuous indexing operations.
FuXi-β	$O(5nd^2 + 2n^2d + 3nd_{FFN}d)$	$O(n^2)$ special function computation and arithmetic operations

Table 2: Dataset statistics.

Dataset	User	Item	Interactions	Avg. Len.
MovieLens-1M	6,041	3,706	1,000,209	165.60
MovieLens-20M	138,493	26,744	20,000,263	144.41
Daily Recomm.	19,252,028	234,488	1,023,711,774	53.17
All scenarios	28,927,689	476,544	1,313,729,225	40.89

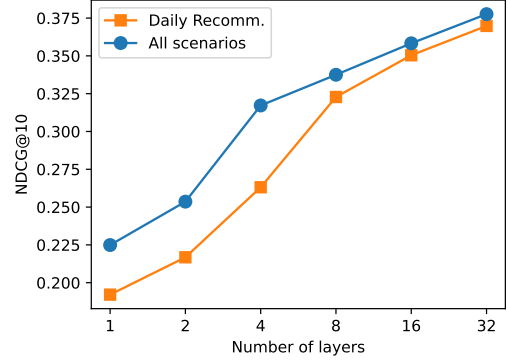
6.2 Performance Comparison

6.2.1 Public Dataset Performance The results of *FuXi- β* and the baseline models on the public datasets are presented in Table 2. We observe the following:

- Generative recommendation models generally outperform traditional models, even when the model has only 2 layers.
- As an early model, SASRec’s structural design may lead to training difficulties. Under our training parameters, increasing the layers to 8 actually resulted in a decline in performance. However, other generative models have successfully increased their layers to 8.
- Although FLASH outperforms the Transformer in NLP, it performs poorly in recommendation tasks. RetNet, despite having equally performance compared to models like LLaMa and HSTU, still lags behind the strongest baseline, *FuXi- α* , due to its lack of adaptation to recommendation tasks.
- FuXi- β* generally exhibits performance similar to *FuXi- α* . When using a 2-layer model, its performance on MovieLens-20M slightly decreases, while in other cases, it shows a modest improvement.

6.2.2 Industrial Dataset Performance Table 4 shows the performance comparison of our proposed *FuXi- β* against several baseline models using two large-scale private industrial datasets. Our *FuXi- β* model outperforms both *FuXi- α* and HSTU across two datasets. In the first dataset, the NDCG@10 metric has improved by 47.4% and 73.5% respectively, compared to *FuXi- α* and HSTU. In the second dataset, the improvements are 27.2% and 38.2% respectively. This may be attributed to the fact that industrial datasets are more complex than public datasets. The incorporation of queries and keys in the self-attention layer, along with bucketed relative attention bias, increases the model’s complexity, making it prone to overfitting. However, Our framework mitigates the issue of overfitting, resulting in an exceptional performance on the industrial datasets.

6.2.3 Scaling on Industrial Dataset We investigated the performance changes of *FuXi- β* as the model size increases on both industrial datasets, with the results shown in Figure 2. Due to memory constraints, we only increase the number of layers up to 32 layers. Additionally, the performance of *FuXi- β* continuously improves as the number of layers increases. This illustrates that *FuXi- β* exhibits

**Figure 2: Scaling of FuXi- β on Industrial Dataset.**

excellent performance on large-scale datasets while still adhering to the scaling law, which states that the model’s performance continues to improve with the increase in data, model size, and computational resource.

6.3 Efficiency Comparison

6.3.1 Public Datasets. Table 3 presents the total training time for different models over the same number of epochs. Due to the significant structural and performance differences between traditional models and generative recommendation models, we do not consider the efficiency of these traditional models. Although FLASH have a lower complexity [24], it do not exhibit a noticeable speed advantage when the sequence length is short. RetNet exhibits relatively fast training speeds. However, due to its lack of adaptation for recommendation problems, its performance still falls short compared to some recommendation models. The training times for LLaMa, HSTU, and *FuXi- α* are gradually increasing as the models become more complex. *FuXi- β* , when using 2 layers, reduces the training time by 5.51% and 10.6% on two datasets, respectively, compared to *FuXi- α* . When using 8 layers, the reductions in training time are 13.1% and 27.0%, respectively.

6.3.2 Industrial Datasets. Table 4 presents the training times of different models on industrial datasets. For LLaMa, we employed the same relative attention bias as HSTU instead of RoPE [49], which results in a longer training time on industrial datasets compared to HSTU. *FuXi- α* , due to its more complex architecture, lagged behind LLaMa in terms of speed. In contrast, *FuXi- β* , by applying our new framework to simplify its structure, reduced training time by 15.8% and 20.8% on the two datasets compared to *FuXi- α* , respectively.

6.4 Ablation Study of FuXi-β

6.4.1 Component Ablation Study We first investigated the impact of different components on the model. We denote *FuXi- β* -FRAB as

Table 3: The overall performance comparison on public datasets. We use bold to indicate the best result and underline to indicate the second best result. We set the training time of SASRec as 1.0. The training times of other models are expressed as multiples of SASRec’s training time.

Dataset Model	MovieLens-1M						MovieLens-20M					
	NDCG@10	NDCG@50	HR@10	HR@50	MRR	Time	NDCG@10	NDCG@50	HR@10	HR@50	MRR	Time
BPRMF	0.0607	0.1027	0.1185	0.3127	0.0556	-	0.0629	0.1074	0.1241	0.3300	0.0572	-
GRU4Rec	0.1015	0.1460	0.1816	0.3864	0.0895	-	0.0768	0.1155	0.1394	0.3177	0.0689	-
NARM	0.1350	0.1894	0.2445	0.4915	0.1165	-	0.1037	0.1552	0.1926	0.4281	0.0910	-
FLASH	0.1573	0.2144	0.2830	0.5426	0.1341	1.022	0.1496	0.2057	0.2682	0.5226	0.1285	1.210
RetNet	0.1601	0.2184	0.2925	0.5557	0.1352	0.975	0.1687	0.2249	0.2979	0.5526	0.1442	1.034
SASRec	0.1594	0.2187	0.2824	0.5500	0.1375	1.000	0.1553	0.2119	0.2781	0.5353	0.1330	1.000
LLaMa	0.1620	0.2207	0.2926	0.5591	0.1373	0.978	0.1640	0.2206	0.2915	0.5476	0.1402	1.006
HSTU	0.1639	0.2238	0.2969	0.5672	0.1390	1.019	0.1642	0.2225	0.2909	0.5553	0.1410	1.072
FuXi- α	0.1835	0.2429	0.3254	0.5941	0.1557	1.078	0.1954	0.2533	0.3353	0.5969	0.1677	1.126
FuXi- β	0.1848	0.2432	0.3231	0.5866	0.1578	1.019	0.1944	0.2513	0.3325	0.5899	0.1671	1.006
SASRec-Large	0.1186	0.1733	0.2183	0.4671	0.0186	1.000	0.0206	0.0379	0.0412	0.1209	0.0207	1.000
LLaMa-Large	0.1659	0.2257	0.2990	0.5692	0.1408	0.989	0.1842	0.2412	0.3202	0.5776	0.1576	1.022
HSTU-Large	0.1844	0.2437	0.3255	0.5929	0.1568	1.134	0.1995	0.2572	0.3407	0.6012	0.1714	1.187
FuXi- α -Large	<u>0.1934</u>	<u>0.2518</u>	<u>0.3359</u>	<u>0.5983</u>	0.1651	1.230	<u>0.2086</u>	<u>0.2658</u>	<u>0.3530</u>	0.6113	<u>0.1792</u>	1.371
FuXi- β -Large	0.1947	0.2523	0.3428	0.6022	<u>0.1645</u>	1.068	0.2117	0.2677	0.3566	<u>0.6095</u>	0.1818	1.000

Table 4: Performance comparison on Industrial dataset.

Dataset Model	Daily Recommendations						All scenarios					
	NDCG@10	NDCG@50	HR@10	HR@50	MRR	Time	NDCG@10	NDCG@50	HR@10	HR@50	MRR	Time
SASRec	0.0795	0.1488	0.1696	0.4887	0.0707	1.000	0.1638	0.2263	0.2971	0.5767	0.1399	1.000
LLaMa	0.1486	0.2205	0.2881	0.6139	0.1248	1.208	0.2229	0.2795	0.3732	0.6259	0.1919	1.212
HSTU	0.1516	0.2235	0.2921	0.6183	0.1274	1.153	0.2295	0.2859	0.3838	0.6355	0.1972	1.144
FuXi- α	<u>0.1785</u>	<u>0.2488</u>	<u>0.3271</u>	<u>0.6449</u>	<u>0.1513</u>	1.356	<u>0.2493</u>	<u>0.3039</u>	<u>0.4080</u>	<u>0.6512</u>	<u>0.2150</u>	1.323
FuXi- β	0.2631	0.3134	0.4228	0.6497	0.2269	1.142	0.3172	0.3631	0.4740	0.6781	0.2806	1.050

Table 5: Ablation study results on the public datasets

Dataset Model	MovieLens-1M						MovieLens-20M					
	NDCG@10	NDCG@50	HR@10	HR@50	MRR	Time	NDCG@10	NDCG@50	HR@10	HR@50	MRR	Time
FuXi- α	0.1835	0.2429	0.3254	0.5941	0.1557	1.000	0.1954	0.2533	0.3353	0.5969	0.1677	1.000
FuXi- β -FRAB	0.1823	0.2407	0.3244	0.5876	0.1541	0.997	0.1963	0.2530	0.3355	0.5923	0.1685	0.962
FuXi- β -AFTM	0.1797	0.2382	0.3188	0.5819	0.1526	0.994	0.1956	0.2534	0.3352	0.5964	0.1679	0.959
FuXi- β	0.1848	0.2432	0.3231	0.5866	0.1578	0.945	0.1944	0.2513	0.3325	0.5899	0.1671	0.894
FuXi- α -Large	0.1934	0.2518	0.3359	0.5983	0.1651	1.000	0.2086	0.2658	0.3530	0.6113	0.1792	1.000
FuXi- β -FRAB-Large	0.1871	0.2454	0.3327	0.5944	0.1578	0.913	0.2097	0.2658	0.3532	0.6062	0.1803	0.867
FuXi- β -AFTM-Large	0.1923	0.2507	0.3407	0.6030	0.1624	0.982	0.2107	0.2672	0.3550	0.6098	0.1811	0.858
FuXi- β -Large	0.1947	0.2523	0.3428	0.6022	0.1645	0.869	0.2117	0.2677	0.3566	0.6095	0.1818	0.730

Table 6: Comparison of the performance of FRAB using different functions.

Dataset Function	MovieLens-1M			MovieLens-20M		
	NG@50	HR@50	MRR	NG@50	HR@50	MRR
linear	0.2120	0.5345	0.1330	0.2488	0.5869	0.1646
log	0.2350	0.5817	0.1487	0.2617	0.6032	0.1761
exp	0.2500	0.5959	0.1635	0.2658	0.6057	0.1804
sin	0.2367	0.5771	0.1521	0.2526	0.5870	0.1690
mixed	0.2136	0.5378	0.1336	0.2498	0.5844	0.1665
NN	0.2411	0.5806	0.1565	0.2514	0.5819	0.1690
zero	0.2345	0.5776	0.1490	0.2434	0.5765	0.1605
bucket	0.2507	0.6030	0.1624	0.2672	0.6098	0.1811
pow	0.2523	0.6022	0.1645	0.2677	0.6095	0.1818

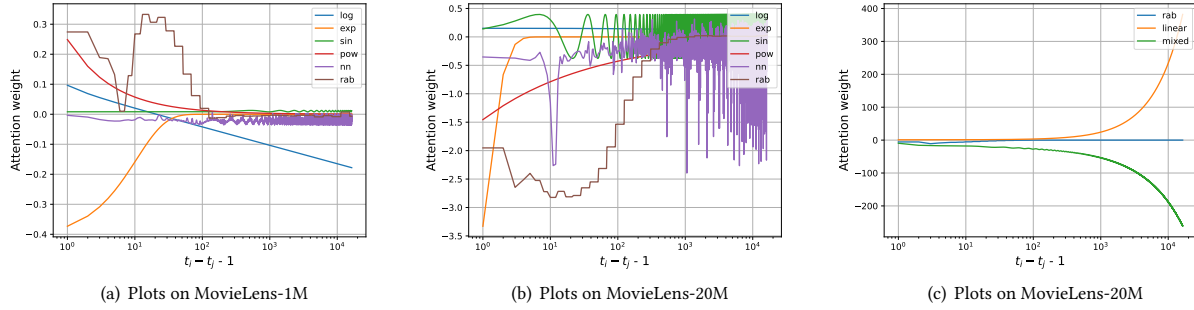
the model that replaces the RAB module in *FuXi- α* with the FRAB module to utilize relative temporal information, and the model that

removes all query and key matrices from *FuXi- α* as *FuXi- β -AFTM*. The experimental results on public datasets are shown in Table 5. Although using FRAB alone and removing queries and keys led to a certain degree of performance degradation on MovieLens-1M, the combination of these two modules still achieved performance close to that of *FuXi- α* . On MovieLens-20M, these modifications resulted in a slight performance enhancement. On MovieLens-1M, they reduced the training time for the Large model by 8.68% and 1.78%, respectively, while on MovieLens-20M, they reduced the training time by 13.3% and 14.2%, respectively. This indicates that in smaller models, the acceleration is mainly due to FRAB, whereas in larger models, both contribute significantly to the acceleration.

6.4.2 Function Variations on Functional Relative Attention Bias To validate the effectiveness of the function employed in our FRAB module, we conducted experiments by replacing it with various

Table 7: Results of the ablation study on the self-attention layer

Dataset	MovieLens-1M						MovieLens-20M					
Model	NDCG@10	NDCG@50	HR@10	HR@50	MRR	Time	NDCG@10	NDCG@50	HR@10	HR@50	MRR	Time
FuXi- β -FRAB-Large	0.1871	0.2454	0.3327	0.5944	0.1578	1.000	0.2097	0.2658	0.3532	0.6062	0.1803	1.000
- query-key attention map	0.1947	0.2523	0.3428	0.6022	0.1645	0.951	0.2117	0.2677	0.3566	0.6095	0.1818	0.842
- positional attention map	0.1800	0.2393	0.3164	0.5833	0.1539	0.951	0.2093	0.2659	0.3527	0.6078	0.1801	0.946
- temporal attention map	0.1741	0.2329	0.3096	0.5750	0.1480	0.990	0.1863	0.2418	0.3218	0.5730	0.1594	0.914

**Figure 3: Variation of attention weights produced by the FRAB module with respect to relative time using different functions.****Table 8: Compatibility analysis on the public datasets. We express the training time as a multiple of the training time for SASRec with the same parameter settings.**

Dataset	MovieLens-1M			MovieLens-20M		
Model	NG@50	HR@50	Time	NG@50	HR@50	Time
LLaMa-Large	0.2257	0.5692	0.989	0.2412	0.5776	1.022
LLaMa- β -Large	0.2354	0.5785	1.016	0.2560	0.6006	0.902
HSTU-Large	0.2437	0.5929	1.134	0.2572	0.6012	1.187
HSTU- β -Large	0.2425	0.5828	0.959	0.2520	0.5940	0.851
FuXi- α -Large	0.2518	0.5983	1.230	0.2658	0.6113	1.371
FuXi- β -Large	0.2523	0.6022	1.068	0.2677	0.6095	1.000

alternative functions. The functions considered for this purpose are as follows:

- (1) Linear function: $f_{lin}(x) = a \cdot x + b$
- (2) Logarithmic function: $f_{log}(x) = a \cdot \log(1 + \exp(b) \cdot x) + c$
- (3) Exponential function: $f_{exp}(x) = a \cdot \exp(-\exp(b) \cdot x)$
- (4) Sine function: $f_{sin}(x) = c \cdot \sin(a \cdot x + b) + d$
- (5) Power function: $f_{pow}(x) = a \cdot (1 + x)^{-b}$, which is used in our module
- (6) Mixed function: $f_{mix}(x) = \frac{f_{lin}(x) + f_{log}(x) + f_{exp}(x) + f_{sin}(x) + f_{pow}(x)}{5}$
- (7) A small 3-layer Multi-Layer Perceptron (MLP) f_{nn} , utilizing sinusoidal functions and SiLU as activation functions.
- (8) Zero function: $f_{zero}(x) = 0$
- (9) Bucket function: f_{bucket} represents the method used in relative attention bias [73].

In the above functions, a , b , c , and d represent learnable parameters unique to each function. Our experimental results on MovieLens-1M are presented in Table 6. From the results, replacing the power function we used with other functions results in varying degrees of performance degradation.

To further investigate the effects of different functions, we visualize the attention weights in the temporal channel with the time difference as the horizontal axis and the attention weight as the vertical axis. The results of the plot are shown in Figure 3. We have the following observations:

- The function f_{sin} varies periodically at a fixed frequency, but it is not effective in distinguishing between items that are far apart and those that are close together.
- The functions f_{mixed} and f_{nn} are difficult to train due to their complex models, resulting in suboptimal performance.
- The functions f_{lin} and f_{log} fail to adequately control the numerical range, leading to unreasonable weights assigned to items that are far apart, thus resulting in poor performance.
- The functions f_{pow} and f_{exp} have absolute values that monotonically decrease, which assigns higher attention weights to recent items. Additionally, the rate of decrease slows down, making it easier to distinguish recent features, leading to good performance. Since f_{pow} can be expressed as $ae^{-b \ln(1+x)}$, it has a slower decay rate compared to f_{exp} . Therefore, f_{pow} is more effective at assigning weights to items over longer distances, which is why it performs slightly better than f_{exp} .

6.4.3 Ablation Study of Simplifying the Self-Attention Layer To investigate whether there is redundancy among the three types of attention maps (query-key, temporal, and positional) in the self-attention layer for recommendation tasks, we empirically conducted ablation experiments based on *FuXi- β -FRAB-Large*. The results are shown in Table 7. Removing the temporal attention map significantly degrades the model's performance. Removing the positional attention map leads to a performance drop on MovieLens-1M. In contrast, removing the query-key attention map not only results in the fastest execution time but also achieves the best performance on both datasets.

6.4.4 Compatibility Analysis We validated our proposed framework across multiple models. Since RoPE used by LLaMa requires multiplying the query and key vectors to encode relative positional information, which is incompatible with the AFTM module, we replaced it with relative attention bias in LLaMa- β . In addition, we applied the FRAB module to process timestamps and removed the query and key matrices. In HSTU- β , we made modifications in a manner similar to FuXi- β . The results are shown in Table 8. For LLaMa, due to the application of fusion operators for acceleration and the low time complexity of RoPE, the training time for LLaMa- β on the MovieLens-1M dataset increased. However, on the MovieLens-20M dataset, the size of the query and key matrices increased so their removal can significantly reduce the training time, resulting in an 11.8% decrease for LLaMa- β . The performance of LLaMa- β was significantly improved due to the addition of temporal information. For HSTU- β , the training time was reduced by 15.5% and 28.3% on the two datasets, respectively, but its performance significantly declined. This decline may be attributed to HSTU's weakened implicit feature interaction, which hinders the effective utilization of information obtained from explicit feature interactions. For FuXi- α , the training time was reduced by 13.1% and 27.0% on the two datasets, respectively, with a slight improvement in accuracy.

7 CONCLUSION

In our work, we focus on simplifying generative recommendation models, and we propose a new framework. On the one hand, we introduce the Functional Relative Attention Bias module, which employs special functions and arithmetic operations to avoid the time-consuming indexing operations in the bucketed relative attention bias, thereby accelerating the process. On the other hand, we found that the query-key attention map in self-attention might yield negative effects in recommendation tasks. Therefore, we removed the query-key attention map from the original self-attention layer and designed a new module called the Attention-free Token Mixer. Based on this framework, we propose a new model, FuXi- β . Comprehensive experiments demonstrate the remarkable efficiency of FuXi- β and its outstanding performance. In the future, we plan to further reduce the complexity of the model and aim to address more complex recommendation tasks.

References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altmenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
- [2] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Singh. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245* (2023).
- [3] Newsha Ardalani, Carole-Jean Wu, Zeliang Chen, Bhargav Bhushanam, and Adnan Aziz. 2022. Understanding scaling laws for recommendation models. *arXiv preprint arXiv:2208.08489* (2022).
- [4] Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* (2020).
- [5] Sung Min Cho, Eunhyeok Park, and Sungjoo Yoo. 2020. MEANTIME: Mixture of attention mechanisms with multi-temporal embeddings for sequential recommendation. In *Proceedings of the 14th ACM Conference on recommender systems*. 515–520.
- [6] Krzysztof Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794* (2020).
- [7] Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066* (2024).
- [8] Zihang Dai. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860* (2019).
- [9] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3.int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems* 35 (2022), 30318–30332.
- [10] Alexey Dosovitskiy. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).
- [11] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).
- [12] Elias Frantar, Saleh Ashkboos, Torsten Hoefer, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323* (2022).
- [13] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International conference on machine learning*. PMLR, 1243–1252.
- [14] 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).
- [15] Xingzhuo Guo, Junwei Pan, Ximei Wang, Baixu Chen, Jie Jiang, and Mingsheng Long. 2023. On the Embedding Collapse when Scaling up Recommendation Models. *arXiv preprint arXiv:2310.04400* (2023).
- [16] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems* 28 (2015).
- [17] Yongqiang Han, Hao Wang, Kefan Wang, Likang Wu, Zhi Li, Wei Guo, Yong Liu, Defu Lian, and Enhong Chen. 2024. Efficient noise-decoupling for multi-behavior sequential recommendation. In *Proceedings of the ACM Web Conference 2024*. 3297–3306.
- [18] Yongqiang Han, Likang Wu, Hao Wang, Guifeng Wang, Mengdi Zhang, Zhi Li, Defu Lian, and Enhong Chen. 2023. Guesr: A global unsupervised data-enhancement with bucket-cluster sampling for sequential recommendation. In *International conference on database systems for advanced applications*. Springer, 286–296.
- [19] Bobby He and Thomas Hofmann. 2023. Simplifying transformer blocks. *arXiv preprint arXiv:2311.01906* (2023).
- [20] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. DeBERTa: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654* (2020).
- [21] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. 2020. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701* (2020).
- [22] B Hidasi. 2015. Session-based Recommendations with Recurrent Neural Networks. *arXiv preprint arXiv:1511.06939* (2015).
- [23] Geoffrey Hinton. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531* (2015).
- [24] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. 2022. Transformer quality in linear time. In *International conference on machine learning*. PMLR, 9099–9117.
- [25] Zhiheng Huang, Davis Liang, Peng Xu, and Bing Xiang. 2020. Improve transformer models with better relative position embeddings. *arXiv preprint arXiv:2009.13658* (2020).
- [26] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.
- [27] Guolin Ke, Di He, and Tie-Yan Liu. 2020. Rethinking positional encoding in language pre-training. *arXiv preprint arXiv:2006.15595* (2020).
- [28] Anton Klenitskiy and Alexey Vasilev. 2023. Turning Dross Into Gold Loss: is BERT4Rec really better than SASRec? In *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys '23)*. ACM, 1120–1125. doi:10.1145/3604915.3610644
- [29] John P Kotter. 2012. Accelerate. *Harvard business review* 90, 11 (2012), 45–58.
- [30] Zhenzhong Lan. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* (2019).
- [31] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM Conference on Information and Knowledge Management*. 1419–1428.
- [32] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th international conference on web search and data mining*. 322–330.

- [33] Aixian Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, et al. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434* (2024).
- [34] Weiwen Liu, Wei Guo, Yong Liu, Ruiming Tang, and Hao Wang. 2023. User Behavior Modeling with Deep Learning for Recommendation: Recent Advances. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1286–1287.
- [35] Chengqiang Lu, Jianwei Zhang, Yunfei Chu, Zhengyu Chen, Jingren Zhou, Fei Wu, Haiping Chen, and Hongxia Yang. 2022. Knowledge distillation of transformer-based language models revisited. *arXiv preprint arXiv:2206.14366* (2022).
- [36] Yingtao Luo, Qiang Liu, and Zhaocheng Liu. 2021. Stan: Spatio-temporal attention network for next location recommendation. In *Proceedings of the web conference 2021*. 2177–2185.
- [37] Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems* 36 (2023), 21702–21720.
- [38] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2017. Mixed precision training. *arXiv preprint arXiv:1710.03740* (2017).
- [39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).
- [40] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. 2023. Rkv: Reinventing rns for the transformer era. *arXiv preprint arXiv:2305.13048* (2023).
- [41] Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Xingjian Du, Teddy Ferdinand, Haowen Hou, et al. 2024. Eagle and finch: Rkv with matrix-valued states and dynamic recurrence. *arXiv preprint arXiv:2404.05892* (2024).
- [42] 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.
- [43] Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. 2020. Fixed encoder self-attention patterns in transformer-based machine translation. *arXiv preprint arXiv:2002.10260* (2020).
- [44] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).
- [45] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155* (2018).
- [46] Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150* (2019).
- [47] Noam Shazeer. 2020. Glue variants improve transformer. *arXiv preprint arXiv:2002.05202* (2020).
- [48] Tingjia Shen, Hao Wang, Chuhan Wu, Jin Yao Chin, Wei Guo, Yong Liu, Huifeng Guo, Defu Lian, Ruiming Tang, and Enhong Chen. 2025. Optimizing Sequential Recommendation Models with Scaling Laws and Approximate Entropy. *arXiv:2412.00430 [cs.AI]* <https://arxiv.org/abs/2412.00430>
- [49] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yufeng Liu. 2024. Reformer: Rethinking self-attention for transformer models. In *International conference on machine learning*. PMLR, 10183–10192.
- [50] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. 2024. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *arXiv preprint arXiv:2404.02905* (2024).
- [51] Zhen Tian, Ting Bai, Zibin Zhang, Zhiyuan Xu, Kangyi Lin, Ji-Rong Wen, and Wayne Xin Zhao. 2023. Directed acyclic graph factorization machines for CTR prediction via knowledge distillation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 715–723.
- [52] Junxiong Tong, Mingjia Yin, Hao Wang, Qiushi Pan, Defu Lian, and Enhong Chen. 2024. MDAP: A Multi-view Disentangled and Adaptive Preference Learning Framework for Cross-Domain Recommendation. In *International Conference on Web Information Systems Engineering*. Springer, 164–178.
- [53] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).
- [54] Hao Wang, Wei Guo, Luankang Zhang, Jin Yao Chin, Yufei Ye, Huifeng Guo, Yong Liu, Defu Lian, Ruiming Tang, and Enhong Chen. 2025. Generative Large Recommendation Models: Emerging Trends in LLMs for Recommendation. *arXiv preprint arXiv:2502.13783* (2025).
- [55] Hao Wang, Yongqiang Han, Kefan Wang, Kai Cheng, Zhen Wang, Wei Guo, Yong Liu, Defu Lian, and Enhong Chen. 2024. Denoising Pre-Training and Customized Prompt Learning for Efficient Multi-Behavior Sequential Recommendation. *arXiv preprint arXiv:2408.11372* (2024).
- [56] Hao Wang, Defu Lian, Hanghang Tong, Qi Liu, Zhenya Huang, and Enhong Chen. 2021. Decoupled representation learning for attributed networks. *IEEE Transactions on Knowledge and Data Engineering* 35, 3 (2021), 2430–2444.
- [57] Hao Wang, Tong Xu, Qi Liu, Defu Lian, Enhong Chen, Dongfang Du, Han Wu, and Wen Su. 2019. MCNE: An end-to-end framework for learning multiple conditional network representations of social network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1064–1072.
- [58] Hao Wang, Mingjia Yin, Luankang Zhang, Sirui Zhao, and Enhong Chen. 2025. MF-GSLAE: A Multi-Factor User Representation Pre-training Framework for Dual-Target Cross-Domain Recommendation. *ACM Transactions on Information Systems* 43, 2 (2025), 1–28.
- [59] Kefan Wang, Hao Wang, Kenan Song, Wei Guo, Kai Cheng, Zhi Li, Yong Liu, Defu Lian, and Enhong Chen. 2025. A Universal Framework for Compressing Embeddings in CTR Prediction. *arXiv preprint arXiv:2502.15355* (2025).
- [60] 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.
- [61] Wenjia Xie, Hao Wang, Luankang Zhang, Rui Zhou, Defu Lian, and Enhong Chen. 2024. Breaking determinism: Fuzzy modeling of sequential recommendation using discrete state space diffusion model. *Advances in Neural Information Processing Systems* 37 (2024), 22720–22744.
- [62] Wenjia Xie, Rui Zhou, Hao Wang, Tingjia Shen, and Enhong Chen. 2024. Bridging User Dynamics: Transforming Sequential Recommendations with Schrödinger Bridge and Diffusion Models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 2618–2628.
- [63] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tiejian Liu. 2020. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*. PMLR, 10524–10533.
- [64] Xiang Xu, Hao Wang, Wei Guo, Luankang Zhang, Wanshan Yang, Runlong Yu, Yong Liu, Defu Lian, and Enhong Chen. 2024. Multi-granularity Interest Retrieval and Refinement Network for Long-Term User Behavior Modeling in CTR Prediction. *arXiv preprint arXiv:2411.15005* (2024).
- [65] Yufei Ye, Wei Guo, Jin Yao Chin, Hao Wang, Hong Zhu, Xi Lin, Yuyang Ye, Yong Liu, Ruiming Tang, Defu Lian, and Enhong Chen. 2025. FuXi- α : Scaling Recommendation Model with Feature Interaction Enhanced Transformer. *arXiv:2502.03036 [cs.IR]* <https://arxiv.org/abs/2502.03036>
- [66] Mingjia Yin, Hao Wang, Wei Guo, Yong Liu, Zhi Li, Sirui Zhao, Zhen Wang, Defu Lian, and Enhong Chen. 2024. Learning partially aligned item representation for cross-domain sequential recommendation. *arXiv preprint arXiv:2405.12473* (2024).
- [67] Mingjia Yin, Hao Wang, Wei Guo, Yong Liu, Suojuan Zhang, Sirui Zhao, Defu Lian, and Enhong Chen. 2024. Dataset regeneration for sequential recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3954–3965.
- [68] Mingjia Yin, Hao Wang, Xiang Xu, Likang Wu, Sirui Zhao, Wei Guo, Yong Liu, Ruiming Tang, Defu Lian, and Enhong Chen. 2023. Apg4sr: A generic framework with adaptive and personalized global collaborative information in sequential recommendation. In *Proceedings of the 32nd ACM international conference on information and knowledge management*. 3009–3019.
- [69] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems* 33 (2020), 17283–17297.
- [70] 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).
- [71] Biao Zhang and Rico Sennrich. 2019. Root Mean Square Layer Normalization. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 12360–12371. <https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html>
- [72] Gaowei Zhang, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Scaling Law of Large Sequential Recommendation Models. *arXiv preprint arXiv:2311.11351* (2023).

- [76] Jiaqing Zhang, Mingjia Yin, Hao Wang, Yawen Li, Yuyang Ye, Xingyu Lou, Junping Du, and Enhong Chen. 2025. TD3: Tucker Decomposition Based Dataset Distillation Method for Sequential Recommendation. *arXiv preprint arXiv:2502.02854* (2025).
- [77] Luankang Zhang, Kenan Song, Yi Quan Lee, Wei Guo, Hao Wang, Yawen Li, Huifeng Guo, Yong Liu, Defu Lian, and Enhong Chen. 2025. Killing Two Birds with One Stone: Unifying Retrieval and Ranking with a Single Generative Recommendation Model. arXiv:2504.16454 [cs.IR] <https://arxiv.org/abs/2504.16454>
- [78] Luankang Zhang, Hao Wang, Suojuan Zhang, Mingjia Yin, Yongqiang Han, Jiaqing Zhang, Defu Lian, and Enhong Chen. 2024. A Unified Framework for Adaptive Representation Enhancement and Inversed Learning in Cross-Domain Recommendation. In *International Conference on Database Systems for Advanced Applications*. Springer, 115–130.
- [79] 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.