LinRec: Linear Attention Mechanism for Long-term Sequential Recommender Systems

Langming Liu
City University of Hong Kong

Liu Cai
Ant Group

Chi Zhang
Harbin Engineering University

Xiangyu Zhao*
City University of Hong Kong

Jingtong Gao
City University of Hong Kong

Wanyu Wang
City University of Hong Kong

Yifu Lv
Ant Group

Wenqi Fan
The Hong Kong Polytechnic University

Yiqi Wang
National University of Defense Technology

Ming He
AI Lab, Lenovo Research

Zitao Liu
Guangdong Institute of Smart Education, Jinan University

Qing Li
The Hong Kong Polytechnic University

ABSTRACT

Transformer models have achieved remarkable success in sequential recommender systems (SRSs). However, computing the attention matrix in traditional dot-product attention mechanisms results in a quadratic complexity with sequence lengths, leading to high computational costs for long-term sequential recommendation. Motivated by the above observation, we propose a novel L2-Normalized **Lin**ear Attention for the Transformer-based Sequential **Rec**ommender Systems (LinRec), which theoretically improves efficiency while preserving the learning capabilities of the traditional dot-product attention. Specifically, by thoroughly examining the equivalence conditions of efficient attention mechanisms, we show that LinRec possesses linear complexity while preserving the property of attention mechanisms. In addition, we reveal its latent efficiency properties by interpreting the proposed LinRec mechanism through a statistical lens. Extensive experiments are conducted based on two public benchmark datasets, demonstrating that the combination of LinRec and Transformer models achieves comparable or even superior performance than state-of-the-art Transformer-based SRS models while significantly improving time and memory efficiency. The implementation code is available online at https://github.com/Applied-Machine-Learning-Lab/LinRec.

CCS CONCEPTS

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

*Xiangyu Zhao is corresponding author. xianzhao@cityu.edu.hk

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07…$15.00
https://doi.org/10.1145/3539618.3591717

KEYWORDS

Efficient Transformer, Sequential Recommender Systems, Linear Complexity, L2 Normalization

ACM Reference Format:
Langming Liu, Liu Cai, Chi Zhang, Xiangyu Zhao, Jingtong Gao, Wanyu Wang, Yifu Lv, Wenqi Fan, Yiqi Wang, Ming He, Zitao Liu, and Qing Li. 2023. LinRec: Linear Attention Mechanism for Long-term Sequential Recommender Systems. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3539618.3591717

1 INTRODUCTION

In recent years, sequential recommender systems (SRSs) have become an increasingly popular and widely-applied technology [7, 17, 22, 41, 49, 55, 57, 64, 74], with applications in various practical scenarios such as social media, e-commerce, and online movie platforms [14, 15, 34, 37, 44, 82–85]. In practice, users' historical interaction sequences are typically long-term, which contain valuable yet unequal information (i.e., different interactions' importance), including more and less recent interactions, for revealing users' actual preferences [67, 71, 72, 76, 77, 79, 86–90, 92]. Therefore, identifying important interactions while not losing valuable information from a sequence [6, 57], thus learning better sequence representation for making next-item recommendations, leads to the problem of *long-term sequential recommendation.*

Towards this purpose, the Transformer architecture [53] has gained significant attention since its capabilities for learning informative long-term sequential patterns among historical user-item interactions. The core component of Transformer-based models is the dot-product attention mechanism [53], which computes the corresponding attention matrix for distinguishing items' importance by a dot-product operation between the *query* and *key* matrices (Query and Key for short), thus learning sequence representations. For example, BERT4Rec [48] uses multi-head self-attention and simultaneously calculates the attention score of all positions. FDSA [75] introduces multiple attention blocks to depict the potential features. SASRec [24] controls the predictions based on only a small amount

of actions by adopting an attention mechanism. However, a significant limitation of these models arises when dealing with long-term sequences, where the sequence length N is far greater than the item embedding size d ($N \gg d$). Due to the dot-product operation in the attention layer, the computational and memory complexity of transformers are $O(N^2)$, resulting in the complexity could be dramatically increased with the problem scale (i.e., sequence lengths) for long-term sequential recommendation [30–32, 80].

In view of the above limitations, several approaches have been proposed to address the high computational costs of Transformer models, including Fixed Patterns (FP), Combination of Patterns (CP), Learnable Patterns (LP), Neural Memory, Low-Rank Methods, Kernels, and Recurrence [51]. Some of these categories, such as FP, LP, and low-Rank, are theoretically less complex than the standard dot-product attention and have been shown to effectively reduce memory and time costs in practice [4, 8, 25, 27, 38, 40, 58, 59]. Notably, there are several efficient transformer models with linear complexity, such as Linformer [58], Linear Transformers [25], and Big Bird [70]. However, these methods may not be well-suited for SRSs, particularly when dealing with long-term sequences, as they often introduce additional steps to improve efficiency while sacrificing accuracy and stability. For instance, to reduce the rank of the attention matrix, Linformer [58] introduces the projection for both Query and Key, which results in additional complexity and impairs accuracy, jeopardizing recommendation performance.

In this paper, we propose an efficient attention mechanism to address the issue of high complexity transformers with linear complexity, called L2-normalized **Lin**ear attention for long-term sequential **Rec**ommender systems (LinRec). Our proposed method aims to create a method for the long-term sequential recommender that not only retains the advantages of attention (such as high accuracy) but also significantly reduces the computational complexity. Additionally, LinRec can be conveniently transplanted to any transformer for sequential recommendation systems, providing flexibility and compatibility. Moreover, LinRec does not introduce additional steps, but directly enhances the attention layer to improve efficiency, preserving both effectiveness and stability. To be more specific, the proposed LinRec mechanism involves three key modifications compared to standard attention: (*i*) changing the dot-product order of the attention mechanism, (*ii*) using row-wise and column-wise normalization methods for Query ($\boldsymbol{Q}$) and Key ($\boldsymbol{K}$) respectively, and (*iii*) adding an activation layer to $\boldsymbol{Q}$ and $\boldsymbol{K}$. These modifications enable LinRec to reduce the complexity from $O(N^2)$ to $O(N)$, while still preserving the attention property and providing sparsity.

The major contributions to our work are four-fold:

- We develop a novel L2 normalized linear attention (LinRec) for long-term sequential recommendations, which reduces the complexity of the attention mechanism from $O(N^2)$ to $O(N)$, while preserving the high accuracy of the attention mechanism;
- We theoretically analyze the proposed LinRec mechanism, including its effectiveness and efficiency, justifying the correctness of our design choice. Moreover, we explain and factorize the attention operation from a statistical perspective, demonstrating the inherent relation of efficient Transformer with probabilities;
- The proposed LinRec mechanism is generally applicable to most transformer models for SRSs, as it can be easily incorporated into existing transformers by replacing the standard dot-product attention, providing flexibility and compatibility;
- Empirical evaluations are conducted on two public benchmark datasets (ML-1m and Gowalla), which demonstrates that LinRec possesses competitive or superior performance than representative Transformer-based SRS models, while greatly reducing computational and memory costs.

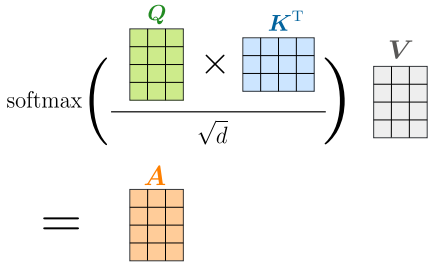


Figure 1: The standard process of Dot-Product Attention.

2 PRELIMINARY

In this section, we briefly introduce and discuss the widely used dot-product attention mechanism in Transformer-based models and the long-term sequential recommendation.

2.1 Dot-Product Attention

The critical part of transformers is the attention layer. The central concept underlying the attention mechanism is that each element in the sequence should learn to collect information from other tokens. Below is a standard dot-product attention. First, we define N as sequence length and d as hidden size. The standard attention mechanism can be written as:

$$\boldsymbol{A} = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d}})\boldsymbol{V},$$

$$where \quad \boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q, \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K, \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V,$$

where $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ is the input sequence matrix, $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V$ are weight matrices for the projections that are learned from the training process, $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{N\times d}$ are Query, Key and Value matrices which represent all N positions' queries, keys, and values, softmax$(\cdot)$ is row-wise softmax. $\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}$ is usually divided by $\sqrt{d}$ for scaling, and $\boldsymbol{A}$ is the output of attention mechanism.

Figure 1 shows that the standard attention mainly includes two dot-product operations, where Cross represents matrix multiplication. One of the highlights of the attention mechanism is that after the dot-product operation of Query, Key, and Value, the dimension of the output matrix will not vary. The dimensional invariance can convey the attention information to the sequence smoothly.

Advantages and Disadvantages. The attention matrix $\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}$ is vital for the attention mechanism. Firstly, this $N \times N$ attention matrix provides scores between elements in the sequence, which helps learn the latent self-correlation in the sequence. In addition, using a matrix that consists of all sequence information reduces sequence operations and alleviates the gradient vanishing problem compared to RNN. However, the disadvantage of attention is often about the complexity problem. The main complexity is generated by the dot-product operation of attention matrix, and the complexity is $O(N^2d)$, quadratic to the sequence length N. The quadratic

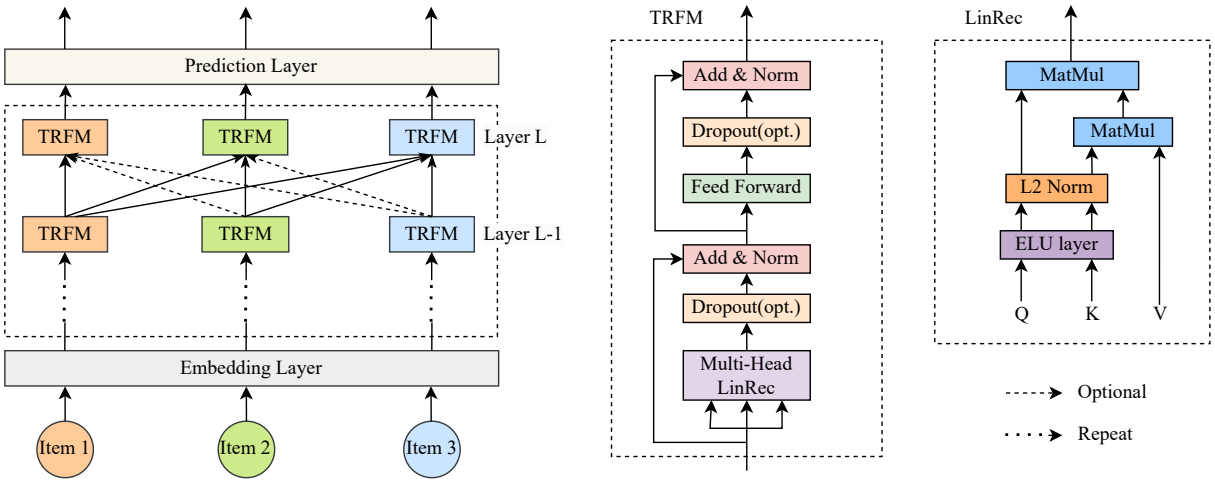


Figure 2: The overall architecture of the proposed model.

complexity will bring great memory and time costs when N is big enough. And the complexity is simplified to $O(N^2)$ when $N \gg d$. Therefore, utilizing a standard attention mechanism when dealing with long-term sequences is impractical. Besides, softmax layer aggregates attention scores on a few positions, which is not conducive to mastering more information from long-term sequences.

2.2 Long-term Sequential Recommender Systems

Long-term sequential recommender systems, i.e., Long-term SRSs, are defined mathematically as systems that take into account a user's historical interactions over an extended period rather than just their recent interactions. The definition of "long-term" can vary depending on the specific application and context. In general, long-term SRSs are characterized by the number of historical interactions that are considered when generating recommendations, denoted as N. A common way to define the threshold between short-term and long-term SRSs is based on the ratio N/d, where d is the dimension of the user or item embeddings, typically 64 to 128. In practice, a ratio of N/d greater than 1.5 is considered long-term, meaning that the number of historical interactions is at least 1.5 times the dimension of the embeddings. For example, if $d = 64$, a long-term SRS would consider at least 100 historical interactions.

3 METHODOLOGY

Given a set of user $\mathcal{U} = \{u_1, u_2, \cdots, u_{|\mathcal{U}|}\}$ and a set of item $\mathcal{V} = \{v_1, v_2, \cdots, v_{|\mathcal{V}|}\}$. Suppose that u_i's historical interaction sequence is $s_i = [v_1^i, \cdots, v_t^i, \cdots, v_{n_i}^i]$, where v_t^i is the t-th item interacted by user u_i, and n_i represents the sequence' length. For each user u_i, LinRec takes u_i's interaction sequence s_i as input, and outputs the top-k items from $\mathcal{V}$ that are most likely to be interacted with in the next time step. As illustrated in Figure 2, our proposed architecture consists of three major components, including the embedding layer to generate item embeddings, the transformer layer to learn sequence representation, and the final prediction layer. In particular, to improve efficiency and applicability for Long-term SRSs, we propose a novel attention mechanism in the transformer layer. In the following, we detail each component.

3.1 Embedding Layer

To jointly train LinRec with other Transformer-based backbones, we build two parameter matrices (i.e., $\boldsymbol{E}^s$ and $\boldsymbol{E}^p$) as embedding look-up tables for items' embedding initialization in a sequence via

$$\boldsymbol{E} = [\boldsymbol{E}_1^s + \boldsymbol{E}_1^p, \boldsymbol{E}_2^s + \boldsymbol{E}_2^p, \cdots, \boldsymbol{E}_N^s + \boldsymbol{E}_N^p]^{\mathrm{T}}, \tag{1}$$

where $\boldsymbol{E}^s, \boldsymbol{E}^p \in \mathbb{R}^{N \times d}$ are trainable matrices mapping items' IDs and position in sequence into d-dimension dense vectors.

3.2 Transformer Layer

With the above embedding layer, we are ready to integrate LinRec into existing Transformer-based recommenders to learn sequence representations. As mentioned in Sec. 2, traditional dot-product attention methods are inherently sub-optimal since they typically need high computational costs (i.e., $O(N^2 d)$) to calculate the attention matrix. To preserve dot-product attention's learning capabilities while reducing its computational cost, we consequently propose the LinRec mechanism for long-term SRSs. Specifically, we first analyze dot-product attentions' properties to derive equivalence conditions that inspire our method design. We then propose an efficient L2 Normalization method to modify the mapping (i.e., Softmax) of the dot-product, thus switching the calculation order (i.e., compute $\boldsymbol{K}^{\mathrm{T}}\boldsymbol{V}$ first) to reduce computational complexity. Moreover, we theoretically analyze the proposed LinRec, justifying its correctness and computational efficiency. Besides, we interpret LinRec's superiority from a statistical perspective attached in Appendix A.

3.2.1 ***Equivalence Conditions.*** Generally, dot-product attention mechanisms [53] calculate the corresponding attention matrix could be formulated as follows:

$$\boldsymbol{A} = \rho(\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}})V \; ; \; \boldsymbol{B} = \rho(\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}), \tag{2}$$

where $\rho(\cdot)$ means the scaling and row-wise Softmax operators. Accordingly, the attention matrix $\boldsymbol{B}$ satisfies the following two properties: (1) *Normalized Property*. Each row of $\boldsymbol{B}$'s elements sums up to 1. (2) *Non-Negative Property*. Each element of $\boldsymbol{B}$ is non-negative. Each row of $\boldsymbol{B}$'s elements is constrained into a range as $[0, 1]$ to easily distinguish the importance of different positions (i.e., element

value is proportional to the corresponding item importance) in a sequence and ensure numerical stability.

In practice, the computational complexity of computing $\boldsymbol{B}$ first could be dramatically increased with the problem scale (i.e., sequence length) [33, 40], which harms dot-product attentions' applicability for long-term sequences. Besides, $\rho(\cdot)$ makes $\rho(\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}})\boldsymbol{V} \neq \rho(\boldsymbol{Q})\rho(\boldsymbol{K}^{\mathrm{T}})\boldsymbol{V}$, which renders a non-trivial technical challenge (i.e., compute $\boldsymbol{B}$ last) for simplifying Eq. (2)'s complexity. To tackle such a challenge, we could devise another mapping instead of $\rho(\cdot)$ to decompose Eq. (2). Thus we could compute $\boldsymbol{K}^{\mathrm{T}}\boldsymbol{V}$ first to reduce computation complexity without jeopardizing attention's learning capabilities. Formally, we define such a mapping as follows:

$$\boldsymbol{A}' = \rho_1(\boldsymbol{Q})\rho_2(\boldsymbol{K})^{\mathrm{T}}\boldsymbol{V}\ ;\ \boldsymbol{B}' = \rho_1(\boldsymbol{Q})\rho_2(\boldsymbol{K})^{\mathrm{T}}, \tag{3}$$

where $\rho_1(\cdot)$ and $\rho_2(\cdot)$ are different mappings from $\mathbb{R}^{N\times d}$ to $\mathbb{R}^{N\times d}$, decomposing the original derivation (i.e., $\rho(\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}})$) into two parts (i.e., $\rho_1(\boldsymbol{Q})\rho_2(\boldsymbol{K})^{\mathrm{T}}$) for subsequent calculation. Therefore, we could specify the i-th row of $\boldsymbol{B}'$ via

$$\boldsymbol{B}'_i = \Big(\sum_{j=1}^{d} \boldsymbol{Q}^{\rho}_{ij}\boldsymbol{K}^{\rho}_{1j}, \sum_{j=1}^{d} \boldsymbol{Q}^{\rho}_{ij}\boldsymbol{K}^{\rho}_{2j}, \cdots, \sum_{j=1}^{d} \boldsymbol{Q}^{\rho}_{ij}\boldsymbol{K}^{\rho}_{Nj}\Big), \tag{4}$$

where $\boldsymbol{Q}^{\rho} = \rho_1(\boldsymbol{Q}), \boldsymbol{K}^{\rho} = \rho_2(\boldsymbol{K})$. Ideally, to satisfy the above requirements, we could derive two equivalence conditions as follows:

$$\textbf{Condition (1).}\ \sum_{m=1}^{N}\sum_{j=1}^{d} \boldsymbol{Q}^{\rho}_{ij}\boldsymbol{K}^{\rho}_{mj} \leq 1, \qquad \forall i = 1, 2, \cdots, N;$$

$$\textbf{Condition (2).}\ \sum_{j=1}^{d} \boldsymbol{Q}^{\rho}_{ij}\boldsymbol{K}^{\rho}_{mj} \geq 0, \qquad \forall i, m = 1, 2, \cdots, N.$$

Ultimately, an appropriate design of the above mappings could enable us to compute $(\boldsymbol{K}^{\rho})^{\mathrm{T}}\boldsymbol{V}$ first and achieve $\boldsymbol{O}(Nd^2)$ complexity. This is because $\boldsymbol{K}$ and $\boldsymbol{V}$ are both $N \times d$ matrices. Besides, omitting d ($N >> d$) and utilizing linear complexity mappings (i.e., $\rho_1(\cdot)$ and $\rho_2(\cdot)$) for long-term SRSs, we could achieve approximated linear complexity (i.e., $\boldsymbol{O}(N)$). However, linear complexity mappings are hard to satisfy the above conditions strictly. Toward this end, we further relax the above conditions and design the two mappings for approximating dot-product attention's learning capabilities (i.e., generate a comparable attention matrix for identifying items' importance) while significantly reducing complexity. Functionally speaking, such a proper mapping design should have the following requirements: (1) $\rho_1(\cdot)$ and $\rho_1(\cdot)$ do not introduce additional computational costs, and (2) they have to maintain attention's identifying capabilities and the numerical stability, thus learning better sequence representations to make recommendations for long-term SRSs efficiently and effectively.

3.2.2 *L2 Normalized Linear Attention Mechanism (LinRec)*.

According to the above conditions, we further derive an equivalent condition to provide theoretically designing insights. Mathematically, Condition (1) is equivalent to the following formulation:

$$\sum_{j=1}^{d} \boldsymbol{Q}^{\rho}_{ij} \sum_{m=1}^{N} \boldsymbol{K}^{\rho}_{mj} \leq 1, \tag{5}$$

and its sufficient condition is straightforward:

$$\textbf{Condition (3).}\ \sum_{j=1}^{d} \boldsymbol{Q}^{\rho}_{ij} \leq 1, \quad \sum_{m=1}^{N} \boldsymbol{K}^{\rho}_{mj} \leq 1.$$

Therefore, it is formally equivalent to Lp Normalization families, which constrains the summations in a small range. The Lp Normalization families represent a trade-off between the performance and computational cost of serving as mapping functions: a small value of p could introduce fewer computational costs with relatively inaccuracy approximation. Specifically, we leverage row- and column-wise L2 Normalization methods to perform the two mappings. This is because L2 Normalization is more efficient than L1 Normalization and more effective than others (e.g., $p = \infty$). For i-th row of $\boldsymbol{Q}$ (i.e., $\boldsymbol{Q}_i = (\boldsymbol{Q}_{i1}, \cdots, \boldsymbol{Q}_{id})$) and the j-th column of $\boldsymbol{K}$ (i.e., $\boldsymbol{K}_j = (\boldsymbol{K}_{1j}, \cdots, \boldsymbol{K}_{Nj})^{\mathrm{T}}$), we have:

$$\begin{aligned}
\rho_1(\boldsymbol{Q}_i) &= \frac{\boldsymbol{Q}_i}{\sqrt{d}\|\boldsymbol{Q}_i\|_2} = \frac{1}{\sqrt{d}\|\boldsymbol{Q}_i\|_2}(\boldsymbol{Q}_{i1}, \cdots, \boldsymbol{Q}_{id}),\\
\rho_2(\boldsymbol{K}_j) &= \frac{\boldsymbol{K}_j}{\sqrt{N}\|\boldsymbol{K}_j\|_2} = \frac{1}{\sqrt{N}\|\boldsymbol{K}_j\|_2}(\boldsymbol{K}_{1j}, \cdots, \boldsymbol{K}_{Nj})^{\mathrm{T}},
\end{aligned} \tag{6}$$

where $\|\cdot\|_2$ represents the L2 Normalization.

Proof: by Cauchy-Schwarz inequality [61], we obtain:

$$\begin{aligned}
(\boldsymbol{Q}_{i1} + \cdots + \boldsymbol{Q}_{id})^2 &= \Big(\sqrt{1\cdot\boldsymbol{Q}^2_{i1}} + \cdots + \sqrt{1\cdot\boldsymbol{Q}^2_{id}}\Big)^2\\
&\leq (1 + \cdots + 1)(\boldsymbol{Q}^2_{i1} + \cdots + \boldsymbol{Q}^2_{id})\\
&= d(\boldsymbol{Q}^2_{i1} + \cdots + \boldsymbol{Q}^2_{id})\\
&= d\|\boldsymbol{Q}_i\|_2^2,
\end{aligned}$$

then we can have:

$$\boldsymbol{Q}_{i1} + \cdots + \boldsymbol{Q}_{id} \leq \sqrt{d}\|\boldsymbol{Q}_i\|_2. \tag{7}$$

Therefore, $\rho_1(\cdot)$ and $\rho_2(\cdot)$ satisfy Condition (3), ensuring the normalized property of the generated attention matrix $\boldsymbol{B}'$.

However, such normalization methods do not necessarily satisfy the non-negative condition. We consequently incorporate the Rectified Linear Unit (ReLU) [16] function families to eliminate negative values' influence. Although utilizing the standard ReLU function could strictly satisfy non-negativity, such an activated function may meet zero-gradient error [18, 28, 66], which leads to an unstable training process. Besides, the ReLU function could meet inevitable information loss issues by fully setting negative elements to zero since each item should contribute to representation learning in a sequence. Therefore, we use a variation of the ReLU families, ELU function [10], to perform activation and control negative values. Specifically, we formulate such a function via

$$\mathrm{elu}(x) := \begin{cases} x, & \text{if } x \geq 0,\\ e^x - 1, & \text{if } x < 0. \end{cases} \tag{8}$$

Accordingly, utilizing the $\mathrm{elu}(\cdot, \alpha = 1)$ on a matrix x (i.e., $\boldsymbol{Q}$ and $\boldsymbol{K}$), most elements of x are non-negative, and others would be constrained to be smaller as compared to linear mappings, while improving negative value robustness and ensuring linear complexity. Furthermore, leveraging the proposed L2 Normalization and ELU activation for traditional dot-product attention mechanisms satisfies the aforementioned requirements:

- It reduces the computational complexity of traditional dot-product attention mechanisms to $O(N)$ for long-term SRSs.
- It does not introduce additional computational costs since both the L2 Normalization and $\text{elu}(\cdot, \alpha = 1)$ are linear complexity operators that do not require other trainable parameters.
- It could distinguish different items' importance in a sequence by comparing attention scores. This is because items' importance is still proportional to the corresponding element values in $\boldsymbol{B}'$.
- It could ensure a relatively stable learning process. Since the L2 Normalization decreases the variance of attention scores by constraining the matrix's row-wise summations less than or equal to 1. Besides, the ELU function could prevent the calculation process from zero-gradient errors, thus ensuring numerical stability.

Towards this end, we formulate the final architecture of the proposed **L2 Normalized Linear Attention** (LinRec) mechanism to learn sequence representations as follows:

$$A'(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \rho_1\big(\text{elu}(\boldsymbol{Q})\big)\Big(\rho_2\big(\text{elu}(\boldsymbol{K})\big)^{\mathrm{T}}\boldsymbol{V}\Big), \tag{9}$$

where the row-wise mapping is defined as $\rho_1(\boldsymbol{Q}_i) = \frac{1}{\sqrt{d}\|\boldsymbol{Q}_i\|_2}\boldsymbol{Q}_i$ for $\forall i \in [N]$, and the column-wise mapping is defined as $\rho_2(\boldsymbol{K}_j) = \frac{1}{\sqrt{N}\|\boldsymbol{K}_j\|_2}\boldsymbol{K}_j$ for $\forall j \in [d]$, where $\boldsymbol{Q}_i$ is i-th row of $\boldsymbol{Q}$ and $\boldsymbol{K}_j$ is j-th col of $\boldsymbol{K}$. Therefore, we could integrate Eq. (9) into existing Transformer-based recommenders [12, 13, 35, 68] to generate sequence representations as follows:

$$\begin{aligned}
\text{head}_i &= \boldsymbol{A'}(\boldsymbol{H}^l\boldsymbol{W}_Q^{(i)}, \boldsymbol{H}^l\boldsymbol{W}_K^{(i)}, \boldsymbol{H}^l\boldsymbol{W}_V^{(i)}),\\
\text{MH}(\boldsymbol{H}^l) &= \text{Concat}(\text{head}_1, \cdots, \text{head}_h)\boldsymbol{W}_O,\\
\boldsymbol{S}^{l-1} &= \text{LayerNorm}(\boldsymbol{H}^{l-1} + \text{Dropout}(\text{MH}(\boldsymbol{H}^{l-1}))),\\
\boldsymbol{H}^l &= \text{LayerNorm}(\boldsymbol{S}^{l-1} + \text{Dropout}(\text{FNN}(\boldsymbol{S}_{l-1})))),\\
\boldsymbol{H}^1 &= \boldsymbol{E}\ ;\ \boldsymbol{H} = \boldsymbol{H}^L\boldsymbol{W}_L + \boldsymbol{b}_L,
\end{aligned} \tag{10}$$

where $\boldsymbol{W}_Q^{(i)}, \boldsymbol{W}_K^{(i)}, \boldsymbol{W}_V^{(i)} \in \mathbb{R}^{d\times d}$ are weight matrices at head i, $\boldsymbol{W}_O$ is weight matrix at multi-head (MH) block, LayerNorm refers to layer normalization function [3], $\boldsymbol{H}^l$ is hidden value at layer l ($l = 1, \cdots, L$) which is iteratively generated until L, $\text{FNN}(\cdots)$ is Feed-Forward Network, $\boldsymbol{W}_L \in \mathbb{R}^{hd\times d}, \boldsymbol{b}_L \in \mathbb{R}^d$ are weight and bias respectively. Eventually, we get sequence representation $\boldsymbol{H} \in \mathbb{R}^{N\times d}$.

3.3 Prediction and Model Optimization

After obtaining item representations (i.e., $\boldsymbol{H} \in \mathbb{R}^{N\times d}$) in a sequence, we make the next-item recommendation by calculating a probability distribution of the next-item of the whole item set. At time t, for each candidate item v_i, we can calculate its recommendation score:

$$\boldsymbol{z}_i = \boldsymbol{H}_t(\boldsymbol{e}_i^s)^{\mathbf{T}}, \tag{11}$$

where $\boldsymbol{H}_t$ it the t-th item's representation in a sequence, performing the sequence representation, $\boldsymbol{e}_i^s \in \mathbb{R}^d$ is v_i's embedding. Consequently, the recommended probability of that the next-item being v_i, $\hat{y}_i$, could be computed as follows:

$$\hat{y}_i = \frac{\exp(z_i)}{\sum_{v_j\in\mathcal{V}}\exp(z_j)}. \tag{12}$$

Table 1: Statistics of the datasets.

Datasets	# Users	# Items	# Interactions	# Sparsity
ML-1M	6,041	3,884	1,000,209	95.74%
Gowalla	64,116	164,533	2,018,421	99.98%

Therefore, we formulate the sequential recommendation task as minimizing the cross-entropy of the recommendation results $\hat{y}$ to measure the difference between the prediction and ground truth as

$$\mathcal{L}(y, \hat{y}) = y\log(\hat{y}) + (1-y)(1-\log(\hat{y})). \tag{13}$$

The training target is to obtain the best parameters of the network (including $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V$ and other parameters outside the attention layer). We will use Adam [26] optimizer to optimize the network, a stochastic method using moment estimation.

3.4 In-depth Analyses

In this section, we theoretically analyze the proposed LinRec in terms of its advantages and complexity.

3.4.1 ***Comparison with Other Normalization Methods.*** An intuitive way to implement Eq. (3) is to leverage the Softmax function twice (i.e., row- and column-wise) to perform mapping functions $\rho_1(\cdot)$ and $\rho_2(\cdot)$ via

$$\rho_1(\boldsymbol{Q}) = \rho(\boldsymbol{Q})\ ;\ \rho_2(\boldsymbol{K}) = \rho(\boldsymbol{K}), \tag{14}$$

where $\rho(\cdot)$ means the row- and column-wise Softmax for generating $\rho_1(\boldsymbol{Q})$ and $\rho_2(\boldsymbol{K})$, respectively. The Softmax function contains an exponential operation and a weight distribution operation for a $\mathbb{R}^d$ (or $\mathbb{R}^N$) sequence(vector). For a vector $\boldsymbol{x} = (x_1, \cdots, x_n)$, we could formulate it as follows:

$$\rho(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}. \tag{15}$$

It is evident that $\rho(\cdot)$ can satisfy Condition (3). Nevertheless, a fundamental property of $\rho(\cdot)$ is that the exponential function aggregates the weight into only a few positions of the sequence, which may exaggerate importance (i.e., long-tailed attention scores) of a few items in a sequence ignoring the others, thus loss useful information from the entire sequence for long-term SRSs. In contrast to such a method, the proposed L2 Normalization could generate relatively smooth attention scores to better capture information.

3.4.2 ***Model Complexity Analysis.*** We analyze the computational complexity of the LinRec mechanism (i.e., Eq. (9)) from several key components: (i) Firstly, the computational cost of $\text{elu}(\cdot, \alpha = 1)$ function is activating elements in $\boldsymbol{Q}$ and $\boldsymbol{K}$, which are two $N \times d$ matrices. Thus ELU's complexity is $O(Nd)$; (ii) Secondly, $\rho_1(\cdot)$ and $\rho_2(\cdot)$ perform L2 Normalization. Therefore, their total complexity is $O(Nd)$; (iii) Thirdly, the matrix multiplication operator's complexity is $O(Nd^2)$. Towards this end, the time complexity of LinRec is $O(Nd^2)$. For long-term SRSs, it could significantly reduce the time complexity of Transformer-based recommenders (we omit d since $N >> d$) from $O(N^2)$ to $O(N)$. In conclusion, our model could achieve comparable efficiency with state-of-the-art lightweight Transformer-based sequential recommendation techniques (e.g., Linformer [58], Linear Transformers [25], and Big Bird [70]).

Table 2: Overall performance comparison.

Datasets	Metrics	Bert4Rec		CORE		FDSA		SASRec		SASRecF	
		w/o	w	w/o	w	w/o	w	w/o	w	w/o	w
ML-1M	Recall@10	0.6975	0.6997	0.5406	0.6088	0.7166	0.7159	**0.7212**	0.7209	0.7099	0.7113
	MRR	0.3700	0.3699	0.2051	0.2792	0.4272	0.4299	0.4282	**0.4301**	0.4230	0.4249
	NDCG@10	0.4483	0.4488	0.2835	0.3570	0.4964	0.4983	0.4974	**0.4997**	0.4921	0.4933
Gowalla	Recall@10	0.8717	0.8739	0.9190	**0.9242**	0.8981	0.8987	0.9174	0.9171	0.9077	0.9072
	MRR	0.5886	0.5907	0.6948	**0.7031**	0.6247	0.6358	0.6690	0.6723	0.6179	0.6400
	NDCG@10	0.6567	0.6589	0.7492	**0.7569**	0.6907	0.6995	0.7293	0.7317	0.6880	0.7048

Results of backbone models with (w) and without (w/o) LinRec mechanism have been shown. All improvements are **statistically significant** (i.e., two-sided t-test with $p < 0.05$) over backbone models, except Recall of SASRec. In each row, the best result is bold, while the second-best result is underlined.

4 EXPERIMENTS

In this section, we aim to answer the following research questions:

- **RQ1**: How does integrating LinRec into Transformer-based recommenders perform compared with the original ones?
- **RQ2**: Does LinRec outperforms other state-of-the-art efficient Transformer variants for sequential recommendation?
- **RQ3**: How is the scalability of the LinRec mechanism?
- **RQ4**: How do different components contribute to LinRec?
- **RQ5**: How is the interpretation ability of LinRec?

4.1 Experimental Settings

4.1.1 ***Datasets and Evaluation Metrics.*** To evaluate the effectiveness of the proposed LinRec, we conduct experiments on two publicly available datasets: (1) **ML-1M**: it contains users' ratings (about one million) on movies, and is a popular dataset for sequential recommendation. (2) **Gowalla**: it contains users' check-ins collected by SNAP group. The statistical information is presented in Table 1. Identical to the previous studies [1, 24, 43, 63, 69], we group user interactions in chronological order on all datasets and split data by the leave-one-out strategy. For example, for an input sequence $s_i = [v_1^i, \cdots, v_t^i, \cdots, v_{n_i}^i, v_{n_i+1}^i, v_{n_i+2}^i, v_{n_i+3}^i]$, we use $([v_1^i, \cdots, v_{n_i}^i], v_{n_i+1}^i)$ for training, $([v_1^i, \cdots, v_{n_i+1}^i], v_{n_i+2}^i)$ for validation, and $([v_1^i, \cdots, v_{n_i+2}^i], v_{n_i+3}^i)$ for testing. Moreover, a variety of common evaluation metrics, including *Recall*, *Mean Reciprocal Rank* (MRR), and *Normalized Discounted Cumulative Gain* (NDCG), are used for top-k ($k = 10$) recommendations performance evaluation.

4.1.2 ***Baselines.*** To demonstrate the effectiveness of our proposed method, we compare LinRec with a wide range of representative state-of-the-art Transformer-based recommenders, including traditional (i.e., $O(N^2d)$ complexity) and efficient (i.e., $O(Nd^2)$ complexity) Transformer-based methods.

Traditional Transformer: (1) **BERT4Rec** [48] uses bidirectional Transformer to fuse sequence information and a mask & fill self-supervised task to enhance sequence representations. (2) **CORE** [20]: leverages a simple yet efficient framework to unify representation space, thus generating better representations bridging the inconsistency between items and sessions. (3) **FDSA** [75]: equips different self-attention blocks to learn item- and feature-wise representations, thus enhancing sequence representation learning. (4) **SASRec** [24]: uses the traditional self-attention block (i.e., multi-head attention) for generating sequence representation. (5) **SASRecF**[1]: improves the learning capabilities of SASRec by introducing item features as additional contextual information.

Efficient Transformer: (1) **Linear Transformer** [25]: rearranges the computational operation of the dot product in the self-attention mechanism. The authors find that the modified process contains a repetitious operation that they can reuse to reduce complexity. (2) **Efficient Attention** [47]: separates the dot-product attention into two processes conducted by different scaling mappings.

4.1.3 ***Implementation Details.*** Identical to previous studies [20, 24, 48], the trainable parameters are initialized with a Gaussian distribution. We optimize LinRec with other Transformer-based recommenders with Adam [26] and use the default learning rate of 0.001 and default mini-batch size of 2,048 (we decrease the mini-batch size on Gowalla to 512 to avoid GPU memory errors). Suggesting by the original papers [20, 24, 25, 47, 48, 75], we set the hyper-parameters for all models, including Transformer layer as $L = 2$, attention head as $h = 8$, and inner size (e.g., FNN layers) as 256. In particular, we set dimension size as $d = 128, 64$ and maximum sequence length $N = 200, 100$ on ML-1M and Gowalla datasets, respectively. We further pad short-term sequences (i.e., those with $n_i < N$) by zero to ensure $n_i > d$ for long-term sequential recommendation. The maximum number of training epochs is 100. Moreover, we adopt the early-stopping training strategy if the NDCG@10 performance on the validation set decreases for 10 continuous epochs. We implement our model[2] in PyTorch 1.13, Python 3.7.15, and RecBole 1.1.1 [81].

4.2 Traditional Transformer Comparison (RQ1)

To demonstrate the effectiveness of LinRec for improving recommendation performance, we integrate LinRec into other representative Transformer-based recommenders (e.g., BERT4Rec and SASRec). We report the main experimental results in Table 2, where we can draw a few interesting observations as follows:

- Generally, integrating with LinRec, Transformer-based recommenders perform significantly better than the original ones in most cases, demonstrating the effectiveness of the proposed

[1] https://www.recbole.io/docs/index.html
[2] https://github.com/Applied-Machine-Learning-Lab/LinRec

Table 3: Performance comparison with state-of-the-art efficient Transformer-based methods.

Datasets	Model	Recall@10	MRR	NDCG@10
ML-1M	LT+SASRec	0.6575	0.3359	0.4124
	EA+SASRec	0.6873	0.4094	0.4760
	LinRec+SASRec	**0.7209**	**0.4301**	**0.4997**
	Imprv.	7.1%~9.6%	9.6%~28.0%	8.8%~21.7%
Gowalla	LT+SASRec	0.9113	0.6627	0.7230
	EA+SASRec	0.9139	0.6577	0.7198
	LinRec+SASRec	**0.9171**	**0.6723**	**0.7317**
	Imprv.	0.4%~0.6%	1.4%~2.2%	1.2%~1.6%

All improvements are statistically significant (i.e., two-sided t-test with $p < 0.05$) over baseline. In each row, the best result is bold.

method. We attribute such improvements to LinRec could improve the long-term information capturing capabilities of Transformer by generating a relatively smooth attention matrix.

- Comparing with other recommenders, integrating LinRec into SASRec (LinRec+SASRec) consistently yields best performance on ML-1M, while LinRec+CORE resulting the best performance on Gowalla. Such results present that CORE is more suitable for sparse datasets (i.e., short-term sequences), while equipping with the proposed LinRec could significantly improve the performance on both ML-1M and Gowalla datasets, again demonstrating the superiority of LinRec.

4.3 Efficient Transformer Comparison (RQ2)

To demonstrate the superiority of LinRec over state-of-the-art efficient Transformer methods, we take the SASRec as a backbone, and report the experimental results on ML-1M and Gowalla datasets in Table 3. All improvements are statistically significant by performing two-sided t-test with $p < 0.05$. Accordingly, we have a few observations as follows:

- In general, the proposed LinRec+SASRec consistently outperforms other approaches on ML-1M and Gowalla datasets. In particular, LinRec+SASRec obtains an average of about 10% improvement over other baselines on ML-1M dataset. Such results generally demonstrate the superiority of the proposed LinRec.
- Comparing the other methods (i.e., LT+SASRec and EA+SASRec) with the original SASRec model, integrating with such methods does not achieve consistent improvements. It is because these methods do not satisfy traditional dot-product attention conditions, which harms Transformer's learning abilities. In contrast, the proposed LinRec satisfies the properties and generates relatively smooth attention scores for sequence representation learning, which can yield better performance.

4.4 Scalability Study (RQ3)

To investigate the model efficiency, we evaluate the computational cost, including GPU memory and Time, of the proposed LinRec as compared to traditional Transformer baselines.

Efficiency of Different Backbones. In Table 4, we observe that LinRec mechanism exactly reduces the GPU memory and Time cost,

Table 4: Efficiency comparison.

Datasets	Model	GPU memory (GB)		Time cost (s/epoch) Training		Evaluation	
		w/o	w	w/o	w	w/o	w
ML-1M	BERT4Rec	20.74G	10.42G	246s	157s	56s	37s
	CORE	19.49G	6.79G	109s	33s	38s	16s
	FDSA	35.22G	13.87G	270s	99s	79s	36s
	SASRec	21.07G	8.72G	121s	44s	41s	20s
	SASRecF	22.71G	10.33G	141s	64s	46s	23s
Gowalla	BERT4Rec	20.10G	19.46G	483s	455s	516s	454s
	CORE	3.75G	2.59G	126s	94s	570s	377s
	FDSA	4.81G	3.48G	209s	116s	834s	550s
	SASRec	3.75G	2.68G	112s	90s	434s	384s
	SASRecF	3.85G	3.22G	117s	93s	445s	400s

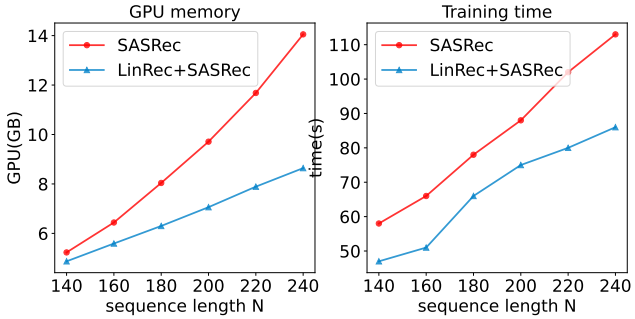


Figure 3: Parameter sensitivity.

demonstrating the high efficiency of LinRec. For example, LinRec performs extraordinarily to improve efficiency on ML-1M, which reduces the GPU memory and Time cost to approximately one-third for most baseline models. Such a result generally conforms to the theoretical analysis of linear complexity (i.e., LinRec could reduce complexity from $O(N^2d)$ to $O(Nd^2)$).

Efficiency of Different Sequence Lengths. To investigate the training cost of different sequence lengths for long-term SRSs, we tune maximum sequence lengths in $\{140, 160, \cdots, 240\}$ according to the hidden size (i.e., $d = 128$, the lengths of long-term sequences typically larger than such a dimension size) settings. From Figure 3, we see the LinRec+SASRec considerably reduces the computational complexity of SASRec consistently in all cases. This is highly desirable in practice, especially for long-term sequential recommendation, where user historical interaction sequences are stored for a long time, thus learning user preferences comprehensively.

4.5 Ablation Study (RQ4)

To verify the contribution of each component of the proposed LinRec, we conduct an ablation study with two variants of the LinRec+SASRec (i.e., Eq. (9)) over the Gowalla dataset, including (1) *w/o L2 norm*: without the L2 normalization layer, and (2) *w/o ELU*: without the ELU activation layer. Figure 4 shows the performances of different variants in terms of Recall@10, MRR, and NDCG@10. It can be observed that each component contributes to performance. The L2 normalization is particularly crucial for LinRec, justifying

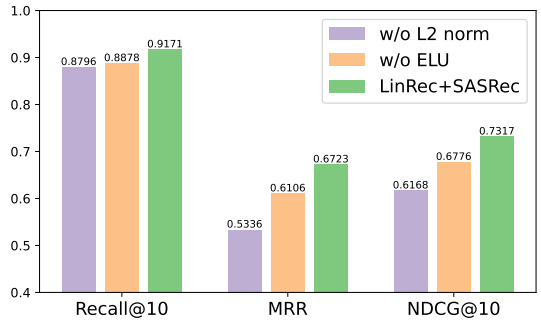


Figure 4: The results of ablation study.

that satisfying the normalized property (i.e., attention score summations less than or equal to 1) is the most critical part of preserving the learning capabilities of attention mechanisms. Besides, the ELU activation layer is indispensable for achieving encouraging results. Furthermore, LinRec+SASRec consistently improves performance in all cases, confirming the correctness of our design choice.

4.6 Case Study (RQ5)

Finally, we conduct a case study to visualize the attention scores and illustrate how the attention matrix learned in LinRec can help to capture more information in long-term sequences. In Figure 5, we show a user whose historical interaction sequence length is 50 from the ML-1M dataset, where the darker the color is, the higher the attention score is. Moreover, we pad the sequence with zero to ensure its length is larger than the embedding dimension, and we omit the padded part for simplicity. Comparing the heatmaps generated by SASRec and LinRec+SASRec, the proposed LinRec tends to generate relatively smooth attention scores, which can capture more information, including more recent or less recent items. In contrast, SASRec tends to attend to a few items, which may exaggerate such items' importance and leads to an inevitable information-losing issue for long-term sequence learning.

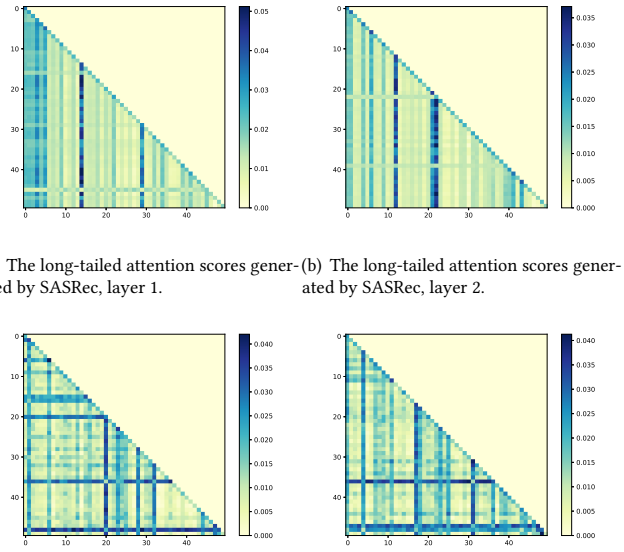
Specifically, comparing with Figure 5(a) and Figure 5(b), SASRec assigns larger attention scores on recent items, which is consistent with the observations in the original paper [24], showing its limited capabilities of learning long-term sequential patterns. In contrast, comparing with Figure 5(c) and Figure 5(d), LinRec can gradually attend to both short- and long-term patterns with increasing Transformer layers. In conclusion, this case aligns with our motivation that LinRec can further improve traditional Transformer-based recommenders' learning capabilities for long-term SRSs.

5 RELATED WORKS

In this section, we concisely review the Transformer-based SRSs and efficient Transformers to discuss the differences between the proposed LinRec mechanism and the related ones.

5.1 Transformer-based SRSs

In view of the Transformers' superior capabilities of learning sequence representations, various studies [5, 20, 24, 48, 62, 73, 75] leverage Transformers to learn sequence representation. The key



(a) The long-tailed attention scores generated by SASRec, layer 1.

(b) The long-tailed attention scores generated by SASRec, layer 2.

(c) The relatively smooth attention scores generated by LinRec+SASRec, layer 1.

(d) The relatively smooth attention scores generated by LinRec+SASRec, layer 2.

Figure 5: Heatmap of attention scores of different methods.

idea is that Transformers can distinguish different items' importance, capturing users' long- and short-term preferences to learn better sequence representations. Specifically, ATTRec [73] applies attention mechanism and metric embedding to absorb users' long- and short-term information, respectively. SASRecc [24] captures long- and short-term sequential dynamics based on a multi-head attention mechanism. BERT4Rec [48] is proposed to address the issue that a unidirectional model is hard to discover latent features of interaction sequences. In addition, the Cloze objective is used to improve the efficiency of the training process. FDSA [75] utilizes additional feature-level sequences in Transformer to better extract sequential patterns. BST [5] uses the Transformer model to learn the sequential nature of users' behaviors. SSE-PT [62] applies a personalized Transformer into self-attentive neural network architectures to improve ranking performance. CORE [20] utilizes a representation-consistent encoder and a robust distance measuring method to improve traditional sequential encoders' (e.g., self-attention mechanism) capabilities of learning representations.

While these methods are effective, they often employ traditional dot-product attention mechanisms, which result in an inefficient learning process, particularly when calculating attention matrices.

5.2 Efficient Transformers

To tackle the high complexity issue of Transformers, a new line of research has started to analyze and propose various efficient methods to reduce the inherently computational complexity. Such methods mainly aim to leverage approximated algorithms to simplify the attention matrix computing process (e.g., sparse matrix computing) in self-attention mechanisms [51].

The earliest research line of efficient Transformers, named fixed patterns [2, 4, 8, 11, 36, 40], designs attention matrices' sparse architecture based on specific patterns such as blocks and strides. The block-wise models [38, 40], one approach of fixed patterns,

separate the attention matrix into several blocks and use a random masking matrix to decide whether to retain each block. Another famous method of fixed patterns concerns striding approaches [4, 8], where several strides separate the masking matrix in equal intervals. Moreover, the Axial transformers [19] provides a new view of applying attention mechanisms for multidimensional tensors. Unlike fixed patterns, learnable patterns [27, 45, 50, 54, 59] consider the masking matrices can be learned. Cluster-Former [59] uses the clustering method to learn the classes of positions in the sequence and separate the sequence into chunks. Then they perform the transformers in each short chunk to obtain efficiency. Reformers [27] improve the efficiency by utilizing the LSH attention, where they use locality-sensitive hashing at each round.

Neural memory [4, 23, 29, 36, 42, 46, 52] method tries to utilize information about multiple positions in the sequence and their relevant connections. Set Transformer [29] detects and utilizes the interactions of positions in the sequence. They apply the inducing points to derive their particular attention mechanism. Longformer [4] uses the sliding window to capture the local context, meanwhile introducing global attention to aggregate valuable information from selected positions. Another recently famous approach of efficient Transformers introduces low-rank approximations to reduce the complexity and simplify models [9, 21, 25, 39, 47, 58, 60, 65, 91]. Based on the spectrum results and theoretical analysis that self-attention is a low-rank matrix, the Linformer [58] derives a brief attention mechanism for reducing complexity. They use linear projections matrix to reduce the dimension of Key and Value matrices. The Linear transformers [25] introduce a kernel-like method. Specifically, the authors disassemble and reassemble the dot-product operation of attention mechanisms. Then they find a part of the operation repeated in each round that can be reused for efficient computing. Besides, some studies further explore leveraging efficient model retraining [78] or sequence chunking [56] approaches to enhance computational efficiency for long-term SRSs. They mainly focus on transferring knowledge or editing data-level structures, which do not undermine our technical contributions of reducing the computational complexity of traditional Transformer-based SRSs (e.g., dot-product attention mechanisms) when calculating attention matrices.

Despite existing efficient Transformer methods decreasing the computational complexity, they either need to sacrifice more accuracy or perform low efficiency. In contrast, the proposed LinRec effectively preserves attention mechanisms' advantageous properties and reduces computational costs without jeopardizing performance.

6 CONCLUSION

In this paper, we studied the problem of long-term sequential recommendation from a new perspective–how to reduce the computational complexity of traditional Transformer-based SRSs models raised by the dot-product operations. We theoretically analyze the core properties of the dot-product attention mechanism for distinguishing items' importance in sequences. Accordingly, we propose a novel LinRec mechanism possessing linear complexity $O(N)$ while capturing more information from long-term sequences, thus generating better sequence representations for making recommendations efficiently and effectively. Comprehensive experiments demonstrate that LinRec has the excellent capability of improving efficiency while keeping accuracy. Significantly, LinRec outperforms two state-of-the-art efficient Transformer-based methods. In addition, we provide adequate theories and discussions to support our LinRec mechanism, where two highlights are the equivalence conditions and the statistical interpretation. The equivalence conditions provide theoretical insights for our design choice and can assist us in generating other efficient attention mechanisms for tasks other than long-term SRSs. The Statistical Interpretation that all elements have statistical meaning provides a solid foundation for the proposed LinRec mechanism.

A STATISTICAL INTERPRETATION

Statistically, Condition (1) means the probabilities of all positions in a row add up to no more than 1, while Condition (2) means the probabilities of all positions are more than or equal to 0. Therefore, similar to dot-product attention, the attention scores of LinRec satisfy the corresponding probability properties. Considering each position i in a sequence, we denote i's attention to position k as $\mathcal{A}_{ik}$, and the corresponding independent probability as $\mathrm{P_r}(\mathcal{A}_{ik})$. Then Conditions (1) and (2) can be rewritten in probability format:

$$\sum_{k=1}^{N} \mathrm{P_r}(\mathcal{A}_{ik}) \leq 1 \; ; \; \mathrm{P_r}(\mathcal{A}_{ik}) \geq 0, \quad \forall k. \tag{16}$$

Then we instead consider separating attention into d sub-events by d latent features, e.g., $\mathcal{B}_{i1}, \cdots, \mathcal{B}_{id}$, which respect to the hidden states of attention mechanism. And we assume that the sub-events $\mathcal{B}_{ij}$ are independent. In this way, we can define the probability $\mathrm{P_r}(\mathcal{B}_{ij})$ and the conditional probability $\mathrm{P_r}(\mathcal{A}_{ik}|\mathcal{B}_{ij})$. Therefore, we could rewrite the probability applying Bayes theorem as follows

$$\mathrm{P_r}(\mathcal{A}_{ik}) = \sum_{j=1}^{d} \mathrm{P_r}(\mathcal{B}_{ij}) \mathrm{P_r}(\mathcal{A}_{ik}|\mathcal{B}_{ij}). \tag{17}$$

We then substitute probabilities by elements of $\boldsymbol{Q}^{\rho}$ and $\boldsymbol{K}^{\rho}$ as

$$\begin{aligned} \mathrm{P_r}(\mathcal{B}_{ij}) = \boldsymbol{Q}^{\rho}_{ij} \; ; \; \mathrm{P_r}(\mathcal{A}_{ik}|\mathcal{B}_{ij}) = \boldsymbol{K}^{\rho}_{kj}, \\ \mathrm{P_r}(\mathcal{A}_{ik}) = \sum_{j=1}^{d} \boldsymbol{Q}^{\rho}_{ij} \boldsymbol{K}^{\rho}_{kj} = \boldsymbol{B}'_{ik}, \end{aligned} \tag{18}$$

where $\boldsymbol{B}'_{ik}$ (in Eq. (3)) is the attention score of position i to position k. Also, we can rewrite Condition (3) as

$$\sum_{j=1}^{d} \mathrm{P_r}(\mathcal{B}_{ij}) \leq 1 \; ; \; \sum_{k=1}^{N} \mathrm{P_r}(\mathcal{A}_{ik}|\mathcal{B}_{ij}) \leq 1. \tag{19}$$

Thus, all components of LinRec have statistical meaning. In addition, we observe that Eq. (18) holds for any i. Then we reveal an interesting property of the LinRec mechanism:

$$\mathrm{P_r}(\mathcal{A}_{i_1 k}|\mathcal{B}_{i_1 j}) = \mathrm{P_r}(\mathcal{A}_{i_2 k}|\mathcal{B}_{i_2 j}) = \boldsymbol{K}^{\rho}_{kj}. \tag{20}$$

Accordingly, such a shared conditional probability can significantly reduce parameter numbers, resulting in high-efficiency computing.

ACKNOWLEDGMENTS

This work was supported by Ant Group Research Fund. It was also partially supported by APRC - CityU New Research Initiatives (No.9610565, Start-up Grant for New Faculty of City University of Hong Kong), SIRG - CityU Strategic Interdisciplinary Research Grant (No.7020046, No.7020074), CityU-HKIDS Early Career Research Grant (No.9360163), InnoHK initiative, The Government of the HKSAR, and Laboratory for AI-Powered Financial Technologies.

REFERENCES

[1] M Mehdi Afsar, Trafford Crump, and Behrouz Far. 2022. Reinforcement learning based recommender systems: A survey. *Comput. Surveys* 55, 7 (2022), 1–38.

[2] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding long and structured inputs in transformers. *arXiv preprint arXiv:2004.08483* (2020).

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[4] Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* (2020).

[5] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior sequence transformer for e-commerce recommendation in alibaba. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*. 1–4.

[6] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 108–116.

[7] Yongjun Chen, Zhiwei Liu, Jia Li, Julian McAuley, and Caiming Xiong. 2022. Intent contrastive learning for sequential recommendation. In *Proceedings of the ACM Web Conference 2022*. 2172–2182.

[8] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509* (2019).

[9] Krzysztof Choromanski, Valerii Likhosherstov, 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).

[10] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* (2015).

[11] Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *Advances in neural information processing systems* 33 (2020), 4271–4282.

[12] Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. 2021. Transformers4rec: Bridging the gap between nlp and sequential/session-based recommendation. In *Proceedings of the 15th ACM Conference on Recommender Systems*. 143–153.

[13] Hanwen Du, Hui Shi, Pengpeng Zhao, Deqing Wang, Victor S Sheng, Yanchi Liu, Guanfeng Liu, and Lei Zhao. 2022. Contrastive Learning with Bidirectional Transformers for Sequential Recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 396–405.

[14] Wenqi Fan, Tyler Derr, Xiangyu Zhao, Yao Ma, Hui Liu, Jianping Wang, Jiliang Tang, and Qing Li. 2021. Attacking Black-box Recommendations via Copying Cross-domain User Profiles. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 1583–1594.

[15] Yingqiang Ge, Shuchang Liu, Ruoyuan Gao, Yikun Xian, Yunqi Li, Xiangyu Zhao, Changhua Pei, Fei Sun, Junfeng Ge, Wenwu Ou, et al. 2021. Towards Long-term Fairness in Recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 445–453.

[16] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 315–323.

[17] Ruining He, Wang-Cheng Kang, Julian J McAuley, et al. 2018. Translation-based Recommendation: A Scalable Method for Modeling Sequential Behavior.. In *IJCAI*. 5264–5268.

[18] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).

[19] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. 2019. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180* (2019).

[20] Yupeng Hou, Binbin Hu, Zhiqiang Zhang, and Wayne Xin Zhao. 2022. Core: simple and effective session-based recommendation within consistent representation space. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 1796–1801.

[21] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. 2022. Transformer quality in linear time. In *International Conference on Machine Learning*. PMLR, 9099–9117.

[22] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 505–514.

[23] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. 2021. Perceiver: General perception with iterative attention. In *International conference on machine learning*. PMLR, 4651–4664.

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

[25] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*. PMLR, 5156–5165.

[26] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[27] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451* (2020).

[28] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. *Advances in neural information processing systems* 30 (2017).

[29] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*. PMLR, 3744–3753.

[30] Muyang Li, Zijian Zhang, Xiangyu Zhao, Wanyu Wang, Minghao Zhao, Runze Wu, and Ruocheng Guo. 2023. AutoMLP: Automated MLP for Sequential Recommendations. In *Proceedings of the Web Conference 2023*.

[31] Muyang Li, Xiangyu Zhao, Chuan Lyu, Minghao Zhao, Runze Wu, and Ruocheng Guo. 2022. MLP4Rec: A Pure MLP Architecture for Sequential Recommendations. In *31st International Joint Conference on Artificial Intelligence and the 25th European Conference on Artificial Intelligence (IJCAI-ECAI 2022)*. International Joint Conferences on Artificial Intelligence, 2138–2144.

[32] Jiahao Liang, Xiangyu Zhao, Muyang Li, Zijian Zhang, Qian Li, Wanyu Wang, Haochen Liu, Ming He, Yiqi Wang, and Zitao Liu. 2023. MMMLP: Multi-modal Multilayer Perceptron for Sequence Recommendation. In *Proceedings of the Web Conference 2023*.

[33] Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198* (2018).

[34] Shuchang Liu, Qingpeng Cai, Bowen Sun, Yuhao Wang, Ji Jiang, Dong Zheng, Kun Gai, Peng Jiang, Xiangyu Zhao, and Yongfeng Zhang. 2023. Exploration and Regularization of the Latent Action Space in Recommendation. In *Proceedings of the Web Conference 2023*.

[35] Zhiwei Liu, Ziwei Fan, Yu Wang, and Philip S Yu. 2021. Augmenting sequential recommendation with pseudo-prior items via reversely pre-training transformer. In *Proceedings of the 44th international ACM SIGIR conference on Research and development in information retrieval*. 1608–1612.

[36] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*. 10012–10022.

[37] Ziru Liu, Jiejie Tian, Qingpeng Cai, Xiangyu Zhao, Jingtong Gao, Shuchang Liu, Dayou Chen, Tonghao He, Dong Zheng, Peng Jiang, et al. 2023. Multi-Task Recommendations with Reinforcement Learning. In *Proceedings of the Web Conference 2023*.

[38] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. In *International conference on machine learning*. PMLR, 4055–4064.

[39] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. 2021. Random feature attention. *arXiv preprint arXiv:2103.02143* (2021).

[40] Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. 2019. Blockwise self-attention for long document understanding. *arXiv preprint arXiv:1911.02972* (2019).

[41] Ruihong Qiu, Zi Huang, Hongzhi Yin, and Zijian Wang. 2022. Contrastive learning for representation degeneration problem in sequential recommendation. In *Proceedings of the fifteenth ACM international conference on web search and data mining*. 813–823.

[42] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507* (2019).

[43] Shaina Raza and Chen Ding. 2022. News recommender system: a review of recent progress, challenges, and opportunities. *Artificial Intelligence Review* (2022), 1–52.

[44] Zhaochun Ren, Zhi Tian, Dongdong Li, Pengjie Ren, Liu Yang, Xin Xin, Huasheng Liang, Maarten de Rijke, and Zhumin Chen. 2022. Variational Reasoning about User Preferences for Conversational Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 165–175.

[45] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics* 9 (2021), 53–68.

[46] Michael Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. 2021. Tokenlearner: Adaptive space-time tokenization for videos. *Advances in Neural Information Processing Systems* 34 (2021), 12786–12797.

[47] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. 2021. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 3531–3539.

[48] 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.

[49] Jiaxi Tang, Francois Belletti, Sagar Jain, Minmin Chen, Alex Beutel, Can Xu, and Ed H. Chi. 2019. Towards neural mixture recommender for long range dependent user sequences. In *The World Wide Web Conference*. 1782–1793.

[50] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020. Sparse sinkhorn attention. In *International Conference on Machine Learning*. PMLR, 9438–9447.

[51] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *Comput. Surveys* 55, 6 (2022), 1–28.

[52] Yi Tay, Vinh Q Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2021. Charformer: Fast character transformers via gradient-based subword tokenization. *arXiv preprint arXiv:2106.12672* (2021).

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[54] Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. 2020. Fast transformers with clustered attention. *Advances in Neural Information Processing Systems* 33 (2020), 21665–21674.

[55] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of the 38th International ACM SIGIR conference on Research and Development in Information Retrieval*. 403–412.

[56] Ruijie Wang, Zheng Li, Danqing Zhang, Qingyu Yin, Tong Zhao, Bing Yin, and Tarek Abdelzaher. 2022. RETE: retrieval-enhanced temporal event forecasting on unified query product evolutionary graph. In *Proceedings of the ACM Web Conference 2022*. 462–472.

[57] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. 2019. Sequential recommender systems: challenges, progress and prospects. *arXiv preprint arXiv:2001.04830* (2019).

[58] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768* (2020).

[59] Shuohang Wang, Luowei Zhou, Zhe Gan, Yen-Chun Chen, Yuwei Fang, Siqi Sun, Yu Cheng, and Jingjing Liu. 2020. Cluster-former: Clustering-based sparse transformer for long-range dependency encoding. *arXiv preprint arXiv:2009.06097* (2020).

[60] Genta Indra Winata, Samuel Cahyawijaya, Zhaojiang Lin, Zihan Liu, and Pascale Fung. 2020. Lightweight and efficient end-to-end speech recognition using low-rank transformer. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 6144–6148.

[61] Hui-Hua Wu and Shanhe Wu. 2009. Various proofs of the Cauchy-Schwarz inequality. *Octogon mathematical magazine* 17, 1 (2009), 221–229.

[62] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT: Sequential recommendation via personalized transformer. In *Fourteenth ACM Conference on Recommender Systems*. 328–337.

[63] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: a survey. *Comput. Surveys* 55, 5 (2022), 1–37.

[64] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive learning for sequential recommendation. In *2022 IEEE 38th international conference on data engineering (ICDE)*. IEEE, 1259–1273.

[65] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. 2021. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 14138–14148.

[66] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853* (2015).

[67] Haoran Yang, Hongxu Chen, Sixiao Zhang, Xiangguo Sun, Qian Li, Xiangyu Zhao, and Guandong Xu. 2023. Generating Counterfactual Hard Negative Samples for Graph Contrastive Learning. In *Proceedings of the Web Conference 2023*.

[68] Yuhao Yang, Chao Huang, Lianghao Xia, Yuxuan Liang, Yanwei Yu, and Chenliang Li. 2022. Multi-behavior hypergraph-enhanced transformer for sequential recommendation. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 2263–2274.

[69] Enming Yuan, Wei Guo, Zhicheng He, Huifeng Guo, Chengkai Liu, and Ruiming Tang. 2022. Multi-Behavior Sequential Transformer Recommender. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1642–1652.

[70] 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.

[71] Chi Zhang, Rui Chen, Xiangyu Zhao, Qilong Han, and Li Li. 2023. Denoising and Prompt-Tuning for Multi-Behavior Recommendation. In *Proceedings of the Web Conference 2023*.

[72] Chi Zhang, Yantong Du, Xiangyu Zhao, Qilong Han, Rui Chen, and Li Li. 2022. Hierarchical Item Inconsistency Signal Learning for Sequence Denoising in Sequential Recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2508–2518.

[73] Shuai Zhang, Yi Tay, Lina Yao, and Aixin Sun. 2018. Next item recommendation with self-attention. *arXiv preprint arXiv:1808.06414* (2018).

[74] Shengyu Zhang, Dong Yao, Zhou Zhao, Tat-Seng Chua, and Fei Wu. 2021. Causerec: Counterfactual user sequence synthesis for sequential recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 367–377.

[75] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. 2019. Feature-level Deeper Self-Attention Network for Sequential Recommendation.. In *IJCAI*. 4320–4326.

[76] Weinan Zhang, Xiangyu Zhao, Li Zhao, Dawei Yin, Grace Hui Yang, and Alex Beutel. 2020. Deep Reinforcement Learning for Information Retrieval: Fundamentals and Advances. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2468–2471.

[77] Xiaoyu Zhang, Xin Xin, Dongdong Li, Wenxuan Liu, Pengjie Ren, Zhumin Chen, Jun Ma, and Zhaochun Ren. 2023. Variational Reasoning over Incomplete Knowledge Graphs for Conversational Recommendation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 231–239.

[78] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. 2020. How to retrain recommender system? A sequential meta-learning method. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1479–1488.

[79] Kesen Zhao, Xiangyu Zhao, Zijian Zhang, and Muyang Li. 2022. MAE4Rec: Storage-saving Transformer for Sequential Recommendations. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2681–2690.

[80] Kesen Zhao, Lixin Zou, Xiangyu Zhao, Maolin Wang, et al. 2023. User Retention-oriented Recommendation with Decision Transformer. In *Proceedings of the Web Conference 2023*.

[81] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, et al. 2021. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4653–4664.

[82] Xiangyu Zhao. 2022. Adaptive and automated deep recommender systems. *ACM SIGWEB Newsletter* Spring (2022), 1–4.

[83] Xiangyu Zhao, Changsheng Gu, Haoshenglun Zhang, Xiwang Yang, Xiaobing Liu, Hui Liu, and Jiliang Tang. 2021. DEAR: Deep Reinforcement Learning for Online Advertising Impression in Recommender Systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 750–758.

[84] Xiangyu Zhao, Long Xia, Jiliang Tang, and Dawei Yin. 2019. Deep reinforcement learning for search, recommendation, and online advertising: a survey. *ACM SIGWEB Newsletter* Spring (2019), 1–15.

[85] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep Reinforcement Learning for Page-wise Recommendations. In *Proceedings of the 12th ACM Recommender Systems Conference*. ACM, 95–103.

[86] Xiangyu Zhao, Long Xia, Lixin Zou, Hui Liu, Dawei Yin, and Jiliang Tang. 2020. Whole-Chain Recommendations. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1883–1891.

[87] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1040–1048.

[88] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Dawei Yin, Yihong Zhao, and Jiliang Tang. 2017. Deep Reinforcement Learning for List-wise Recommendations. *arXiv preprint arXiv:1801.00209* (2017).

[89] Xiangyu Zhao, Xudong Zheng, Xiwang Yang, Xiaobing Liu, and Jiliang Tang. 2020. Jointly learning to recommend and advertise. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3319–3327.

[90] Zhi Zheng, Zhaopeng Qiu, Hui Xiong, Xian Wu, Tong Xu, Enhong Chen, and Xiangyu Zhao. 2022. DDR: Dialogue Based Doctor Recommendation for Online Medical Service. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4592–4600.

[91] Chen Zhu, Wei Ping, Chaowei Xiao, Mohammad Shoeybi, Tom Goldstein, Anima Anandkumar, and Bryan Catanzaro. 2021. Long-short transformer: Efficient transformers for language and vision. *Advances in Neural Information Processing Systems* 34 (2021), 17723–17736.

[92] Lixin Zou, Long Xia, Yulong Gu, Xiangyu Zhao, Weidong Liu, Jimmy Xiangji Huang, and Dawei Yin. 2020. Neural Interactive Collaborative Filtering. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 749–758.