mathematics

MDPI

Article

Blin: A Multi-Task Sequence Recommendation Based on Bidirectional KL-Divergence and Linear Attention

Yanfeng Bai, Haitao Wang * and Jianfeng He

Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China; kustbyf@stu.kust.edu.cn (Y.B.); jfenghe@kust.edu.cn (J.H.)
* Correspondence: 12309177@kust.edu.cn

Abstract: Sequence recommendation is a prominent research area within recommender systems, focused on predicting items that users may be interested in by modeling their historical interaction sequences. However, due to data sparsity, user interaction sequences in sequence recommendation are typically short. A common approach to address this issue is filling sequences with zero values, significantly reducing the effective utilization of input space. Furthermore, traditional sequence recommendation methods based on self-attention mechanisms exhibit quadratic complexity with respect to sequence length. These issues affect the performance of recommendation algorithms. To tackle these challenges, we propose a multi-task sequence recommendation model, Blin, which integrates bidirectional KL divergence and linear attention. Blin abandons the conventional zero-padding strategy, opting instead for random repeat padding to enhance sequence data. Additionally, bidirectional KL divergence loss is introduced as an auxiliary task to regularize the probability distributions obtained from different sequence representations. To improve the computational efficiency compared to traditional attention mechanisms, a linear attention mechanism is employed during sequence encoding, significantly reducing the computational complexity while preserving the learning capacity of traditional attention. Experimental results on multiple public datasets demonstrate the effectiveness of the proposed model.

Keywords: sequence recommendation; self-attention mechanism; consistency training; data augmentation

MSC: 68T07; 68T01

Citation: Bai, Y.; Wang, H.; He, J. Blin: A Multi-Task Sequence Recommendation Based on Bidirectional KL-Divergence and Linear Attention. *Mathematics* **2024**, *12*, 2391. https://doi.org/10.3390/math12152391

Academic Editors: Nazim Choudhury and Matloob Khushi

Received: 13 July 2024
Revised: 26 July 2024
Accepted: 29 July 2024
Published: 31 July 2024

1. Introduction

Finding content that interests users quickly and accurately amid vast amounts of information has always been a challenging problem. To address this issue, recommender systems have become indispensable tools for online users to filter their preferred information based on their historical behavior interactions. Sequential recommendation [1], which dynamically models user preferences over time, has proven effective and has garnered widespread research attention.

In recent years, with the rapid advancement of deep learning, a range of sequential recommendation methods have emerged, encompassing architectures such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), graph neural networks (GNNs), and Transformers [2–8]. Among these, Transformer-based methods for sequential recommendation have gained significant attention due to their exceptional performance in sequence encoding [5–8]. However, current research still faces several challenges: (1) Sequence padding space utilization issues and data sparsity problems: most sequence recommendation methods handle the padding of original short sequences with zeros [9], which do not contribute meaningful information and, thus, result in inefficient space utilization. Additionally, in the field of sequence recommendation, data sparsity is a prevalent issue.

User interaction sequences are typically short and sparse, making it challenging for models to accurately capture user preferences and behaviors due to the limited data available. This sparsity further exacerbates the inefficiency of traditional zero-padding methods, as the padding does not add valuable information and fails to address the underlying lack of data. (2) The computational complexity of attention mechanisms: when dealing with long sequences, the computational complexity of dot product operations in attention layers increases quadratically with sequence length. (3) The limitation of a single loss function: most models rely solely on a single loss function to update their model parameters, which poses certain limitations to learning user preference representations from interaction sequences.

In response to these issues, this paper proposes a multi-task sequential recommendation model named Blin, which integrates bidirectional KL divergence and linear attention. Blin departs from the traditional zero-padding of original short sequences by employing a method called random repeat padding (RandomPad for short) to enhance the original sequence data. This involves independently applying RandomPad twice to generate two padded sequences, which are jointly input into the model. Additionally, a linear attention mechanism is utilized for the padded sequences in the model to significantly reduce the computational complexity of dot product operations in the attention layers, while preserving the learning capability of traditional attention mechanisms. Finally, bidirectional KL divergence loss is added as an auxiliary task for regularization [10,11], aiming to align or make similar the probability distributions derived from the representations of different padded sequences. This assists the model in learning the desired target distributions.

The main contributions of this paper are as follows:

- Blin adopts the RandomPad method to replace traditional zero-padding, thereby alleviating data sparsity issues and improving the effective utilization of input space.
- Blin employs a linear attention mechanism that reduces the computational complexity of the attention dot product operations from quadratic O(n^2) to near-linear O(n), while ensuring the accuracy of attention mechanisms for long sequences.
- Blin introduces bidirectional KL divergence loss as an auxiliary task for sequential recommendation, aiming to regularize the probability distributions derived from different representations of padded sequences. This loss, combined with the sequence recommendation loss, jointly updates the model parameters to achieve enhanced user preference representations.
- Experimental evaluations on multiple public datasets, compared with representative baseline methods, demonstrate that Blin achieves performance improvements over baseline methods to varying degrees.

2. Related Work

2.1. Sequence-Based Recommendation

Sequence recommendation tasks involve predicting users' next behavioral preferences based on their historical interaction behaviors. Early research, including pioneering methods based on Markov chains [12], initially explored simple, low-order sequential dependencies. However, these approaches were constrained by assumptions, making them inadequate for handling complex situations involving high-order dependencies. With the rapid development of deep learning, a variety of sequence recommendation methods have emerged. Among these, methods based on self-attention mechanisms stand out for their excellent performance and have received extensive research attention. Kang et al. [5] were pioneers in integrating self-attention mechanisms into sequence recommendation, effectively capturing user preference representations and achieving an improved performance over previous methods. FDSA [13] introduces multiple attention blocks to depict potential features. Sun et al. [6] proposed BERT4Rec, which utilizes bidirectional attention mechanisms to model user interaction sequences and employs the Cloze objective to improve the efficiency of the training process. Li et al. [14] introduced TiSASRec, incorporating time interval information into attention mechanisms to capture the impacts of different time intervals on item prediction. Furthermore, an increasing number of studies are combin-

ing contrastive learning with attention mechanisms [15–18]. For instance, CL4Srec [15] utilizes a contrastive learning framework to obtain self-supervised signals from original user sequences and proposes three data augmentation methods to construct these signals. CoSeRec [16] builds upon CL4SRec by integrating item similarity information into contrastive learning objectives to maximize sequence consistency enhancement.

2.2. Linear Attention Mechanisms

Traditional attention mechanisms incur high computational costs, posing challenges in model design and practical applications. To address the high computational costs associated with Transformers, recent studies have adopted linear attention mechanisms as a solution. Specifically, linear attention replaces the Softmax function in traditional dot product attention with a designed kernel function. In this approach, attention can be computed more efficiently by first calculating $\mathbf{K}^{\mathrm{T}}\mathbf{V}$. While linear attention is highly efficient in terms of computational complexity, designing a linear attention module with the same learning capability as Softmax attention remains a challenge. This issue has led to a series of emerging studies. The Performer [19] approximates Softmax operations using orthogonal random features, while Li et al. [20] proposed using a first-order Taylor expansion to approximate the expanded form of the Softmax function. K et al. [21] utilized simple feature mapping represented by the elu function as a kernel function. Shen et al. [22] introduced Efficient Attention, applying Softmax separately to $\mathbf{Q}$ and $\mathbf{K}$ to ensure that each row of $\mathbf{Q}\mathbf{K}^{\mathrm{T}}$ summed to 1. Hydra Attention [23] selects cosine similarity as a kernel function, reducing the computational complexity to O(nd).

However, the methods mentioned above still face several challenges. Sequential recommendation models typically process input sequences into a fixed length for ease of model training. Most sequence recommendation models handle user sequences by padding with zeros [9], where these padded zeros do not contribute to the model training, leading to inefficient utilization of the input space and difficulty in learning long-term user behavior representations. Furthermore, models based on self-attention mechanisms suffer from high computational costs [20–25]. Although current research on linear attention mechanisms has significantly reduced computational complexity, these methods often sacrifice accuracy. In other words, existing research has not effectively preserved the learning capability of traditional attention mechanisms.

The Blin model proposed in this paper addresses these challenges by employing RandomPad to effectively increase the utilization of input space for model input sequences. Additionally, improvements are made to the linear attention mechanism by incorporating a DWC module [24,25], which enhances the ability to capture local features while maintaining effective attention. Moreover, Blin introduces bidirectional KL divergence loss as an auxiliary task to regularize the probability distributions derived from different representations of padded sequences, aiding the model in learning more aligned target distributions.

3. Methodology

In this section, we first introduce the problem definition of sequential recommendation. Then, we provide a detailed description of each component of the Blin model, including RandomPad, the linear attention mechanism, and the bidirectional KL divergence. The overall framework of the model is illustrated in Figure 1.

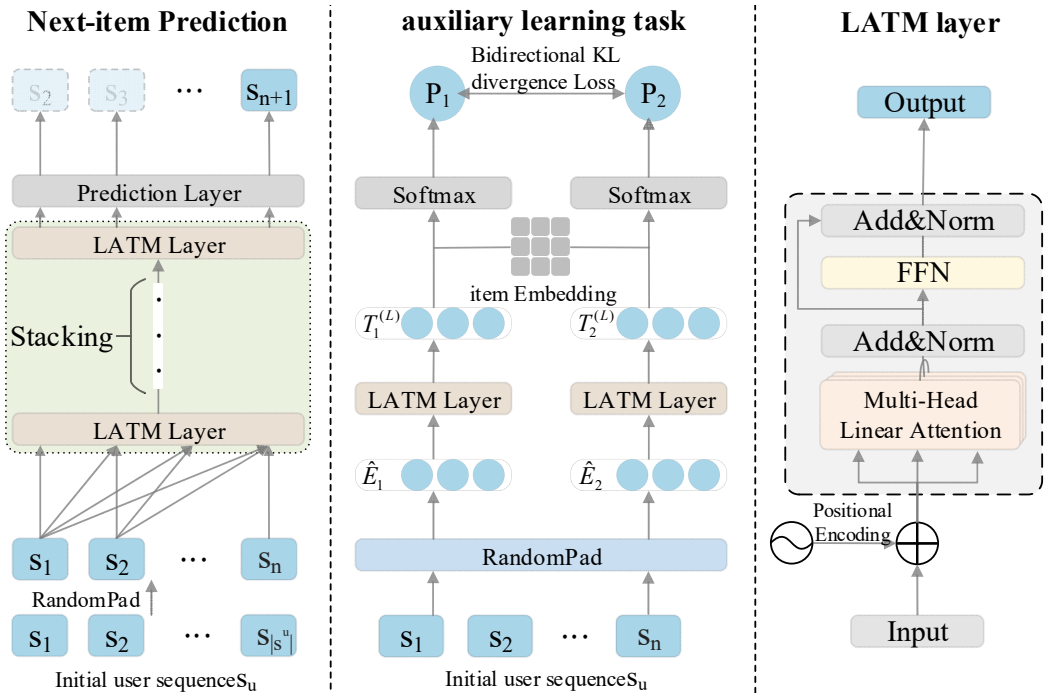

Figure 1. Architecture of the Blin.

3.1. Problem Statement

In the task of sequence recommendation, let **U** denote the set of users and **V** denote the set of items. For each user u ∈ **U**, there exists a time-ordered user interaction sequence $\mathbf{S}_u = [s_1^u, s_2^u, \ldots, s_t^u, \ldots, s_{|S^u|}^u]$, where $|S^u|$ denotes the number of interactions in sequence $\mathbf{S}_u$ and s_t^u represents the item interacted by user u at time t. The task of sequence recommendation is to predict a Top-N candidate list of items that a user may find interesting at the next time step, based on their historical interaction sequence $\mathbf{S}_u$. given by:

$$\begin{aligned} &\arg max \quad P\left(v_{|\mathbf{S}^u|+1} = v^* | \mathbf{S}_u\right) \\ &v^* \in \mathbf{V} \end{aligned} \tag{1}$$

This equation can be understood as calculating the probability for all candidate items and selecting the highest one for recommendation.

3.2. RandomPad

Most common sequence models can only handle fixed-length sequences. Additionally, due to the issue of data sparsity, the user sequences in sequential recommendations are often short. Therefore, sequence padding is a frequently used technique in training sequence models [5]. Before model training, the maximum sequence length n that the model can handle is defined. For sequences with an initial length greater than n, the most recent n items are selected as the input sequence. If the initial sequence length is less than n, padding is required on the left side of the sequence.

The general operation of sequence padding can be described as follows:

$$SeqOper(\mathbf{S}_u) = SeqOper\left(\left[s_1, s_2, s_3, \ldots, s_{|S^u|}\right]\right) = \begin{cases} T(\mathbf{S}_u), & |S^u| < n, \\ \mathbf{S}_u, & |S^u| = n, \\ [s_{c+1}, s_{c+2}, s_{c+3}, \ldots, s_n], & |S^u| > n, \end{cases} \tag{2}$$

where $|\mathbf{S}^u|$ represents the initial sequence length, SeqOper($\mathbf{S}_u$) indicates the preprocessing operation on sequence $\mathbf{S}_u$ before training, and T($\mathbf{S}_u$) represents the padding of the initial sequence $\mathbf{S}_u$. The usual method for sequence padding is zero-padding, where $c = |\mathbf{S}^u| - n$ represents the number of items that need to be truncated from the initial sequence.

Traditional sequence padding typically uses zero-padding to extend the initial sequence length to the maximum sequence length n:

$$Z(\mathbf{S}_u) = Z\left(\left[s_1, s_2, s_3, \ldots, s_{|\mathbf{S}^u|}\right]\right) = [0, 0, \ldots, s_{k+1}, s_{k+2}, s_{k+3}, \ldots, s_n] \tag{3}$$

where $Z(\mathbf{S}_u)$ indicates zero-padding for sequence $\mathbf{S}_u$, and $k = n - |\mathbf{S}^u|$ represents the number of zeros that need to be added, with $|\mathbf{S}^u| < n$.

Although traditional zero-padding is simple to implement, it can lead to a waste of input space. Blin uses RandomPad instead of traditional zero-padding. A comparison between traditional zero-padding and RandomPad is shown in Figure 2. The specific definition of the RandomPad padding method is as follows:

$$Ra\mathit{nd}omPa\mathit{d}(\alpha, \mathbf{S}_u) = \underbrace{[\mathbf{S}_u|0|\mathbf{S}_u|0|\mathbf{S}_u|\ldots|0|\mathbf{S}_u|]}_{(\alpha+1)\mathbf{S}_u} \tag{4}$$

where $\text{RandomPad}(\alpha, \mathbf{S}_u)$ represents the random repeat padding for $\mathbf{S}_u$, $|\mathbf{S}^u| < n$ and $\alpha \in [1, \max]$ represents the randomly generated number of padding repetitions. $\max = \lfloor n / |\mathbf{S}^u| \rfloor$ represents the maximum number of repetitions that can be applied to the initial sequence. When $\max = 0$, it means that the initial sequence is a long sequence, and there is not enough space to accommodate the maximum sequence length n. In this case, the number of padding times $\alpha = 0$. Any remaining space after RandomPad is filled with zeros. Adding zeros between repeated sequences prevents the tail of the initial user sequence from predicting the head.

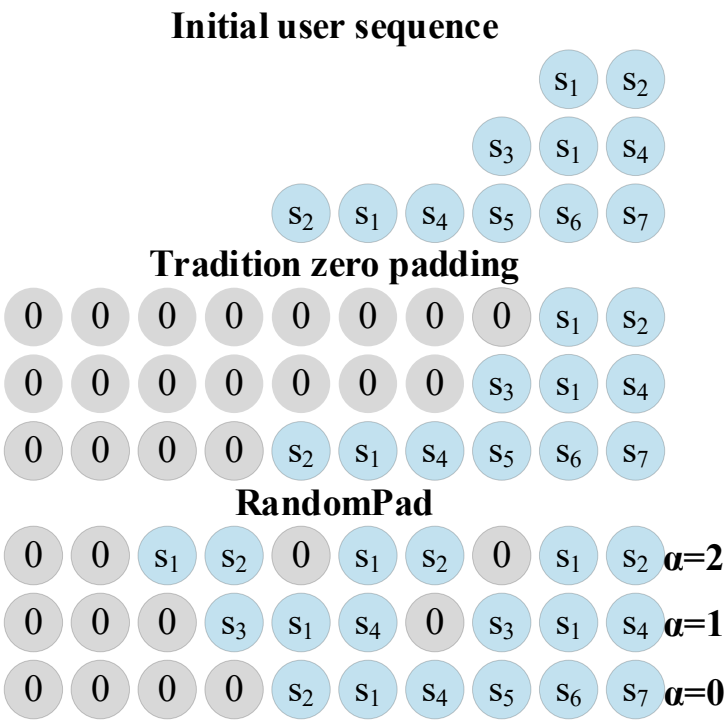

Figure 2. Comparison of sequence padding (n = 10).

3.3. Embedding Layer

First, we constructed an embedding table $\mathbf{M} \in R^{|V| * d}$ using the set of projects, where $|V|$ denotes the number of projects in the set and d represents the embedding dimension. We converted the user training sequence $\mathbf{S}_u = [s_1^u, s_2^u, \ldots, s_t^u, \ldots, s_{|\mathbf{S}^u|}^u]$ into a fixed-length sequence $\mathbf{S}_u = [s_1, s_2, \ldots, s_n]$, where n is the maximum length the model can handle. If the sequence length exceeded n, we considered only the most recent n user interactions. If the

sequence length was less than n, we used RandomPad to pad the left side of the sequence until its length reached n. Next, we retrieved the project embedding matrix $\mathbf{E} \in \mathrm{R}^{n*d}$ corresponding to the user interaction sequence $\mathbf{S}_\mathrm{u}$ by querying the project embedding table $\mathbf{M}$, where $\mathrm{E_i} = \mathrm{M_{si}}$. For positional embedding, we introduced a learnable positional embedding $\mathbf{P} \in \mathrm{R}^{n*d}$. By combining project embeddings with positional embeddings, we obtained the final sequence embedding:

$$\hat{\mathbf{E}} = \begin{bmatrix} \mathbf{M_{s_1}} + \mathbf{P_1} \\ \mathbf{M_{s_2}} + \mathbf{P_2} \\ \cdots \\ \mathbf{M_{s_n}} + \mathbf{P_n} \end{bmatrix} = \begin{bmatrix} \mathbf{s_1} \\ \mathbf{s_2} \\ \cdots \\ \mathbf{s_n} \end{bmatrix} \tag{5}$$

3.4. Transformer Layer

Utilizing the sequence embeddings obtained from the previous section, we input them into the Transformer module. We stacked multiple Transformer layers to capture hierarchical dependencies within the input sequence, aiming to better learn the representations of user preferences. However, traditional dot product attention methods involve high computational costs in the attention matrix calculation process. To mitigate these costs while ensuring model effectiveness, Blin employs a linear attention mechanism by designing a unique kernel function to replace the Softmax function in the self-attention mechanism. This changes the computation order in the attention dot product operation, from first calculating $\mathbf{QK}^\mathrm{T}$ to first calculating $\mathbf{K}^\mathrm{T}\mathbf{V}$, thereby significantly reducing the computational cost. The computational complexity of traditional dot product attention is reduced from $\mathrm{O}(n^2d)$ to $\mathrm{O}(nd^2)$. In scenarios where $n \gg d$, this complexity can be approximated as linear $\mathrm{O}(n)$.

3.4.1. Definition of Traditional Dot Product Attention Method

The key component of the Transformer is self-attention, which effectively captures dependencies between different positions in a sequence. Each element in the sequence is weighted based on the importance of other elements. Typically, this is computed using the dot product attention function [26]:

$$\mathbf{Q} = \mathbf{EW}_\mathrm{Q}, \mathbf{K} = \mathbf{EW}_\mathrm{K}, \mathbf{V} = \mathbf{EW}_\mathrm{V}$$

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{QK}^\mathrm{T}}{\sqrt{d}}\right)\mathbf{V} \tag{6}$$

where $\mathrm{E} \in \mathrm{R}^{n\times d}$ represents the input sequence matrix, $\mathbf{Q} = \mathbf{EW}^\mathrm{Q}, \mathbf{K} = \mathbf{EW}^\mathrm{K}, \mathbf{V} = \mathbf{EW}^\mathrm{V}$ respectively, denote queries, keys, and values. $\left\{\mathbf{W}^\mathrm{Q}, \mathbf{W}^\mathrm{K}, \mathbf{W}^\mathrm{V}\right\} \in \mathbf{R}^{d\times d}$ are three projection matrices, with $\sqrt{d}$ as a scaling factor. The softmax function is applied row-wise to $\mathbf{QK}^\mathrm{T}$.

3.4.2. Generalization of Kernel-Based Dot Product Attention

Assuming matrix $\mathbf{D}$ represents the output of Equation (6), where the ith row of $\mathbf{D}$ is denoted as $\mathbf{D}_i$, Equation (6) can be generalized for attention computation under any similarity function:

$$\mathbf{D}_i = \left(\frac{\mathrm{Sim}(\mathbf{Q}_i, \mathbf{K}_m)\mathbf{V}_i}{\sum_{j=1}^{n}\mathrm{Sim}(\mathbf{Q}_i, \mathbf{K}_j)}\right)_{m=1}^{n} \tag{7}$$

where $\mathbf{Q}_i = (\mathrm{Q}_{i1}, \mathrm{Q}_{i2}, \ldots, \mathrm{Q}_{id})$ represents the ith row of matrix $\mathbf{Q}$, $\mathrm{K}_j = \left(\mathrm{K}_{j1}, \mathrm{K}_{j2}, \ldots, \mathrm{K}_{jd}\right)^\mathrm{T}$ represents the jth column of matrix $\mathbf{K}^\mathrm{T}$, and $\mathrm{V}_i = (\mathrm{V}_{i1}, \mathrm{V}_{i2}, \ldots, \mathrm{V}_{id})$ represents the ith row of matrix $\mathbf{V}$, where $i \in (1, 2 \ldots, n)$. $\mathrm{f} = (\mathrm{f}(x))_{x=1}^{n}$ denotes the content of vector f as $\mathrm{f} = (\mathrm{f}(1), \mathrm{f}(2), \ldots, \mathrm{f}(n))$. Equation (6) is equivalent to Equation (7) using Softmax attention, where the similarity measure is $\mathrm{Sim}(\mathbf{Q}, \mathbf{K}) = \exp(\mathbf{QK}^\mathrm{T}/\sqrt{d})$. $\mathrm{Sim}(\mathbf{Q}, \mathbf{K})$ calculates the similarity between all queries and keys, and since $\mathbf{Q}, \mathbf{K} \in \mathrm{R}^{n\times d}$, the product of $\mathbf{Q}$ and $\mathbf{K}$ leads to $\mathrm{O}(n^2)$ computational complexity. Recent studies have aimed to reduce

computational costs by introducing carefully designed kernel functions as approximations of the original similarity function. Specifically, $\mathrm{Sim}(\mathbf{Q}_i, \mathbf{K}_j)$ can be further generalized as $\mathrm{Sim}(\mathbf{Q}_i, \mathbf{K}_j) = \phi(\mathbf{Q}_i)\,\phi(\mathbf{K}_j)^{\mathrm{T}}$, transforming Equation (7) into:

$$\mathbf{D}_i = \left(\frac{\phi(\mathbf{Q}_i)\phi(\mathbf{K}_m)^{\mathrm{T}}\mathbf{V}_i}{\sum_{j=1}^{n} \phi(\mathbf{Q}_i)\phi(\mathbf{K}_j)^{\mathrm{T}}} \right)_{m=1}^{n} \tag{8}$$

Equation (8) continues to simplify:

$$\mathbf{D}_i = \left(\frac{\phi(\mathbf{Q}_i)\phi(\mathbf{K}_m)^{\mathrm{T}}\mathbf{V}_i}{\phi(\mathbf{Q}_i)\sum_{j=1}^{n} \phi(\mathbf{K}_j)^{\mathrm{T}}} \right)_{m=1}^{n} \tag{9}$$

It can be observed that, by extending the original similarity function $\mathrm{Sim}(\mathbf{Q}_i, \mathbf{K}_j)$ to $\phi(\mathrm{Q}_i)\,\phi(\mathrm{K}_j)^{\mathrm{T}}$, the computation of attention can prioritize the calculation of $\mathrm{K}^{\mathrm{T}}\mathrm{V}$. Since $\mathbf{K}^{\mathrm{T}} \in \mathrm{R}^{d\times n}$ and $\mathbf{V} \in \mathrm{R}^{n\times d}$, the complexity of the product between $\mathbf{K}^{\mathrm{T}}$ and $\mathbf{V}$ is $\mathrm{O}(d^2)$. In scenarios where $n \gg d$, the computational complexity of kernel-based dot product calculation can be approximated as $\mathrm{O}(n)$, significantly reducing the computational cost compared to traditional dot product attention.

3.4.3. Linear Attention Mechanism

From the above reasoning, it is evident that achieving linear computational complexity in attention mechanisms hinges on determining an appropriate kernel function, specifically using $\phi(\mathbf{Q})\,\phi(\mathbf{K})^{\mathrm{T}}$ to replace the traditional dot product attention operation $\mathrm{Sim}(\mathbf{Q}, \mathbf{K})$. Blin employs an effective method of attention to replace traditional dot product attention, specifically using the Softmax function as the kernel function ($\phi(\mathbf{Q})\phi(\mathbf{K})^{\mathrm{T}} = \sigma_{\mathrm{row}}(\mathbf{Q})\sigma_{\mathrm{col}}(\mathbf{K})^{\mathrm{T}}$). This transforms the single Softmax operation in traditional dot product attention into two separate Softmax operations. Verified to be a straightforward and effective replacement for traditional dot product attention, the following equation describes the effective attention mechanism:

$$\mathrm{E}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \sigma_{\mathrm{row}}(\mathbf{Q})\left(\sigma_{\mathrm{col}}(\mathbf{K})^{\mathrm{T}}\mathbf{V}\right) \tag{10}$$

where $\sigma_{\mathrm{row}}(\cdot)$ and $\sigma_{\mathrm{row}}(\cdot)$ apply the Softmax function along each row or column of the matrix, respectively. Effective attention essentially selects the Softmax function as the kernel function, transforming the single Softmax operation in traditional dot product attention into two separate Softmax operations. This transformation changes the execution order of dot product operations from $\mathbf{Q}\mathbf{K}^{\mathrm{T}}\mathbf{V}$ to $\mathbf{Q}(\mathbf{K}^{\mathrm{T}}\mathbf{V})$, thereby reducing computational costs. Although this transformation is not entirely equivalent, in practice, their effects are very similar. The main propertes of $\sigma_{\mathrm{row}}(\mathbf{Q}\mathbf{K}^{\mathrm{T}})$ are that the sum of each row is 1 and the elements in each row are non-negative. The matrix $\sigma_{\mathrm{row}}(\mathbf{Q})\sigma_{\mathrm{col}}(\mathbf{K})^{\mathrm{T}}$ also possesses these properties, which has been confirmed in previous studies.

The Softmax function tends to globally attend to the entire sequence, but its global nature may lead to neglecting local information when capturing long-range dependencies. Additionally, the exponential nature of Softmax may cause attention weights to overly focus on a few items in the sequence, exaggerating their importance and neglecting others, resulting in information loss. To address the limitations of the Softmax function, this paper proposes a simple and effective improvement: integrating a Depthwise Convolution (DWC) module into the attention matrix. The DWC module helps the model to capture local structures and relationships more comprehensively within the sequence, enabling the attention mechanism to compute attention weights based not only on exponential functions, but also on richer local features. Figure 3 contrasts traditional dot product attention with the linear attention mechanism. The final output representation of the linear attention mechanism is:

$$\mathbf{D} = \sigma_{\mathrm{row}}(\mathbf{Q})\left(\sigma_{\mathrm{col}}(\mathbf{K})^{\mathrm{T}}\mathbf{V}\right) + \mathrm{DWC}(\mathbf{V}) \tag{11}$$

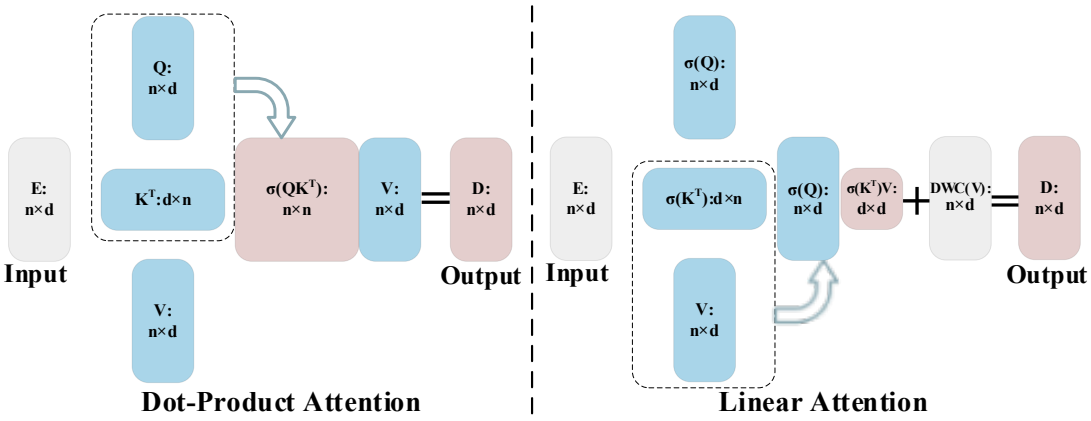

Figure 3. Comparison between dot product attention and linear attention.

Here, $\mathbf{Q} = \mathbf{E}\mathbf{W}^{\mathrm{Q}}, \mathbf{K} = \mathbf{E}\mathbf{W}^{\mathrm{K}}, \mathbf{V} = \mathbf{E}\mathbf{W}^{\mathrm{V}}$ represent queries, keys, and values, respectively, where $\left\{\mathbf{W}^{\mathrm{Q}}, \mathbf{W}^{\mathrm{K}}, \mathbf{W}^{\mathrm{V}}\right\} \in \mathrm{R}^{d \times d}$ are three projection matrices. $\mathbf{D}$ denotes the output representation of the attention layer. Additionally, multiple attention functions can be used in parallel to enhance model expressiveness $\mathbf{D} \leftarrow \mathrm{MultiHead}(\mathbf{E})$.

Feedforward Neural Network: Since the linear attention layer primarily operates on linear projections, a feedforward neural network is introduced to impart non-linear characteristics to the model:

$$\mathbf{T}_i = \mathrm{FFN}(\mathbf{D}_i) = \mathrm{ReLU}\left(\mathbf{D}_i\mathbf{W}^{(1)} + \mathbf{b}^{(1)}\right)\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \tag{12}$$

Here, $\mathbf{W}^{(1)}, \mathbf{W}^{(2)} \in \mathrm{R}^{d \times d}$ are weight matrices and $\mathbf{b}^{(1)}, \mathbf{b}^{(2)} \in \mathrm{R}^{d}$ are learnable bias vectors. $\mathbf{D}_i$ and $\mathbf{T}_i$ represent the behavioral representations of the ith item in $\mathbf{D}$ and $\mathbf{T}$, respectively.

Finally, to accurately capture deep representations of user behavior, a strategy of stacking Transformer layers is employed, where the definition of the Lth layer is as follows:

$$\mathbf{D}^{(\mathrm{L})} = \mathrm{LA}\left(\mathbf{T}^{(\mathrm{L}-1)}\right) \tag{13}$$

$$\mathbf{T}_i^{(\mathrm{L})} = \mathrm{FFN}\left(\mathbf{D}_i^{(\mathrm{L})}\right) \tag{14}$$

Here, $\mathbf{D}^{(1)} = \mathbf{D}$, $\mathbf{T}^{(1)} = \mathbf{T}$, LA represents the linear attention layer. In addition, residual connections [27], dropout [28], and layer normalization techniques [29] can be applied to ensure stability and expedite convergence during training. Here, this paper succinctly summarizes the final output of the Transformer layer as $\mathbf{T}^{(\mathrm{L})}$.

3.5. Bidirectional KL Divergence Loss

Using the item-embedding method from Section 3.3, two sequence embeddings $\hat{\mathbf{E}}_1$ and $\hat{\mathbf{E}}_2$, which have been padded using RandomPad, are formed. These embeddings, $\hat{\mathbf{E}}_1$ and $\hat{\mathbf{E}}_2$, are then input into the Transformer architecture based on the linear attention mechanism, as described in Section 3.4. This process ultimately yields two output representations ${\mathbf{T}_1}^{(\mathrm{L})}$ and ${\mathbf{T}_2}^{(\mathrm{L})}$. To understand the relationship between users and items in sequence recommendation, a similarity function (such as the dot product) can be used to measure the distance between user output representations and item representations. Specifically, for the user output representations ${\mathbf{T}_1}^{(\mathrm{L})}$ and ${\mathbf{T}_2}^{(\mathrm{L})}$, we compute corresponding similarity

probability distributions $P(\mathbf{T}_1{}^{(L)})$ and $P(\mathbf{T}_2{}^{(L)})$ to predict the next item of interest for user u. The specific formula is as follows:

$$P\left(\mathbf{T}^{(L)};\alpha\right) = \frac{\exp\left(\mathbf{T}^{(L)}e^{+}_{t+1}\right)}{\exp\left(\mathbf{T}^{(L)}e^{+}_{t+1}\right) + \sum_{e^{-}_{t+1}\in \mathbf{V}} \exp\left(\mathbf{T}^{(L)}e^{-}_{t+1}\right)} \tag{15}$$

$$\mathcal{L}_{seq} = -\log P\left(\mathbf{T}^{(L)};\alpha\right) \tag{16}$$

Here, e^{+}_{t+1} represents a positive sample, e^{-}_{t+1} represents a randomly sampled negative sample from the item set $\mathbf{V}$, and α represents all trainable parameters in the model. $\mathcal{L}_{seq}$ represents the sequence loss function.

After applying Equation (15) to the user output representations $\mathbf{T}_1{}^{(L)}$ and $\mathbf{T}_2{}^{(L)}$, we obtain two different similarity probability distributions $P(\mathbf{T}_1)$ and $P(\mathbf{T}_2)$. Blin introduces bidirectional KL divergence loss to regularize these two different similarity distributions. Figure 4 illustrates a specific example, showing the process of an auxiliary learning task executed by Blin for a simple user interaction sequence $\mathbf{S}_u = [s_1, s_2]$. The numbers ①–⑥ indicate the sequence of steps. Before introducing the bidirectional KL divergence loss, let us first define the KL divergence. For two probability distributions $P(x)$ and $Q(x)$, the KL divergence is defined as:

$$D_{KL}(P||Q) = \sum_x P(x)\log\left(\frac{P(x)}{Q(x)}\right) \tag{17}$$

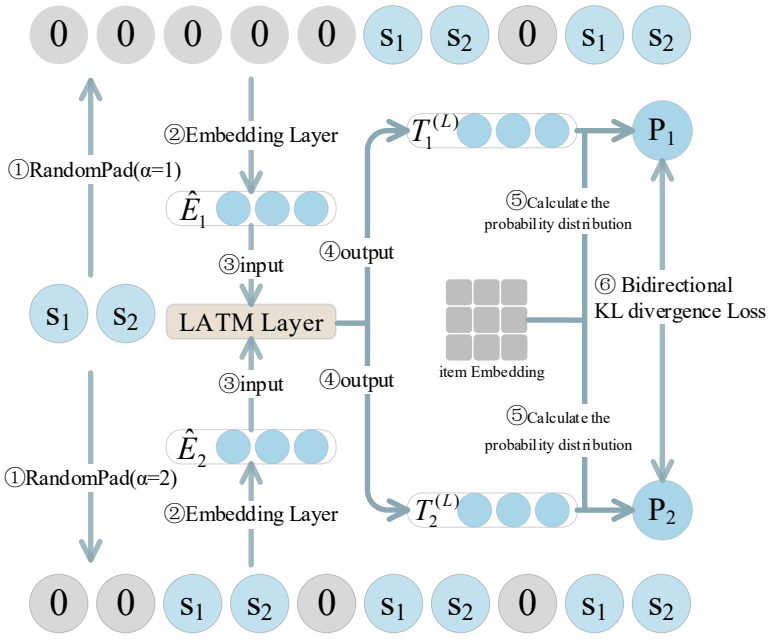

Figure 4. Auxiliary learning task, where LATM Layer denotes the Transformer layer that replaces traditional dot product attention with a linear attention mechanism.

KL divergence measures the expected amount of additional information required to represent $P(x)$ using the probability distribution $Q(x)$ given the probability distribution $P(x)$, and is commonly used to measure the difference between two probability distributions. While bidirectional KL divergence can take into account the two-way difference between two distributions at the same time and is more comprehensive in terms of optimization objectives, bidirectional KL divergence loss is defined as follows:

$$\mathcal{L}_{bkl} = \frac{1}{2}\left(D_{KL}\left(P\left(\mathbf{T_1}^{(L)};\alpha\right)\middle|\middle|D_{KL}\left(P\left(\mathbf{T_2}^{(L)};\alpha\right)\right) + D_{KL}\left(P\left(\mathbf{T_2}^{(L)};\alpha\right)\middle|\middle|D_{KL}\left(P\left(\mathbf{T_1}^{(L)};\alpha\right)\right)\right) \tag{18}$$

Bidirectional KL divergence loss helps in measuring the difference between two probability distributions. By minimizing this metric, the difference between the two distributions can be reduced. In other words, using the bidirectional KL divergence loss combines the KL divergences between the two distributions and their mutual information, aiding the model in learning a target distribution that better aligns with expectations.

3.6. Model Training

As shown in Figure 1, the Blin model adopts a multitask training strategy, jointly optimizing the sequence recommendation prediction task and auxiliary learning tasks. The sequence loss and the final training objective are defined as follows:

$$\mathcal{L}_{\text{seq}} = -\frac{1}{2}\left(\log \mathrm{P}\left(\mathbf{T_1}^{(\mathrm{L})}; \alpha\right) + \log \mathrm{P}\left(\mathbf{T_2}^{(\mathrm{L})}; \alpha\right)\right) \tag{19}$$

$$\mathcal{L} = \mathcal{L}_{\text{seq}} + \lambda \mathcal{L}_{\text{bkl}} \tag{20}$$

where $\mathcal{L}_{\text{seq}}$ is the sequence loss, $\mathcal{L}$ is the total loss function of the model, and λ is the hyperparameter controlling the intensity of the bidirectional KL divergence loss.

4. Experiments

4.1. Datasets

To validate the effectiveness of the model, experiments were conducted on four widely used public datasets, including three short-sequence datasets (Beauty, Sports, Yelp) and one long-sequence dataset (ML-1M). A brief introduction of these datasets is provided below:

- **Amazon:** The Amazon dataset is a large-scale dataset that records user reviews of products on the Amazon website, making it a classic dataset for recommendation systems. We selected Beauty and Sports as two different datasets from the Amazon dataset.
- **Yelp:** The Yelp dataset is a well-known open-source dataset obtained through a business platform.
- **MovieLens-1M (ML-1M):** ML-1M is a dense movie recommendation dataset widely used for evaluating recommendation algorithms.

For all the datasets, to ensure data quality, preprocessing was performed by removing users and items with fewer than five interactions. The interaction records of each user were aggregated and sorted based on timestamps to obtain each user's interaction sequence. For each user, their two most recent interactions were used as the test set and validation set, respectively, while their remaining interactions were used as the training set. Table 1 provides the statistics of the four training sets.

Table 1. Dataset Statistics.

Dataset	Users	Items	Interactions	Sparsity
Beauty	22,363	12,101	198,502	99.73%
Sports	35,958	18,357	296,337	99.95%
Yelp	30,431	20,033	316,354	99.95%
ML-1M	6041	3417	999,611	95.16%

4.2. Evaluation Metrics

This study employs two widely used evaluation metrics in recommendation systems: Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG).

- **HR@K:** HR@K is a metric used to measure the recall rate. Specifically, HR@K evaluates the proportion of the top K positions in the recommendation list that successfully "hit" the user's true preference items.
- **NDCG@K:** NDCG@K focuses more on the ranking performance of recommendations. In NDCG@K, not only should the recommended items match the user's true interests, but the items ranked higher should also match the user's higher interest levels.

4.3. Comparison Methods

We select the following models as baseline models for comparison:

- **GRU4Rec [3]:** An RNN-based method that introduces Gated Recurrent Units (GRUs) to explore the dependencies between items in a sequence.
- **Caser [2]:** A CNN-based method that uses horizontal and vertical convolution to model users' dynamic preferences.
- **SASRec [5]:** The first method to introduce Transformers into sequential recommendation, modeling user behavior sequences based on a unidirectional self-attention mechanism.
- **TiSASRec [14]:** Builds on SASRec by incorporating absolute positions and the time intervals between items for sequence modeling.
- **CL4SRec [15]:** Proposes various data augmentation methods to construct contrastive learning tasks, adding a contrastive learning objective to the original SASRec objective.
- **LinRec [30]:** A Transformer-based sequential recommendation model that proposes L2-normalized linear attention mechanisms to reduce the computational cost of traditional attention.
- **MStein [31]:** Calculates the Wasserstein distance between augmented sequences as a self-supervised learning framework for mutual information in sequential recommendation.

4.4. Experimental Details

We implemented Blin and the baseline methods using the PyTorch framework. For all methods, the training batch size was fixed at 256, and the Adam optimizer with a learning rate of 0.001 was used. The embedding dimension size was set to 64, and the maximum sequence length was set to 100. For methods involving attention mechanisms, the number of attention heads and attention layers were both set to 2. The dropout rate for each dataset was set to 0.5. Other hyperparameters for each baseline were configured according to the original papers. For the hyperparameters in our proposed Blin method, λ was set to 2.

4.5. Overall Performance

The performances of all the methods on the three datasets are shown in Table 2, with the best results in each row highlighted in bold. From a comparison of the experimental results, the following conclusions can be drawn:

Table 2. Overall performance, the best result for each row is indicated in bold.

Datasets	Metric	GRU4Rec	Caser	SASRec	TiSASRec	CL4SRec	Linrec	MStein	Blin
Beauty	HR@10	0.1601	0.1498	0.2775	0.2788	0.2931	0.2772	0.3184	**0.3319**
	HR@20	0.2172	0.2012	0.3572	0.3606	0.3648	0.3558	0.3876	**0.4051**
	NDCG@10	0.1013	0.0966	0.1752	0.1794	0.1906	0.1774	0.2122	**0.2218**
	NDCG@20	0.1152	0.1078	0.1961	0.1989	0.2102	0.1978	0.2294	**0.2446**
Sports	HR@10	0.1526	0.1453	0.2763	0.2782	0.2854	0.2766	0.2988	**0.3112**
	HR@20	0.2322	0.2221	0.3769	0.3794	0.3780	0.3762	0.3941	**0.4102**
	NDCG@10	0.0824	0.0813	0.1574	0.1581	0.1739	0.1597	0.1824	**0.1911**
	NDCG@20	0.1037	0.1015	0.1836	0.1846	0.1964	0.1859	0.2067	**0.2158**
Yelp	HR@10	0.2569	0.2546	0.4243	0.4244	0.4462	0.4239	0.4772	**0.4894**
	HR@20	0.4273	0.4252	0.5988	0.5902	0.6003	0.5978	0.6281	**0.6403**
	NDCG@10	0.1253	0.1234	0.2349	0.2448	0.2652	0.2366	0.2846	**0.2944**
	NDCG@20	0.1684	0.1666	0.2796	0.2803	0.3048	0.2842	0.3235	**0.3369**
ML-1M	HR@10	0.3322	0.3145	0.5588	0.5609	0.5811	0.5626	**0.6068**	0.5701
	HR@20	0.4301	0.4028	0.6747	0.6842	0.7024	0.6807	**0.7252**	0.6884
	NDCG@10	0.1823	0.1726	0.3022	0.3084	0.3232	0.3068	**0.3371**	0.3124
	NDCG@20	0.2269	0.2165	0.3846	0.3928	0.4142	0.3892	**0.4308**	0.3985

Firstly, it can be observed that Transformer-based methods (SASRec and TiSASRec) consistently outperform traditional machine learning methods (GRU4Rec and Caser). This result highlights the superiority of self-attention mechanisms in capturing the global dependencies in sequence modeling. Additionally, methods based on contrastive learning (CL4SRec and MStein) consistently perform better than other methods. This is because data augmentation effectively mitigates the data sparsity problem, and contrastive learning methods capture more meaningful features in the sequence by maximizing mutual information, thereby improving the performance of sequential recommendations.

The proposed Blin method demonstrates a superior performance across all evaluation metrics and datasets when compared to all baseline methods on the short-sequence datasets (Beauty, Sports, and Yelp). Unlike the other baseline methods, Blin employs a RandomPad strategy for padding during the sequence-filling stage instead of traditional zero-padding. This approach improves the utilization of the input space compared to traditional methods. To address the increased computational cost due to padding enhancement, the linear attention mechanism is used to significantly reduce the computational complexity of traditional dot product operations. Additionally, the DWC module is incorporated to help the model capture richer local features. Ultimately, the bidirectional KL divergence loss regularizes the probability distributions obtained from different sequence representations, helping the model to learn a target distribution that better aligns with expectations.

It is noteworthy that Blin's performance on the ML-1M dataset did not meet expectations. This may have been due to the limitations of Blin in handling long-sequence datasets. The RandomPad padding strategy loses its effectiveness when dealing with long sequences. In the ML-1M dataset, the average number of user interactions exceeds 160, and even with the maximum sequence length set to 200, most user interaction sequences still lack sufficient space for RandomPad padding. Additionally, RandomPad padding might interfere with the model's ability to capture long-term user preferences, which also affects Blin's performance. Nonetheless, Blin still outperforms other models in terms of its overall performance.

4.6. Ablation Study

To validate the effectiveness of different components of Blin, this paper constructs the following variants and conducts ablation experiments. All experimental variants are evaluated on the Beauty dataset, and the experimental results are shown in Table 3.

Table 3. Ablation study on Beauty dataset.

Model	HR@10	NDCG@10
(A) Blin-ZC	0.3167	0.2088
(B) Blin-D	0.3278	0.2196
(C) Blin-DWC	0.3256	0.2178
(D) Blin-BKL	0.3198	0.2114
(E) Blin	0.3319	0.2218

- **Blin-ZC:** Removes the RandomPad padding method and replaces it with traditional zero-padding, using item cropping as the sequence data augmentation method.
- **Blin-D:** Replaces the linear attention mechanism used in this paper with a traditional attention mechanism.
- **Blin-DWC:** Removes the DWC module from the linear attention mechanism.
- **Blin-BKL:** Removes the auxiliary learning task and relies solely on sequence loss for model training.

By comparing (A) and (E), it is observed that removing the RandomPad padding method significantly reduces the model performance. This is because RandomPad utilizes idle input space to provide more samples for the model and pays more attention to long-tail items, thereby helping the model to learn better user representations. Comparing (B–C)

with (E), it is evident that replacing the linear attention mechanism with a traditional dot product attention mechanism results in a decreased model performance. This is primarily because the DWC module can alleviate the information loss issue inherent in Softmax, helping the model to obtain richer local features and, thus, compute better attention weights. By comparing (D) and (E), it is found that removing the auxiliary learning task from the Blin model leads to a significant performance decrease. This indicates that bidirectional KL divergence loss contributes significantly to model performance; regularizing the probability distributions obtained from different sequence outputs via bidirectional KL divergence loss helps the model to learn target distributions that better match expectations.

4.7. Hyperparameter Study

In this section, the impact of the auxiliary learning task on the model performance is studied by sequentially setting parameter λ to {0.1, 0.5, 1, 2, 3}. HR@10 and NDCG@10 are used as evaluation metrics, and experiments are conducted on the Beauty and Sports datasets. The experimental results are shown in Figure 5.

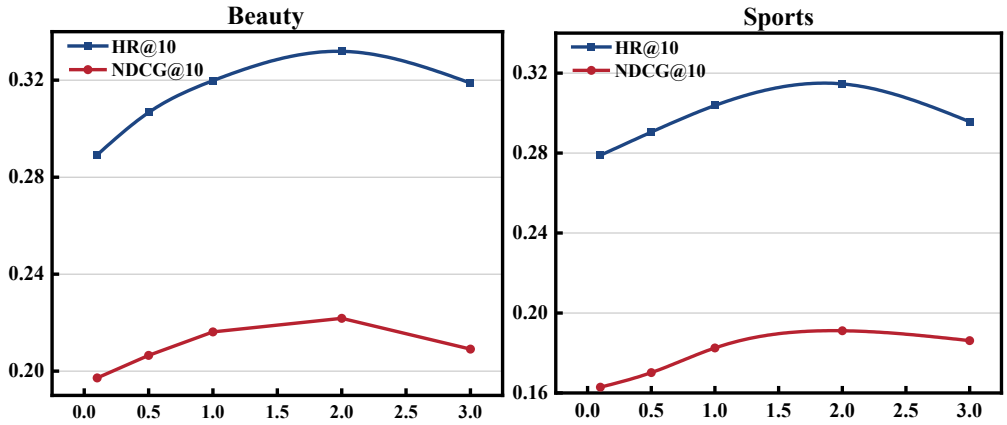

Figure 5. Impact of λ values on model performance.

By observing the changes in the curves of Figure 5, it is observed that the value of λ exhibits a trend of an increasing performance followed by a decrease. Specifically, the model performance peaks when λ is around 2. This result further confirms the effectiveness of the auxiliary learning task; through bidirectional KL divergence loss, the model indeed learns more valuable target distributions.

4.8. Computational Cost Analysis

To study the improvement in the model computational efficiency by the linear attention mechanism discussed in Section 3.4.3, we integrate the linear attention mechanism into the classic Transformer sequence models SASRec and BERT4Rec. GPU memory usage and training time are selected as computational cost indicators, with the training time measured by the time taken for the model to run 10 epochs. All experiments are conducted on the ML-1M dataset, and the final experimental results are shown in Figure 6.

From Figure 6, it is observed that models using the linear attention mechanism significantly reduce the GPU memory usage and training time compared to the initial models. This indicates that the linear attention mechanism used in the Blin model effectively mitigates the high computational costs associated with traditional dot product operations.

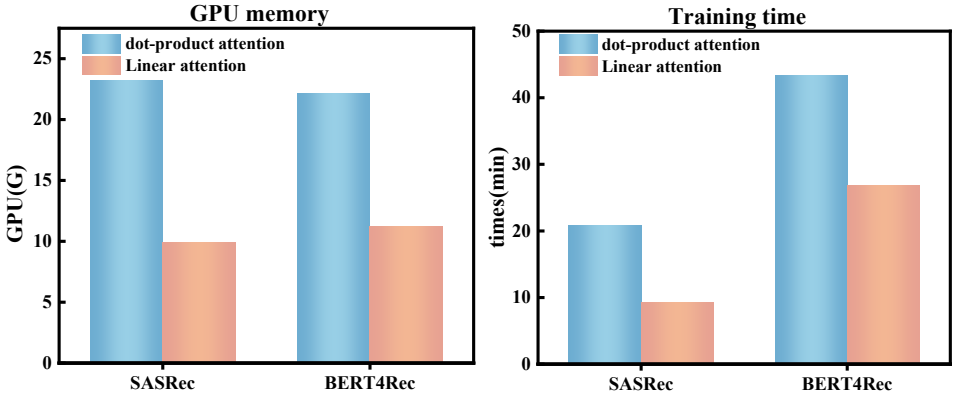

Figure 6. Comparison of computational costs.

5. Conclusions and Outlook

5.1. Conclusions

The Blin model proposed in this paper significantly improves the issues of data sparsity, sequence padding space utilization, the computational complexity of attention mechanisms, and the limitations of a single loss function in sequence recommendation through innovative RandomPad and linear attention mechanisms. By employing the RandomPad padding method, Blin overcomes the inefficiency of traditional zero-padding strategies and addresses data sparsity. Combined with the linear attention mechanism, Blin reduces the computational complexity of handling long sequences from $O(n^2)$ to nearly linear $O(n)$, greatly enhancing computational efficiency while maintaining traditional attention learning capabilities. The introduction of bidirectional KL divergence loss as an auxiliary task enhances the regularization effect of probability distributions across different sequence representations, further improving the accuracy of the model predictions.

The innovation of the Blin model lies not only in its technical breakthroughs, but also in its effective response to longstanding issues in sequence recommendation systems. Blin's success validates the feasibility and effectiveness of introducing multitask learning and linear attention mechanisms in sequence recommendation tasks. By integrating random repetitive padding and linear attention mechanisms, Blin offers new insights for addressing data sparsity and computational complexity issues.

5.2. Outlook

Despite the significant progress Blin has made in addressing data sparsity and computational complexity in sequence recommendation, there are areas that warrant further exploration. Firstly, when the initial user sequence length is greater than or equal to half of the maximum sequence length that the Blin model can handle, the padding times α will always be zero, rendering the RandomPad padding method ineffective. Considering the limitations of the RandomPad padding method in handling long sequences, future research should focus on improving Blin's padding strategy for long sequence datasets. Secondly, with the increasing application of graph neural networks in sequence recommendation, integrating graph neural networks with the Blin model to leverage graph structure information for enhancing sequence modeling capabilities is a promising direction to explore. Finally, extensive research on auxiliary learning in sequence recommendation has demonstrated its effectiveness in improving model performance, highlighting the significant potential of optimizing recommendation systems through additional tasks. Future research can further explore and improve the design of auxiliary tasks in sequence recommendation to enhance the performance and applicability of sequence recommendation models.

Author Contributions: Methodology, Y.B.; validation, Y.B. and H.W.; formal analysis, Y.B.; investigation, Y.B. and H.W.; writing—original draft preparation, Y.B.; writing—review and editing, Y.B. and H.W.; supervision, H.W. and J.H.; All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by the National Natural Science Foundation of China under Grant 82160347.

Data Availability Statement: This study utilized publicly available datasets for experimental research.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wang, S.; Hu, L.; Wang, Y.; Cao, L.; Sheng, Q.Z.; Orgun, M. Sequential recommender systems: Challenges, progress and prospects. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, 10–16 August 2019; pp. 6332–6338.
2. Tang, J.; Wang, K. Personalized top-n sequential recommendation via convolutional sequence embedding. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, Los Angeles, CA, USA, 5–9 February 2018; pp. 565–573.
3. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-based Recommendations with Recurrent Neural Networks. *arXiv* **2015**, arXiv:1511.06939.
4. Zhang, S.; Chen, L.; Wang, C.; Li, S.; Xiong, H. Temporal Graph Contrastive Learning for Sequential Recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 20–27 February 2024; Volume 38, pp. 9359–9367.
5. Kang, W.C.; McAuley, J. Self-attentive sequential recommendation. In Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), Singapore, 17–20 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 197–206.
6. Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; Jiang, P. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 1441–1450.
7. Dang, Y.; Yang, E.; Guo, G.; Jiang, L.; Wang, X.; Xu, X.; Sun, Q.; Liu, H. TiCoSeRec: Augmenting data to uniform sequences by time intervals for effective recommendation. *IEEE Trans. Knowl. Data Eng.* **2023**, *36*, 2686–2700. [CrossRef]
8. Dang, Y.; Yang, E.; Guo, G.; Jiang, L.; Wang, X.; Xu, X.; Sun, Q.; Liu, H. Uniform sequence better: Time interval aware data augmentation for sequential recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; Volume 37, pp. 4225–4232.
9. Dang, Y.; Liu, Y.; Yang, E.; Guo, G.; Jiang, L.; Wang, X.; Zhao, J. Repeated Padding as Data Augmentation for Sequential Recommendation. *arXiv* **2024**, arXiv:2403.06372.
10. Adler, A.; Tang, J.; Polyanskiy, Y. Quantization of random distributions under KL divergence. In Proceedings of the 2021 IEEE International Symposium on Information Theory (ISIT), Virtual, 12–20 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 2762–2767.
11. Chong, L.; Liu, X.; Zheng, R.; Zhang, L.; Liang, X.; Li, J.; Wu, L.; Zhang, M.; Lin, L. CT4Rec: Simple yet Effective Consistency Training for Sequential Recommendation. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Long Beach, CA, USA, 6–10 August 2023; pp. 3901–3913.
12. Rendle, S.; Freudenthaler, C.; Schmidt-Thieme, L. Factorizing personalized markov chains for next-basket recommendation. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010; pp. 811–820.
13. Zhang, T.; Zhao, P.; Liu, Y.; Sheng, V.S.; Xu, J.; Wang, D.; Liu, G.; Zhou, X. Feature-level deeper self-attention network for sequential recommendation. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, 10–16 August 2019; pp. 4320–4326.
14. Li, J.; Wang, Y.; McAuley, J. Time interval aware self-attention for sequential recommendation. In Proceedings of the 13th International Conference on Web Search and Data Mining, Houston, TX, USA, 3–7 February 2020; pp. 322–330.
15. Xie, X.; Sun, F.; Liu, Z.; Wu, S.; Gao, J.; Zhang, J.; Ding, B.; Cui, B. Contrastive learning for sequential recommendation. In Proceedings of the 2022 IEEE 38th International Conference on Data Engineering (ICDE), Virtual, 9–12 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1259–1273.
16. Liu, Z.; Chen, Y.; Li, J.; Yu, P.S.; McAuley, J.; Xiong, C. Contrastive self-supervised sequential recommendation with robust augmentation. *arXiv* **2021**, arXiv:2108.06479.
17. Qiu, R.; Huang, Z.; Yin, H.; Wang, Z. Contrastive learning for representation degeneration problem in sequential recommendation. In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, Tempe, AZ, USA, 21–25 February 2022; pp. 813–823.
18. Zhang, Y.; Liu, Y.; Xu, Y.; Xiong, H.; Lei, C.; He, W.; Cui, L.; Miao, C. Enhancing sequential recommendation with graph contrastive learning. *arXiv* **2022**, arXiv:2205.14837.
19. Choromanski, K.; Likhosherstov, V.; Dohan, D.; Song, X.; Gane, A.; Sarlos, T.; Hawkins, P.; Davis, J.; Mohiuddin, A.; Kaiser, L.; et al. Rethinking Attention with Performers. *arXiv* **2020**, arXiv:2009.14794.

20. Li, R.; Su, J.; Duan, C.; Zheng, S. Linear attention mechanism: An efficient attention for semantic segmentation. *arXiv* **2020**, arXiv:2007.14902.
21. Katharopoulos, A.; Vyas, A.; Pappas, N.; Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020; pp. 5156–5165.
22. Shen, Z.; Zhang, M.; Zhao, H.; Yi, S.; Li, H. Efficient attention: Attention with linear complexities. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Virtual, 5–9 January 2021; pp. 3531–3539.
23. Bolya, D.; Fu, C.Y.; Dai, X.; Zhang, P.; Hoffman, J. Hydra attention: Efficient attention with many heads. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; Springer Nature: Cham, Switzerland, 2022; pp. 35–49.
24. Han, D.; Pan, X.; Han, Y.; Song, S.; Huang, G. Flatten transformer: Vision transformer using focused linear attention. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 1–6 October 2023; pp. 5961–5971.
25. Guo, J.; Chen, X.; Tang, Y.; Wang, Y. SLAB: Efficient Transformers with Simplified Linear Attention and Progressive Re-parameterized Batch Normalization. *arXiv* **2024**, arXiv:2405.11582.
26. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
27. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
28. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
29. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer normalization. *arXiv* **2016**, arXiv:1607.06450.
30. Liu, L.; Cai, L.; Zhang, C.; Zhao, X.; Gao, J.; Wang, W.; Lv, Y.; Fan, W.; Wang, Y.; He, M.; et al. 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, Taipei, Taiwan, 23–27 July 2023; pp. 289–299.
31. Fan, Z.; Liu, Z.; Peng, H.; Yu, P.S. Mutual wasserstein discrepancy minimization for sequential recommendation. In Proceedings of the ACM Web Conference 2023, Austin, TX, USA, 30 April–4 May 2023; pp. 1375–1385.