

Article

Feature-Interaction-Enhanced Sequential Transformer for Click-Through Rate Prediction

Quan Yuan ¹, Ming Zhu ², Yushi Li ^{2,*}, Haozhe Liu ² and Siao Guo ²¹ School of Mechanical Engineering and Electronic Information, China University of Geosciences, Wuhan 430074, China; yuanquan_cug@163.com² Hubei Key Laboratory of Smart Internet Technology, School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China; zhuming@mail.hust.edu.cn (M.Z.); hiliuhaozhe@163.com (H.L.); g553690204@163.com (S.G.)

* Correspondence: liyushi@hust.edu.cn

Abstract: Click-through rate (CTR) prediction plays a crucial role in online services and applications, such as online shopping and advertising. The performance of CTR prediction can have a direct impact on user experience and the revenue of the online platforms. For CTR prediction models, self-attention-based methods have been widely applied to this field. Recent works generally adopted the Transformer architecture, where the self-attention mechanism can capture the global dependencies of the user's historical interactions and predict the next item. Despite the effectiveness of self-attention methods in modeling sequential user behaviors, most sequential recommenders hardly exploit feature interaction techniques to extract high-order feature combinations. In this paper, we propose a Feature-Interaction-Enhanced Sequence Model (FESeq), which integrates feature interaction and the sequential recommendation model in a cascading structure. Specifically, the interacting layer in FESeq is an automatic feature engineering step for the Transformer model. Then, we add a linear time interval embedding layer and a positional embedding layer to the Transformer in the sequence-refiner layer to learn both the time intervals and the position information in the user's sequence behaviors. We also design an attention-based sequence pooling layer that can model the relevance of the user's historical behaviors and the target item representation through scaled bilinear attention. Our experiments show that the proposed method beats all the baselines on both public and industrial datasets.



Citation: Yuan, Q.; Zhu, M.; Li, Y.; Liu, H.; Guo, S. Feature-Interaction-Enhanced Sequential Transformer for Click-Through Rate Prediction. *Appl. Sci.* **2024**, *14*, 2760. <https://doi.org/10.3390/app14072760>

Received: 2 March 2024

Revised: 19 March 2024

Accepted: 22 March 2024

Published: 26 March 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: click-through-rate prediction; feature interaction; sequential recommendation; sequence pooling; self-attention

1. Introduction

The click-through rate (CTR) prediction task is a binary classification problem that measures the likelihood of a user clicking an item. A higher CTR indicates that a higher percentage of the recommended items are clicked by the users, leading to more revenues for the advertising platforms. Therefore, research in CTR prediction has gained attention from academia and industry [1,2]. Currently, two mainstream methods exist to solve the CTR prediction task in recommender systems: feature interaction and sequential recommendation. While both methods aim to provide personalized recommendations by mining useful information from the user-item interactions, they process the feature fields differently.

Feature interaction methods combine features that describe the current interaction and generate high-order feature combinations relevant to the prediction task. As a specific research field in non-sequential recommendation systems, feature interaction has evolved tremendously. Previous works on feature interaction methods [3,4] adopted the inner product operation of individual features, which managed to extract simple low-order feature crosses. Wide and Deep [5] first proposed a two-stream structure, which

complements a linear model with a deep neural network (DNN) to observe both low-order and high-order feature combinations. Since then, the two-stream architecture has become the paradigm of many current state-of-the-art (SOTA) models, such as DeepFM [6], DCN [7], and FinalMLP [8]. Recent studies achieve feature interaction through the attention network [9,10] and self-attention mechanism [11]. The multi-stream structure is a plausible solution to further improve the performance of the feature interaction model, which integrates multiple low-order models and a high-order model in a parallel manner [12]. However, feature interaction methods cannot take advantage of the user's historical behavior sequences and thus fail to extract the user's temporal interests and behavioral patterns.

Sequential recommendation methods can model the sequential dependencies of the user's historical behaviors. By capturing the underlying sequential patterns along the timeline in the user's actions, such as clicks, purchases, and ratings, sequential recommenders achieve better performance than feature interaction methods in real-world scenarios, such as e-commerce [13], media platforms [14], and news recommendations [15]. Previous sequential recommenders focused mainly on traditional methods, such as Markov chains (MC) and hidden Markov models [16,17], which calculate the user state transition probability based on the previous N-order interactions. More sophisticated deep learning methods, such as Recurrent Neural Networks (RNNs) [18] and Convolutional Neural Networks (CNNs) [19], then emerged and have been vigorously applied in this field. A multitude of sequential recommendation research nowadays focuses on the Transformer architecture [20]. The self-attention mechanism in the Transformer calculates the relevance of different user behavior pairs to capture the long-term dependencies of the user's sequence data [21,22], making self-attention-based sequence recommenders perform much better than MC-based and previous deep-learning-based methods. Sequence recommenders focus on mining the dynamics of the user's historical interactions but ignore the high-order features in the user's sequential interactions.

In short, either a pure feature interaction model or a pure sequential recommender has its disadvantages in processing non-sequential and sequential feature fields. Therefore, endeavors have been made to integrate feature interaction and sequential recommendation in CTR prediction. For example, JointCTR [23] manages to combine feature interaction models and a sequential recommendation model in a parallel manner. Yet, the parallel architecture treats feature interaction and sequential recommendations separately, making it suboptimal. Actually, two limitations can be found in the parallel architecture:

- First, the feature interactions of sequential features are ignored. For example, the performance of sequential recommendation in the gaming scenario is constrained without feature interactions between the player's current number of coin stocks and the current number of virtual gadget stocks in each historical behavior.
- Second, it cannot learn the feature combinations of non-sequential and sequential features, such as a combination between the user's gender and the genre sequence of the interacted movies in a movie recommendation.

To overcome the limitations of the parallel architecture, we aim to integrate feature interaction and sequential recommendation in a cascading manner. We obtain high-order features from non-sequential and sequential feature fields, combine them with the original features, and feed them into the sequence recommendation model for user behavior representation learning.

In this paper, we propose a Feature-Interaction-Enhanced Sequence model (FESeq) that considers feature interaction as an incipient feature engineering step for the Transformer model. The high-order features learned from the feature interaction model can further enhance the expressive power of the sequential recommender. Besides, given that linear time intervals and positions in a user sequence are useful for learning the user's temporal interest, we introduce a linear time interval embedding layer and a learned positional embedding layer to the Transformer. Inspired by the explicit user-to-item relevance modeling in DMR [24] and the attention-based sequence pooling in DIN [13], we design a novel attention-based sequence pooling layer that aggregates the Transformer output sequence

through attention. Using scaled bilinear attention [25], the attention score in the sequence pooling layer explicitly measures the similarity between the user's historical interactions and the target item. We believe that attention-based sequence pooling is better than simple target pooling and average pooling because it can adaptively learn the relevance of the user's historical behaviors to the target item and assign higher weights to more relevant behaviors. We formulate the recommendation ranking problem as a click-through-rate prediction task. The experimental results reveal that our model is superior to the SOTA CTR prediction methods on public and industrial datasets.

Our contributions with this paper are as follows:

- We propose a novel sequential model named FESeq that makes feature interaction an automatic feature engineering step for the sequential model. The feature interaction in FESeq captures high-order feature combinations and further enhances the representation learning of sequential user interactions.
- We introduce a learned positional embedding and a linear time interval embedding in the Transformer to model the timeline interest from the user's historical interactions.
- We design a new sequence pooling method based on scaled bilinear attention that explicitly learns the similarities between each user interaction and the target item.
- Our extensive experiments validate the effectiveness of the proposed FESeq model. After integrating feature interaction into sequence feature learning in a cascading manner, FESeq beats all SOTA baselines, including a JointCTR framework that jointly trains feature interaction and sequence feature learning modules in parallel (the AUC increases by +1.46% on Ele.me and +0.30% on Bundle).

The following sections of the paper are organized as follows: Related works are reviewed in Section 2, and our proposed architecture is presented in Section 3. Experimental results and analyses are included in Section 4, and finally, we conclude our work in Section 5.

2. Related Work

2.1. Feature Interaction

Feature interaction is a technique to extract feature combinations from the individual features, which helps increase the expressive power of the recommendation model.

Previous works proposed feature interaction through inner product operations, which included a factorization machine (FM) [3] and product-based neural networks (PNNs) [4]. The FM projects the original features into low-dimensional vectors and calculates the inner product to learn second-order feature interactions. The PNN uses a product layer to learn second-order feature interactions and stacks multiple fully connected layers to further extract higher-order feature interactions. Yet, due to the layer depth scalability challenge, both the FM and the product layer cannot learn higher-order feature interactions. Other than explicit inner product operations, Deep Crossing [26] implicitly learns high-order feature crosses through DNNs with residual units, but the model cannot learn low-order linear feature interactions. NFM [27] stacks multiple DNN layers after the FM model and achieves more expressive power in learning than the FM model. AFM [9] combines the FM with an attention network to aggregate the output of the FM.

Other than the DNN, innovations in feature interaction layer architectures have improved the scalability of feature interaction models and achieved effective higher-order feature interactions. DCN [7] designs a cross network to perform efficient feature interaction in linear complexity and allows for the growth of the stacking layer depth. xDeepFM [28] proposed a novel Compressed Interaction Network (CIN) that combines outer-product and convolution pooling operations to learn explicit high-order feature interactions. Ref. [11] proposed AutoInt, which adopts a residual self-attention network that automatically identifies high-order feature interactions and ensures exponential degree growth. DCN V2 [29] improves the cross network in DCN to learn both point-wise and vector-wise feature interactions. In [30], a self-attention network with a disentangled self-attention operation decouples the unary attention weights from the pairwise ones. FINAL [31] achieves exponential polynomial degree growth through layers of multiplicative interaction.

Since AutoInt [11] explicitly learns high-order feature combinations through a self-attention mechanism and achieves exponential polynomial degree growth, a few self-attention interacting layers can suffice and provide explainability through the attention scores. Therefore, we adopt the AutoInt model to extract high-order feature interactions.

2.2. Sequential Recommendation

Sequential recommenders model the sequential user behaviors and predict the next item based on the historical information from the user's action.

Conventional methods include pattern mining and MC. Yap [32] proposed a personalized sequence pattern mining approach to discover the frequent sequence pattern from the user interaction sequence, but it neglects more fine-grained information in the sequence. Fossil [17] is a high-order MC model that learns local dynamics in the user sequence.

Previous deep learning methods include RNNs and CNNs. RNN-based sequence models can capture long-term information from the past sequence and predict future items. Ref. [18] applies the Gated Recurrent Unit (GRU) to sequential recommendations and uses ranking loss to train GRUs. Ref. [33] uses Long Short-Term Memory to model the temporal dynamics of user behaviors and item features. Ref. [34] proposed a Hierarchical RNN that learns short-term information about each user session through a session-level RNN and long-term dependencies of all sessions from a user through a user-level RNN. CNNs can model the sequential patterns by using convolutional filters. Caser [19] transforms the user action sequence into an "image" and processes the image data by using horizontal and vertical convolutional filters to model point-level and union-level patterns, respectively. Ref. [35] applies a stack of 1D dilated CNNs to enlarge the receptive field and capture long-range item dependencies directly from the sequence data.

Recent progress in sequential recommendations can be attributed to enhancements in model architectures and pretraining techniques from the Natural Language Processing (NLP) field [36], such as the attention network and Transformer. The idea of attention network architecture [37] is to attend to the historical user interactions most relevant to the target item. DIN directly measures the attention between the interacted items and the target item and aggregates the representations of the interacted items with the attention scores. DIEN improves DIN by using a GRU to extract the user's interest and a GRU with an attentional update gate (AUGRU) to learn the evolution of the user's interest. DMR [24] uses attention pooling to learn the representation of the user's historical interactions.

The breakthrough in Transformer architectures in NLP also fuels novel Transformer-based methods in sequential recommendations. For example, AtRank [38] leverages a multi-head self-attention mechanism to learn the latent representation of heterogeneous user behaviors and the attention network to combine multiple user behavior representations. SASRec [21] utilizes a Transformer decoder to predict the next item based on the user's historical actions. BST [22] captures long-term dependencies from the user's historical behaviors by using a Transformer encoder and represents the position information as the linear time intervals between the user's historical interaction and the current interaction. BERT4Rec [39] adopts the vanilla BERT model and a masked item prediction training task to predict the next item. SSE-PT [40] novelly introduces user embeddings to the vanilla Transformer and adds a stochastic shared embedding regularization term to avoid overfitting. TiSASRec [41] proposes the relative time interval embedding and designs a self-attention mechanism that combines the absolute position and relative time interval information. DMIN [42] refines the user's sequence behaviors through Transformer layers and models the user's multiple interests through multi-head self-attention, where each head learns a dimension of the user's latent interest.

Since the Transformer backbone has become mainstream in current CTR prediction models that deal with sequential user data, we adopt the Transformer to model the sequential user behaviors.

3. Proposed Model

3.1. Overall Architecture

To utilize high-order feature interactions for representation learning of the sequential user behaviors, we propose a Feature-Interaction-Enhanced Sequential model (FESeq). The overall architecture of the model is shown in Figure 1, which consists of five components:

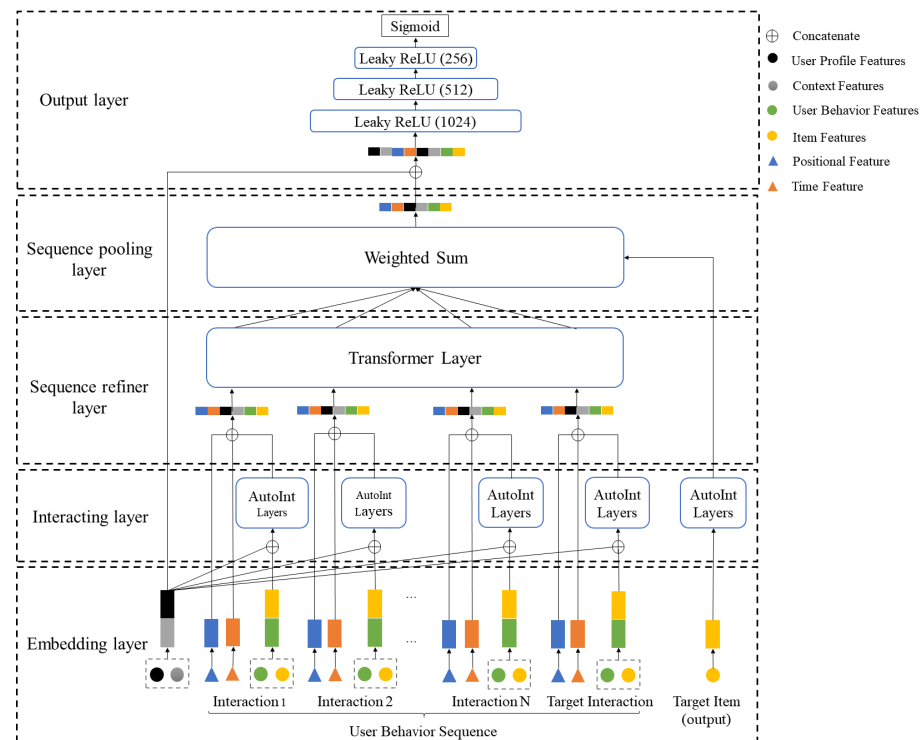


Figure 1. The architecture of FESeq. The embedding layer embeds the input features into a low-dimensional fixed-length embedding. Then, the interacting layer extracts high-order feature interactions from all feature fields. The sequence-refiner layer further refines the user's behavior sequence through a Transformer layer. At the sequence pooling layer, scaled bilinear attention pools the Transformer output through explicit user–item similarity calculation. An MLP network in the output layer predicts the CTR probability.

- *Embedding layer:* It is used to represent the input feature as a dense embedding representation. We use the input feature embedding layer and the output item embedding layer to represent all original features in the input data and the target item features, respectively, as is shown in Figure 1. The original features include non-sequential features (the user profile and context) and sequential features (the positions, time intervals, user behaviors, and interacted items). The target item features are all the last element of the interacted item feature sequences.
- *Interacting layer:* It is proposed to extract high-order feature interactions from all the original feature fields by using the self-attention mechanism. The details of this layer and how we view feature interaction as a form of an automatic feature engineering step for the input of the sequential model are in Section 3.2.
- *Sequence-refiner layer:* It refines the user's behavior sequence using the Transformer layer. Unlike the vanilla Transformer, we add the positional embedding and the linear time interval embedding to the input of the Transformer layer. More details on this layer are presented in Section 3.3.
- *Sequence pooling layer:* It is designed to pool the output sequence of the Transformer model based on the relevance of each user behavior to the target item. The details on the pooling methods and the attention score calculations are introduced in Section 3.4.

- **Output layer:** In this layer, we concatenate the output of the sequence pooling layer with the embeddings of non-sequential features and utilize three MLP layers with LeakyReLU hidden activation to reduce the dimension of the output features. Since the task of CTR prediction is a binary classification problem, a sigmoid function will project the output of the MLP layers into CTR probability.

3.2. Feature Interaction as an Automatic Feature Engineering Step for Sequential Recommendation

We propose to stack the feature interaction layers and the sequential model to enhance the expressiveness of the sequential model. We introduce an interacting layer to extract high-order features from the original ones. Besides, we find that the input feature fields of the interacting layer should include both sequential features and non-sequential features because a feature interaction model such as AutoInt can capture the relevance between two heterogeneous feature fields. For example, in [11], AutoInt assigns significant influences between the genre of the movie, a sequential feature field in the sequential recommendation setting, and the gender of the user, a non-sequential feature field. To combine non-sequential and sequential feature fields, the non-sequential fields are transformed into sequential fields. Specifically, we broadcast non-sequential feature embeddings e_{ns} to sequences $E_{ns} = (e_{ns}, e_{ns}, \dots, e_{ns})$, which have the same length as sequential features, and concatenate E_{ns} with sequential feature embeddings E_s along the feature field dimension to generate embeddings from all feature fields E_{in} , which are then the input of the interacting layer. We choose self-attention layers in AutoInt [11] as the core part of the interacting layer. The i -th AutoInt layer based on self-attention requires the following operations:

$$Q = E_{in}W^Q, K = E_{in}W^K, V = E_{in}W^V \quad (1)$$

$$\tilde{E}_{in} = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \quad (2)$$

$$E_{in}^i = \text{ReLU}(\tilde{E}_{in}^{i-1} + E_{in}^{i-1}) \quad (3)$$

where $W^Q, W^K, W^V \in R^{d \times d}$, and d are the dimensions of query, key, and value. According to Figure 1, for the interacting layer with n AutoInt layers, the output of the last AutoInt layer E_{in}^n becomes the input of the Transformer layer. Therefore, FESeq unifies the feature interaction and the sequential model by stacking two types of layers together.

The interacting layer can be considered as an automatic feature engineering step before the sequential modeling. Due to the residual connection in each AutoInt layer, the output of the last AutoInt layer E_{in}^n maintains the original features and generates as high as 2^n -order feature combinations. The AutoInt layers engineer new high-order features from the provided feature fields. This feature engineering process can enhance the relevance of different original features and provide informative feature combinations for the sequential modeling of user behaviors. The feature engineering step is automatic because, after the joint training of the interacting layer and the sequential model, the interacting layer can automatically learn high-order features useful to the predictive sequential model.

Here are four details about the interacting layer not shown in Equation (3) for simplicity.

- In general, feature fields at different positions of a user behavior sequence tend to share similar high-order feature combinations. Thus, we introduce parameter sharing so that the input fields at each position of the user behavior sequence share the same parameters in the interacting layer.
- Since the interacting layer preserves the dimension, we can further stack multiple layers to extract higher-order feature interaction.
- The AutoInt model can achieve base-2 exponential degree growth. Therefore, stacking a few layers (no more than three layers) is sufficient to extract high-order feature interactions in real-world settings.

- Input feature interaction layers and output item feature interaction layers are designed to extract feature interactions from input feature embeddings and output item embeddings, respectively. We denote the output of the input feature interaction layers as E_{in}^M , where M is the number of the input feature interaction layers and the output of the output item feature interaction layers as E_{out}^N , where N is the number of the output item feature interaction layers.

3.3. A Position and Time-Aware Sequence-Refiner Layer Based on Transformer

The sequence-refiner layer learns a deep-level representation of the user behaviors by capturing long-range dependencies of the user behavior sequence. The core of this module is a position and time-aware Transformer layer. To model the temporal interest of the user's behavior sequence, both a positional embedding and a linear time interval embedding are concatenated with E_{in}^M and become the input to the Transformer layer, which is shown in Figure 2.

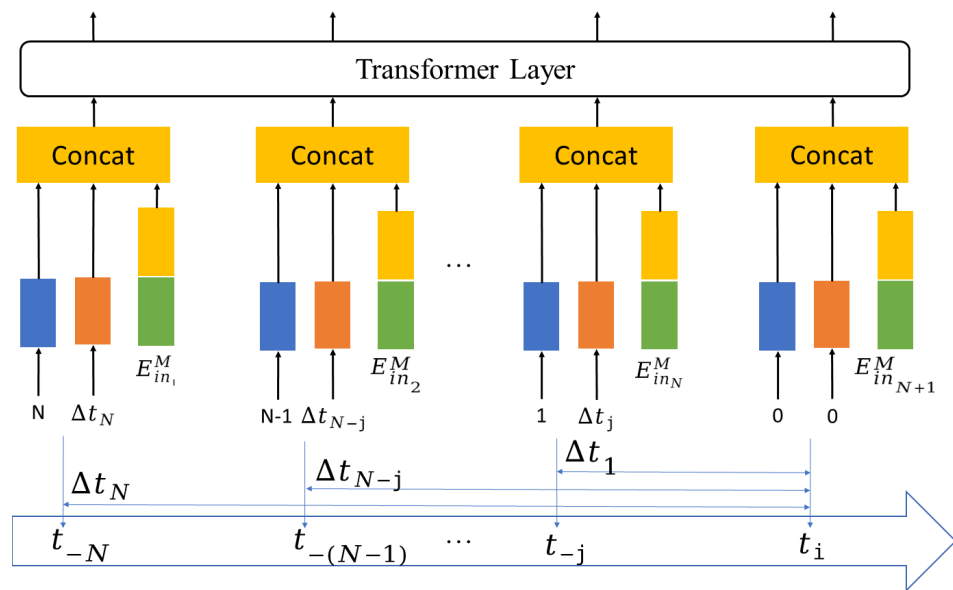


Figure 2. Illustration of the position and time-aware sequence-refiner layer, where both positional and linear time embeddings are concatenated with the user behavior sequence.

Positional embedding: Since self-attention models do not contain any position-aware structures, such as the recurrent structure in RNNs and convolution filter operations in CNNs, we need to add a learned positional embedding $p \in R^d$ to the feature embedding E_{in}^M in Section 3.2, where M represents the number of input feature interaction layers. The positional embedding p is obtained from a look-up table $P \in R^{n \times d}$ that encodes the position ids to a fixed-length embedding. As more recent behaviors have more influence on the target item prediction, we assign position ids to the user behaviors in a reverse time order.

Linear time interval embedding: As is shown in [41], positional embedding fails to represent the time interval information between neighboring behaviors. However, user behaviors closer to the current timestamp tend to have greater impacts on the current prediction. Therefore, we propose to insert linear time interval embedding $ti \in R^d$ into the input feature embeddings E_{in}^M . Linear time interval embedding embeds the time interval between the timestamp of the historical user behavior t_{-i} and the current timestamp t_0 , $\Delta t_i = t_0 - t_{-i}$, through time embedding $Ti \in R^{m \times d}$. Finally, the input embeddings of the Transformer layer in the sequence-refiner layer are represented as

$$\hat{E} = \begin{bmatrix} E_{in_1}^M + p_n + ti_n \\ E_{in_2}^M + p_{n-1} + ti_{n-1} \\ \dots \\ E_{in_n}^M + p_1 + ti_1 \end{bmatrix} \quad (4)$$

The Transformer layer is causal and contains a multi-head self-attention layer and point-wise feed-forward networks.

Multi-head self-attention layer: The calculations in the multi-head self-attention layer are almost the same as Equation (2), except that we use multiple self-attention heads. The process is then described as

$$S = MH(\hat{E}) = \text{concat}(\text{head}_1, \text{head}_2, \dots, \text{head}_h)W^H \quad (5)$$

$$\text{head}_i = \text{Attention}(\hat{E}W^Q, \hat{E}W^K, \hat{E}W^V) \quad (6)$$

where $W^Q, W^K, W^V \in R^{d \times d}$, and $W^H \in R^{hd \times d}$ are projection matrices. h is the number of attention heads. Apart from more efficient parallel computations, multi-head self-attention has a stronger expressive power of learning the user behavior representation than single-head self-attention. As the input sequence is projected to representations in multiple latent subspaces, each subspace can effectively learn a type of the user's heterogeneous behavior or the user's latent interest.

Causality: The causal Transformer has been applied in tasks such as neural machine translation and next-item recommendation. The $i + 1$ -th element is predicted based on the i -th element in the sequence. Each item i only attends to the previous item j ($i > j$) and cannot see the future items, which effectively eliminates the information leakage problem. While our proposed model predicts the target item based on all historical user behaviors, making causality not a requirement, our experiment reveals that a causal Transformer in a sequence-refiner layer leads to a better CTR prediction performance than a non-causal Transformer. A possible explanation is that the sequence pooling layer in Section 3.4 calculates the relevance of all user behaviors and the target item. To model the influence of each behavior separately, any future behaviors should not blend in with the historical behavior.

Point-Wise Feed-Forward Network: According to [22], we add a point-wise feed-forward network (FFN) after the self-attention layer. We also use dropout layers to avoid overfitting. The activation function of the FFN is LeakyReLU. The output of the self-attention layer S' is represented as

$$S' = \hat{E} + \text{Dropout}(MH(\hat{E})) \quad (7)$$

and the output of the two-layer FFN is represented as

$$F = S' + \text{Dropout}(\text{LeakyReLU}(S'W^{(1)} + b^{(1)}) + W^{(2)} + b^{(2)}) \quad (8)$$

where $W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}$ are learnable network parameters. We only use one layer of the Transformer based on the experimental results in [22].

3.4. A Sequence Pooling Layer Based on Scaled Bilinear Attention and User–Item Similarity

Since the sequence-refiner layer preserves the dimension of the input and output data, the output of the sequence-refiner layer is still sequence data. Therefore, pooling is needed to aggregate the sequence data for downstream MLP network prediction. Current popular pooling methods include *concat pooling* (a concatenation of all behaviors in a sequence [22]), *target pooling* (the target behavior as the pooling result), *mean pooling* (the mean of all behaviors in a sequence), and attention-based weighted-sum pooling [24]. While being simple, the first three pooling methods have disadvantages. Concatenation can preserve

all sequence behavior representations, but the output of the concat pooling layer is an extremely high-dimensional vector, thus increasing the number of training parameters. Our experiments also demonstrate that concat pooling makes the network training unstable. Worse still, the model cannot even learn from training data from scratch. Target pooling ensures that the pooling output is a low-dimensional vector, which not only preserves the target behavior representation but also includes the connections of all historical behaviors to the target behavior. However, target pooling assumes that only the target behavior has a strong influence on target item prediction and eliminates all direct influences from historical behaviors. Mean pooling assumes that all user behaviors in a sequence have equal influences on the target item prediction. Either assumption is suboptimal for modeling the influence of each user behavior on the target item: although the target behavior tends to have the most importance, the influences of recent historical behaviors are not negligible. Attention-based weighted sum pooling utilizes an attention network to adaptively learn the influences of all user behaviors on the target item and combine all user behaviors in a sequence by using the attention scores. The attention-based weighted sum pooling method has the following advantages:

- Attention pooling has weaker inductive biases compared with target pooling and mean pooling. Therefore, the model is not constrained by fixed and sometimes flawed prior assumptions.
- The attention weights in the attention-pooling method can explicitly represent the similarity between the user behavior and the target item, which is the core idea of collaborative filtering. Therefore, attention pooling is suitable for the CTR prediction task.

Next, we will introduce the attention-based weighted sum pooling method. As is shown in Figure 3, the attention network requires the query, key, and value. The key and value can be obtained from two linear projections of the sequence-refiner-layer output F . The query is obtained from a different linear projection of the target item embedding e_{out}^N in Section 3.2. The projections to the query, key, and value are represented as

$$q = W^q e_{out}^N, K = FW^K, V = FW^V \quad (9)$$

where $W^q, W^K, W^V \in R^{d \times d_h}$ are three linear projection matrices. The linear projections unify the dimensions of the user behavior and the target item into the same hidden dimension d_h and improve the flexibility of the attention network. We pass q, K , and V through a dropout layer and a PReLU activation layer to avoid overfitting. Then, the attention-pooling process is described as follows:

$$u = \text{PReLU}\left(\sum_{t=1}^n (\alpha_t V_t)W + b\right) \quad (10)$$

$$\alpha_t = g(q, K_t) \quad (11)$$

where the function $g(\cdot, \cdot)$ represents the equation of the attention score α_t . K_t and V_t represent the key and value of the t -th user behavior, respectively. W and b are learnable training parameters that reduce the hidden dimension d_h to the input dimension d . Notably, the attention score α_t reflects the similarity between the t -th user behavior and the target item. Through attention pooling, the user behaviors more similar to the target item have higher attention scores and are assigned higher coefficients when all user behaviors are combined.

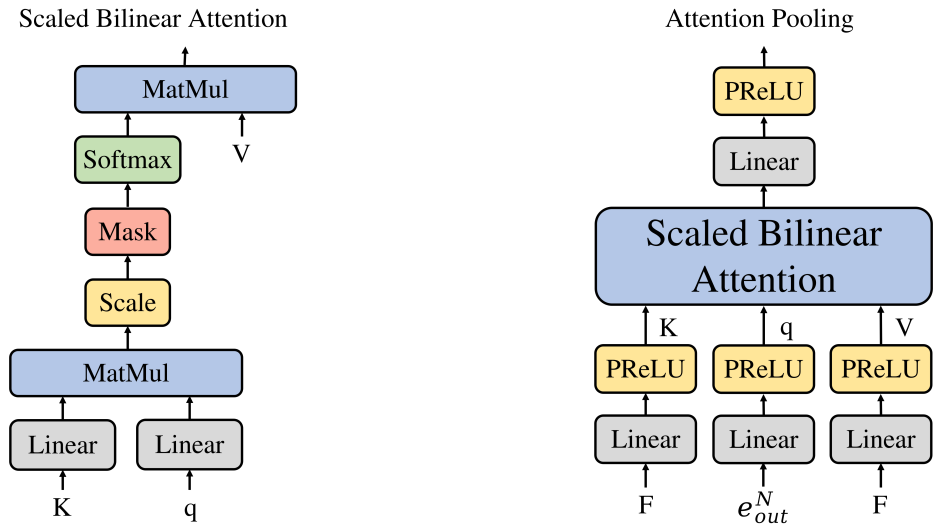


Figure 3. (left) Scaled bilinear attention. (right) Attention pooling is based on scaled bilinear attention, where key and value are obtained from the sequence-refiner-layer output F and the query is generated from the target item embedding e_{out}^N .

Next, we will introduce scaled bilinear attention that calculates attention scores in attention pooling. In [25], bilinear attention networks (BAN) are proposed to solve the multi-modal learning task. Similar to image and text, which are two different modalities, the user behaviors and the target item can also be viewed as two distinct data types in the recommender system. Therefore, we use bilinear attention to calculate the relevance between the query and key and model the relationship between the user and the target item. Similar to scaled-dot attention, we add a scale factor for training stability. The scaled bilinear attention is calculated as follows:

$$g(q, K_t) = \frac{q^T W K_t}{\sqrt{d_h}} \quad (12)$$

where $W \in R^{d_h \times d_h}$ is initialized as an identity matrix.

3.5. Loss Function

The loss function of the task is the cross-entropy loss, which is formulated as

$$L = -\frac{1}{N} \sum_{(x,y) \in D} (y \log(p(x)) + (1-y) \log(1-p(x))) + \lambda \sum_l \|W_l\|_2^2 \quad (13)$$

where D represents all the training samples. $y \in \{0, 1\}$ denotes whether the user clicks the item. $p(x)$ is the output CTR prediction probability with the input sample x , and λ is the L_2 regularization parameter.

4. Experiments

In this section, we introduce the experiment settings and present our experimental results. For the experimental results, we first compare the proposed model with current SOTA baselines on both public and industrial datasets. Then, we conduct ablation studies to verify the efficacy of each part on the proposed model. We also carry out hyperparameter tuning experiments to explore how parameters affect model performance, such as the embedding dimension and the interacting layer depth. To observe how the interacting layer learns the relevance between two feature pairs and how the sequence pooling layer attends to all user behaviors for the CTR prediction task, we provide the visualization results of the attention matrix in the interacting layer and the attention scores in the sequence pooling layer.

4.1. Experiment Settings

4.1.1. Datasets

Public Dataset Ele.me (<https://tianchi.aliyun.com/dataset/131047>, 21 March 2024) contains user click logs sampled from a commercial food delivery website in 3 days, with a total of 1.23 million samples. The label of the dataset represents whether the user clicks the food ad or not. We sort the dataset according to the timestamp and split it into the training set, validation set, and test set with a split ratio of 8:1:1.

Industrial dataset Bundle is a bundle recommendation user interaction dataset collected from an online game server from 1 November 2021 to 30 December 2021. The label of the dataset shows whether the user buys the bundle or not. We also sort the dataset according to the timestamp and split it into the training set, validation set, and test set with a split ratio of 8:1:1. The whole dataset contains 4.19 million samples. The statistics of the datasets are shown in Table 1.

Table 1. Statistics of the datasets.

Dataset	# Users	# Items	# Samples	# Sequence Fields	# Non-Sequence Fields
Ele.me	1,013,338	449,440	1,228,589	14	9
Bundle	118,931	11	4,193,418	24	9

4.1.2. Baseline Models

To confirm the superiority of the proposed model, we compare it with eleven representative feature interaction baselines in the past seven years, six representative sequential recommendation baselines in the past five years, and one representative baseline that combines feature interactions and sequential recommendations. The following models are representative feature interaction baselines:

- **Wide and Deep** [5]: uses a parallelization of a linear model and a DNN model to extract both the low-order and the high-order feature combinations.
- **IPNN** [4]: learns two-order feature interactions based on inner products and high-order feature interactions by stacking multiple layers of the DNN network.
- **DeepFM** [6]: an end-to-end model similar to Wide and Deep except that it replaces the linear model with an FM model and shares the same input for both the wide and the deep part.
- **DCN** [7]: consists of cross network layers and a DNN network in parallel.
- **AutoInt** [11]: it stacks multi-head self-attention layers to extract high-order feature interactions.
- **ONN** [43]: proposes operation-aware embedding layers to learn diverse representations of each feature field for different operations.
- **DCNv2** [29]: improves the cross network in DCN, enabling both point-wise and vector-wise feature interactions.
- **DESTINE** [30]: adopts disentangled self-attention layers to extract high-order feature interactions.
- **DualMLP** [8]: a well-tuned two-stream MLP model.
- **FinalMLP** [8]: an enhanced two-stream MLP model combined with stream-specific feature gating and bilinear interaction fusion layers.
- **FINAL** [31]: Takes advantage of multiplicative feature gating for soft feature selection and multiplicative feature interactions to achieve exponential degree growth. Self knowledge distillation is used to supervise the outputs of two FINAL blocks.

The following models are representative sequential baselines:

- **DIN** [13]: introduces the attention mechanism by adaptively attending to historical user behaviors to learn the user's interest.

- **SASRec** [21]: Utilizes the causal Transformer model for CTR prediction. The latent factor model in the prediction layer is replaced with an MLP model for binary classification.
- **DIEN** [44]: based on AUGRU to capture the evolution of the user interest relevant to the target item.
- **BST** [22]: utilizes the Transformer encoder layer to capture the sequence information of the user's behaviors.
- **TiSASRec** [41]: Embeds both the absolute positions and the relative time intervals of the user interaction sequences and designs a time-interval-aware self-attention mechanism to capture the temporal dependencies of the user action. The latent factor model in the prediction layer is replaced with an MLP model for binary classification.
- **DMR** [24]: An item-to-item network is to learn the user temporal interest representation and the implicit user-item relevance. A user-to-item network is to learn the explicit user-item similarity by using the inner product.

The following model is a representative baseline that unifies feature interaction and sequential recommendation:

JointCTR [23]: A joint framework that combines a logistic regression model, an FM model, a multi-head attention network, and a sequential feature learning model based on DIEN in a parallel manner.

4.1.3. Implementation Details

We implement all the baselines and the proposed FESeq model based on FuxiCTR, an open-source CTR prediction library [45]. We train the model with the Adam optimizer [46]. We use two input feature interaction layers, two target item feature interaction layers, and one Transformer layer for both datasets. The batch size is 4096 and the embedding regularization parameter is 0.1. The dropout rate is 0.1 for both datasets, and the number of MLP hidden units is [1024, 512, 256]. The maximum linear time interval is 4000 for both datasets. For head setting, we set the number of heads to eight for the sequence-refiner layer and one for the interacting layer. The rest of our parameter settings are shown in Table 2. All experiments are conducted with a single RTX-3090 Ti GPU.

Table 2. Hyperparameter settings.

Dataset	Ele.me	Bundle
learning rate	10^{-3}	10^{-4}
maximum sequence length	10	8
sequence pooling hidden dimension	128	32

4.1.4. Evaluation Metrics

We evaluate the test data by using AUC and Logloss. AUC is formulated as

$$AUC = \frac{\sum_{i \in I^+} \sum_{j \in I^-} \delta(\hat{p}_i, \hat{p}_j)}{|I^+||I^-|} \quad (14)$$

where I^+ is the positive sample set and I^- is the negative sample set. \hat{p}_i is the predicted probability that the item i is clicked. And, $\delta(\cdot, \cdot)$ is defined as

$$\delta(i, j) = \begin{cases} 1, & i > j \\ 0.5, & i = j \\ 0, & i < j \end{cases} \quad (15)$$

Logloss is formulated in Equation (13) when the regularization parameter λ is zero. In general, a higher AUC and lower Logloss indicate a better CTR performance.

4.2. Performance Comparison

Table 3 presents the performance comparison results between the baseline models in Section 4.1.2 and the proposed model in Section 3 on two datasets. For simplicity, we denote feature interaction models as type I, sequence recommendation models as type II, and unified models as type III.

Table 3. Performance comparison results. The bold number denotes the best value in the column and the underlined number denotes the second best value in the column.

Type	Models	Ele.me		Bundle	
		AUC	Logloss	AUC	Logloss
I	Wide and Deep	0.5489	0.0950	0.8761	0.0111
	IPNN	0.5535	0.0932	0.8724	0.0126
	DeepFM	0.5460	0.0935	0.8750	0.0115
	DCN	0.5495	0.0932	0.8779	0.0112
	AutoInt	0.5392	0.0950	0.8783	0.0114
	ONN	0.5432	0.0938	0.8783	0.0114
	DCNv2	0.5426	0.0937	0.8759	0.0113
	DESTINE	0.5444	0.0944	0.8749	0.0113
	DualMLP	0.5431	0.0943	0.8788	0.0112
	FinalMLP	0.5497	0.0933	0.8791	<u>0.0111</u>
	FINAL	0.5528	0.0933	0.8796	0.0113
II	DIN	0.5988	0.0924	0.8684	0.0115
	SASRec	0.5867	0.0927	0.8578	0.0114
	DIEN	0.5569	0.0931	0.8743	0.0123
	BST	0.5497	0.0933	0.8766	0.0112
	TiSASRec	0.5914	0.0925	0.8768	0.0122
	DMR	<u>0.6039</u>	<u>0.0921</u>	<u>0.8798</u>	0.0112
III	JointCTR	0.5562	0.0930	<u>0.8882</u>	0.0109
	FESeq	0.6127	0.0918	0.8912	0.0117

According to the results, we obtain the following observations:

- Among all eleven feature interaction baselines, FINAL performs the best. Several reasons are present to explain this. First, the multiplicative feature interaction in the FINAL model can achieve both point-wise and vector-wise feature interactions. However, except for DCNv2, the other feature interaction baselines can only achieve either point-wise or vector-wise feature interaction. Second, the FINAL model simulates a fast exponentiation algorithm to achieve exponential degree growth of any base. Therefore, two factorized interaction layers can reach a high polynomial degree. However, AutoInt and DESTINE can only achieve base-2 exponential degree growth, and the rest of the feature interaction baselines cannot achieve exponential degree growth after stacking multiple layers. Third, the multiplicative feature gating mechanism in the implementation of FINAL also achieves a form of soft feature selection and further improves the model performance.
- Among all six sequential recommendation baselines, DMR has the best performance because it not only leverages the attention network to capture the dynamics in the user behavior sequence but also uses the dot product to explicitly extract the static user–item similarity and further match the user with the target item. Other sequential baselines mainly focus on improving the representation learning of the user behavior but fail to obtain direct relations between the user behavior and the target item.
- Compared with the feature interaction and sequence recommendation baselines, FESeq complements the advantages of two types of CTR prediction models. It can learn higher-order feature interactions, enhance the relevance of the original features, and increase the power of sequence feature learning in the sequential model. Compared with JointCTR, the proposed model extracts higher-order features from heterogeneous

feature fields, further enhancing the expressiveness of the model. Therefore, FESeq is superior to all the baselines in terms of the CTR prediction performance (+1.46% in the AUC and −0.32% in Logloss on the dataset Ele.me; +0.30% in the AUC on the dataset Bundle).

- The CTR prediction performance on the dataset Bundle is significantly better than the dataset Ele.me in the proposed model. The primary reason is that the dataset Bundle contains more user behavior feature fields than the dataset Ele.me, making it easy for FESeq to learn the user's preferences from rich user behavior data on Bundle. Second, the label in the dataset Bundle represents whether the user purchased the game bundle, while that in the dataset Ele.me shows whether the user clicked the item ad. Compared with the click behavior, the purchase behavior more obviously implies the user's item preferences. Therefore, the proposed model can more accurately predict the target item on the dataset Bundle than on the dataset Ele.me.

4.3. Ablation Studies

To further analyze how each part of the proposed model affects the performance and verify the efficacy of each module, we design the following ablation studies.

4.3.1. Influence of Interacting Layer

Table 4 shows the performance comparison of the proposed model with and without an interacting layer on both datasets. “FESeq w/o Int” represents the proposed model without the interacting layer. According to the results, we can see that without the interacting layer, the proposed model performs worse on both datasets. The result shows the advantage of combining feature interaction with the sequential model hierarchically.

Table 4. Ablation studies of models with and without interacting layer. The bold number denotes the best value in the column.

Datasets	Evaluation	FESeq	FESeq w/o Int
Ele.me	AUC	0.6127	0.5882
	Logloss	0.0918	0.0932
Bundle	AUC	0.8912	0.8649
	Logloss	0.0117	0.0120

4.3.2. Influence of Non-Sequential Features on Feature Interaction

Recall that in Section 3.2, we introduce the input feature interaction layers that perform feature interaction on input feature fields, which include both sequential features and non-sequential features. We want to see if adding non-sequential features to the input fields can improve the performance. According to Table 5, “FESeq w/o Non-seq Feats”, which represents the proposed model without adding non-sequential features to input feature interaction layers, does not perform well. The result is reasonable because after removing the non-sequential features, the input feature interaction layers can only perform feature interaction on sequential feature fields, which hinders the learning capability of the interacting layer.

Table 5. Ablation studies of models with and without non-sequential features added to input feature interaction layers. The bold number denotes the best value in the column.

Datasets	Evaluation	FESeq	FESeq w/o Non-Seq Feats
Ele.me	AUC	0.6127	0.5980
	Logloss	0.0918	0.0923
Bundle	AUC	0.8912	0.8780
	Logloss	0.0117	0.0113

4.3.3. Influence of Positional Embedding and Linear Time Interval Embedding

Table 6 shows the performance comparison of FESeq without positional embedding or linear time interval embedding. “FESeq w/o Pos” represents the proposed model without positional embedding, and “FESeq w/o Time” is the proposed model without linear time interval embedding. We can obtain two observations:

- Both positional embedding and linear time interval embedding can help improve the model performance because the two types of embeddings reflect the order and the time intervals of the behaviors in a sequence, respectively, and can help extract the temporal dynamics from the user behaviors.
- The influence of positional embedding on the model performance is more significant than linear time interval embedding. On the one hand, positional embedding encodes the order relations of different behaviors from a user and can show the logic of causality in the user behaviors. On the other hand, notice in Table 2 that the maximum sequence length of both datasets is short (10 on the dataset Ele.me and 8 on the dataset Bundle), which leads to insignificant time interval differences in different user behaviors for the same user. Therefore, compared with positional embedding, linear time interval embedding has a smaller impact on the model performance.

Table 6. Ablation studies of models without positional embedding or linear time interval embedding in sequence-refiner layer. The bold number denotes the best value in the column.

Datasets	Evaluation	FESeq	FESeq w/o Pos	FESeq w/o Time
Ele.me	AUC	0.6127	0.5757	0.6040
	Logloss	0.0918	0.0928	0.0921
Bundle	AUC	0.8912	0.8866	0.8902
	Logloss	0.0117	0.0111	0.0123

4.3.4. Influence of Causal Masks

The causal Transformer predicts the $(i + 1)$ -th item based on all previous i items in the sequence. Therefore, each item in a sequence cannot connect to the future items. In a non-causal Transformer, each item in a sequence can attend to both the previous and the future context information to learn a global representation of the sequence. In the implementation, the difference between a causal Transformer and a non-causal Transformer is that an upper triangular causal mask matrix is added to mask some attention scores in the attention matrix so that each item can only attend to the previous items. We implement a model “FESeq w/o Causal” without the causal masks in the sequence-refiner layer. The performance comparison in Table 7 shows that using a causal Transformer in the sequence-refiner layer can further improve the recommendation performance.

Table 7. Ablation studies of models with and without causal masks in sequence-refiner layer. The bold number denotes the best value in the column.

Datasets	Evaluation	FESeq	FESeq w/o Causal
Ele.me	AUC	0.6127	0.6124
	Logloss	0.0918	0.0920
Bundle	AUC	0.8912	0.8905
	Logloss	0.0117	0.0108

4.3.5. Influence of Sequence Pooling Methods

In Table 8, we compare the performance of the proposed model with different sequence pooling methods. “Target Pooling” represents the proposed model with target pooling in Section 3.4, and “Mean Pooling” is the proposed model with mean pooling in Section 3.4. The proposed model uses the attention-based weighted sum pooling method by default.

The results show that attention-based weighted sum pooling is superior to simple target pooling and mean pooling in the proposed model.

Table 8. Ablation studies of models with different sequence pooling methods in sequence pooling layer. The bold number denotes the best value in the column.

Datasets	Evaluation	FESeq	Target Pooling	Mean Pooling
Ele.me	AUC	0.6127	0.6084	0.6016
	Logloss	0.0918	0.0923	0.0922
Bundle	AUC	0.8912	0.8820	0.8838
	Logloss	0.0117	0.0108	0.0109

4.3.6. Influence of Attention Score Calculation Methods

Apart from scaled bilinear attention, we also implement two methods to calculate attention scores in the sequence pooling layer, which are the scaled-dot product and DIN attention:

- **Scaled-dot product** In [20], the scaled-dot product calculates the similarity between the query q and key K_t :

$$g(q, K_t) = \frac{q^T K_t}{\sqrt{d_h}} \quad (16)$$

where the scale factor $\sqrt{d_h}$ can avoid excessively large dot product results. The scaled-dot product can simply and directly represent the user–item similarity.

- **DIN attention** In the DIN model [13], attention scores are calculated via MLP networks with Dice activation. In our model, the inputs of MLP are the query q , key K_t , and the outer-product of the query and key. The MLP networks then reduce the input data to attention scores. The DIN attention is calculated as

$$g(q, K_t) = \text{Dropout}(\text{Dice}(\text{Concat}(q, K_t, q \otimes K_t)W^{(1)} + b^{(1)})) + W^{(2)} + b^{(2)} \quad (17)$$

Compared with the scaled-dot product and scaled bilinear attention, the drawback of MLP-based DIN attention is that it cannot explicitly represent the similarity between the user behavior and the target item. The scaled-dot product is a simple method to calculate the attention score and model the user–item similarity, but it fails to learn more complex user–item relations without additional learning parameters. Compared with the scaled-dot product, scaled bilinear attention can capture more complex information about the relations between the user behavior and the target item.

As is shown in Table 9, the proposed model uses scaled bilinear attention by default. The model “Scaled Dot Attention” uses scaled-dot attention to calculate the attention scores, and the model “DIN Attention” uses DIN attention for the attention score calculation. The results verify that using scaled bilinear attention for attention score calculations in the sequence pooling layer achieves the best performance. An explanation for the results is that MLP-based DIN attention cannot explicitly represent the similarity between the user behavior and the target item and that the scaled-dot product fails to learn complex user–item relations without additional learning parameters.

Table 9. Ablation studies of models with different attention score calculation methods in sequence pooling layer. The bold number denotes the best value in the column.

Datasets	Evaluation	FESeq	Scaled Dot Attention	DIN Attention
Ele.me	AUC	0.6127	0.6124	0.5987
	Logloss	0.0918	0.0920	0.0939
Bundle	AUC	0.8912	0.8868	0.8821
	Logloss	0.0117	0.0109	0.0112

4.3.7. Influence of Scale Factor

Related works [20,41] show that a scale factor should normalize the attention scores in the dot product of the self-attention layer to avoid high dot product values. To verify the influence of the scale factor on bilinear attention in a sequence pooling layer, we implement a model “FESeq w/o Scale”, where the scale factor normalization is removed. According to Table 10, adding a scale factor to bilinear attention in a sequence pooling layer can enhance the recommendation performance.

Table 10. Ablation studies of models with and without scale factor in sequence pooling layer. The bold number denotes the best value in the column.

Datasets	Evaluation	FESeq	FESeq w/o Scale
Ele.me	AUC	0.6127	0.5926
	Logloss	0.0918	0.0926
Bundle	AUC	0.8912	0.8874
	Logloss	0.0117	0.0109

4.4. Hyperparameter Tuning

Next, we modify the parameter settings to gain insight into the proposed model.

4.4.1. Effect of Embedding Dimension

According to Figure 4, through the evaluation metrics of the AUC and Logloss, we can see that the proposed model performs the best when the embedding dimension d is 16. Since the number of heads in the sequence-refiner layer is eight, d should be a multiple of eight. A lower d will decrease the number of model parameters and limit the model’s expressive power. However, a higher d will lead to overfitting and thus a decline in performance.

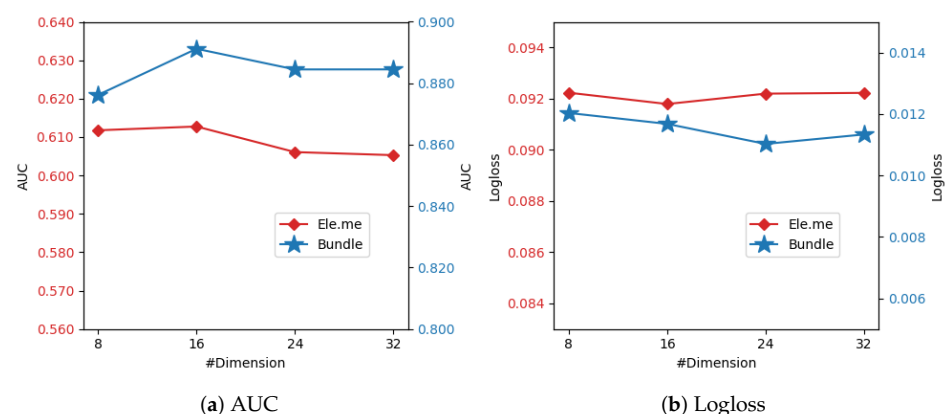


Figure 4. Effect of the embedding dimension on AUC and Logloss.

4.4.2. Effect of Feature Interaction Depth

Feature interaction stacks multiple layers to change the layer depth and learn high-order feature interactions. In our proposed model, we focus on how the input feature interaction layer depth and the target item feature interaction layer depth affect the performance.

The experiment results are concluded in Table 11. We observe that the model performs the best in terms of both the AUC and Logloss when both the input feature interaction layer depth and the target item feature interaction layer depth are two. Therefore, both the input feature interaction layers and the target item feature interaction layers are necessary. When the interacting layer depth is lower than two, the feature interaction can only extract low-order simple feature interactions, and the model performance is limited. When the interacting layer depth is three, the feature interaction can extract as high as eight-order fea-

ture interactions, but the recommendation performance tends to decrease, which indicates that extremely high-order feature combinations are not useful for CTR prediction.

Table 11. Effect of the input feature interaction layer depth and the target item feature interaction layer depth on the model performance. The bold number denotes the best value in the column.

Layers of Input FI	Layers of Target FI	Ele.me		Bundle	
		AUC	Logloss	AUC	Logloss
1	0	0.6088	0.0929	0.8836	0.0111
1	1	0.6092	0.0923	0.8817	0.0111
2	0	0.6142	0.0923	0.8796	0.0109
2	1	0.6065	0.0921	0.8851	0.0126
2	2	0.6127	0.0918	0.8912	0.0117
3	0	0.6068	0.0929	0.8879	0.0109
3	1	0.6078	0.0924	0.8857	0.0121
3	2	0.6053	0.0922	0.8845	0.0113
3	3	0.6091	0.0919	0.8818	0.0114

4.5. Visualization Results

In this section, we show the visualization results of the proposed model. First, the interacting layer uses the explainable AutoInt model. Therefore, we present the attention weight matrix heatmap of the last AutoInt layer to help explain the useful feature combinations that the AutoInt layer learns. Second, we observe the attention scores in the sequence pooling layer to explain how the sequence pooling layer learns the influence of different user behaviors on the target item prediction.

4.5.1. Attention Weight Matrix in Interacting Layer

Figures 5 and 6 represent the attention weight matrix in the last AutoInt layer of the input feature interaction layers on both datasets, respectively. The results are the average on the test samples. For the dataset Ele.me, we find that the interacting layer can learn meaningful feature combinations, such as $\langle rank_7, rank_30, rank_90 \rangle$, and $\langle avg_price, geohash12 \rangle$ (i.e., the red rectangle). It is reasonable that the item rank of the recent 7 days $rank_7$, the item rank of the recent 30 days $rank_30$, and the item rank of the recent 90 days $rank_90$ are correlated with each other. A user's average purchase price avg_price is also related to the user's geolocation $geohash12$. Likewise, for the dataset Bundle, the input feature interaction layers can also learn the relations between different user state features, such as $\langle tournament_rank, island_no \rangle$, $\langle life_time, island_no \rangle$, and $\langle friend_cnt, island_no \rangle$ (i.e., the red rectangle). They are explainable rules in the online game setting because a higher rank of the user's current island level $island_no$ indicates a higher rank in the tournament, a longer duration of the user using the game application, and a higher number of friends in the game platform.

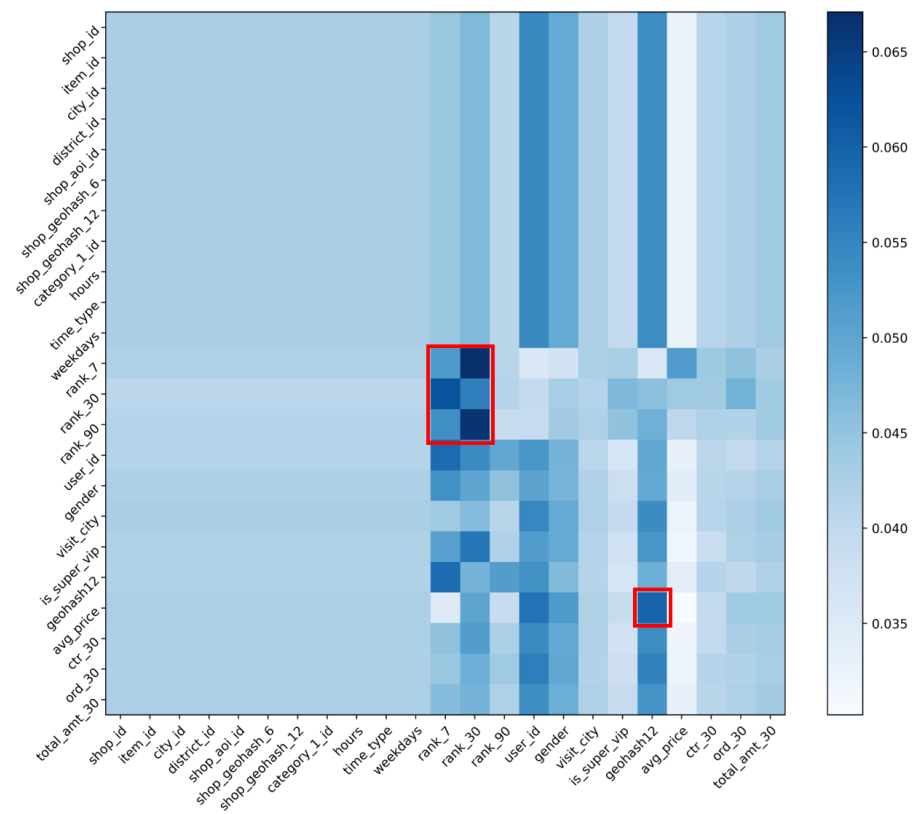


Figure 5. Visualization result of the attention weight matrix in the last input feature interaction layer on dataset Ele.me.

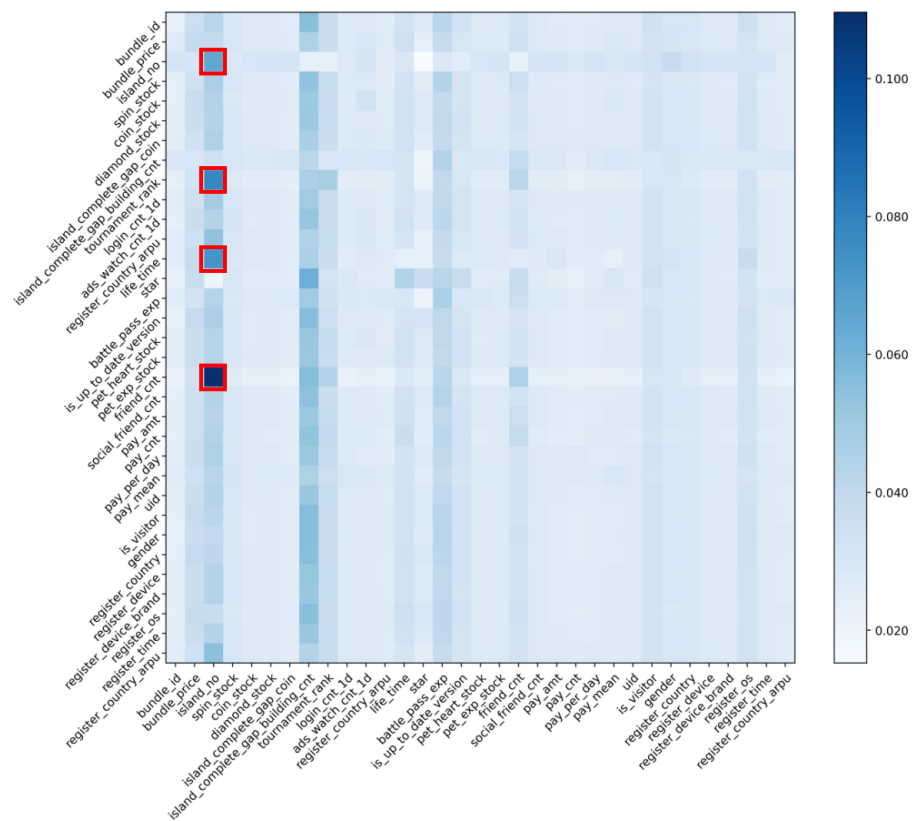


Figure 6. Visualization result of the attention weight matrix in the last input feature interaction layer on dataset Bundle.

4.5.2. Attention Scores in Sequence Pooling Layer

Figures 7 and 8 represent the attention scores in the sequence pooling layer on both datasets. The results are averaged on the test samples. We do interpolation smoothing on the attention score heatmaps. According to the heatmaps of the two datasets, we can obtain the following conclusions:

- More recent user behaviors obtain higher attention weights, which shows that more recent user behaviors tend to be more important for the target item prediction task.
- Compared with target pooling, the attention pooling in the heatmaps attends to both the current user behavior and a small portion of historical behaviors informative for the prediction task. Therefore, the visualization results can demonstrate the superiority of attention pooling to target pooling in the sequence pooling layer.

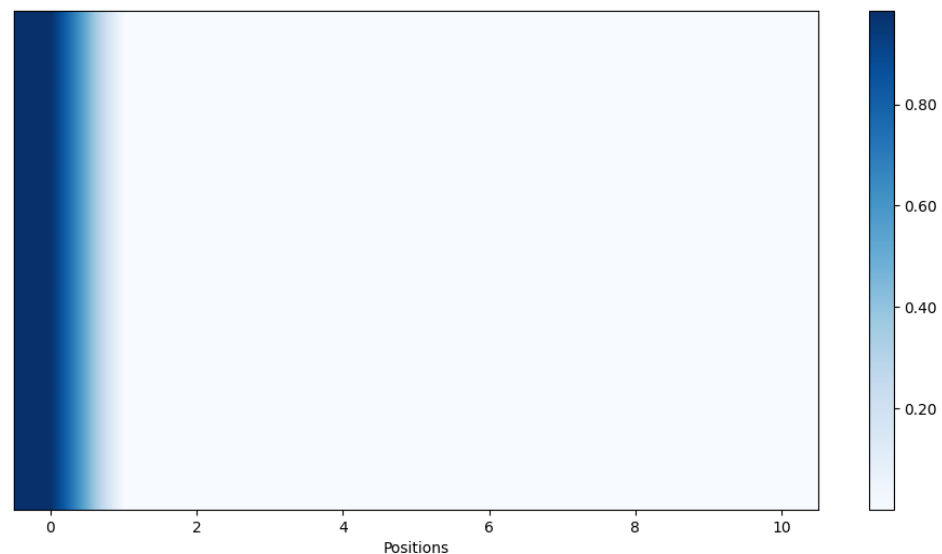


Figure 7. Visualization result of the attention scores in sequence pooling layer on dataset Ele.me.

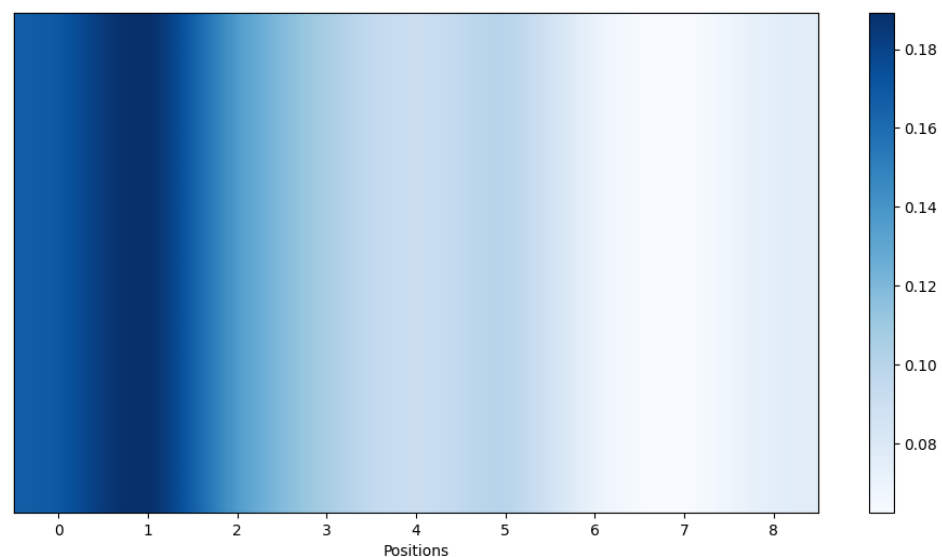


Figure 8. Visualization result of the attention scores in sequence pooling layer on dataset Bundle.

5. Conclusions

In this paper, we present a novel CTR prediction model that combines feature interactions with sequence recommendations in a cascading manner. Specifically, the model incorporates feature interactions as a feature engineering step into the Transformer se-

quence model to provide high-order features for the Transformer. Besides, positional embeddings and linear time interval embeddings are introduced to the Transformer layer to model the positional and time interval information of user behaviors. We also novelly pool the Transformer layer output with scaled bilinear attention that explicitly learns the relevance between the user's historical behaviors and the target item. Extensive experimental results on both public and private datasets show that our proposed model outperforms SOTA baselines on the CTR prediction task. Ablation studies confirm that the interacting layer can extract high-order information from both sequential and non-sequential features before the Transformer model. Besides, we observe the effectiveness of embedding positional and timestamp information and maintaining causality in the Transformer layer. The experiments further reveal the importance of attention pooling after the Transformer layer. Among the various attention mechanisms investigated, scaled bilinear attention is superior in modeling user-item relevance. Finally, our visualization results regarding feature interaction weights and attention-pooling weights provide good model explainability. In the future, we plan to replace the self-attention interacting layer with more recent SOTA feature interaction modules to enhance the model performance.

Author Contributions: Conceptualization, S.G.; methodology, Q.Y. and M.Z.; resources, M.Z.; software, Q.Y. and S.G.; writing—original draft preparation, Q.Y. and H.L.; writing—review and editing, H.L. and Y.L.; funding acquisition, Q.Y.; supervision, Y.L. and M.Z.; visualization, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (Grant No. 62071189).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Our data and source code are available at <https://github.com/liuhaozhe6788/FESeq>, accessed on 21 March 2024. The dataset Bundle is available from the corresponding author upon reasonable request, this data is not publicly available due to privacy.

Acknowledgments: The author wishes to extend sincere appreciation to Bideng Zhu for providing access to the Bundle dataset, which was crucial for this study. Gratitude is also owed to He Xingshun He for his expert assistance with data visualization, which greatly enhanced the understanding and presentation of the research findings.

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

Abbreviations

The following abbreviations are used in this manuscript:

MC	Markov chains
FM	Factorization machine
PNN	Product-based neural networks
GRU	Gated Recurrent Unit
AUGRU	GRU with an attentional update gate
AUC	Area under the curve
CNN	Convolutional Neural Network
CTR	Click-through rate
DIN	Deep interest network
DNN	Deep neural network
FESeq	Feature-Interaction-Enhanced Sequence Model
MLP	Multilayer perceptron
RNN	Recurrent Neural Network
SOTA	State-of-the-art

References

- Meng, Z.; Zhang, J.; Li, Y.; Li, J.; Zhu, T.; Sun, L. A General Method for Automatic Discovery of Powerful Interactions in Click-Through Rate Prediction. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, 11–15 July 2021; Association for Computing Machinery: New York, NY, USA, 2021; SIGIR '21; pp. 1298–1307. [\[CrossRef\]](#)
- Ouyang, W.; Zhang, X.; Ren, S.; Li, L.; Zhang, K.; Luo, J.; Liu, Z.; Du, Y. Learning Graph Meta Embeddings for Cold-Start Ads in Click-Through Rate Prediction. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, 11–15 July 2021; Association for Computing Machinery: New York, NY, USA, 2021; SIGIR '21; pp. 1157–1166. [\[CrossRef\]](#)
- Rendle, S. Factorization Machines. In Proceedings of the 2010 IEEE International Conference on Data Mining, Sydney, Australia, 13–17 December 2010; pp. 995–1000. [\[CrossRef\]](#)
- Qu, Y.; Fang, B.; Zhang, W.; Tang, R.; Niu, M.; Guo, H.; Yu, Y.; He, X. Product-Based Neural Networks for User Response Prediction over Multi-Field Categorical Data. *ACM Trans. Inf. Syst.* **2018**, *37*, 1–35. [\[CrossRef\]](#)
- Cheng, H.T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. Wide & Deep Learning for Recommender Systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016; Association for Computing Machinery: New York, NY, USA, 2016; DLRS 2016; pp. 7–10. [\[CrossRef\]](#)
- Guo, H.; TANG, R.; Ye, Y.; Li, Z.; He, X. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, Melbourne, Australia, 19–25 August 2017; pp. 1725–1731. [\[CrossRef\]](#)
- Wang, R.; Fu, B.; Fu, G.; Wang, M. Deep & Cross Network for Ad Click Predictions. In Proceedings of the ADKDD'17, Halifax, NS, Canada, 14 August 2017; Association for Computing Machinery: New York, NY, USA, 2017; ADKDD'17. [\[CrossRef\]](#)
- Mao, K.; Zhu, J.; Su, L.; Cai, G.; Li, Y.; Dong, Z. FinalMLP: An Enhanced Two-Stream MLP Model for CTR Prediction. *Proc. Aaaai Conf. Artif. Intell.* **2023**, *37*, 4552–4560. [\[CrossRef\]](#)
- Xiao, J.; Ye, H.; He, X.; Zhang, H.; Wu, F.; Chua, T.S. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, Melbourne, Australia, 19–25 August 2017; pp. 3119–3125. [\[CrossRef\]](#)
- Liu, B.; Zhu, C.; Li, G.; Zhang, W.; Lai, J.; Tang, R.; He, X.; Li, Z.; Yu, Y. AutoFIS: Automatic Feature Interaction Selection in Factorization Models for Click-Through Rate Prediction. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, CA, USA, 6–10 July 2020; Association for Computing Machinery: New York, NY, USA, 2020; KDD '20; pp. 2636–2645. [\[CrossRef\]](#)
- Song, W.; Shi, C.; Xiao, Z.; Duan, Z.; Xu, Y.; Zhang, M.; Tang, J. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; Association for Computing Machinery: New York, NY, USA, 2019; CIKM '19; pp. 1161–1170. [\[CrossRef\]](#)
- Yan, C.; Chen, Y.; Wan, Y.; Wang, P. Modeling Low- and High-Order Feature Interactions with FM and Self-Attention Network. *Appl. Intell.* **2021**, *51*, 3189–3201. [\[CrossRef\]](#)
- Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; Gai, K. Deep Interest Network for Click-Through Rate Prediction. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; Association for Computing Machinery: New York, NY, USA, 2018; KDD '18; pp. 1059–1068. [\[CrossRef\]](#)
- Wu, L.; Sun, P.; Fu, Y.; Hong, R.; Wang, X.; Wang, M. A Neural Influence Diffusion Model for Social Recommendation. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France, 21–25 July 2019; Association for Computing Machinery: New York, NY, USA, 2019; SIGIR'19; pp. 235–244. [\[CrossRef\]](#)
- An, M.; Wu, F.; Wu, C.; Zhang, K.; Liu, Z.; Xie, X. Neural News Recommendation with Long- and Short-term User Representations. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 336–345. [\[CrossRef\]](#)
- Garcin, F.; Dimitrakakis, C.; Faltings, B. Personalized News Recommendation with Context Trees. In Proceedings of the 7th ACM Conference on Recommender Systems, Hong Kong, China, 12–16 October 2013; Association for Computing Machinery: New York, NY, USA, 2013; RecSys '13; pp. 105–112. [\[CrossRef\]](#)
- He, R.; McAuley, J. Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM), Barcelona, Spain, 12–15 December 2016; pp. 191–200. [\[CrossRef\]](#)
- Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-based Recommendations with Recurrent Neural Networks. *arXiv* **2015**, arXiv:1511.06939.

19. 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, Marina Del Rey, CA, USA, 5–9 February 2018; Association for Computing Machinery: New York, NY, USA, 2018; WSDM '18; pp. 565–573. [\[CrossRef\]](#)
20. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.u.; Polosukhin, I. Attention is All you Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
21. 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.
22. Chen, Q.; Zhao, H.; Li, W.; Huang, P.; Ou, W. 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, Anchorage, Alaska, 5 August 2019; Association for Computing Machinery: New York, NY, USA, 2019; DLP-KDD '19. [\[CrossRef\]](#)
23. Yan, C.; Li, X.; Chen, Y.; Zhang, Y. JointCTR: A Joint CTR Prediction Framework Combining Feature Interaction and Sequential Behavior Learning. *Appl. Intell.* **2022**, *52*, 4701–4714. [\[CrossRef\]](#)
24. Lyu, Z.; Dong, Y.; Huo, C.; Ren, W. Deep Match to Rank Model for Personalized Click-Through Rate Prediction. *Proc. Aaai Conf. Artif. Intell.* **2020**, *34*, 156–163. [\[CrossRef\]](#)
25. Kim, J.H.; Jun, J.; Zhang, B.T. Bilinear Attention Networks. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018; NIPS'18; pp. 1571–1581.
26. Shan, Y.; Hoens, T.R.; Jiao, J.; Wang, H.; Yu, D.; Mao, J. Deep Crossing: Web-Scale Modeling without Manually Crafted Combinatorial Features. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; Association for Computing Machinery: New York, NY, USA, 2016; KDD '16; pp. 255–262. [\[CrossRef\]](#)
27. He, X.; Chua, T.S. Neural Factorization Machines for Sparse Predictive Analytics. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; Association for Computing Machinery: New York, NY, USA, 2017; SIGIR '17; pp. 355–364. [\[CrossRef\]](#)
28. Lian, J.; Zhou, X.; Zhang, F.; Chen, Z.; Xie, X.; Sun, G. XDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; Association for Computing Machinery: New York, NY, USA, 2018; KDD '18; pp. 1754–1763. [\[CrossRef\]](#)
29. Wang, R.; Shivanna, R.; Cheng, D.; Jain, S.; Lin, D.; Hong, L.; Chi, E. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-Scale Learning to Rank Systems. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; Association for Computing Machinery: New York, NY, USA, 2021; WWW '21; pp. 1785–1797. [\[CrossRef\]](#)
30. Xu, Y.; Zhu, Y.; Yu, F.; Liu, Q.; Wu, S. Disentangled Self-Attentive Neural Networks for Click-Through Rate Prediction. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, Virtual Event, Australia, 1–5 November 2021; Association for Computing Machinery: New York, NY, USA, 2021; CIKM '21; pp. 3553–3557. [\[CrossRef\]](#)
31. Zhu, J.; Jia, Q.; Cai, G.; Dai, Q.; Li, J.; Dong, Z.; Tang, R.; Zhang, R. FINAL: Factorized Interaction Layer for CTR Prediction. In Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, Taipei, Taiwan, 23–27 July 2023; Association for Computing Machinery: New York, NY, USA, 2023; SIGIR '23; pp. 2006–2010. [\[CrossRef\]](#)
32. Yap, G.E.; Li, X.L.; Yu, P.S. Effective Next-Items Recommendation via Personalized Sequential Pattern Mining. In Proceedings of the 17th International Conference on Database Systems for Advanced Applications—Volume Part II, Busan, Republic of Korea, 15–19 April 2012; Springer: Berlin/Heidelberg, Germany, 2012; DASFAA'12; pp. 48–64. [\[CrossRef\]](#)
33. Wu, C.Y.; Ahmed, A.; Beutel, A.; Smola, A.J.; Jing, H. Recurrent Recommender Networks. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, Cambridge, UK, 6–10 February 2017; Association for Computing Machinery: New York, NY, USA, 2017; WSDM '17; pp. 495–503. [\[CrossRef\]](#)
34. Quadrana, M.; Karatzoglou, A.; Hidasi, B.; Cremonesi, P. Personalizing Session-Based Recommendations with Hierarchical Recurrent Neural Networks. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; Association for Computing Machinery: New York, NY, USA, 2017; RecSys '17; pp. 130–137. [\[CrossRef\]](#)
35. Yuan, F.; Karatzoglou, A.; Arapakis, I.; Jose, J.M.; He, X. A Simple Convolutional Generative Network for Next Item Recommendation. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, Melbourne, Australia, 11–15 February 2019; Association for Computing Machinery: New York, NY, USA, 2019; WSDM '19; pp. 582–590. [\[CrossRef\]](#)
36. de Souza Pereira Moreira, G.; Rabhi, S.; Lee, J.M.; Ak, R.; Oldridge, E. Transformers4Rec: Bridging the Gap between NLP and Sequential / Session-Based Recommendation. In Proceedings of the 15th ACM Conference on Recommender Systems, Amsterdam, The Netherlands, 27 September–1 October 2021; Association for Computing Machinery: New York, NY, USA, 2021; RecSys '21; pp. 143–153. [\[CrossRef\]](#)
37. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
38. Zhou, C.; Bai, J.; Song, J.; Liu, X.; Zhao, Z.; Chen, X.; Gao, J. ATRank: An Attention-Based User Behavior Modeling Framework for Recommendation. *arXiv* **2017**, arXiv:1711.06632.
39. 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; Association for Computing Machinery: New York, NY, USA, 2019; CIKM '19; pp. 1441–1450. [\[CrossRef\]](#)

40. Wu, L.; Li, S.; Hsieh, C.J.; Sharpnack, J. SSE-PT: Sequential Recommendation Via Personalized Transformer. In Proceedings of the 14th ACM Conference on Recommender Systems, Virtual Event, Brazil, 22–26 September 2020; Association for Computing Machinery: New York, NY, USA, 2020; RecSys '20; pp. 328–337. [[CrossRef](#)]
41. 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; Association for Computing Machinery: New York, NY, USA, 2020; WSDM '20; pp. 322–330. [[CrossRef](#)]
42. Xiao, Z.; Yang, L.; Jiang, W.; Wei, Y.; Hu, Y.; Wang, H. Deep Multi-Interest Network for Click-through Rate Prediction. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, Virtual Event, Ireland, 19–23 October 2020; Association for Computing Machinery: New York, NY, USA, 2020; CIKM '20; pp. 2265–2268. [[CrossRef](#)]
43. Yang, Y.; Xu, B.; Shen, S.; Shen, F.; Zhao, J. Operation-aware Neural Networks for user response prediction. *Neural Netw.* **2020**, *121*, 161–168. [[CrossRef](#)] [[PubMed](#)]
44. Zhou, G.; Mou, N.; Fan, Y.; Pi, Q.; Bian, W.; Zhou, C.; Zhu, X.; Gai, K. Deep Interest Evolution Network for Click-through Rate Prediction. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; AAAI'19/IAAI'19/EAAI'19. [[CrossRef](#)]
45. Zhu, J.; Liu, J.; Yang, S.; Zhang, Q.; He, X. Open Benchmarking for Click-Through Rate Prediction. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, Virtual Event, Australia, 1–5 November 2021; Association for Computing Machinery: New York, NY, USA, 2021; CIKM '21; pp. 2759–2769. [[CrossRef](#)]
46. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015; Conference Track Proceedings; 2015.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.