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

Haozhe Liu[1], Yushi Li[1], Siao Guo[1], Ming Zhu[1*], Bideng Zhu[2]

[1]School of Electronic Information and Communications, Huazhong University of Science and Technology (HUST), Wuhan, China.
[2]FOREVER9, Shenzhen, China.

*Corresponding author(s). E-mail(s): zhuming@hust.edu.cn;
Contributing authors: hiliuhaozhe@163.com; liyushi@hust.edu.cn;
g553690204@163.com; nicolas@aladinfun.com;

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. Our data and source code are available at https://github.com/liuhaozhe6788/FESeq.

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 Meng et al (2021); Ouyang et al (2021). 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 of feature interaction methods Rendle (2010); Qu et al (2018) adopted the inner product operation of individual features, which managed to extract simple low-order feature crosses. Wide & Deep Cheng et al (2016) 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 Guo et al (2017), DCNWang et al (2017) and FinalMLPMao et al (2023). Recent studies achieve feature interaction through the attention network Xiao et al (2017); Liu et al (2020) and self-attention mechanism Song et al (2019). 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 Yan et al (2021). 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 Zhou et al (2018), media platforms Wu et al (2019), and news recommendations An et al (2019). Previous sequential recommenders focused mainly on traditional methods, such as Markov chains (MC) and hidden Markov models Garcin et al (2013); He and McAuley (2016), which calculate the user state transition probability based on the previous N-order interactions. More sophisticated deep learning methods, such as Recurrent Neural Networks (RNN) Hidasi et al (2015) and Convolutional Neural Networks (CNN) Tang and Wang (2018), then emerged

and have been vigorously applied in this field. A multitude of sequential recommendation research nowadays focuses on the Transformer architecture Vaswani et al (2017). Self-attention mechanism in Transformer calculates the relevance of different user behavior pairs to capture the long-term dependencies of the user's sequence data Kang and McAuley (2018); Chen et al (2019), 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 Yan et al (2022) manages to combine feature interaction models and a sequential recommendation model in a parallel manner. Yet, the parallel architecture treats feature interaction and sequential recommendation separately, making it suboptimal. Actually, two limitations can be found in the parallel architecture:

- First, 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 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 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 **F**eature Interaction-**E**nhanced **Seq**uence 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 Lyu et al (2020) and the attention-based sequence pooling in DIN Zhou et al (2018), we design a novel attention-based sequence pooling layer that aggregates the Transformer output sequence through attention. Using scaled bilinear attention Kim et al (2018), 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 to 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 factorization machine (FM) Rendle (2010) and product-based neural networks (PNN) Qu et al (2018). FM projects the original features into low-dimensional vectors and calculates the inner product to learn second-order feature interactions. 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 FM and the product layer cannot learn higher-order feature interactions. Other than explicit inner product operations, Deep Crossing Shan et al (2016) implicitly learns high-order feature crosses through DNN with residual units, but the model cannot learn low-order linear feature interactions. NFM He and Chua (2017) stacks multiple DNN layers after the FM model and achieves more expressive power in learning than the FM model. AFM Xiao et al (2017) combines FM with an attention network to aggregate the output of FM.

Other than DNN, innovations in feature interaction layer architectures have improved the scalability of feature interaction models and achieved effective higher-order feature interactions. DCN Wang et al (2017) designs a cross network to perform efficient feature interaction in linear complexity and allows the growth of stacking layer depth. xDeepFM Lian et al (2018) proposed a novel Compressed Interaction Network (CIN) that combines outer-product and convolution pooling operations to learn explicit high-order feature interactions. Song et al (2019) proposed AutoInt, which adopts a residual self-attention network that automatically identifies high-order feature interactions and ensures exponential degree growth. DCN V2 Wang et al (2021) improves the cross network in DCN to learn both point-wise and vector-wise feature interactions. In Xu et al (2021), a self-attention network with a disentangled self-attention operation decouples the unary attention weights from the pairwise ones. FINAL Zhu et al (2023) achieves exponential polynomial degree growth through layers of multiplicative interaction.

Since AutoInt 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 Yap et al (2012) 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 He and McAuley (2016) is a high-order MC model that learns local dynamics in the user sequence.

Previous deep learning methods include RNN and CNN. RNN-based sequence models can capture long-term information from the past sequence and predict future items. Hidasi et al (2015) applies Gated Recurrent Unit (GRU) to sequential recommendation and uses ranking loss to train GRU. Wu et al (2017) uses Long Short-Term Memory to model the temporal dynamics of user behaviors and item features. Quadrana et al (2017) proposed 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. CNN can model the sequential patterns using convolutional filters. Caser Tang and Wang (2018) transforms the user action sequence into an "image" and processes the image data using horizontal and vertical convolutional filters to model point-level and union-level patterns respectively. Yuan et al (2019) 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 recommendation can be attributed to enhancements in model architectures and pretraining techniques from the Natural Language Processing (NLP) field de Souza Pereira Moreira et al (2021), such as attention network and Transformer. The idea of attention network architecture Bahdanau et al (2014)

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 GRU to extract the user's interest and GRU with an attentional update gate (AUGRU) to learn the evolution of the user's interest. DMR Lyu et al (2020) 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 recommendation. For example, AtRank Zhou et al (2017) 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 Kang and McAuley (2018) utilizes a Transformer decoder to predict the next item based on the user's historical actions. BST Chen et al (2019) captures long-term dependencies from the user's historical behaviors 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 Sun et al (2019) adopts the vanilla BERT model and a masked item prediction training task to predict the next item. SSE-PT Wu et al (2020) novelly introduces user embeddings to the vanilla Transformer and adds a stochastic shared embedding regularization term to avoid overfitting. TiSASRec Li et al (2020) proposes the relative time interval embedding and designs a self-attention mechanism that combines absolute position and relative time interval information. DMIN Xiao et al (2020) 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 **F**eature Interaction-**E**nhanced **Seq**uential model (FESeq). The overall architecture of the model is shown in Figure 1, which consists of five components:

- *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 (user profile and context) and sequential features (positions, time intervals, user behaviors, and interacted items). The target item features are all the last element of the interacted item feature sequences.

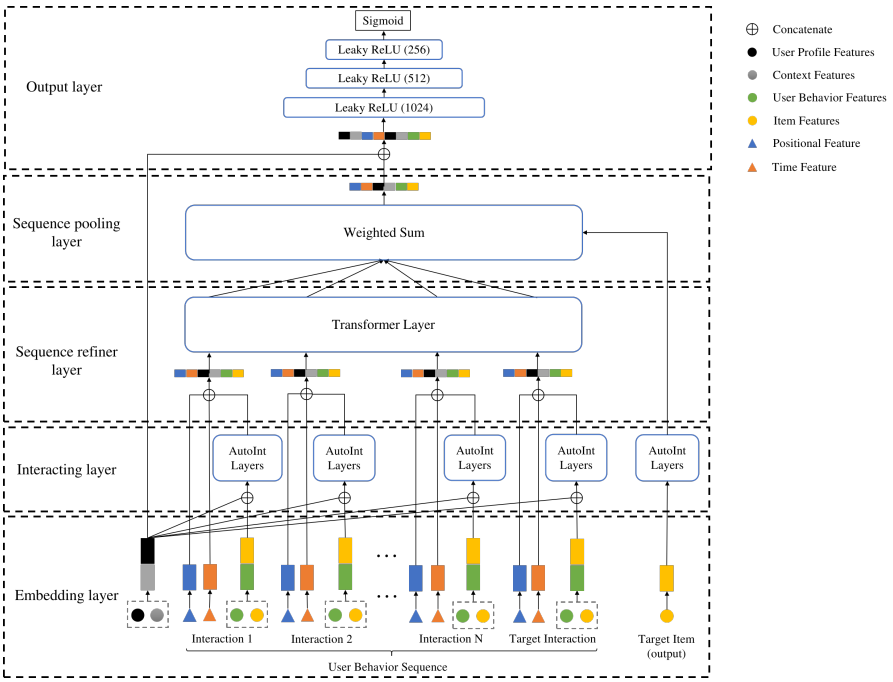


Fig. 1 The architecture of FESeq. The embedding layer embeds the input features into a low-dimensional fixed-length embedding. Then, the interacting layer extracts high-order feature interactions from all feature fields. The sequence refiner layer further refines the user's behavior sequence through 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.

- *Interacting Layer*: It is proposed to extract high-order feature interactions from all the original feature fields using the self-attention mechanism. The details of this layer and how we view feature interaction as a form of 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 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 Song et al (2019), 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 $E_{ns} = (e_{ns}, e_{ns}, ..., e_{ns})$, 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 interacting layer. We choose self-attention layers in AutoInt Song et al (2019) as the core part of the interacting layer. The i-th AutoInt layer based on self-attention requires the following operations:

$$Q = E_{in}W^Q, K = E_{in}W^K, V = E_{in}W^V \tag{1}$$

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

$$E^i_{in} = ReLU(\tilde{E}^{i-1}_{in} + E^{i-1}_{in}) \tag{3}$$

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, if the number of AutoInt layers for the input features E_{in} is n, the output of the last AutoInt layer E^n_{in} 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^n_{in} 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 Eq. 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

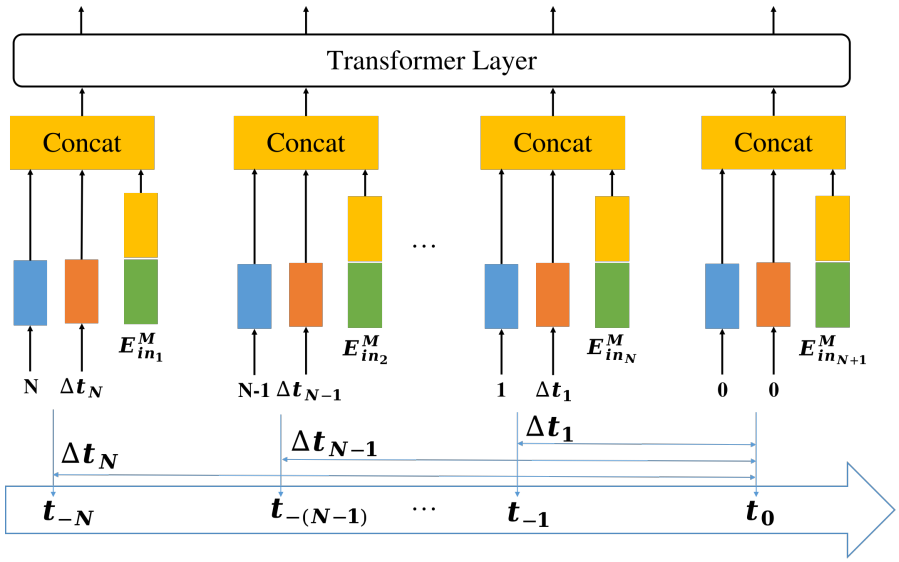


Fig. 2 Illustration of the position and time-aware sequence refiner layer, where both positional and linear time embeddings are concatenated with the user behavior sequence.

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.

Positional Embedding: Since self-attention models do not contain any position-aware structures, such as the recurrent structure in RNN and convolution filter operations in CNN, 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 influences 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 Li et al (2020), 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:

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

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 Eq. 2, except that we use multiple self-attention heads. The process is then described as:

$$S = MH(\hat{E}) = \text{concat}(head_1, head_2, ..., head_h)W^H \tag{5}$$

$$head_i = \text{Attention}(\hat{E}W^Q, \hat{E}W^K, \hat{E}W^V) \tag{6}$$

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 sequence refiner layer leads to 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 Chen et al (2019), 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 FFN is LeakyReLU. The output of the self-attention layer S' is represented as:

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

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

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

where $W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}$ are learnable network parameters. We only use one layer of Transformer based on the experimental results in Chen et al (2019)

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* (concatenation of all behaviors in a sequence Chen et al (2019)), *target pooling* (target behavior as the pooling result), *mean pooling* (mean of all behaviors in a sequence), and attention-based weighted-sum pooling Lyu et al (2020). 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 using the attention scores. 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.

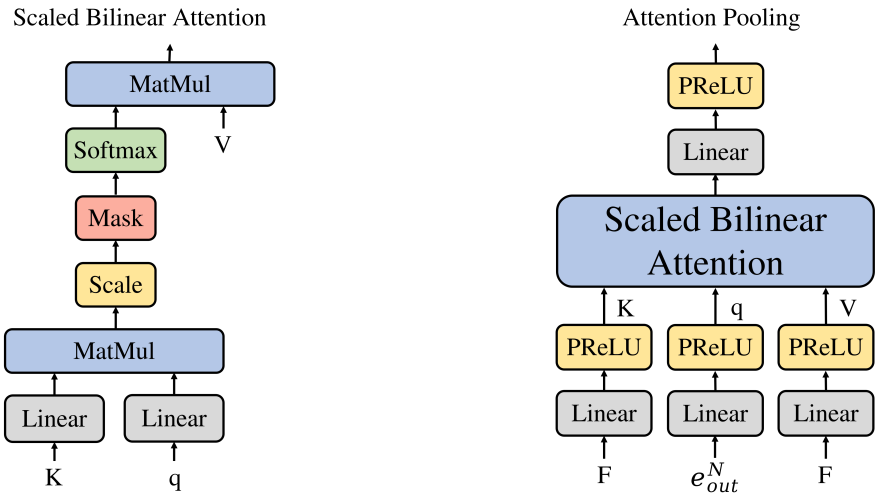


Fig. 3 (left) Scaled Bilinear Attention. (right) Attention Pooling is based on scaled bilinear attention, where key and value are obtained from the sequence refiner layer output F and the query is generated from the target item embedding e^N_{out}.

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

$$q = W^q e^N_{out}, K = FW^K, V = FW^V \tag{9}$$

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:

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

$$\alpha_t = g(q, K_t) \tag{11}$$

where function $g(\cdot,\cdot)$ represents the equation of the attention score α_t. K_t and V_t represent key and value of the t-th user behavior. 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.

Next, we will introduce scaled linear attention that calculates attention scores in attention pooling. In Kim et al (2018), bilinear attention networks (BAN) are proposed to solve the multimodal 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 query and key and model the relationship between the user and the target item. We add a scale factor $\sqrt{d_h}$ to avoid large values. Scaled bilinear attention is calculated as follows:

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

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:

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

where D represents all 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[1] 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 2021-11-1 to 2021-12-30. 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 datasets are shown in Table 1. The sparsity of the dataset is defined as $1 - \frac{\text{Interactions}}{\text{Users} \times \text{Items}}$, where Interactions is the number of non-empty values in a user-item interaction matrix. Ele.me is a highly sparse dataset, while Bundle is an extremely dense dataset.

Table 1 Statistics of the Datasets

Dataset	# Users	# Items	# Samples	# Sequence Fields	# Non-sequence Fields	# Sparsity
Ele.me	1013338	449440	1228589	14	9	over 99.99%
Bundle	118931	11	4193418	24	9	73.69%

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 interaction and sequential recommendation. The following models are representative feature interaction baselines:

[1] https://tianchi.aliyun.com/dataset/131047

- **Wide & Deep** Cheng et al (2016): Uses a parallelization of a linear model and a DNN model to extract both the low-order and the high-order feature combinations.
- **IPNN** (Qu et al, 2018): Learns two-order feature interactions based on inner product and high-order feature interactions by stacking multiple layers of the DNN network.
- **DeepFM** Guo et al (2017): An end-to-end model similar to Wide & 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** Wang et al (2017): Consists of cross network layers and DNN network in parallel.
- **AutoInt** Song et al (2019): It stacks multi-head self-attention layers to extract high-order feature interactions.
- **ONN** Yang et al (2020): Proposes operation-aware embedding layers to learn diverse representations of each feature field for different operations.
- **DCNv2** Wang et al (2021): Improves the cross network in DCN, enabling both point-wise and vector-wise feature interactions.
- **DESTINE** Xu et al (2021): Adopts disentangled self-attention layers to extract high-order feature interactions.
- **DualMLP** Mao et al (2023): A well-tuned two-stream MLP model.
- **FinalMLP** Mao et al (2023): An enhanced two-stream MLP model combined with stream-specific feature gating and bilinear interaction fusion layers.
- **FINAL** Zhu et al (2023): 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** Zhou et al (2018): Introduces the attention mechanism by adaptively attending to historical user behaviors to learn the user's interest.
- **SASRec** Kang and McAuley (2018): 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** Zhou et al (2019): Based on AUGRU to capture the evolution of the user interest relevant to the target item.
- **BST** Chen et al (2019): Utilizes the Transformer encoder layer to capture the sequence information of the user's behaviors.
- **TiSASRec** Li et al (2020): 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** Lyu et al (2020): 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 using the inner product.

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

JointCTR Yan et al (2022): 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 Zhu et al (2021). We train the model with the Adam optimizerKingma and Ba (2015). 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 8 for the sequence refiner layer and 1 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 using AUC and Logloss. AUC is formulated as:

$$AUC = \frac{\sum_{i \in I^+} \sum_{j \in I^-} \delta(\hat{p_i}, \hat{p_j})}{|I^+|\,|I^-|} \tag{14}$$

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:

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

Logloss is formulated in Eq.13 when the regularization parameter λ is zero. In general, higher AUC and lower Logloss indicate 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 & Deep	0.5489	0.0950	0.8761	<u>0.0111</u>
	PNN	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 get 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, 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 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 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 CTR prediction performance (+1.46% in AUC and -0.32% in Logloss on dataset Ele.me; +0.30% in AUC on dataset Bundle).
- The CTR prediction performance on dataset Bundle is significantly better than dataset Ele.me in the proposed model. The primary reason is that dataset Bundle contains more user behavior feature fields and a lower sparsity than 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 dataset Bundle represents whether the user purchased the game bundle, while that in 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 dataset Bundle than on 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 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

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

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 non-sequential features, the input feature interaction layers can only perform feature interaction on sequential feature fields, which hinders the learning capability of 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 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

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 get 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 dataset Ele.me and 8 on dataset Bundle), which leads to insignificant time interval differences of different user behaviors for the same user. Therefore, compared with positional embedding, linear time interval embedding has a smaller impact on the model performance.

4.3.4 Influence of causal masks

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**

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 for learning 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 sequence refiner layer. The performance comparison in Table 7 shows that using a causal Transformer in sequence refiner layer can further improve the recommendation performance.

4.3.5 Influence of sequence pooling methods

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

According to Table 8, we compare the performance of the proposed model with different sequence pooling methods. "Target Pooling" represents the proposed model with target pooing 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.

4.3.6 Influence of attention score calculation methods

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

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

- **Scaled dot product** In Vaswani et al (2017), scaled dot product calculates the similarity between query q and key K_t:

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

  where the scale factor $\sqrt{d_h}$ can avoid excessively large dot product results. Scaled dot product can simply and directly represent the user-item similarity.
- **DIN attention** In DIN model Zhou et al (2018), attention scores are calculated via MLP networks with Dice activation. In our model, the inputs of MLP are query q, key K_t, and the outer product of query and key. MLP networks then reduce the input data to attention scores. DIN attention is calculated as:

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

Compared with 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. 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

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 attention score calculation. The results verify that using scaled bilinear attention for attention score calculation in 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 scaled dot-product fails to learn complex user-item relations without additional learning parameters.

4.3.7 Influence of scale factor

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**

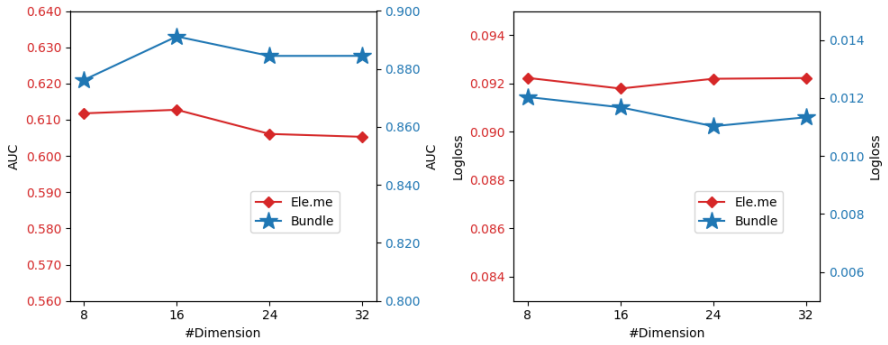
Related works Li et al (2020); Vaswani et al (2017) show that a scale factor should normalize the attention scores in the dot product of the self-attention layer to avoid high dot product values. To verify the influence of the scale factor on bilinear attention in sequence pooling layer, we implement 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 sequence pooling layer can enhance the recommendation performance.

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 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 8, d should be a multiple of 8. 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.



(a) AUC (b) Logloss

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

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

4.4.2 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 AUC and Logloss when both the input feature interaction layer depth and the target item feature interaction layer depth are 2. Therefore, both the input feature interaction layers and the target item feature interaction layers are necessary. When interacting layer depth is lower than 2, feature interaction can only extract low-order simple feature interactions, and the model performance is limited. When the interacting layer depth is 3, feature interaction can extract as high as eight-order feature interactions, but the recommendation performance tends to decrease, which indicates that extremely high-order feature combinations are not useful for CTR prediction.

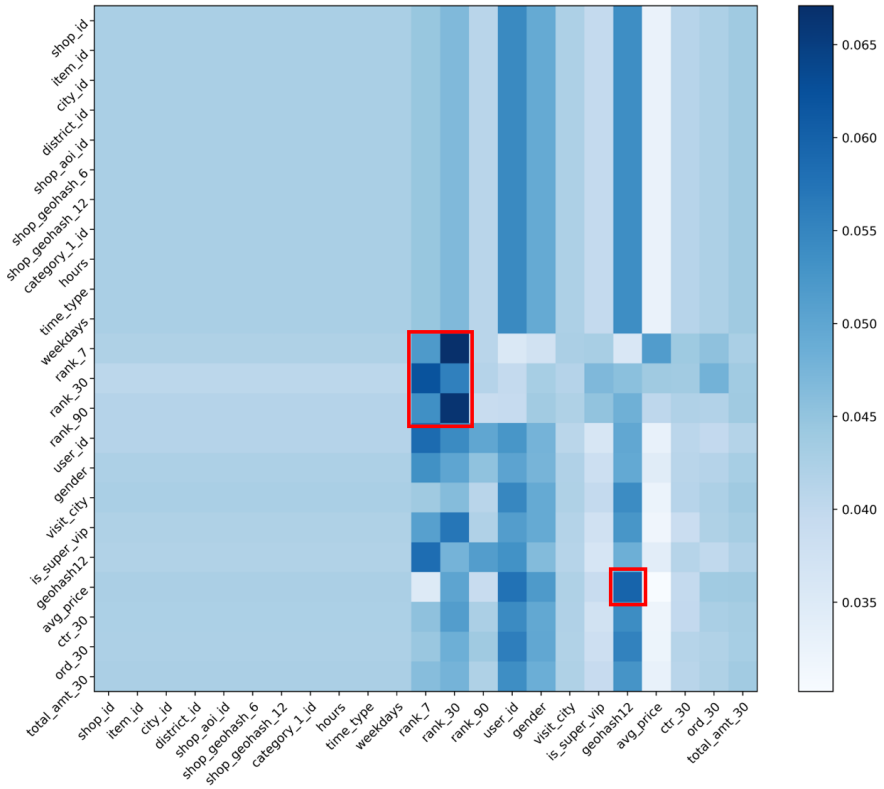


Fig. 5 Visualization result of the attention weight matrix in the last input feature interaction layer on dataset Ele.me.

4.5 Visualization results

In this section, we show the visualization results of the proposed model. First, 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 sequence pooling layer to explain how sequence pooling layer learns the influence of different user behaviors on the target item prediction.

4.5.1 Attention weight matrix in interacting layer

Figure 5 and Figure 6 represent the attention weight matrix in the last AutoInt layer of input feature interaction layers on both datasets respectively. The results are the average on the test samples. For dataset Ele.me, we find that interacting layer can learn meaningful feature combinations, such as $< rank\_7, rank\_30, rank\_90 >$ and $< avg\_price, geohash12 >$ (i.e., 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

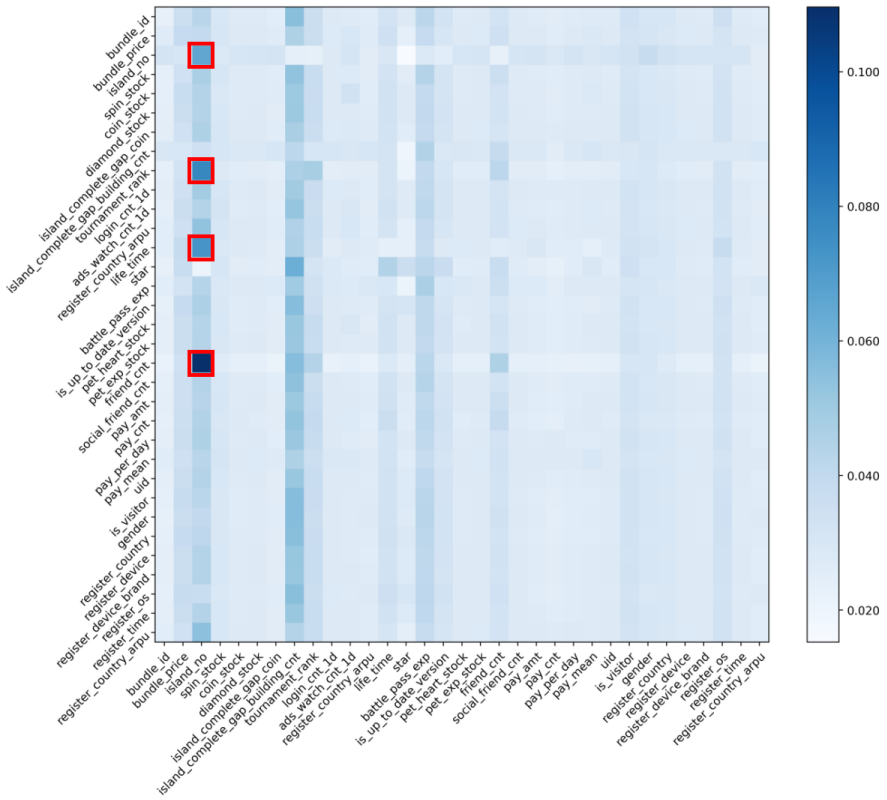


Fig. 6 Visualization result of the attention weight matrix in the last input feature interaction layer on dataset Ele.me.

purchase price *avg_price* is also related to the user's geolocation *geohash*12. Likewise, for dataset Bundle, input feature interaction layers can also learn the relations between different user state features, such as $< tournament\_rank, island\_no >$, $< life\_time, island\_no >$, and $< friend\_cnt, island\_no >$ (i.e., 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.

4.5.2 Attention scores in sequence pooling layer

Figure 7 and Figure 8 represent the attention scores in 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 get the following conclusions:

- More recent user behaviors get higher attention weights, which shows that more recent user behaviors tend to be more important for the target item prediction task.

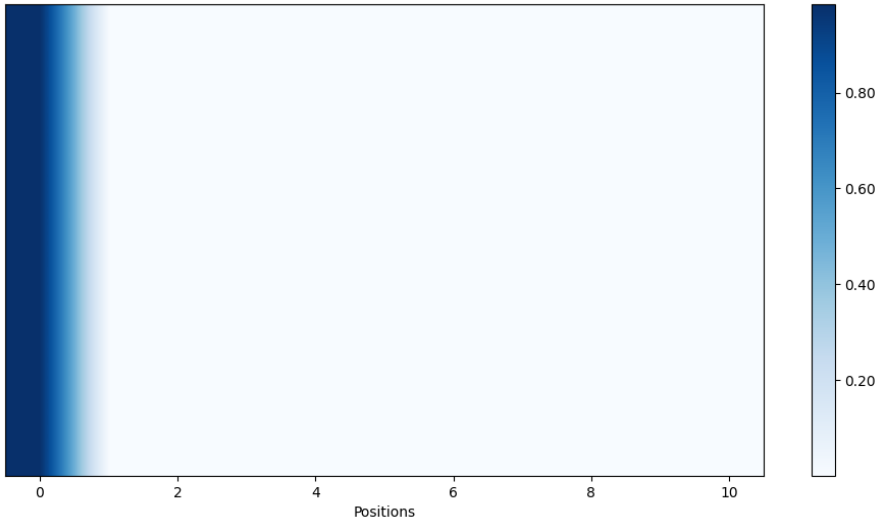


Fig. 7 Visualization result of the attention scores in sequence pooling layer on dataset Ele.me.

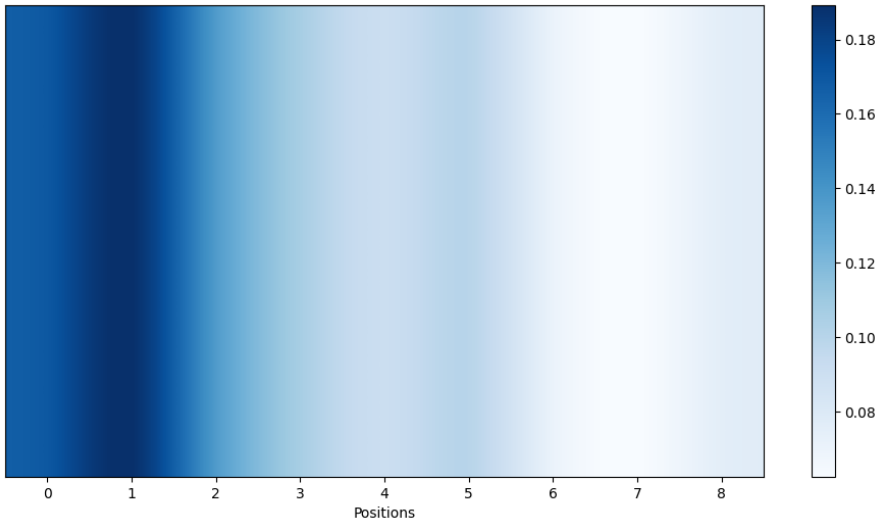


Fig. 8 Visualization result of the attention scores in sequence pooling layer on dataset Bundle.

- The heatmap of dataset Bundle has a larger blue region than that of dataset Ele.me. It implies that dense datasets (e.g. Bundle) need a larger scope of the user's historical behaviors than sparse datasets (e.g. Ele.me).
- 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 to the prediction task. Therefore, the visualization results can demonstrate the superiority of attention pooling to target pooling in sequence pooling layer.

5 Conclusion

In this paper, we present a novel CTR prediction model that combines feature interaction with sequence recommendation in a cascading manner. Specifically, the model incorporates feature interaction as a feature engineering step into the Transformer sequence 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 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 dense and sparse datasets show that our proposed model outperforms SOTA baselines on the CTR prediction task. In the future, we plan to replace the self-attention interacting layer with more recent SOTA feature interaction modules to enhance the model performance.

Declarations

- Funding
  The authors did not receive support from any organization for the submitted work.
- Conflict of interest
  The authors have no conflicts of interest to declare that are relevant to the content of this article.
- Ethics approval
  Not applicable.
- Consent to participate
  The authors consent to participate in the study.
- Consent for publication
  The authors approve the work to be published.
- Availability of data and materials
  Dataset Bundle is available from the corresponding author on reasonable request.
- Code availability
  The source code is available for public access and replication. The code repository can be found at https://github.com/liuhaozhe6788/FESeq. We encourage researchers and interested parties to utilize the code to replicate and build upon the authors' findings.
- Authors' contributions
  Haozhe Liu: Conceptualization; Methodology; Data collection; Writing - original draft preparation; Writing - review and editing; Software. Yushi Li: Conceptualization; Methodology; Writing - review and editing. Siao Guo: Conceptualization; Writing - review and editing. Ming Zhu: Conceptualization; Writing - review and editing. Bideng Zhu: Investigation; Data collection.

References

An M, Wu F, Wu C, et al (2019) Neural news recommendation with long- and short-term user representations. In: Proceedings of the 57th Annual Meeting of

the Association for Computational Linguistics. Association for Computational Linguistics, Florence, Italy, pp 336–345, https://doi.org/10.18653/v1/P19-1033

Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:14090473

Chen Q, Zhao H, Li W, et al (2019) Behavior sequence transformer for e-commerce recommendation in alibaba. In: Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data. Association for Computing Machinery, New York, NY, USA, DLP-KDD '19, https://doi.org/10.1145/3326937.3341261

Cheng HT, Koc L, Harmsen J, et al (2016) Wide & deep learning for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. Association for Computing Machinery, New York, NY, USA, DLRS 2016, p 7–10, https://doi.org/10.1145/2988450.2988454

Garcin F, Dimitrakakis C, Faltings B (2013) Personalized news recommendation with context trees. In: Proceedings of the 7th ACM Conference on Recommender Systems. Association for Computing Machinery, New York, NY, USA, RecSys '13, p 105–112, https://doi.org/10.1145/2507157.2507166

Guo H, TANG R, Ye Y, et al (2017) Deepfm: A factorization-machine based neural network for ctr prediction. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pp 1725–1731, https://doi.org/10.24963/ijcai.2017/239

He R, McAuley J (2016) Fusing similarity models with markov chains for sparse sequential recommendation. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), pp 191–200, https://doi.org/10.1109/ICDM.2016.0030

He X, Chua TS (2017) Neural factorization machines for sparse predictive analytics. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. Association for Computing Machinery, New York, NY, USA, SIGIR '17, p 355–364, https://doi.org/10.1145/3077136.3080777

Hidasi B, Karatzoglou A, Baltrunas L, et al (2015) Session-based recommendations with recurrent neural networks. URL http://arxiv.org/abs/1511.06939

Kang WC, McAuley J (2018) Self-attentive sequential recommendation. In: 2018 IEEE international conference on data mining (ICDM), IEEE, pp 197–206

Kim JH, Jun J, Zhang BT (2018) Bilinear attention networks. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, NIPS'18, p 1571–1581

Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: Bengio Y, LeCun Y (eds) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, URL http://arxiv.org/abs/1412.6980

Li J, Wang Y, McAuley J (2020) Time interval aware self-attention for sequential recommendation. In: Proceedings of the 13th International Conference on Web Search and Data Mining. Association for Computing Machinery, New York, NY, USA, WSDM '20, p 322–330, https://doi.org/10.1145/3336191.3371786

Lian J, Zhou X, Zhang F, et al (2018) Xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Association for Computing Machinery, New York, NY, USA, KDD '18, p 1754–1763, https://doi.org/10.1145/3219819.3220023

Liu B, Zhu C, Li G, et al (2020) 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. Association for Computing Machinery, New York, NY, USA, KDD '20, p 2636–2645, https://doi.org/10.1145/3394486.3403314

Lyu Z, Dong Y, Huo C, et al (2020) Deep match to rank model for personalized click-through rate prediction. Proceedings of the AAAI Conference on Artificial Intelligence 34(01):156–163. https://doi.org/10.1609/aaai.v34i01.5346

Mao K, Zhu J, Su L, et al (2023) Finalmlp: An enhanced two-stream mlp model for ctr prediction. Proceedings of the AAAI Conference on Artificial Intelligence 37(4):4552–4560. https://doi.org/10.1609/aaai.v37i4.25577

Meng Z, Zhang J, Li Y, et al (2021) 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. Association for Computing Machinery, New York, NY, USA, SIGIR '21, p 1298–1307, https://doi.org/10.1145/3404835.3462842, URL https://doi.org/10.1145/3404835.3462842

Ouyang W, Zhang X, Ren S, et al (2021) 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. Association for Computing Machinery, New York, NY, USA, SIGIR '21, p 1157–1166, https://doi.org/10.1145/3404835.3462879, URL https://doi.org/10.1145/3404835.3462879

Qu Y, Fang B, Zhang W, et al (2018) Product-based neural networks for user response prediction over multi-field categorical data. ACM Trans Inf Syst 37(1). https://doi.org/10.1145/3233770

Quadrana M, Karatzoglou A, Hidasi B, et al (2017) Personalizing session-based recommendations with hierarchical recurrent neural networks. In: Proceedings of the Eleventh ACM Conference on Recommender Systems. Association for Computing Machinery, New York, NY, USA, RecSys 17, p 130–137, https://doi.org/10.1145/3109859.3109896

Rendle S (2010) Factorization machines. In: 2010 IEEE International Conference on Data Mining, pp 995–1000, https://doi.org/10.1109/ICDM.2010.127

Shan Y, Hoens TR, Jiao J, et al (2016) 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. Association for Computing Machinery, New York, NY, USA, KDD '16, p 255–262, https://doi.org/10.1145/2939672.2939704

Song W, Shi C, Xiao Z, et al (2019) Autoint: Automatic feature interaction learning via self-attentive neural networks. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. Association for Computing Machinery, New York, NY, USA, CIKM '19, p 1161–1170, https://doi.org/10.1145/3357384.3357925

de Souza Pereira Moreira G, Rabhi S, Lee JM, et al (2021) Transformers4rec: Bridging the gap between nlp and sequential / session-based recommendation. In: Proceedings of the 15th ACM Conference on Recommender Systems. Association for Computing Machinery, New York, NY, USA, RecSys '21, p 143–153, https://doi.org/10.1145/3460231.3474255

Sun F, Liu J, Wu J, et al (2019) Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. Association for Computing Machinery, New York, NY, USA, CIKM '19, p 1441–1450, https://doi.org/10.1145/3357384.3357895

Tang J, Wang K (2018) Personalized top-n sequential recommendation via convolutional sequence embedding. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. Association for Computing Machinery, New York, NY, USA, WSDM '18, p 565–573, https://doi.org/10.1145/3159652.3159656

Vaswani A, Shazeer N, Parmar N, et al (2017) Attention is all you need. In: Advances in Neural Information Processing Systems

Wang R, Fu B, Fu G, et al (2017) Deep & cross network for ad click predictions. In: Proceedings of the ADKDD'17. Association for Computing Machinery, New York, NY, USA, ADKDD'17, https://doi.org/10.1145/3124749.3124754

Wang R, Shivanna R, Cheng D, et al (2021) Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In: Proceedings of the Web Conference 2021. Association for Computing Machinery, New York, NY, USA, WWW '21, p 1785–1797, https://doi.org/10.1145/3442381.3450078

Wu CY, Ahmed A, Beutel A, et al (2017) Recurrent recommender networks. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. Association for Computing Machinery, New York, NY, USA, WSDM '17, p 495–503, https://doi.org/10.1145/3018661.3018689

Wu L, Sun P, Fu Y, et al (2019) A neural influence diffusion model for social recommendation. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. Association for Computing Machinery, New York, NY, USA, SIGIR'19, p 235–244, https://doi.org/10.1145/3331184.3331214

Wu L, Li S, Hsieh CJ, et al (2020) Sse-pt: Sequential recommendation via personalized transformer. In: Proceedings of the 14th ACM Conference on Recommender Systems. Association for Computing Machinery, New York, NY, USA, RecSys '20, p 328–337, https://doi.org/10.1145/3383313.3412258

Xiao J, Ye H, He X, et al (2017) 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, pp 3119–3125, https://doi.org/10.24963/ijcai.2017/435

Xiao Z, Yang L, Jiang W, et al (2020) Deep multi-interest network for click-through rate prediction. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. Association for Computing Machinery, New York, NY, USA, CIKM '20, p 2265–2268, https://doi.org/10.1145/3340531.3412092

Xu Y, Zhu Y, Yu F, et al (2021) Disentangled self-attentive neural networks for click-through rate prediction. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management. Association for Computing Machinery, New York, NY, USA, CIKM '21, p 3553–3557, https://doi.org/10.1145/3459637.3482088

Yan C, Chen Y, Wan Y, et al (2021) Modeling low- and high-order feature interactions with fm and self-attention network. Applied Intelligence 51(6):3189–3201. https://doi.org/10.1007/s10489-020-01951-6

Yan C, Li X, Chen Y, et al (2022) Jointctr: A joint ctr prediction framework combining feature interaction and sequential behavior learning. Applied Intelligence 52(4):4701–4714. https://doi.org/10.1007/s10489-021-02678-8

Yang Y, Xu B, Shen S, et al (2020) Operation-aware neural networks for user response prediction. Neural Networks 121:161–168. https://doi.org/https://doi.org/10.1016/

j.neunet.2019.09.020

Yap GE, Li XL, Yu PS (2012) 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. Springer-Verlag, Berlin, Heidelberg, DASFAA12, p 48–64, https://doi.org/10.1007/978-3-642-29035-0_4

Yuan F, Karatzoglou A, Arapakis I, et al (2019) A simple convolutional generative network for next item recommendation. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining. Association for Computing Machinery, New York, NY, USA, WSDM '19, p 582–590, https://doi.org/10.1145/3289600.3290975

Zhou C, Bai J, Song J, et al (2017) Atrank: An attention-based user behavior modeling framework for recommendation. 1711.06632

Zhou G, Zhu X, Song C, et al (2018) Deep interest network for click-through rate prediction. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Association for Computing Machinery, New York, NY, USA, KDD 18, p 1059–1068, https://doi.org/10.1145/3219819.3219823

Zhou G, Mou N, Fan Y, et al (2019) 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. AAAI Press, Honolulu, Hawaii, USA, AAAI19/IAAI19/EAAI19, https://doi.org/10.1609/aaai.v33i01.33015941

Zhu J, Liu J, Yang S, et al (2021) Open benchmarking for click-through rate prediction. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management. Association for Computing Machinery, New York, NY, USA, CIKM '21, p 2759–2769, https://doi.org/10.1145/3459637.3482486

Zhu J, Jia Q, Cai G, et al (2023) Final: Factorized interaction layer for ctr prediction. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. Association for Computing Machinery, New York, NY, USA, SIGIR '23, p 2006–2010, https://doi.org/10.1145/3539618.3591988