Bidirectional Encoder Representation with Matching Task for Sequential Recommendation

Lingxiao Zhang[1] and **Xiu Li**

Abstract. Characterizing users' interest accurately plays a significant role in an effective recommender system. The sequential recommender system takes successive user-item interactions and dynamic users' preferences into account, which brings powerful hidden representations of users. Conventional methods of such patterns mainly include Markov Chains (MCs) and Recurrent Neural Networks (RNNs). Recently, the use of self-attention mechanisms and bidirectional architectures enables recommendations not only to capture long-term semantics but also to fuse information from both the left and right sides. However, a major limitation can be observed in previous strategies—unstable representations of users. Sequential models only mine the relationship between items that users have interacted rather than forming the whole representation of each user, which is often a constraint on the further improvement of recommendation performance. To address this limitation, we propose a mixed sequential recommendation by combining bidirectional self-attention with direct user-item match, which brings two benefits. For one thing, our model can learn bidirectional semantics from users' behavioral sequences. For another, it produces more stable users' representations and achieves better recommendation performance. Extensive empirical studies demonstrate our approach considerably outperforms various state-of-the-art sequential models.

1 INTRODUCTION

Recommender Systems have been playing a crucial role in the recent two decades. It can help users obtain a more customized and personalized recommendation experience by characterizing users exhaustively and mine their interests precisely. A widely used approach to building quality recommender systems in real applications is collaborative filtering (CF) [1]. But such a method takes users' shopping behaviors as isolated manners, while these usually happen successively in a sequence.

Recently, sequential recommendations based on users' historical interactions have attracted increasing attention. [2-4]. They suggest items that may be of interest to a user by mainly modeling the sequential dependencies over the user-item interactions (e.g., like or purchase items on an online shopping platform) in a sequence. [5] Two basic paradigms of the pattern have proliferated: unidirectional (left-to-right) sequential model and bidirectional sequential model. The former, including Markov Chains (MC) [4, 6-8], Recurrent Neural Networks (RNNs) [2, 9-13] and self-attentive sequential recommendation [14], is more prevalent and close to the order of interactions between users and items in many real-world applications, yet it is not sufficient to learn optimal representations for user behavior sequences. The latter, like Bert4Rec [15], premeditates various unobservable external factors [16] and does not follow a rigid order assumption [17-18], which is beneficial to incorporate context from both sides for sequence representations learning. Nevertheless, previous methods do not form the whole user representation to get unstable recommendation performance, especially for a bidirectional sequential model.

On the other hand, the matching task is the key problem in recommendations. [19] The intrinsic quality of recommendation is matching at the semantic level between user and item [27] after they are represented as real-value vectors that encode rich semantics (e.g., semantically relevant objects should have large similarities). [20-21]. Recently, diverse deep learning methods have been developed for the matching task of recommendation and have shown promising results and demonstrated great potentials for further improvements. [22-23] In this paper, we seek to combine the bidirectional model and the matching task to learn hidden information from users' historical behavioral sequences and form the stable representation of each user.

To avoid information leakage and efficiently train the bidirectional model, the Cloze task [15, 24-25] is adopted to substitute the objective in unidirectional models (i.e., sequentially predicting the next item). Some items in the users' behavioral sequences are masked in certain probability (e.g., replace them with a special token [mask]). Then, the model predicts the ids of those masked items based on their surrounding context, which is a mixture of both the left and right context. Otherwise, the strategy can create more instances in multiple epochs to enhance the scalability of the model. For the matching task, inspired by doc2vec [26], we use a special token [UID] to represent the whole user behavioral sequence and concatenate the user token with several item tokens from a sequence. Then, our model needs to determine whether to match between each users' semantic vector (i.e., the output of the user token) and items' semantic vectors (including positive samples and negative samples) or not. While the user token is unique among datasets, the item tokens are shared and the output of the user token has merged various correlations among items in each sequence. This method can be applied to variable-length pieces of sequences and form the stable user representation.

[1] Shenzhen Graduate School, Tsinghua University, China, email: zhang-lx18@mails.tsinghua.edu.cn

During training time, we combine the Cloze task with matching task as a mixture task trained by Adam [29] and backpropagation. To fix a downside of the Cloze task that it is not consistent with the final task (i.e., sequential recommendation), we append the special token "[mask]" at the end of the input sequence, meaning to predict the last item. To make each user presentation unique, we produce instances that only compute the loss function of the matching task between original user behavioral sequence (no masking) and items. Extensive experiments on four datasets show that our model outperforms various state-of-the-art sequential models consistently.

The contributions of our paper are as follows:

- We propose the method that uses a special token [UID] to represent each user and concatenates [UID] with several item tokens from a sequence during training time. In this way, variable-length pieces of user behavioral sequences can be represented uniquely and stably.
- We combine the Cloze task with the matching task as a mixture task during training.
- We compare our model with state-of-the-art methods on four benchmark datasets.

2 RELATED WORK

In this section, we will briefly introduce several works closely related to ours. We first discuss general recommendation, followed by sequential recommendation and the attention mechanism. Eventually, we review the process of the matching task in recommendation.

2.1 General Recommendation

A widely used approach to building quality recommender systems in real applications is collaborative filtering (CF) [1, 30-31]. There are two types of methods of using users' historical feedbacks (e.g., ratings, clicks, purchases, comments, likes): point-wise [32] and pairwise [33]. Matrix Factorization (MF) is a typical one among various CF methods, and it maps users and items into the same latent vector space and estimates interactions through the inner product their vectors. [30, 34-37]. Additionally, item-based neighborhood methods are proposed, which estimate similarities between a user and an item via measuring the user's similarities with items in her/his interaction history based on a learned item-to-item similarity matrix.

Recently, deep learning techniques have been introduced for recommendation. One line of work seeks to use more auxiliary information (e.g., images [39-40], text [41-42], acoustic [43]) into recommendation systems. Another line of work seeks to replace conventional matrix factorization. For instances, Deep Matrix Factorization Models (DMF) [44] uses Multi-Layer Perceptions (MLP) to factorize model and Neural Collaborative Filtering (NCF) [45] uses Multi-Layer Perceptions (MLP) to estimate user preferences instead of inner product, while AutoRec [46] and CDAE [47] predict users' ratings by Auto-encoder framework.

2.2 Sequential Recommendation

Different from the above methods, sequential recommendation systems consider orders in users' behaviors. Early these works adopt Markov chain models to model the transition matrices over user-item interactions in a sequence to predict the next interaction. For example, FPMC combines an MF term with an item-item transition term to capture long-term preferences and short-term transitions respectively. [4] Besides the first-order MCs, high-order MCs are also adopted to consider more previous items [8, 14].

Recently, the most commonly used deep neural networks for sequential recommendation systems are recurrent neural networks (RNN) [5]. For example, GRU4Rec uses Gated Recurrent Units (GRU) to model click sequences for session-based recommendation [2], an improved version further boosts its Top-N recommendation performance [10], and attention-based GRU(NARM) uses the global encoder and local encoder to emphasis users' purposes. [48]

Apart from RNNs, other deep learning models are also applied in sequential recommendation systems, including convolutional neural networks (CNN). For example, Caser learns sequential patterns through both horizontal and vertical convolutional filters. [49] Memory Network is employed to improve sequential recommendation. [50]

2.3 Attention Mechanism

Attention mechanisms have been widely applied in various tasks, like image captioning [51], machine translation [52], text classification [53]. Recently, the use of attention mechanisms in recommendation has got substantial performance. For example, an attention mechanism is incorporated into GRU to capture both the user's sequential behavior and main purpose in session-based recommendation. [48] Attentional Factorization Machines (AFM) learn the importance of each feature interaction for content-aware recommendation. [54]

Different from taking attention mechanism as an additional component to the original models, Transformer [55] consists of multi-head self-attention layers and feed-forward layers and it is applied in BERT [24], GPT [56] and GPTv2[57] as a feature extractor, dominating on text sequence modeling.

Latterly, there is a surging enthusiasm for applying attention-based neural networks to sequential recommendation. For example, a two-layer Transformer decoder is applied in SASRec to capture user's sequential behaviors in left-to-right order (i.e., Transformer language model). [3] Bert4Rec uses a two-layer Transformer decoder with the help of the Cloze task to achieve bidirectional information mining, [15] which is closely related to our work. In this paper, we get better performance than previous works by adding the matching task.

2.4 Matching Task in Recommendation

There may be no overlap between user features and item features, so user-item matching task tends to be done on the semantic level rather than the superficial feature level. According to the specific technique used, there are two deep learning paradigms for matching in recommendation: methods of representation learning and methods of matching function learning. [19] The former calculates representation and then conducts matching, while the latter constructs basic low-level matching signals at the first step and aggregates matching patterns at the second step.

For methods of representation learning, matching function is a simple inner product or cosine similarity and the results depend on

the available data to describe an user/item. In these methods, appropriate DNNs are just chosen to learn representation, including Auto-Encoder, CNN, RNN, etc. [44, 46, 47] For methods of matching function learning, matching function is directly based on neural network such as MLP [45], CNN [58], Factorization Machines (FM) [58-61], etc. We adopt the former in this paper.

3 METHODOLOGY TITLE

In sequential recommendation, considering a user's interaction sequence $S^{u_i}=[v_1^{u_i},v_2^{u_i},v_3^{u_i},...,v_{|V|}^{u_i}]$, the next item $v_{n+1}^{u_i}$ is predicted, where $v_i^u \in V$ and item set $V=\{v_1,v_2,v_3,...,v_{|V|}\}$, $u_i \in U$ and item set $U=\{u_1,u_2,u_3,...,u_{|U|}\}$. Predicted probability can be formalized as $p(v_{n+1}^{u_i}=v \mid S^{u_i})$. In the section, our model architecture and several detailed modules are introduced. Table 1 shows the notations in the section.

Table 1. Notations

Notation	Description
S^u	Historical interaction sequence for users
U	User set
V	Item set
$L \in \mathbf{N}$	Number of Transformer layers
$N \in \mathbf{N}$	Maximum sequence length
$d \in \mathbf{N}$	Latent vector dimensionality
$h \in \mathbf{N}$	Head number in Transformer
$E_v \in \mathbf{R}^{N \times d}$	Item embedding matrix
$E_p \in \mathbf{R}^{N \times d}$	Position embedding matrix
$E \in \mathbf{R}^{N \times d}$	Input embedding matrix
$H_t^i \in \mathbf{R}^d$	Out of the i-th hidden layer at time step t
$n \in \mathbf{N}$	Number of negative sampling in the matching task

3.1 Model Architecture

Our model architecture is shown as Figure 1, which is made up of embedding layer, transformer layer, and output layer.

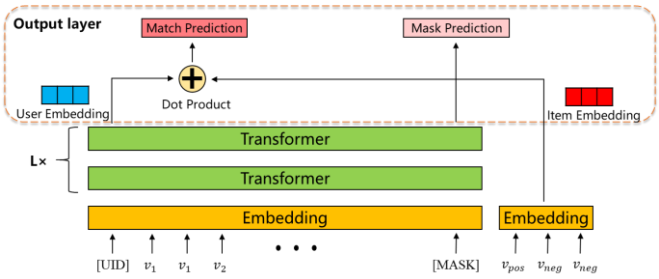

Figure 1. Our model architecture

In the embedding layer, the input sequential items are represented as item embedding and position embedding. Note that in order to build the whole user's representation, we add a special token [UID] at the beginning of the sequence and share the weights from the item embedding with positive and negative items that are used in following matching task. In Transformer layers, we stack L Transformer layers to catch dependencies of items in each sequence. Different from other sequential models such as RNN, self-attention mechanism directly computes dependencies of tokens in sequences rather than through accumulative dependencies in last time. In the output layer, besides predicting masked items, the model also needs to compute matching probability between user embedding and item embedding. It is worth mentioning that for the mask prediction task, we follow the previous method of bidirectional encoder representation (i.e., the Cloze task).

3.2 Embedding Layer

In our model, given a user's interaction sequence S^{u_i}, we set a restriction on the maximum sentence length N to make sure our model can handle. We consider the most recent N-1 actions if the sequence length is greater than N-1 (expect special token [UID] at first of the sequence). The input embedding has two types: user embedding and item embedding. For user embedding, we will capture users' behavioral information via stacking Transformer layers but they are not similar to recurrent modules to be aware of the order of previous items. Therefore, other than item embedding $E_v \in \mathbf{R}^{N \times d}$, learnable position embedding $E_p \in \mathbf{R}^{N \times d}$ is injected. So input representation $E \in \mathbf{R}^{N \times d}$ is made up by summing above two types of embedding:

$$E=E_v+E_p \tag{1}$$

Additionally, we use shared weights from item embedding E_v to map one positive item (i.e., the last item) and n negative items from random sample to item semantic space. A visualization of this construction can be seen in Figure 2.

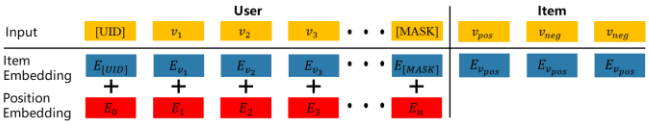

Figure 2. Input representation of our model. The input embedding has two types: user embedding and item embedding. User embedding is the sum of item embedding and position embedding.

3.3 Transformer Layer

Every transformer block is mainly constructed by a multi-head self-attention sub-layer and a feed-forward network. [55] Next, we introduce those details of the Transformer layer.

3.3.1 Multi-head self-attention

This sub-layer consists of several scaled dot-product attentions [55]. Each one can be defined as:

$$\text{Attention}(Q,K,V)=\text{softmax}(\frac{QK^T}{\sqrt{d/h}})V \tag{2}$$

where Q represents the queries of the item vectors, K and V mean the keys and the values of those respectively. The scale factor $\sqrt{d/h}$ is used to make gradients stable, especially when the dimensionality is high.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions [55], which is defined as:

$$\text{MultiHead}(Q,K,V)=\text{Concat}(\text{head}_1,\text{head}_2,...,\text{head}_h)W^o \tag{3}$$

$$\text{head}_i=\text{Attention}(QW_i^Q,KW_i^K,VW_i^V) \tag{4}$$

where the projection matrices $W_i^Q, W_i^K, W_i^V \in \mathbf{R}^{d \times d/h}$ and $W^O \in \mathbf{R}^{d \times d}$ are learnable parameters.

3.3.2 Position-wise feed-forward network

To endow the model with nonlinearity, the point-wise two-layer feed-forward network is applied to the outputs of the self-attention sub-layer, separately and identically at each position. Here, we adopt Gaussian Error Linear Unit (GELU) [62] as the activation function. Its equation is as follows:

$$\mathrm{FFN}(x)=\mathrm{GELU}(xW_1+b_1)W_2+b_1 \quad (5)$$
$$\mathrm{GELU}(x)=x\Phi(x) \quad (6)$$

where $\Phi(x)$ is the cumulative distribution function of the standard of Gaussian distribution, $W_1 \in \mathbf{R}^{d \times 4d}$, $W_2 \in \mathbf{R}^{4d \times d}$, $b_1 \in \mathbf{R}^{4d}$ and $b_2 \in \mathbf{R}^d$ are learnable parameters and shared across all positions. Because of the existence of the feed-forward network, our model can achieve parallel computing, which is amenable to GPU acceleration.

3.3.3 Stacking Transformer Layer

Because we need multiple Transformer layers to learn more complex item transition patterns, in order to avoid overfitting model and vanishing gradient, we employ a residual connection [63], layer normalization [64] and dropout [65] around each of the two sublayers. The process is formulated as follows:

$$g(x) = x + \mathrm{Dropout}(g(\mathrm{LayerNorm}(x))) \quad (7)$$

where $g(x)$ represents the self-attention layer or the feed-forward network. Layer normalization is defined as follows:

$$\mathrm{LayerNorm}(x) = \alpha \odot \frac{x-\mu}{\sqrt{\sigma^2+\varepsilon}} + \beta \quad (8)$$

where $\odot$ is an element-wise product (i.e., the Hadamard product), μ and σ are the mean and variance of x, α and β are learned scaling factors and bias terms.

3.4 Output Layer

In the output layer, we achieve two prediction tasks: masked items prediction and matching prediction. For the former, After L transformer layers hierarchically and adaptively capturing information of the users' behavioral sequences, we get the final output H_t^L, where t means the masked item v_t is at time step t. Activation function Softmax is employed to predict the masked item v_t, of which process is formulated as follows:

$$P_{Mask}(v_t) = \mathrm{Softmax}(h_t^L W_P + b_P) \quad (9)$$

where b_P is a learnable projection bias, W_P is the projection matrix. In order to alleviate overfitting and reducing the model size, we make W_P share weights from the item embedding matrix in the embedding layer.

For the latter, we extract the first final output H_1^L (i.e., the final output of [UID]), and a positive item E_v^{pos} and negative items E_v^{neg} that have been mapped into item semantic space in the embedding layer. Then, we calculate matching scores of positive one and negative ones. These computational equation is shown as follows:

$$Score_{pos} = E_v^{pos} \bullet H_1^L \quad (10)$$
$$Score_{neg} = (E_v^{neg} \bullet H_1^L) \ / n \quad (11)$$

Note that the negative score is divided by the number of negative sampling n to balance positive and negative scores' weights.

3.5 Model Learning

For unidirectional sequential recommendation, the task of predicting the next item tends to be adopted in their models. For example, these models create N-1 samples (like ($[v_1]$, v_2) and ($[v_1, v_2]$, v_3)) from the original length N behavioral sequence. But for bidirectional sequential recommendation, if we also adopt this strategy to train model, these models create (N-1)•(N-1)! samples, which is time-consuming and infeasible. So here, we firstly employ the Cloze task [25] (same as [15, 24]) to efficiently train our model. Different from [15], we add special tokens [UID] at the first of the sequence, which is used in the following matching task. Here is an example consisting of part of an input sequence in our model:

Input: $[\texttt{[UID]}, v_1, v_2, v_3, v_4, v_5] \rightarrow [\texttt{[UID]}, v_1, \texttt{[mask]}_2, v_3, \texttt{[mask]}_4, v_5]$

Labels: $\texttt{[mask]}_2 = v_2, \ \texttt{[mask]}_4 = v_4$

where we randomly mask the proportion ρ of all items in the input sequence (i.e., replace with special token "[mask]"), and our model needs to predict masked items' ids based on their surrounding items. Note that we always mask all of the successive same items at once to prevent the information leakage as far as possible. For the mask task, we define the negative log-likelihood loss for each masked input v_{mask} :

$$Loss_{mask} = \frac{1}{|S_{mask}^u|} \sum_{v_{mask} \in S_{mask}^u} -\log P_{mask}(v_{mask} = v_{mask}^* \mid S^{u'}) \quad (12)$$

where $S^{u'}$ is the masked version for user behavior history S^u, S_{mask}^u is the random masked items in it, v_{mask}^* is the label for the masked item v_{mask}, and the probability $P_{mask}(\bullet)$ is defined in Equation (9). In multiple epochs, we produce different masked samples to train a more powerful model.

It is an ingenious method by the Cloze task to draw upon bidirectional information of the users' behavioral sequences, but the model does not form the whole presentation of each user, which means recommendation is implemented indirectly between user and item. It is easy to cause unstable recommendation performance. Meanwhile, it is difficult to transfer the bidirectional information that model has mined to other learning tasks. So to address the problem, we add a new learning task into our model: the matching task. For this task, we adopt the binary cross-entropy loss as the loss function:

$$Loss_{matching} = -(\log(\sigma(score_{pos} \bullet c)) + \log(1 - \sigma(score_{neg} \bullet c))) \quad (13)$$

where $score_{pos}$ and $score_{neg}$ are defined in Equation (10) and (11) separately, c is a scaling coefficient, which is assigned to 10 by us. In multiple epochs, we randomly generate n negative items for each user sequence. And the total loss is the sum of masked-item loss and matching loss, shown as the following equation:

$$Loss = Loss_{mask} + Loss_{matching} \quad (14)$$

As elaborated above, the Cloze objective results in a mismatch between training and prediction. Here to alleviate the mismatch, we create another type of samples that are only appended the special token "[mask]" at the end of user's behavior sequence and the rest of which are original items. Simultaneously, to enhance our model's power of representation, we also produce samples that only consist of matching samples (no masked items). All in all, we have three types of samples, and the examples are listed as follows:

Mask+Matching: $[[_{[UID]}, v_1, {}_{[mask]_2}, v_3, {}_{[mask]_4}, v_5], [v_{pos}, v_{neg_1}, v_{neg_2}]]$

The last Mask: $[[_{[UID]}, v_1, v_2, v_3, v_4, {}_{[mask]_5}], []]$

Matching: $[[_{[UID]}, v_1, v_2, v_3, v_4, v_5], [v_{pos}, v_{neg_1}, v_{neg_2}]]$

In the prediction stage, we adopt a conventional strategy of sequence prediction, which is the prediction of the last item based on the final hidden representation of the sequence.

3.6 Discussion

Next, the relations of our work and existing methods are discussed conceptually on sequential recommendation.

SASRec: This model is based on the left-to-right unidirectional order of sequence with single head attention and causal attention mask. Its training method is the traditional predict-the-next-item way and adopts the binary cross-entropy loss as the objective function with negative sampling. Although the same objective function is employed as the matching loss function in our model, our model can draw upon the information from both the right side and left side of the sequence.

Bert4Rec: This model also uses the Cloze objective to capture bidirectional information with transformer layers, just the same as our model. Nevertheless, there are still some differences between Bert4Rec and our model: **a)** To produce stable user embedding, we add an extra special token [UID] at the first of the behavioral sequence. **b)** Our model uses a new task called matching task at the training stage to alleviate the mask dilemma besides the Cloze task. **c)** We enrich the composition of samples to enhance the generalization of the model.

4 EXPERIMENTS

4.1 Datasets

We evaluate the proposed model on four representative datasets from three real-world applications, which vary significantly in domains and sparsity.

Amazon [66]: These datasets contain product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 - July 2014. They are separated into 24 categories according to the top level. In this paper, we employ the small subsets of "Beauty" class and "Video Games" class.

Steam [3]: These datasets contain reviews from the Steam video game platform and information about which games were bundled together.

MovieLens [28]: These datasets are stable benchmarks, including several million movie ratings, reviews, and users' information. In the paper, we employ the "MovieLens-1M" version.

For the preprocessing procedure, we use a common strategy from [3, 4, 10, 15]. For all datasets, we transfer all ratings or reviews to implicit feedback (i.e., representing as numeric 1). In the following stage, we group the interaction records by users and arrange them into sequences ordered by timestamp. What is remarkable is that we leave out users and items with fewer than five feedbacks. The statistics of the processed datasets are demonstrated in Table 2.

Table 2. Statistics of processed datasets

Dataset	#users	#items	#actions	Avg. length
Beauty	22363	12101	0.23M	6.88
Video Games	24303	10672	0.26M	7.54
Steam	334730	13047	5.3M	10.59
ML-1M	6040	3416	1.0M	163.5

4.2 Baselines

To verify the effectiveness of our method, we choose the following representative baselines to compare with:

POP: This is a simple baseline that ranks items according to their popularity (i.e., number of interactions).

NCF [45]: This model is a general framework that replaces an inner product with a neural network to learn the matching function.

FPMC [4]: This model combines an MF term with first-order MCs to capture long-term preferences and short-term transitions respectively.

GRU4Rec+ [10]: This model uses GRU with a new cross-entropy loss functions and sampling strategy to model session-based recommendation.

Caser [49]: This model employs CNN in both horizontal and vertical ways to model high-order MCs for sequential recommendation.

SASRec [3]: This model uses a left-to-right Transformer language model to capture users' sequential behaviors.

Bert4Rec [15]: This model uses a two-layer Transformer decoder with the help of the Cloze task to achieve bidirectional sequential information mining.

For GRU4Rec+, Caser, SASRec, we use codes provided by the corresponding authors. For NCF, FPMC, and Bert4Rec, we implement them by using *TensorFlow*. We consider the hidden dimension size d from {8, 16, 32, 64}, the $\ell 2$ regularizer from {1, 0.1, 0.01, 0.001, 0.0001}, and dropout rate from {0, 0.1, 0.2, $\cdots$; 0.9}. All other hyper-parameters (e.g., Markov order in Caser) and initialization strategies are either followed the suggestion from the methods' authors or tuned on the validation sets. All results of baselines are under their optimal hyper-parameter settings.

4.3 Implementation Details

We implement our model with *TensorFlow*. All parameters are initialized by using truncated normal distribution in the range [-0.02, 0.02]. We train the model by using Adam [29] with learning rate of 1e-4, $\beta 1 = 0.9$, $\beta 2 = 0.999$, $\ell 2$ weight decay of 0.01. We set the layer number $L = 2$ and head number $h = 2$ and set the maximum sequence length $N = 200$ for ML-1m, $N = 50$ for Beauty, Video Games, and Steam datasets, which are the same as [3, 15]. We adopt the dimensionality d=32. For the mask proportion ρ, we also employ the same ones with [15] (i.e., $\rho = 0.6$ for Beauty and Video Games, $\rho = 0.4$ for Steam, $\rho = 0.2$ for MovieLens-1M). For the number negative sampling n, we set n=5. All the models are tuned on a single NVIDIA GeForce GTX 1080 Ti GPU with a batch size of 128.

Table 3. Recommendation performance. The best performing method in each row is boldfaced, and the second best method in each row is underlined.

Datasets	Metric	POP	NCF	FPMC	GRU4Rec+	Caser	SASRec	Bert4Rec	Ours	Improv.
Beauty	NDCG@10	0.1793	0.2567	0.2937	0.2354	0.2705	**0.3370**	0.2719	0.3298	-2.1%
	HR@10	0.3363	0.4223	0.4064	0.3943	0.4217	**0.5009**	0.4354	0.4943	-1.3%
	MRR	0.1553	0.2229	0.2773	0.1105	0.2424	0.1865	0.2431	**0.2957**	21.6%
Video Games	NDCG@10	0.2512	0.3778	0.3225	0.4634	0.4137	**0.5163**	0.4738	0.4947	-4.1%
	HR@10	0.4385	0.6031	0.5211	0.7137	0.6307	**0.7320**	0.6861	0.7217	-1.4%
	MRR	0.2151	0.3241	0.2793	0.2379	0.3616	0.3134	0.4119	**0.4396**	6.7%
Steam	NDCG@10	0.4927	0.4996	0.5768	0.5465	0.5950	0.6316	0.6171	**0.6460**	2.2%
	HR@10	0.7556	0.7629	0.8216	0.7986	0.8310	0.8633	0.8440	**0.8760**	1.4%
	MRR	0.4225	0.4295	0.5087	0.5247	0.5292	0.4177	0.5488	**0.5835**	6.3%
ML-1M	NDCG@10	0.2455	0.4094	0.5258	0.5456	0.5483	0.5354	0.5408	**0.5669**	3.3%
	HR@10	0.4458	0.6856	0.7439	0.7514	0.7769	0.7889	0.7546	**0.7896**	0.0%
	MRR	0.2070	0.3398	0.3600	0.4039	0.4728	0.3039	0.4517	**0.4973**	5.1%

4.4 Evaluation Metrics

To evaluate the performances of the recommendation models, we adopt the leave-one-out evaluation (i.e., next item recommendation) task, which is widely used in [3, 15, 10, 45, 49]. For each user, we select the most recent action of his/her behavioral sequence as the test set, treat the second most recent action as the validation set, and utilize the remainder as the train set. Note that during testing, the input sequence is a combination of the train set and the validation set.

For evaluation metrics, we adopt Normalized Discounted Cumulative Gain (NDCG), Hit Ratio (HR) and Mean Reciprocal Rank (MRR) metrics. Note that since we only have one test item for each user, HR@k is equivalent to Recall@k and proportional to Precision@k; MRR is equivalent to Mean Average Precision (MAP). In this work, we report HR and NDCG with k =10. For all metrics, the higher value means the better performance.

To avoid computing heavily on all user-item pairs, we randomly sample 100 negative items and rank these negative items with the ground-truth item for each user. Based on the rankings of these 101 items, the evaluation metrics above can be evaluated.

4.5 Recommendation Performance

Table 3 illustrates the results of all methods on the four datasets. The last column is the improvements of our method relative to the best baseline. From the results, we can summarize that:

Because of just considering the number of interactions, the non-personalized POP method gets the worst performance on all datasets. In general, the sequential methods outperform traditional non-sequential methods such as NCF due to successive sequential information, which explains sequential information is beneficial to the improvement of performance in recommendation systems. Particularly, on sparse dataset Video Games, easy non-neural sequential models based on first-order Markov chain like FPMC even perform worse than the neural model only based collaborative filtering like NCF, for the reason is that these datasets only have little additional sequential information. Thus, compared with sequential information, the neural model having more parameters is magnificent to recommendation performance. Among sequential recommendation baselines, on dense dataset ML-1m, Caser gets better performance than FPMC, which suggests that high-order MCs is useful when input sequences are long. Furthermore, SASRec outperforms RNN (like GRU4Rec+), CNN (like Caser) sequential model on the whole, meaning that the self-attention mechanism is more powerful for sequential feature extraction. On dense dataset ML-1m, Bert4Rec gets better performance than SASRec, suggesting that bidirectional sequential information is beneficial for recommendation system. However, on sparse datasets Beauty, Video Games, Steam, it is hard for Bert4Rec to balance the information to be predicted (i.e., number of masked items) and information from surrounding context (i.e., number of no-masked items). In other words, by reason of the short length of the input sequence, more masked items mean less available context information and vice versa. But for the MRR metric, Bert4Rec basically gets state-of-the-art performance on four benchmark datasets except ours, suggesting that ranks of ground-truth items still improve as a whole even if they do not reach the top 10.

According to the results, it is depicted that except for NDCG@10 and HR@10 in Beauty, Video, and Games, our method improved the best baseline on all four datasets w.r.t. the three metrics, especially, gaining 9.93% MRR improvements (on average) against the strongest baselines. Compared with Bert4Rec, an additional matching task and more abundant samples make our model outperform by a large margin w.r.t. the three metrics, which means the matching task is an important auxiliary tool to improve recommendation performance, especially faced with a situation where the whole input sequence is short. Besides, the output of [UID] can be used in other machine learning tasks as the stable users' representation.

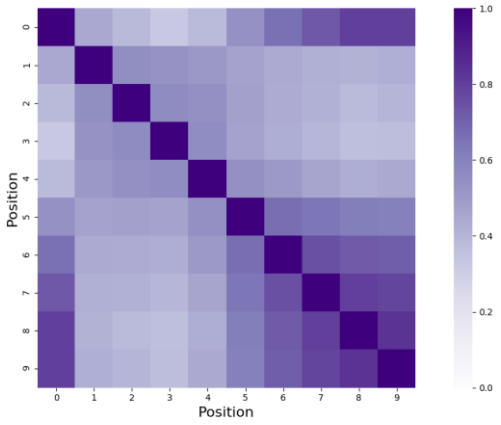

Figure 3. Heat-map of average correlation coefficients of output sequences on Beauty at different positions. The first position "0" denotes "[UID]".

Meanwhile, in order to qualitatively reveal the model's behavior, we visualize average correlation coefficients of output sequences on Beauty of the first 10 items in figure 3. From the result, some

tendencies can be concluded as follows: **a)** The users' representations are more affected by recent behavior, which is consistent with our common sense, for the reason that recent items play a more important role in predicting the future. **b)** Items in our model tend to highlight the items on both sides, especially the surrounding items, which means bidirectional information has been mined successfully.

4.6 Ablation Study

Finally, we use the ablation study to analyze numerous key components of our model to better understand their impacts. Table 4 shows the results of our default version and its variants on all four datasets (with $d = 32$).

Table 4. Ablation analysis (MRR) on four datasets. Bold score indicates performance better than the default version, while " ↓ " indicates performance drop more than 10%.

Architecture	Beauty	V-Games	Steam	ML-1M
Default	0.2957	0.4396	0.5835	0.4973
w/o PE	0.2777	0.3911 ↓	0.5591	0.2955 ↓
Only matching task	0.1770 ↓	0.2487 ↓	0.4933 ↓	0.2284 ↓
Only mask task	0.2431 ↓	0.4119	0.5488	0.4517
1 head ($h = 1$)	0.2818	0.4078	0.5763	0.4611
4 heads ($h = 4$)	0.2826	0.4216	**0.5841**	**0.5012**
1 layer ($L = 1$)	0.2756	0.4273	0.5713	0.4656
3 layers ($L = 3$)	**0.2981**	**0.4435**	**0.5907**	**0.5076**

We introduce the variants and analyze their effect respectively:

w/o PE: Without the position embedding, the sequential model becomes the traditional model based on isolated actions. That is to say, the attention weight on each item depends only on item embedding, which leads to the rapid decline of recommendation performance, especially on dense datasets. This is primarily because long sequences have more noise actions, which brings more serious impacts on prediction if not considering order factors.

Only matching task: The variant just uses the matching task as an objective task. In this way, the recommendation task turns into a naïve matching problem between user sequences and target items. Since the information inside the sequence is not used at all, the decrease caused by this variant is very steep.

Only mask task: The variant just adopts the mask task (i.e., the Cloze task) as an objective task, just like Bert4Rec. Although this variant has less effect on denser datasets owing to the use of bidirectional information in sequences, the recommendation performance witnesses a noticeable decrease on sparse datasets (e.g., Beauty) because the phenomenon of the mask dilemma is more common due to shorter sequential length on sparse datasets. (i.e., more masked items mean less available context information and vice versa.)

Head number *h*: Multi-headed attention can expand the model's ability to focus on different positions, so we observe that long sequence datasets benefit from a larger h (e.g., ML-1M), which means users' multiple interest is mined successfully. Further, short sequence datasets prefer a smaller h due to a lack of enough information.

Number of layers *L*: The results demonstrate that hierarchical Transformer layers can help model learn more complicated item transition patterns, which confirm the validity of the self-attention mechanism.

5 CONCLUSION

Deep bidirectional sequential recommendation architecture has been proposed and caused widespread concern. In this paper, we optimize the existing bidirectional model via a new matching task and the use of the special token [UID] representing users in addition to the Cloze task. This approach alleviates the mask dilemma when the sequence's length is short and provides a stable and unified representation of each user. Extensive experimental results on four real-world datasets indicate that our model outperforms state-of-the-art baselines. In the future, we will fuse heterogeneous interactions (e.g., purchase, review, clicks, comment, etc.) in our model and consider the time factor. Another valuable direction is to handle extremely long dependency in sequence.

REFERENCES

[1] X. Su and T.M. Khoshgoftaar. *A Survey of Collaborative Filtering Techniques*. Advances in Artificial Inteligence (2009).

[2] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. *Session-based Recommendations with Recurrent Neural Networks*. In Proceedings of ICLR, 2016.

[3] Kang, W. Cheng and McAuley, Julian. *Self-Attentive Sequential Recommendation*. In Proceedings of IEEE International Conference on Data Mining (ICDM), IEEE, Singapore, 197–206, 2018.

[4] S. Rendle, C. Freudenthaler, and L.S. Thieme. *Factorizing Personalized Markov Chains for Next-basket Recommendation*. In Proceedings of WWW. ACM, New York, NY, USA, 811–820, 2010.

[5] S. Wang, L. Hu, Y. Wang, L. Cao, Q.Z. Sheng and M. Orgun. *Sequential Recommender Systems: Challenges, Progress and Prospects*. In Proceedings of IJCAI, 2019

[6] F. Garcin, C. Dimitrakakis, and B. Faltings. *Personalized news recommendation with context trees*. In Proceedings of the 7th ACM Conference on Recommender Systems, 105–112, 2013.

[7] S. Feng, X. Li, Y. Zeng, G. Cong, and Y.M. Chee. *Personalized ranking metric embedding for next new poi recommendation*. In Proceedings of the 24th International Joint Conference on Artificial Intelligence, 2069–2075, 2015.

[8] R. He and J. McAuley. *Fusing similarity models with markov chains for sparse sequential recommendation*. In Proceedings of the 16th IEEE International Conference on Data Mining, 191–200, 2016.

[9] S. Hochreiter and J. Schmidhuber. *Long Short-Term Memory*. Neural Computation 9, **8**, 1735–1780, (Nov. 1997).

[10] B. Hidasi and A. Karatzoglou. *Recurrent Neural Networks with Top-k Gains for Session-based Recommendations*. In Proceedings of CIKM. ACM, New York, NY, USA, 843–852, 2018.

[11] F. Yu, Q. Liu, S. Wu, L. Wang, and T Tan. *A Dynamic Recurrent Model for Next Basket Recommendation*. In Proceedings of SIGIR. ACM, New York, NY, USA, 729–732, 2016.

[12] C. Wu, A. Ahmed, A. Beutel, A.J. Smola, and H. Jing. *Recurrent Recommender Networks*. In Proceedings of WSDM. ACM, New York, NY, USA, 495–503, 2017.

[13] M. Quadrana, A. Karatzoglou, and et al. *Personalizing session-based recommendations with hierarchical recurrent neural networks*. In Proceedings of the 11th ACM Conference on Recommender Systems, 130–137, 2017.

[14] R. He, W. Kang, and J. McAuley. *Translation-based Recommendation*. In Proceedings of RecSys. ACM, New York, NY, USA, 161–169, 2017.

[15] F. Sun, J. Liu, and et al. BERT4Rec: *Sequential Recommendation with Bidirectional Encoder Representations from Transformer*. In Proceedings of CIKM, ACM, Beijing, CN, 1441-1450, 2019.

[16] P. Covington, J. Adams, and E. Sargin. *Deep Neural Networks for YouTube Recommendations*. In Proceedings of RecSys. ACM, New York, NY, USA, 191–198, 2019.

[17] L. Hu, L. Cao, and et al. *Diversifying Personalized Recommendation with User-session Context*. In Proceedings of IJCAI. 1858–1864, 2017.

[18] S. Wang, L. Hu, and et al. *Attention-Based Transactional Context Embedding for Next-Item Recommendation*. In Proceedings of AAAI. 2532–2539, 2018.

[19] J. Xu, X. He and H. Li. *Deep Learning for Matching in Search and Recommendation*. In Proceedings of SIGIR, Ann Arbor, MI, USA, 1365–1368, 2018.

[20] X. He, H. Zhang, M.Y. Kan, and T.S. Chua. *Fast Matrix Factorization for Online Recommendation with Implicit Feedback*. In Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, 549–558, 2016.

[21] Y. Koren, R. Bell, and C. Volinsky. *Matrix Factorization Techniques for Recommender Systems*. Computer 42, 8, 30–37, (Aug. 2009).

[22] B. Hidasi, A. Karatzoglou, and et al. *DLRS 2017: Second Workshop on Deep Learning for Recommender Systems*. In Proceedings of RecSys, ACM, New York, NY, USA, 370–371, 2017.

[23] L. Pang, Y. Lan, and et al. *Text Matching As Image Recognition*. In Proceedings AAAI, AAAI Press, 2793–2799, 2016.

[24] J. Devlin, M.W. Chang, K. Lee, and K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. CoRR abs/1810.04805 (2018).

[25] W.L. Taylor. Cloze procedure: *A New Tool for Measuring Readability*. Journalism Bulletin, **30**, 415–433, 1953.

[26] Q. Le and T. Mikolov. *Distributed Representations of Sentences and Documents*. arXiv:1405.4053(2014)

[27] J. Gonzalo, H. Li, and et al. 2014. *SIGIR 2014 Workshop on Semantic Matching in Information Retrieval*. In Proceedings of SIGIR, ACM, New York, NY, USA, 1296–1296, 2014.

[28] F.M. Harper and J.A. Konstan. *The MovieLens Datasets: History and Context*. ACM Trans. Interact. Intell. Syst. 5, **4**, Article 19, 19 (Dec. 2015).

[29] Kingma, Diederik P. and Ba, Jimmy. *Adam: A Method for Stochastic Optimization*. In Proceedings of ICLR, San Diego, 2015

[30] Y. Koren and R. Bell. *Advances in Collaborative Filtering*. In Recommender Systems Handbook. Springer US, Boston, MA, 145–186, 2015.

[31] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based Collaborative Filtering Recommendation Algorithms. In Proceedings of WWW. ACM, New York, NY, USA, 285–295, 2015.

[32] Y. Hu, Y. Koren, and C. Volinsky, *Collaborative filtering for implicit feedback datasets*. In Proceedings of ICDM, 2008.

[33] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. *BPR: bayesian personalized ranking from implicit feedback*. In Proceedings of UAI, 2009.

[34] R. Salak. and A. Mnih. *Probabilistic Matrix Factorization*. In Proceedings of NIPS. Curran Associates Inc., USA, 1257–1264, 2017.

[35] Y. Koren, R. Bell, and C. Volinsky. *Matrix Factorization Techniques for Recommender Systems*. Computer 42, **8**, 30–37, (Aug. 2009).

[36] F. Ricci, L. Rokach, B. Shapira, and P. Kantor. Recommender systems handbook. Springer US, 2011.

[37] Y. Koren and R. Bell. *Advances in collaborative filtering*. Recommender Systems Handbook. Springer, 2011.

[38] S. Kabbur, X. Ning, and G. Karypis. *Fism: factored item similarity models for top-n recommender systems*. In Proceedings of SIGKDD, 2013.

[39] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu. *Convolutional Matrix Factorization for Document Context-Aware Recommendation*. In Proceedings of RecSys. ACM, New York, NY, USA, 233–240, 2016.

[40] H. Wang, N. Wang, and D.Y. Yeung. *Collaborative Deep Learning for Recommender Systems*. In Proceedings of KDD. ACM, New York, NY, USA, 1235–1244, 2015.

[41] W.C. Kang, C. Fang, Z. Wang, and J. McAuley. *Visually-Aware Fashion Recommendation and Design with Generative Image Models*. In Proceedings of ICDM. IEEE Computer Society, 207–216, 2017.

[42] S. Wang, Y. Wang, J. Tang, K. Shu, S. Ranganath, and H. Liu. *What Your Images Reveal: Exploiting Visual Contents for Point-of-Interest Recommendation*. In Proceedings of WWW. 391–400, 2017.

[43] A.v.d. Oord, S. Dieleman, and B. Schrauwen. *Deep content-based music recommendation*. In Proceedings of NIPS. 2643–2651, 2013.

[44] H.J. Xue, X. Dai, J. Zhang, S. Huang and J. Chen. *Deep Matrix Factorization Models for Recommender Systems*. In Proceeding IJCAI, 3203-3209, 2017

[45] X. He, L. Liao, H. Zhang, L. Nie, Xia Hu, and T.S. Chua. *Neural Collaborative Filtering*. In Proceedings of WWW. 173–182, 2017.

[46] S. Sedhain, A. Krishna Menon, S. Sanner, and L. Xie. *AutoRec: Autoencoders Meet Collaborative Filtering*. In Proceedings of WWW. ACM, New York, NY, USA, 111–112, 2015.

[47] Y. Wu, C. DuBois, A.X. Zheng, and M. Ester. *Collaborative Denoising Auto-Encoders for Top-N Recommender Systems*. In Proceedings of WSDM. ACM, New York, NY, USA, 153–162, 2016.

[48] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma. *Neural Attentive Session-based Recommendation*. In Proceedings of CIKM. ACM, New York, NY, USA, 1419–1428, 2017.

[49] J. Tang and K. Wang. *Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding*. In Proceedings of WSDM. 565–573, 2018.

[50] J. Huang, W.X. Zhao, and et al. *Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks*. In Proceedings of SIGIR. ACM, New York, NY, USA, 505–514, 2018.

[51] K. Xu, J. Ba, and et al. *Show, attend and tell: Neural image caption generation with visual attention*. In Proceedings of ICML, 2015.

[52] D. Bahdanau, K. Cho, and Y. Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. In Proceedings of ICLR, 2015.

[53] Z. Yang, D. Yang, and et al. *Hierarchical Attention Networks for Document Classification*. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: 1480-1489, 2016.

[54] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu and T.S. Chua. *Attentional factorization machines: Learning the weight of feature interactions via attention networks*. In Proceedings of IJCAI, 2017.

[55] A. Vaswani, N. Shazeer, and et al. *Attention is All you Need*. In Proceedings of NIPS. Curran Associates, Inc., 5998–6008, 2017.

[56] A. Radford, K. Narasimhan, T. Salimans and I. Sutskever. *Improving Language Understanding by Generative Pre-Training*.

[57] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever. *Language Models are Unsupervised Multitask Learners.*

[58] X. He, X. Du, X. Wang, and et al. *Outer Product-based Neural Collaborative Filtering*. In Proceedings of IJCAI, 2227-2233, 2018.

[59] X. He, T.S Chua. *Neural Factorization Machines for Sparse Predictive Analytics*. In Proceedings of SIGIR, 2017.

[60] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, T.S. Chua. *Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks*. In Proceedings of IJCAI, 2017.

[61] H. Guo, R. Tang, Y. Ye, Z. Li, X. He. *DeepFM: A Factorization-Machine based Neural Network for CTR Prediction*. In Proceedings of IJCAI, 2017.

[62] D. Hendrycks, K. Gimpel. *Gaussian Error Linear Units*. arXiv:1606.08415 (2016)

[63] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. In Proceedings of CVPR. 770–778, 2016.

[64] L. Jimmy Ba, R. Kiros, and G. Hinton. *Layer Normalization*. CoRR abs/1607.06450 (2016).

[65] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. J. Mach. Learn. Res. 15, **1**, 1929–1958, (Jan. 2014).

[66] J. McAuley, C. Targett, Q. Shi, and A.v.d Hengel. *Image-Based Recommendations on Styles and Substitutes*. In Proceedings of SIGIR. ACM, New York, NY, USA, 43–52, 2015.