Sequence and Distance Aware Transformer for Recommendation Systems

Runqiang Zang, Meiyun Zuo, Jilei Zhou, Yining Xue, Keman Huang*
School of Information, Renmin University of China, Beijing, China
Email: {zangrunqiang, zuomy, zhoujilei, xueyn, keman}@ruc.edu.cn

***Abstract*—Transformer has achieved admirable success in sequential tasks. However, the model only considers the order of items in the sequence, not the relative distances, which weakens the relevance between items. To this end, we propose a novel Sequence and Distance Aware Transformer (SDAT) for recommendation systems. Specifically, we first apply the Transformer to handle the interaction between the items effectively. Then, Gated Recurrent Unit can be designed to aggregate information on an item-by-item basis in sequential information, meanwhile, we adopt the attention mechanism to focus on items with smaller time intervals to indicate high relevance. We also add a time gain function to augment the influence weight of recent items. Finally, the processing result of the time information of our integrated items replaces the positional encoding representation of the original Transformer. Extensive experiments on three real-world datasets show that SDAT outperforms state-of-the-art methods.**

***Keywords*—recommendation systems; time-aware; attention; transformer**

I. Introduction

In the age of information explosion, recommendation systems(RSs) determine user interests by analyzing users' historical behavior data, and recommend items that may be of interest to these users. Existing RSs rely heavily on explicit feedback (i.e., user ratings). However, these data types may be extremely sparse. Many users did not even leave any explicit feedback on the online platforms. In addition, explicit feedback generated by users (e.g., reviews) is not all true and objective, which greatly impacts RSs. To overcome this obstacle, researchers have attempted to enhance RSs by incorporating implicit feedback, such as browsing, purchase, and click-through interactive items, which are authentic and easier to collect. Implicit feedback usually happens successively in a sequence rather than an isolated manner [1], each implicit feedback item has a certain dependence on its position [2] in the sequence.

Motivated by the sequential dependencies that commonly exist in implicit feedback data, the sequential recommendation has emerged with increasing attention, which aims to predict the next item that he/she might interact [3]. In the early stage, sequential recommendation mainly explores frequent patterns, itemsets, sequences that frequently appeared in the products purchased by the user, and analyzes which items are often purchased at the same time when shopping, but the sequence

*Meiyun Zuo is the corresponding author.

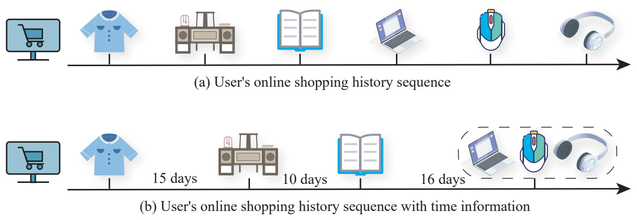
(a) User's online shopping history sequence
15 days
10 days
16 days
(b) User's online shopping history sequence with time information

Fig. 1. An example of a user's online shopping record, (a) the shopping record is only sequential, (b) the shopping record with time information, including the user's shopping basket item.

information of the purchased item in the ongoing conversation is ignored. Later work investigates the use of Markov chains [4], which means the transition of each state in the sequence only depends on the previous n states.

Since Recurrent Neural Network (RNN) and its variants (e.g., Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), etc.) [5] demonstrate their effectiveness in processing sequence data, RNN-based models have attracted increasing interest in sequential recommender systems. However, RNN-based models face the gradient vanishing and exploding problem and struggle to capture long-term dependencies in the historical sequences. To overcome these obstacles, a new model named Transformer [6, 7] has achieved admirable success in the sequence tasks. Different from RNN-based methods, Transformer can access any part of the history regardless of distance, making it potentially more suitable for grasping recurring patterns with long-term dependencies [8].

Despite their prevalence, previous Transformer-based models are not sufficient to learn sequential user behaviors. Although they can learn long-range dependencies between items, previous models implement with no specific means to take the user's short-term behavior into account. A user's choice of items depends on not only long-term historical preferences but also recent short-term preferences(e.g., the recently interacted items). Therefore, considering short-term preference is desirable and necessary.

Previous models mainly focus on the order information, which ignores the valuable patterns underlying the "distance" between items. Here, the "distance" refers to the time interval between items. For example, A historical order made by a user on the online platform is shown in Fig. 1, a user has just purchased a item, and perhaps the user's purchase intention

2021 IEEE International Conference on Web Services (ICWS) | 978-1-6654-1681-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICWS53863.2021.00027

is continuous, then the next item the user wants to purchase may be affected by the current item. However, if the user intends to purchase the next item a long time later, since the time between the two purchases is very long, there may be no correlation between the two items. Another common scenario is that users often purchase multiple items at the same time in the form of a shopping basket, and the sequential recommendation system is no longer applicable. It can be seen that time information is essential, which means that items with small time intervals, especially the items with no time interval (i.e., items are purchased together), are likely to be highly correlated [9].

To address the aforementioned restrictions, we introduce a new recommendation model called SDAT, which adopts **S**equence and **D**istance **A**ware **T**ransformer to the sequential recommendation. Specifically, for the restriction of the left-to-right unidirectional model, Transformer is applied in our study to derive long-term patterns of users over time. To augment the importance of recent items, we use exponential time gain function to construct an item time representation on the time series of the items. To consider the distance information between items, we apply GRU-Attention(GRU-Att) mechanism to the time intervals to calculate the relative distance between items. Finally, the time interval representation and the item time representation are used as weights, which are multiplied by the sequence of continuous representations, and input results into the model to predict the probability of purchasing the next item. Our main contributions are listed as follows:

- We propose that the position embedding in the Transformer model is augmented by the sequence and distance between items, i.e., time sequence features and time interval features, which are complementary and generalized.
- We use the attention-based GRU model to process distance information, in order to selectively retain and utilize the dependencies between items, and to make Transformer model have the ability to learn the interaction between items more flexibly.
- The time gain factor is added to the sequence information of the item, with which our model can capture both the short-term and the long-term preferences of users.

II. RELATED WORK

A. Recommendation Systems

Sequential recommendation predicts the next item that may be clicked or purchased based on the user's implicit feedback of the items in the anonymous sequence. For example, Sarwar et al. [10] compared the difference between the item correlation and the cosine similarity for computing item-item similarities. Rendle et al. [4] used the decomposable personalized Markov chain method, combining the first-order Markov chain and matrix factorization (FPMC) to simulate personalized sequential behavior.

Recently RNN has been employed to capture previously interaction items for current interest prediction. Hidasi et al. [11] applied recurrent neural networks (RNN) to RSs, and it not only proved that RNN could be applied to sequential recommendation problems, but also can design RNN with GRU Model (GRU4Rec) and this problem was regarded as time series prediction. Hu et al. [12] proposed a shallow neural network structure, by calculating the relative distance between context items, the relevance between items is captured. Liu et al. [13] utilized STAMP to emphasize short-term memory priority and used attention network to model the conversation, which improved the importance of the last interactive item in the target conversation. Several recent studies also investigated its capability in session-based recommendation. SR-GNN [14] utilized the complementarity between self-attention and the graph neural network to augment the recommendation performance. CoSAN [8] with self-attention for session-based recommendation, was proposed to learn long-range dependencies between collaborative items and generate collaborative session representation.

There exists a time-aware RSs similar to the sequential recommendation. Time dynamic information is integrated into the time series model to provide the basis for user preference refinement. For example, Session-based Temporal Graph model [15] was proposed to model user's long-term and short-term preferences over time. TeRec [16] was proposed for temporal RSs over tweet streams, which considers shifts of users' interests and popularity of topics as time passes by, and it aims to provide suitable topics for users at the right time. Liu et al. [17] provided a time dynamic model that integrates the user interests and evolving preferences in a specific period of time. These models help understand the user's preferences fully but ignore the multiple items that the user interacts with simultaneously.

Our research is inspired by the above literature review: It is important to improve the last interactive item. RNN applied to recommendation system can handle sequence relationships. Attention mechanism makes it easy to memorize very long-range sequence dependencies in RNN, and helps RNN to concentrate on important parts of inputs. Furthermore, attention mechanism can make the model obtain higher accuracy. In this work, we utilize self-attention to model dependencies and the importance of user behavioral patterns in purchasing combination products.

B. Transformer

Transformer is a strong sequence parallel modeling tool; for instance, SASRec [18] sought to identify which items are relevant from a user's action history, and uses them to predict the next item. Since the Transformer uses sinusoidal position embedding [6] and learns absolute position embedding, it does not explicitly model relative position information in its structure. The result is that the Transformer can only learn the sequence of each item in the sequence, which is an obvious weakness. Li et al. [19] argued that the distance between two items should be considered in Transformer. TiSASRec [19] modelled the relative time intervals and absolute positions among items to predict future interactions, This model not only considers the absolute positions of items like SASRec, but the relative time intervals between any two items. Although the model has achieved good experimental results, it can still process relative distance information in more detail.

Dehghani et al. [20] used a self-attention mechanism to exchange information across all positions in the sequence, thereby generating a vector representation for each position that is informed by the representations of all other positions at the previous time-step. Dai et al. [7] proposed a novel neural architecture Transformer-XL that enables learning dependency beyond a fixed-length without disrupting temporal coherence, the model consists of a segment-level recurrence mechanism and a novel positional encoding scheme. The previous methods were to remedy the loss of relative position information caused by the sum of word embedding and position embedding. These studies provide a meaningful reference for the optimization of the model.

III. SEQUENCE AND DISTANCE AWARE TRANSFORMER

A. Problem Statement

We sort the historical item information of each user by time and connect all the user's items. Therefore, the time series structure of the order is composed of the item sequence and the order time series, and each item corresponds to the order time one by one.

The task of our model is to learn the order sequence of users and predict an item that the user interacts with the next time. Therefore, we mask the last item before recommending, but keep the corresponding time of the last item. As shown in Fig.2, we let $Item = \{i_1, i_2, \ldots, i_{|u|-1}\}$ denote a set of items and $Lable = i_{|u|}$ as our target item, $|u|$ denotes the number of actions in the user's behavior sequence; $Time = \{t_1, t_2, \ldots, t_{|u|-1}, t_{|u|}\}$ represent a set of timestamps of the items, where $t_{|u|-1}$ means a user interacted with an item $i_{|u|-1}$ at time $t_{|u|-1}$; all other events have non-negative delta-timestamps. $\Delta = \{\Delta_1, \Delta_2, \ldots, \Delta_{|u|-1}\}$ denote the time interval between adjacent items in Item. That is, $\Delta_{|u|-1}$ is the time distance between $t_{|u|-1}$ and $t_{|u|}$. We merge $Item, Time, \Delta$ and $Lable$ in sequences and input them into our model.

We project $Item, Time$, and Δ as three kinds of embedding spaces, then integrate these embeddings and input them into the Transformer. $Lable$ will be compared with the output of the SDAT.

B. Item Representation Layer

In this study, we adopt SDAT to the sequential recommendation. The overview architecture is shown in Fig. 2. We can see that SDAT is built upon the popular Transformer encoder layer. The key difference between SDAT and Transformer is that we handle positional encoding more appropriately. In the following parts, we introduce the key components of SDAT: Item representation layer, Transformer encoder layer, and Prediction layer.

Our model is based on the Transformer encoding blocks [6]. Given a user $Item = \{i_1, i_2, \ldots, i_{|u|-1}\}$, we use a look-up table to convert each item to a dense embedding vector $i \in \mathbb{R}^{d_{Item}}$ and combine them to construct a matrix representation item. Here, d_{Item} is the dimensionality of item representations.

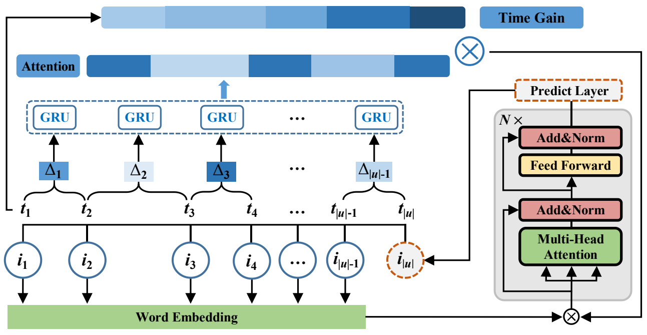

Fig. 2. The architecture of our proposed SDAT.

$$item \leftarrow lookup(Item). \tag{1}$$

The original Transformer model contains no recurrence and no convolution, and sequence information is very important. Therefore, sin function and cos function are used to encode the sequence to represent the relative position of different items [6]. However, the results of the superposition of sin function and cos function are fixed values, which represent only the absolute information of position. To deal with the relative position information of items, we introduce time intervals to quantify the relative position, thereby enhancing the ability of the Transformer to obtain relative position information.

C. Time Interval Embedding

Time Interval Embedding is an important part of our model and highly related to the item representation embedding. Each time interval represents the distance between two adjacent items. In order to find the time interval embedding, we use the method of embedding GRU in attention. GRU can maintain the order between items, and attention can describe the distance between adjacent items.

GRU [11] is most commonly used for RSs. On our own industrial dataset, we find that GRU is slightly better than LSTM and it runs faster. A GRU aggregates information on a token-by-token basis in sequential order; each token is associated with a hidden vector. Based on this important feature, we use GRU to calculate the time interval information between items in the recommendation process. GRU can analyze the time of items in the sequence one by one. The time factor will change with the position of items in the sequence, which means that an item in the sequence will produce different significant effects compared with other items.

Based on the above important properties, we design the GRU with attention to calculate the time interval information in the recommendation process and solve the disadvantage of items' absolute positional encoding. We also introduce the exponential time gain function to optimize the RSs.

Given a time series $\Delta = \{\Delta_1, \Delta_2, \ldots, \Delta_{|u|-1}\}$, Δ denote the current input; and $H_{t-1} = (h_1, h_2, \ldots, h_{|u|-2})$ denote the previous hidden states. In this paper, we formulate the model with GRU, whose equations are as follows:

$$r_t = sigmoid(W_r[h_{t-1}, \Delta_t]), \tag{2}$$

$$z_t = sigmoid(W_z[h_{t-1}, \Delta_t]), \tag{3}$$

$$\widetilde{h_t} = tanh(W_{\widetilde{h}}[r_t \odot h_{t-1}, \Delta_t]), \tag{4}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \tag{5}$$

where $\odot$ is the element-wise(Hadamard) product with the gate value. r_t and z_t represent reset gate and update gate respectively. Through the sigmoid function, the data can be transformed into values in the range of 0-1 to act as a gating signal. The reset gate controls how much information of the previous states are written to the current candidate set $\widetilde{h_t}$. The smaller the reset gate, the less information of the previous states are written. On the other hand, with the help of the reset gate in the GRU model, the time information will have different importance, and the corresponding item will have different importance; meanwhile, it also reflects that the user's interest will change with time.

The update gate is used to control the extent to which the state information of the previous time is brought into the current state. The larger the value of the update gate, the more state information of the previous time is brought in.

The $\widetilde{h_t}$ here mainly contains the currently entered Δ_t data. Adding the $\widetilde{h_t}$ to the current hidden state $\widetilde{h_t}$ is equivalent to remembering the state of the current moment. It represents the information of the whole sequence before the current position. We think that in addition to considering the sequence, the time effect of the sequence should also be considered, that is, the time interval between adjacent items in the sequence represents the degree of association between the items. So we add the time interval attention here. It depends on both itself and the past hidden states, i.e., at time interval step t, the model computes the relation between Δ_t and $\Delta = \{\Delta_1, \Delta_2, \ldots, \Delta_{t-1}\}$ through $\tilde{h}_{t-1}$ and h_t with an attention layer:

$$a^{\Delta_t} = \tanh\left(\left(h_t + \tilde{h}_{t-1}\right)\Delta_t\right), \tag{6}$$

where a^{Δ_t} represents the attention weight of Δ_t. This yields a probability distribution over the hidden state vectors of the previous tokens [21]. They are fixed values, but in the real world, as time goes by, when a user interacts with new items, the weight of the old items should be changed. That is, the attention weight of each old time interval should also be appropriately changed.

$$s^{\Delta_t} = \frac{v^t a^{\Delta_t}}{\sum_{i=1}^{t} v^t a^{\Delta_i}}, \tag{7}$$

where v^t is a weight matrix for optimization. Eq. (7) maps attention weight a^{Δ_t} to $[0, 1]$, and constrains the sum of the attention weights to be 1, the attention weight of the old-time interval is dynamically updated. s^{Δ_t} represent the new attention weight of Δ_t. We can compute an adaptive summary vector h_t for the hidden states denoted by s^{Δ_i} and h:

$$h_t = \sum_{i=1}^{t} s^{\Delta_i} h_i. \tag{8}$$

The dynamic adaptive change values h_t ensures that the attention vectors are modified at each time interval step, increasing the computational accuracy and stability of the attention generation process. Then, we use the new h_t in the next GRU cell as a new input.

We use attention for inducing relations between time intervals. These relations are differentiable, and it is part of the representation learning of the items. We treat attention as an optimized sub-module in the model by stacking multiple hidden layers in an alternating manner, and it also has an item-by-item structured relational reasoning module.

The choice of using time interval attention reflects our first intuition: Increasing the recommendation probability of combined products. That is to say, it is a very common phenomenon for users to buy multiple products at the same time, so the items in the sequence are not in order, but this situation is not suitable for GRU's left-to-right modeling idea. On the other hand, we have considered the Transformer model without positional encodings, but the result is even worse. The Transformer cannot identify the items interacted at the same time. Therefore, we add attention mechanism based on GRU modeling and focus on the items with small-time intervals, especially the items with no time interval, and attention mechanism gives them the greatest weight.

D. *Exponential Time Gain Embedding*

The attention mechanism is not likely going to be sufficient for RSs. The reason is that the user's recent behavior has a greater impact than the previous one [22]. When a user is about to purchase an item or service, historical items from a long time ago are unlikely to be highly relevant. This consideration reflects the common situation of RSs. Therefore, we take $Time = \{t_1, t_2, \ldots, t_{|u|-1}\}$ as input and add a multiplicative exponential time gain term to each attention scores as:

$$gain\_attention_t = -s^{\Delta_t} \odot \exp^{\frac{Time_t}{\mathrm{mean}(Time)}}. \tag{9}$$

As shown in Fig. 2, $\Delta_1 < \Delta_2$, we can get $s^{\Delta_1} < s^{\Delta_2}$, but it's the opposite of what we want. We take the opposite weight to the attention result s^{Δ_t}. We leverage the properties of this function to expand our models by increasing the importance of the items over time.

The choice of using time information reflects our second intuition: The exponential time gain function can enhance the weight of recent records and reduce the influence weight of early records. In other words, the weight of time information not only needs to deal with the relative time distance between items but depends on the timestamp corresponding to the item.

Using the generated $gain\_attention$ as the weights of the user sequence, the embedding vector X of all items in the user sequence is obtained as follows:

$$X = gain\_attention \odot item. \tag{10}$$

E. Transformer Encoder Layer

To learn the global dependency between the items and the time location representation, we employ the self-attention layers composed of self-attention blocks and a feed-forward networks.

1) Self-Attention Blocks

Given the input word representation $X \in \mathbb{R}^{L \times dh}$, where L is the sequence length, d is the input dimension of each head and h is the number of attention heads, we use the linear projection to acquire the query Q, key K and value V. Denoting splitted inputs for i-th head as $X_i \in \mathbb{R}^{L \times dh}$ where $i \in \{1, \ldots, h\}$, single-head self-attention can be calculated as:

$$\begin{gathered} Q_i, K_i, V_i = X_i W_q, X_i W_k, X_i W_v, \\ \text{Attention}\left(Q_i, K_i, V_i\right) = \operatorname{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i, \end{gathered} \tag{11}$$

where learnable weights $\{W_q, W_k, W_v\} \in \mathbb{R}^{d \times d}, \sqrt{d_k}$ is a scaling factor to prevent the effect of large values. We concatenate H independent attention heads and perform attention operation in parallel. It can be calculated as:

$$\begin{gathered} \text{Multi-head}(Q, K, V) = Concat\left(\text{head}_1, \ldots, \text{head}_h\right), \\ \text{where } \text{head}_i = \text{Attention}\left(QW_Q^i, KW_K^i, VW_V^i\right). \end{gathered} \tag{12}$$

The final value produced by Eq.(12) represents a summary of past items of information on multiple time scales. Previous work has shown that it is beneficial to jointly attend to information from different representation subspaces at different positions. We use multi-head attention to process multiple time scales of information at different locations effectively. On the contrary, with a single attention head, averaging inhibits the performance of the model [6]. Therefore, we use h independent attention heads to concatenate the final output values into a vector and pass it to the next layer.

We also use several sub-layers, including one for layer normalization, one for dropout, a fully-connected feed-forward layer, and a residual connection layer [6, 23] in each encoder.

2) Feed-Forward Network

Since self-attention is a linear model, to endow the model nonlinearity, we apply a fully connected feed-forward network to each position in the sequence and consider interactions between different latent dimensions. A fully connected feed-forward network consists of two linear transformations with a ReLU activation in between. Given a sequence of vectors $[\text{head}_1, \ldots, \text{head}_n]$, the computation of a position-wise FFN sub-layer on any head is defined as:

$$\operatorname{FFN}\left(\text{head}_i\right) = \operatorname{ReLU}\left(\text{head}_i W_1 + b_1\right) W_2 + b_2, \tag{13}$$

where W_1, W_2, b_1, and b_2 are parameters. We also use normalization to normalize the features of each input and use dropout to avoid overfitting [8].

F. Prediction Layer

The last component of the SDAT method predicts the user's interactive item at time step $t_{|u|}$, i.e. our model predicts the masked item $i_{|u|}$ according to the feed-forward network output. SDAT optimizes the output result through the Adam optimizer [6]. SDAT generates the recommendation probability of each item through a fully connected network and a softmax function. It should be emphasized that the item set with small time interval in the sequence, the attention model will calculate that the items in the set have strong relevance, especially the item set with time interval equal to 0 is regarded as combination or binding items. Furthermore, even if it is an item that the user has not interacted with, our model will refer to the item set of other users and recommend suitable and new item to the user.

It is worth mentioning that our model does not have decoder blocks like the original Transformer. Because our model is used in RSs, compared with the translation task, it does not have the feature of one-to-one translation. Moreover, we have done additional experiments and found that the performance is a negligible reduction after removing the decoder, and the training time is greatly improved due to the reduced complexity of the model.

IV. EXPERIMENTS

In this section, we sort the sequences by time, take the last item as the ground truth, and generate the sequence representation for the remaining part. In other words, each user has at least two interactive items. Furthermore, we add $zero$-padding to the right side of the sequence to ensure that each user has the same sequence length. And then, we compare our model with the start-of-the-art baselines and analyze the results. Finally, we explore the role of components and the impact of sequence length on our model.

A. Evaluation Metrics

RSs generate a recommendation list that usually contains K items sorted by predicted scores for each sequence. The recommendation list should contain the actual items that the user interacts with next. We adopt Hit Rate (HR@K), Mean Reciprocal Rank (MRR@K), and Normalized Discounted Cumulative Gain (NDCG@K) [24] to evaluate the performance.

HR@k: It is short for Hit Ratio, which shows whether the target item is in the recommended list or not. It does not take into account the actual ranking of the items.

MRR@k: It is short for Mean Reciprocal Rank, which gives the reciprocal of the ranking in the result as its accuracy and then takes the average of all the questions.

NDCG@k: It is short for Normalized Discounted Cumulative Gain, which considers the HR and the orders of ranking. The results of high relevance will affect the final indicator score more than the results of general relevance.

The above three evaluation Metrics values range from 0 to 1, with 1 being a perfect score.

B. Datasets

We study the effectiveness of our proposed model SDAT on three real-world datasets, i.e., Yihaohugong, Taobao, and Netflix.

Yihaohugong is a company that focuses on providing service items for the elderly via its APP in the newly-developing industry of elderly service and expects to get the algorithm help for the RSs through cooperating with our laboratory. The company can provide desensitization information about users purchasing elderly service items from January 01, 2019 to April 05, 2021. The dataset is organized similarly to MovieLens, i.e., each line represents a specific user-item interaction, which consists of user ID, item ID, and timestamp.

Taobao User Behavior [25] is a dataset of user behaviors from Taobao, for recommendation problems with implicit feedback. We select the purchase behavior data from the original Taobao user behavior data set from November 25 to December 3, 2017. We choose the product category as items.

Netflix is an American global provider of streaming films and television series [26]. The Netflix dataset is a subset of the original data set released for the purpose of the Netflix Prize. The Netflix dataset consists of 100 million ratings by 480,189 users to 17,770 movies from 1999 to 2006. Because the Netflix dataset is quite large and training on a subset of the data can still achieve good results, we randomly use 1/10 of movies.

We filter out users with more than two interactive items and less than 100 interactive items on the three datasets. The statistical information of the three datasets after pre-processing is shown in Table. 1 and Fig. 3.

TABLE I
THE STATISTICS OF DATASETS

Dataset	#Users	#Items	#Interactions	Avg.len
Yihaohugong	2,737	127	62,968	23.0
Taobao	286,610	6,946	2,015,839	7.03
Netflix	351,288	1,777	7,348,908	20.9

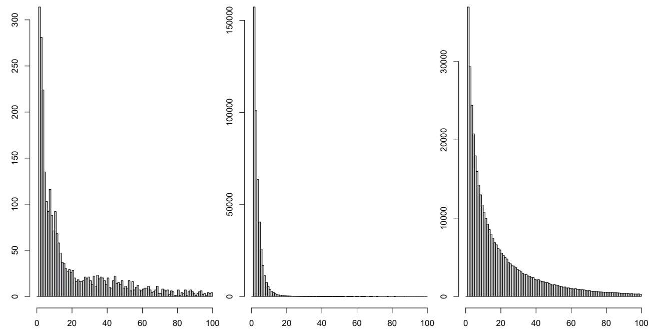

Fig. 3. Frequency (y-axis) vs. the sequence length (x-axis) on Yihaohugong(left), Taobao(middle), and Netflix(right) datasets.

C. Baselines

We compare our model with the following methods.

1) FPMC [4]: This method contains both a common Markov chain and the normal matrix factorization for the next item recommendation.

2) GRU4Rec [11]: This method uses GRU with a ranking loss function to model user sequences for session-based recommendations.

3) STAMP [13]: This method is a short-term memory priority model which captures the user's long-term preference from previous clicks and the current interest of the last clicks in a session.

4) SASRec [18]: This method is a self-attention-based Transformer model which seeks to identify which items are 'relevant' from a user's action history and use them to predict the next item.

5) TiSASRec [19]: It is based on SASRec, which models the absolute position and time interval of items in the sequence.

6) AttRec [27]: This method utilizes a self-attention mechanism to infer the item-item relationship from the user's historical interactions. The model captures the user's long-term preference from previous clicks and the current interest of the last clicks in a session.

7) Caser [28]: This method models a user's information in the current session with the help of Recurrent Neural Networks (RNNs) and an attention mechanism and exploits collaborative information to better predict the intent of current sessions by investigating neighborhood sessions.

8) SR-GNN [14]: This method models session sequences as directed graphs and uses the graph neural network to learn the item for recommendations.

9) Bert4rec [24]: This method employs deep bidirectional self-attention to model user behavior sequences.

D. Performance Comparisons

In this section, we compare the overall performance of our model with some related baselines. Table 2 shows the experimental results of all methods under the HR@5, MRR@5, and NDCG@5 evaluation indicators for three datasets.

Our model improves the performance significantly against all the baselines and achieves state-of-the-art performance. The performance of the FPMC method on the three datasets maintains a stable prediction effect without fluctuating results.

Judging from the results of the Yihaohugong dataset, GRU4Rec has better performance than AttRec, while the results on the Taobao and Netflix datasets are the opposite. The advantage of AttRec is that in the long-distance sequence. The length of user-item interaction sequences in Taobao data set is generally short, leading to a decrease in the model's performance based on the attention mechanism.

Because Taobao data set is sparser than Yihaohugong and Netflix datasets, this causes Caser to perform worse than GRU4Rec and STAMP. STAMP mainly focuses on recent items, which is one of the main reasons for the good performance. It shows that prioritizing items recently have an important influence on RSs. Furthermore, SASRec performs distinctly better than STAMP and GRU4Rec, suggesting that the attention mechanism is a more powerful method for the sequential recommendation. SR-GNN uses GNN to capture more complex and implicit connections between items and achieve great experimental results.

The performance of SASRec, TiSASRec, and Bert4rec is relatively close to our method. We think it is because the concept is similar to SASRec, TiSASRec, and Bert4rec, and

TABLE II
THE PERFORMANCE OF DIFFERENT MODELS ON THE THREE DATASETS, AND THE BEST MODEL IN EACH COLUMN IS IN BOLD.

	Yihaohugong			**Taobao**			**Netflix**		
	HR@5	**NDCG@5**	**MRR@5**	**HR@5**	**NDCG@5**	**MRR@5**	**HR@5**	**NDCG@5**	**MRR@5**
FPMC	0.1583	0.1049	0.0864	0.1357	0.0714	0.0493	0.1484	0.0913	0.0725
GRU4Rec	0.2334	0.1749	0.2277	0.1893	0.0976	0.0665	0.1797	0.1183	0.0977
STAMP	0.3100	0.2618	0.2455	0.3164	0.1736	0.1272	0.1992	0.1415	0.1223
SASRec	0.5286	0.4102	0.3995	0.3359	0.2233	0.1846	0.3789	0.2192	0.1677
TiSASRec	0.5370	0.4257	0.4020	0.3438	0.2331	0.1953	0.3906	0.2757	0.2367
AttRec	0.2232	0.1878	0.1739	0.2023	0.1039	0.0707	0.1875	0.1234	0.1021
Caser	0.2296	0.1815	0.1621	0.1740	0.0877	0.0587	0.1562	0.1071	0.0913
SR-GNN	0.4994	0.3581	0.3115	0.3242	0.2093	0.1702	0.4023	0.2431	0.1915
Bert4rec	0.5538	0.3814	**0.4242**	0.3594	0.2324	0.1899	0.3945	0.2187	0.1605
SDAT	**0.5763**	**0.4511**	0.4113	**0.3750**	**0.2656**	**0.2281**	**0.4219**	**0.2936**	**0.2514**

TABLE III
ABLATION STUDY FOR SDAT(HR@5 AND MRR@5). TIME GAIN FUNCTION AND GRU-ATT OF SDAT GAIN PERFORMANCE BENEFITS.

	Yihaohugong		**Taobao**		**Netflix**	
	HR	**MRR**	**HR**	**MRR**	**HR**	**MRR**
Transformer	0.494	0.304	0.277	0.139	0.305	0.167
-GRU-Att	0.543	0.376	0.305	0.160	0.339	0.183
-Time Gain	0.563	0.403	0.324	0.170	0.367	0.196
SDAT	**0.576**	**0.411**	**0.375**	**0.228**	**0.422**	**0.251**

both are based on Transformer. SASRec emphasizes attention and negative samples; based on SASRec, TiSASRec designed a novel time interval perception self-attention mechanism to learn the weight of different items. Bert4rec focuses on bidirectional sequence construction. We emphasize sequence order and attention. The difference is that the GRU-Att used by SDAT processes the time interval information, which more accurately considers the order and relative distance between items. The adaptive function is more suitable for real-world user orders that are constantly superimposed without recalculating each user. Another difference is that we have strengthened the importance of recent items.

On the one hand, it implies that our thinking is correct; that is, time information has a significant influence, SDAT can more adequately learn item embeddings. On the other hand, it is worth considering the common phenomenon of the sequential recommendation, such as the recent item plays a vital role in the sequential recommendation.

E. Ablation Experiment

Table 3 shows the performance of two key components on the three datasets. We find that the GRU-Att after ablation has the most significant impact, the Time Gain function has a relatively general impact, especially in large datasets such as Taobao and Netflix. The above results indirectly show that the sequential mode plays a positive role in the recommendation.

We study the impact of the hidden dimension d on recommendation performance. Fig. 4 shows the comparison of the sequence length between Transformer and SDAT under HR@5 for the hidden dimension d of 60, 120, and 180. We have observed some phenomena from this figure. The most obvious phenomenon is that as the dimensionality increases, the performance of the Transformer tends to converge, and there is a significant difference in performance from SDAT. A larger hidden dimension does not necessarily lead to better model performance, and a suitable hidden dimension is 120.

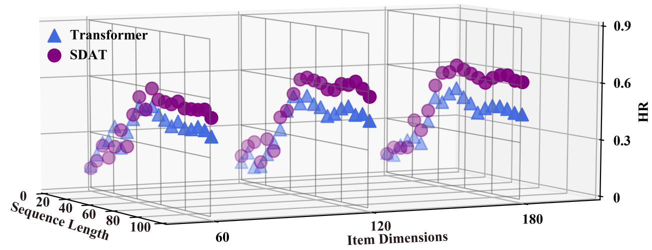

Fig. 4. HR@5(z-axis) effectiveness of sequence length(x-axis) and item dimensions(y-axis) on Yihaohugong datasets.

F. Impact of the sequence length

We also investigate the effect of the sequence length on the model's recommendation performances. Following the method of literature [13, 14], we use the average length of the Yihaohugong as the basis for dividing short-term and long-term sequences of users. Items containing less than 23 are regarded as short-term, while others are regarded as long-term. Considering the size of the dataset and the proportion of long sequences, we choose Yihaohugong as the dataset for the model to evaluate the length of the sequence.

As Fig. 4 shows recommendation performances with different lengths on Yihaohugong, as the sequence length increases, our model performs better than the original Transformer model in different item dimensions. The time gain function and time interval are effective in the short sequence, because SDAT and Transformer have comparable performance. Still, due to the few influencing factors in the short sequence, SDAT has no obvious advantage.

When the sequence length is greater than 23, the performance of SDAT on the long sequence is significantly better than that of the Transformer. The reason may be that the long sequence contains more potential information about the user characteristics, and SDAT can capture and utilize them more efficiently. On the one hand, it is due to the positive impact of the time gain function on recent items. On the other hand, the combination of GRU-Att highlights the items with short time intervals in the sequence, and from another perspective, it filters out sequence some possible noise items.

The size of the data set and the distribution of unbalanced items also affect the model's performance. We randomly select sub-samples of Taobao and Netflix datasets according to the ratio of 1/10, 1/5, 1/2. We find that the performance advantage of SDAT compared to Transformer increases with the sample improvement. We sample some sequences with obvious performance gaps and find that some of them are concentrated on items with lower label frequencies.

V. CONCLUSION

In this paper, we present an extension to Transformer encoder blocks that can be used to incorporate relative time distance information for sequences, which improves performance for RSs, and it can also be transplanted to other sequence models. SDAT considers not only the importance of recent items but also the combined items with high relevance. These considerations are consistent with the reality of the RSs, and help users quickly find interesting and high-quality items. Our experimental results show that SDAT performs the state-of-the-art methods on three real-world datasets in terms of HR, MRR, and NDCG.

ACKNOWLEDGMENT

This work was supported by the Key Projects of Philosophy and Social Sciences Research of Chinese Ministry of Education [grant number 19JZD021], the National Natural Science Foundation of China [grant number 71771210].

REFERENCES

[1] Y. Zhang, N. Gao, and J. Chen, "A practical defense against attribute inference attacks in session-based recommendations," in *2020 IEEE International Conference on Web Services, ICWS*, 2020, pp. 355–363.

[2] M. Sun, J. Yuan, Z. Song, Y. Jin, X. Lu, and X. Wang, "POEM: position order enhanced model for session-based recommendation service," in *2020 IEEE International Conference on Web Services, ICWS*, 2020, pp. 126–133.

[3] M. Wang, P. Ren, L. Mei, Z. Chen, J. Ma, and M. de Rijke, "A collaborative session-based recommendation approach with parallel memory modules," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 345–354.

[4] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *Proceedings of the 19th International Conference on World Wide Web, WWW*, 2010, pp. 811–820.

[5] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 5:1–5:38, 2019.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2017, pp. 5998–6008.

[7] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL*, vol. 1, 2019, pp. 2978–2988.

[8] A. Luo, P. Zhao, Y. Liu, F. Zhuang, D. Wang, J. Xu, J. Fang, and V. S. Sheng, "Collaborative self-attention network for session-based recommendation," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI*, 2020, pp. 2591–2597.

[9] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai, "What to do next: Modeling user behaviors by time-lstm," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*, 2017, pp. 3602–3608.

[10] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the Tenth International World Wide Web Conference, WWW*, 2001, pp. 285–295.

[11] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *4th International Conference on Learning Representations, ICLR*, 2016.

[12] L. Hu, L. Cao, S. Wang, G. Xu, J. Cao, and Z. Gu, "Diversifying personalized recommendation with user-session context," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*, 2017, pp. 1858–1864.

[13] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, "STAMP: short-term attention/memory priority model for session-based recommendation," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD*, 2018, pp. 1831–1839.

[14] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, vol. 33, no. 1, 2019, pp. 346–353.

[15] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun, "Temporal recommendation on graphs via long- and short-term preference fusion," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010, pp. 723–732.

[16] C. Chen, H. Yin, J. Yao, and B. Cui, "Terec: A temporal recommender system over tweet stream," *Proc. VLDB Endow.*, vol. 6, no. 12, pp. 1254–1257, 2013.

[17] Y. Liu, C. Liu, B. Liu, M. Qu, and H. Xiong, "Unified point-of-interest recommendation with temporal interval assessment," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1015–1024.

[18] W. Kang and J. J. McAuley, "Self-attentive sequential recommendation," in *IEEE International Conference on Data Mining, ICDM*, 2018, pp. 197–206.

[19] J. Li, Y. Wang, and J. McAuley, "Time interval aware self-attention for sequential recommendation," in *Proceedings of the 13th International Conference on Web Search and Data Mining, WSDM*, 2020, pp. 322–330.

[20] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser, "Universal transformers," in *7th International Conference on Learning Representations, ICLR*, 2019.

[21] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP*, 2016, pp. 551–561.

[22] C. Pu, Z. Wu, H. Chen, K. Xu, and J. Cao, "A sequential recommendation for mobile apps: What will user click next app?" in *2018 IEEE International Conference on Web Services, ICWS*, 2018, pp. 243–248.

[23] A. Ghosh, N. T. Heffernan, and A. S. Lan, "Context-aware attentive knowledge tracing," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020, pp. 2330–2339.

[24] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1441–1450.

[25] H. Zhu, X. Li, P. Zhang, G. Li, J. He, H. Li, and K. Gai, "Learning tree-based deep model for recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD*, 2018, pp. 1079–1088.

[26] J. Bennett, C. Elkan, B. Liu, P. Smyth, and D. Tikk, "Kdd cup and workshop 2007," *SIGKDD Explor. Newsl.*, vol. 9, no. 2, pp. 51–52, 2007.

[27] S. Zhang, Y. Tay, L. Yao, and A. Sun, "Next item recommendation with self-attention," 2019.

[28] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *Proceedings of the 11th ACM International Conference on Web Search and Data Mining, WSDM*, 2018, pp. 565–573.