Dual Sequential Network for Temporal Sets Prediction

Leilei Sun[1], Yansong Bai[1], Bowen Du[1*], Chuanren Liu[2], Hui Xiong[3], Weifeng Lv[1]
[1]SKLSDE and BDBC Lab, Beihang University, Beijing 100083, China
[2]Department of Business Analytics and Statistics, University of Tennessee, Knoxville, USA
[3]Department of Management Science and Information Systems, Rutgers University, Newark, USA
[1]{leileisun,easonwhite96,dubowen,lwf}@buaa.edu.cn, [2]cliu89@utk.edu, [3]hxiong@rutgers.edu

ABSTRACT

Many sequential behaviors such as purchasing items from time to time, selecting courses in different terms, collecting event logs periodically could be formalized as sequential sets of actions or elements, namely temporal sets. Predicting the subsequent set according to historical sequence of sets could help us make better producing, scheduling, or operating decisions. However, most of the existing methods were designed for predicting time series or temporal events, which could not be directly used for temporal sets prediction due to the difficulties of multi-level representations of items and sets, complex temporal dependencies of sets, and evolving dynamics of sequential behaviors. To address these issues, this paper provides a novel sets prediction method, called DSNTSP (Dual Sequential Network for Temporal Sets Prediction). Our model first learns both item-level representations and set-level representations of set sequences separately based on a transformer framework. Then, a co-transformer module is proposed to capture the multiple temporal dependencies of items and sets. Last, a gated neural module is designed to predict the subsequent set by fusing all the multi-level correlations and multiple temporal dependencies of items and sets. The experimental results on real-world data sets show that our methods lead to significant and consistent improvements as compared to other methods.

CCS CONCEPTS

• **Information systems → Data stream mining**.

KEYWORDS

Temporal Sets Prediction, Set Embedding, Deep Neural Network

ACM Reference Format:
Leilei Sun[1], Yansong Bai[1], Bowen Du[1*], Chuanren Liu[2], Hui Xiong[3], Weifeng Lv[1]. 2020. Dual Sequential Network for Temporal Sets Prediction. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401124

* Corresponding Author.

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401124

1 INTRODUCTION

In practice, many sequential behaviors could be formulated as temporal sets. For example, a customer purchases a basket of items at a visit of a retail store or online store, a student selects a set of courses in each semester, or a patient takes a combination of drugs every day. Different from previous temporal data such as time series or temporal events, temporal sets are a new type of temporal data that consists of sequential sets, where each set contains a number of actions or elements with the same timestamp, see Figure 1. Prediction of temporal sets could help us make better producing, scheduling, or operating decisions. However, most of the existing methods were designed for predicting time series or temporal events[3, 15], which could not be directly used for temporal sets prediction.

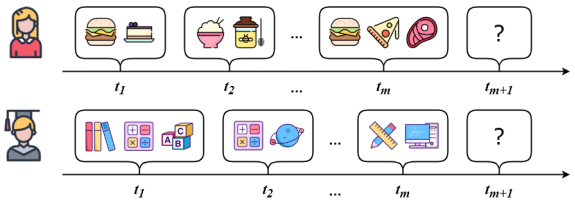


Figure 1: Illustration of temporal sets prediction.

Recently, some methods for temporal sets prediction have been proposed. Yu et al. [36] proposed a dynamic recurrent model for next basket prediction, where a max-pooling operation was used to extract a representation of set according to the items it contains and a recurrent architecture was used to learn the temporal dependence of the sequential sets. Hu et al. [11] provided a set-to-set methods for temporal sets prediction, where average-pooling was used to get a representation of set by aggregating all the elements within it, and a set-based attention module was designed to predict the subsequent set. It can be known that these methods first adopt a pooling-based set embedding module to get representations of sets and then utilize a recurrent neural network to capture the temporal dependencies of sets.

Though these methods provide valuable insights to solve the problem of temporal sets prediction, there are still a lot of problems needs to be investigated in a further step. First, most of the existing methods mainly focus on set-level sequence representation, while there are actually multiple semantic correlations among items, for instance, some items appear frequently within a same set, while the other items are always purchased in different sets, which means these items may have alternative or complementary relationships. It is no doubt that we could achieve a better representation of temporal sets by taking these item-item, item-set, and set-set correlations into account, and which will finally improve the set prediction accuracy.

But how to learn a multi-level representation of a sequence of sets is a challenging problem. Second, it is difficult to capture the temporal dependencies among sequential sets, some sets may reflect an individual's regular purchases or actions, they have periodic dependencies; while the other sets reflect an individual's successive purchases or actions, they are strongly affected by the most recent sets. In this case, how to find the most valuable information for further set prediction according to historical sequence of sets is also a challenging problem. Third, the predicted set may be a possible combination of all the available items or actions, which is affected by all the correlations and temporal dependencies mentioned above. So how to generate a set prediction result by fusing all the multi-level correlations and multiple temporal dependencies among items and sets is a difficult task either.

To address the above issues, this paper proposes a novel temporal sets prediction method, namely, Dual Sequential Network for Temporal Sets Prediction (DSNTSP). To capture the multi-level representations of set sequences, an item-level representation learning module is first proposed, which adopts a transformer-based framework to learn both intra-set and inter-set item correlations. Then, a set-level representation learning module is provided, which could turn a set with unfixed-size into a vector with fixed length. Simultaneously, which could achieve a semantic representation of a set by considering both contextual preceding and subsequent sets with multi-head attentions. Third, a co-transformer learning module is proposed to learn the multiple temporal dependence of items and sets, for instance, sequential associations from item to item, from set to item, from set to set, or from item to set. Last, a gated neural module is designed to fuse all the multi-level correlations and temporal dependencies into a fixed hidden state, and predicts the subsequent set according to the current state.

In summary, our contributions could be summarized as follows:

- We provide a multi-level representation method for temporal sets, which could capture the multi-level correlations of items and sets. By considering these correlations, a semantic representation of set sequence could be achieved, and the set prediction accuracy could be improved consequently.
- We propose a co-transformer module to capture the temporal dependencies of items and sets and present a gated neural module to ensemble the multi-level correlations and multiple temporal dependencies of items and sets.
- We conduct experiments on four real-world data sets, which shows that our method outperforms all the baseline methods.

The rest of the paper is organized as follows. Section 2 surveys the related work. Section 3 presents the definitions and formalization. Section 4 introduces the proposed model. We present experiments in section 5 and conclude this research in section 6.

2 RELATED WORK

This section reviews the existing literature related to our work.

Temporal Sets Prediction. Recently, there has been a surge of interest in temporal sets prediction. Most of the existing methods use pooling method to represent sets and use recurrent based method to capture temporal dependencies. Yu et al. [36] proposed a dynamic recurrent model for next basket prediction, which uses max pooling operations to get representations of baskets and then feed the sequence of baskets into the Recurrent Neural Network (RNN) structure. Factorizing Personalized Markov Chains (FPMC) [25] proposed by Rendle et al. is a traditional method for next basket recommendation. Benson et al. [2] proposed the Correlated Repeated Unions (CRU) model to learn the repeat behaviors in the set sequences. Sets2Sets [11] is an encoder-decoder framework which takes sequential sets as input, uses the average pooling to encode the sets, and provides a repeated-element-specified element-element interaction component to further improve the performance.

Set Embedding. Existing set embedding methods for temporal sets prediction [11, 36] only use some kind of pooling operations to get the representations of sets, which could not capture the rich semantic information among the items and sets. Vinyals et al. [31] designed a LSTM-based network with memories coupled to encode sets. Zaheer et al. [37] proposed a fundamental architecture called DeepSets based on permutation invariance and equivariance properties. Murphy et al. [22] proposed a general method for pooling operations on sets, namely Janossy Pooling, which expresses a permutation-invariant function as the average of a permutation-sensitive function applied to all reorderings of the input sequence. Lee et al. [14] introduced the Set Transformer, an attention-based set-input neural network architecture. Skianis et al. [27] proposed a neural network architecture, namely RepSet, which learns the set representations by solving a series of network flow problems. Meng et al. [20] develop a hierarchical sequence-attention framework to learn inductive Set-of-Sets (SoS) embeddings that are invariant to set permutations.

Temporal Dependence Learning. Recurrent Neural Network (RNN) [16] is a class of deep neural network used for sequential data, which has been applied in many tasks such as speech recognition [8], machine translation [1] and image caption [32]. Long Short-term Memory (LSTM) [10] is a type of RNN which can help with the vanishing gradient problem. Cho et al. [5] proposed a slightly simplified version of LSTM called Gated Recurrent Units (GRU). Recently, RNNs equipped with different attention mechanism [1, 18, 21, 34] have been proposed to improve the performance by guiding the networks to focus on particular parts of the sequence. Shen et al. [26] proposed a Directional Self-Attention Network (DiSAN) to process sentences, which is based solely on the attention mechanism, without any RNN/CNN structure. Transformer is first introduced by [30] for machine translation task in natural language processing, which is built on multi-head self-attention. Like RNNs, Transformers are used to handle sequential data. Recently, transformer has become the basic building block of most state-of-the-art architectures in NLP [6, 29, 33]. There is also other research use temporal graph to model the temporal dependence among sequences [17].

Sequential Recommendation. Temporal sets prediction can be applied to sequential recommendation. Hidasi et al. [9] proposed a GRU-based neural network for sequential recommendation, which is the first model that applied RNN to sequential recommendation. Yao et al. [35] developed a novel POI recommendation method which incorporates the degree of temporal matching between users and POIs. Quadrana et al. [24] proposed a hierarchical RNN based model to address the problem of personalizing session-based recommendation. Zhou et al. [39] proposed a Deep Interest Evolution Network to model interest evolving process for click-through rate prediction, which is an improvement of Deep Interest Network.

Recently, self-attention mechanism has been widely used in sequential recommendation. Zhou et al. [38] proposed an attention based recommendation framework, called ATRank, which considers heterogeneous user behaviors. Chen et al. [4] uses the transformer model to capture the sequential signals underlying user's behavior sequences for recommendation in Alibaba. Feng et al. [7] proposed a novel CTR model called DSIN, which uses self-attention mechanism with bias encoding to extract user's interests in each session. Sun et al. [28] proposed a deep bidirectional sequential model called BERT4REC for sequential recommendation. Lv et al. [19] developed a self-attention based sequential deep matching model to capture user's dynamic preferences by combining short-term sessions and long-term behaviors.

3 FORMALIZATION

Let $U = \{u_1, \cdots, u_{|U|}\}$ denote a collection of users and $V = \{v_1, \cdots, v_{|V|}\}$ indicate all the available items. For a user, we use a tuple (S_i, t_i) to represent a user-set interaction, where $S_i \subset V$ is a set of items interacted by the user at time t_i. We formulate the temporal sets prediction problem as follows:

Definition 3.1. **Temporal Sets Prediction Problem**: For a user, given the user's historical behaviors represented as a sequence of sets, $TS = \{(S_1, t_1), \cdots, (S_m, t_m)\}$, the target of temporal sets prediction is to achieve the subsequent set S_{m+1} according to the given sequence, that is, $S_{m+1} = f(TS)$.

A key problem in temporal sets prediction is how to represent the sets as low-dimensional structured vectors because the sets could be various in cardinality and usually permutation-invariant. To consider the above characteristics, we define the set embedding function as follows:

Definition 3.2. **Set Embedding**: A function $f: \mathcal{P}(V) \rightarrow \mathbb{R}^d$ is a mapping from a set S to a size-fixed vector $\mathbf{s}$, where $\mathbf{s}$ is invariant to any permutation of the items in the set. Formally, for any permutation π of a set $S \subset V$, $f(S) = f(S_\pi)$ if $S = \{v_1, \cdots, v_{|S|}\}$ and $S_\pi = \{v_{\pi(1)}, \cdots, v_{\pi(|S|)}\}$.

4 PROPOSED METHOD

In this section, we present our model for temporal sets prediction, namely DSNTSP (Dual Sequential Network for Temporal Sets Prediction), whose architecture is shown in Figure 4. Our DSNTSP first learns the item-level representations and the set-level representations respectively by using multi-head self-attention mechanism. Then we develop a co-transformer module to learn the temporal dependencies of sets from two sequences. Finally, a gated neural module is designed to fuse the item-level and set-level representations learned in previous steps. In the following parts, we introduce the above modules one by one.

4.1 Item-level Representation Learning

Existing temporal sets prediction methods mainly focus on learning set-level representation for temporal sets, which makes it difficult to achieve satisfactory performance, because the items in sets could be correlated with each other not only in the same set, but also among different sets. In order to learn the multiple relationships, we first propose an Item-level Sequential Model (ISM) to learn items

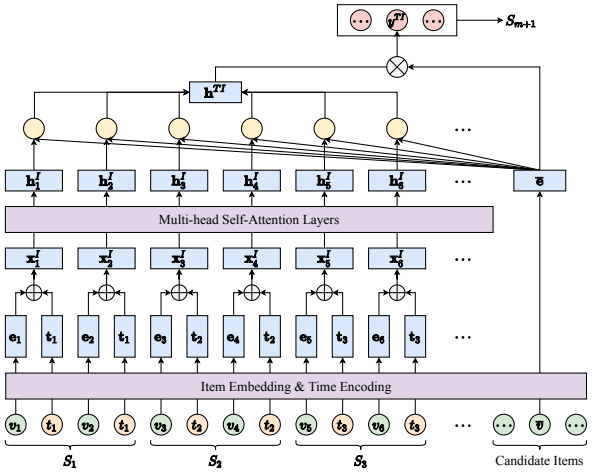


Figure 2: Item-level representation learning.

relationships among different sets, whose architecture is shown in the Figure 2. ISM can learn the item-level representations of set sequences by using multi-head self-attention mechanism. Formally, given the user-set interactions $TS = \{(S_1, t_1), \cdots, (S_m, t_m)\}$, we can get user-item interactions $TI = \{(v_1, t_{\phi(1)}), \cdots, (v_n, t_{\phi(n)})\}$, where $v_i \in S_{\phi(i)}$ is the item interacted by the user at time $t_{\phi(i)}$. Let m denote the size of TS and n denote the size of TI, thus we could conclude that $n = \sum_{i=1}^{m} |S_i|$. The user-item interactions TI is used as the input of ISM, which is composed of three parts: an item embedding and time encoding layer, multi-head self-attention layers and an item-oriented attention layer. Next, we will introduce these three parts step by step.

4.1.1 Item Embedding and Time Encoding Layer. We first employ an item embedding layer to embed each item into a low-dimensional vector. Formally, let $\mathbf{W}_e \in \mathbb{R}^{d \times |V|}$ denote the embedding matrix for items, where d is the embedding dimension. Each input item v_i is encoded as a one-hot column vector $\mathbf{v}_i \in \mathbb{R}^{|V|}$, where only the v_i-th value is 1 and other values are zeros. Then $\mathbf{e}_i = \mathbf{W}_e \mathbf{v}_i$ denotes the embedding vector for item v_i.

In order to the make use of time information in the set sequences, we apply time encoding to the item embedding vectors, which is similar as the positional encoding method mentioned in [30]. Formally, the time encoding of item v_i is computed as

$$\mathbf{t}_{\phi(i)}[2j] = \sin\left(\left(t_{m+1} - t_{\phi(i)}\right)/10000^{2j/d}\right), \tag{1}$$

$$\mathbf{t}_{\phi(i)}[2j+1] = \cos\left(\left(t_{m+1} - t_{\phi(i)}\right)/10000^{2j/d}\right), \tag{2}$$

where $\mathbf{t}_{\phi(i)} \in \mathbb{R}^d$, $\mathbf{t}_{\phi(i)}[j]$ represents the j-th value of the time encoding vector, and t_{m+1} is the prediction time. Then, for each item v_i, its representation $\mathbf{x}_i^I$ is constructed by summing the corresponding item embedding vector and time encoding:

$$\mathbf{x}_i^I = \mathbf{e}_i + \mathbf{t}_{\phi(i)}, \tag{3}$$

where $\mathbf{x}_i^I \in \mathbb{R}^d$, which will be fed into the multi-head self-attention layers to capture the temporal dependencies later.

4.1.2 Multi-head Self-attention Layers. The recurrent neural network (RNN) is the most commonly used model for temporal data mining, which usually processes the sequence data in order. However, for temporal sets prediction problem, the items in the same set usually do not have an inherent ordering, making it unable to be easily handled by the RNN. Recently, multi-head self-attention mechanism has been widely used to process the sequence data, such as natural languages[6, 29, 30, 33]. Unlike RNN, self-attention mechanism doesn't require the sequence to be processed in order and it can use the positional encoding to preserve the order information, making it very convenient to learn the item-level representations of the set sequences.

Formally, We set H heads for our multi-head self-attention layers. Considering the item v_i, the correlation between the item v_i and the item v_j for the head h is defined as:

$$\alpha_{ij}^h = \frac{\exp\left(\left(\mathbf{W}_h^Q \mathbf{x}_i^I\right)^\top \mathbf{W}_h^K \mathbf{x}_j^I / \sqrt{d/H}\right)}{\sum_{k=1}^n \exp\left(\left(\mathbf{W}_h^Q \mathbf{x}_i^I\right)^\top \mathbf{W}_h^K \mathbf{x}_k^I / \sqrt{d/H}\right)}, \tag{4}$$

where $\mathbf{W}_h^Q, \mathbf{W}_h^K \in \mathbb{R}^{d/H \times d}$ are two learnable matrices. Once the attention score is obtained, we apply the multi-head self-attention upon the item v_i as:

$$\mathbf{z}_i^I = \text{LayerNorm}\left(\mathbf{x}_i^I + \mathbf{W}^O \Big\|_{h=1}^{H} \left\{\sum_{j=1}^n \alpha_{ij}^h \mathbf{W}_h^V \mathbf{x}_j^I\right\}\right), \tag{5}$$

where $\mathbf{z}_i^I \in \mathbb{R}^d$, $\mathbf{W}_h^V \in \mathbb{R}^{d/H \times d}$, $\mathbf{W}^O \in \mathbb{R}^{d \times d}$, $\|$ represents the concatenation operation and LayerNorm$(\cdot)$ is the standard normalization layer. Following, we add a point-wise feed-forward network to further enhance the model with non-linearity. The hidden state of item v_i is computed as:

$$\mathbf{h}_i^I = \text{LayerNorm}\left(\mathbf{z}_i^I + \text{FNN}\left(\mathbf{z}_i^I\right)\right), \tag{6}$$

where $\mathbf{h}_i^I \in \mathbb{R}^d$, FNN$(\cdot)$ represents the point-wise feed-forward network. The learnable parameters in Equation (4, 5) and (6) are shared across all items in the set sequence. Usually we stack multiple multi-head self-attention layers to model more complex temporal correlations in the set sequences.

4.1.3 Item-oriented Attention Layer. Let $\mathbf{H}^I = [\mathbf{h}_1^I, \cdots, \mathbf{h}_n^I] \in \mathbb{R}^{d \times n}$ denote the output of multi-head self-attention layers, where $\mathbf{h}_i^I \in \mathbb{R}^d$ is the hidden representation of the item v_i. Then, we could get the sequence representation from these item hidden representations. A popular method is to aggregate item hidden representations in $\mathbf{H}^I$ by some kind of pooling operations such as max pooling or average pooling. However, using a fixed sequence representation is not able to capture the complex dynamic patterns in the sequence. It is found that only a few number of items in the sequences are actually used to model user's preference by this method. Inspired by [39], an item-oriented attention layer is designed to make each candidate item in the item set V choose the most relevant items to use, which is formulated as follows:

$$\beta_i = \frac{\exp\left(\left(\mathbf{W}_I^Q \bar{\mathbf{e}}\right)^\top \mathbf{W}_I^K \mathbf{h}_i^I / T\right)}{\sum_{k=1}^n \exp\left(\left(\mathbf{W}_I^Q \bar{\mathbf{e}}\right)^\top \mathbf{W}_I^K \mathbf{h}_k^I / T\right)}, \tag{7}$$

$$\mathbf{h}^{TI} = \sum_{i=1}^n \beta_i \mathbf{W}_I^V \mathbf{h}_i^I, \tag{8}$$

where $\mathbf{W}_I^Q, \mathbf{W}_I^K, \mathbf{W}_I^V \in \mathbb{R}^{d \times d}$ are learnable matrices, $\bar{\mathbf{e}} \in \mathbb{R}^d$ is the embedding of the candidate item $\bar{v}$. For one candidate item, we calculate the similarity weight between each item representation and the candidate item embedding. Then the corresponding weights are normalized using the softmax function. We add temperature parameter T into softmax which will change the probability distribution of the output. Increasing T makes the distribution softer, and vise versa. Finally, the item-level representation vector $\mathbf{h}^{TI} \in \mathbb{R}^d$ for the candidate item $\bar{v}$ is a weighted sum of the item representations.

With the item-level representation $\mathbf{h}^{TI}$ and the candidate item embedding $\bar{\mathbf{e}}$, we can compute the probability of the user interacting with the target item $\bar{v}$ at a specific time t_{m+1} as

$$y^{TI} = \text{Sigmoid}\left(\bar{\mathbf{e}}^\top \mathbf{h}^{TI} + b^{TI}\right), \tag{9}$$

We will use the above equation to calculate a score for each item in the item set V. In order to speed up the calculations, we manipulate the whole item embedding matrix to calculate the sequence representations for all items:

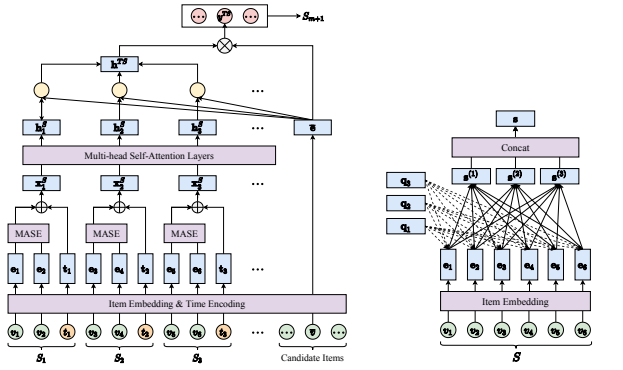
$$\mathbf{H}^{TI} = \mathbf{W}_I^V \mathbf{H}^I \text{Softmax}\left(\left(\mathbf{W}_I^K \mathbf{H}^I\right)^\top \mathbf{W}_I^Q \mathbf{W}_e / T\right), \tag{10}$$

where $\mathbf{H}^{TI} \in \mathbb{R}^{d \times |V|}$ contains item-level representation for each item in V. Then we can use these item-level representations to compute scores for all items.

4.2 Set-level Representation Learning

We also develop a Set-level Sequential Model (SSM) for temporal sets prediction, which could learn the set-level representations of set sequences. The architecture of SSM is shown in the Figure 3(a). The SSM consists of four parts: an item and time encoding layer, a multi-head attention based set embedding layer, multi-head self-attention layers and an item-oriented attention layer.

In order to learn the set-level representations, we need to compute a vector representation for each set in the set sequences. According to the Definition 3.2, the set embedding method has to be invariant to permutations of elements in the set. Existing set embedding methods for temporal sets prediction [11, 36] just aggregate embedding representations of items in the set by some kind of pooling operations such as max pooling or average pooling. However, the pooling-based set embedding method assumes that items in the set are equally important, which is not reasonable in practice. Therefore, we proposed a Multi-head Attention Based Set Embedding Method (MASE), whose architecture is shown in the Figure 3(b). Formally, given a set $S = \{v_1, \cdots, v_{|S|}\}$, we embed each item in the set S into a low-dimensional vector and then get a set of item embedding vectors $\mathbf{S} = [\mathbf{e}_1, \cdots, \mathbf{e}_{|S|}] \in \mathbb{R}^{d \times |S|}$. We set K trainable queries $\mathbf{q}_1, \cdots, \mathbf{q}_K \in \mathbb{R}^{d/K}$ to extract multifaceted



(a) Set-level Sequential Model

(b) Multi-head Attention Based Set Embedding

Figure 3: Set-level representation learning.

features from the set. We first calculate similarity between these queries and each item embedding vectors in the set S:

$$\lambda_{ij} = \frac{\exp\left(\mathbf{q}_i^\top \mathbf{W}_{S,i}^Q \mathbf{e}_j\right)}{\sum_{k=1}^{n} \exp\left(\mathbf{q}_i^\top \mathbf{W}_{S,i}^Q \mathbf{e}_k\right)}, \tag{11}$$

where $\mathbf{W}_{S,i}^Q \in \mathbb{R}^{d/K \times d}$, λ_{ij} represents the attention weight of v_j for the query $\mathbf{q}_i$. Following, we can get a set representation $\mathbf{s}^{(i)}$ for the query $\mathbf{q}_i$ by computing a weighted sum of item embedding vectors:

$$\mathbf{s}^{(i)} = \sum_{j=1}^{|S|} \lambda_{ij} \mathbf{W}_{S,i}^V \mathbf{e}_j, \tag{12}$$

where $\mathbf{s}^{(i)} \in \mathbb{R}^{d/K}$, $\mathbf{W}_{S,i}^V \in \mathbb{R}^{d/K \times d}$. Finally, we concatenate these set representations and get our final representation by using an output matrix as follows:

$$\mathbf{s} = \mathbf{W}_S^O \Big\|_{i=1}^{K} \left\{\mathbf{s}^{(i)}\right\}, \tag{13}$$

where $\mathbf{W}_S^O \in \mathbb{R}^{d \times d}$, $\mathbf{s} \in \mathbb{R}^d$ is the vector representation of set S.

Given the user-set interactions $TS = \{(S_1, t_1), \cdots, (S_m, t_m)\}$, we can get a sequence of set representations $\mathbf{TS} = [\mathbf{s}_1, \cdots, \mathbf{s}_m]$, where $\mathbf{s}_i \in \mathbb{R}^d$ is the set embedding vector of the set S_i. Then we can apply time encoding to set embedding vectors in **TS** as:

$$\mathbf{x}_i^S = \mathbf{s}_i + \mathbf{t}_i, \tag{14}$$

where $\mathbf{t}_i$ is computed by Equation (1). The sequence of set embedding vectors together with their time encodings is further fed into the multi-head self-attention layers to capture temporal dependencies of sets. After that we can get a sequence of set representations

$$\mathbf{H}^S = [\mathbf{h}_1^S, \cdots, \mathbf{h}_m^S],$$

where $\mathbf{h}_i^S \in \mathbb{R}^d$, and then we can learn the set-level representations via an item-oriented attention layer, which is described in the subsection 4.1.3. Finally we can use the set-level representations learned in previous step to predict the probability of each item interacted by the user in the future.

4.3 Dual Sequential Fusion

In this subsection, we introduce our Dual Sequential Network for Temporal Sets Prediction (DSNTSP), which can take advantage of both item-level representations and set-level representations of set sequences to better solve the temporal sets prediction problem. The architecture of DSNTSP is shown in the Figure 4.

Given the user-set interactions $TS = \{(S_1, t_1), \cdots, (S_m, t_m)\}$ and the user-item interactions $TI = \{(v_1, t_{\phi(1)}), \cdots, (v_n, t_{\phi(n)})\}$ derived from TS, we can use the networks described in the subsection 4.1 and 4.2 to learn the item-level representations and the set-level representations separately. It should be noted that these two networks share the same item embedding matrix. In order to learn the temporal dependencies of sets from both two sequences, we propose a novel co-transformer module. Co-transformer is a general model that can learn the temporal dependencies between two sequences with different lengths and different elements. We replace the multi-head self-attentions layers described earlier with our co-transformer module. Finally, we design a gated neural module to fuse the item-level representations and set-level representations learned in previous step.

4.3.1 Co-Transformer. The multi-head self-attention can learn the temporal dependencies of the sequence by taking this sequence as queries, keys and values at the same time. Inspired by this, given an item representation sequence $\mathbf{X}^I = [\mathbf{x}_1^I, \cdots, \mathbf{x}_n^I]$ and a set representation sequence $\mathbf{X}^S = [\mathbf{x}_1^S, \cdots, \mathbf{x}_m^S]$, we can take one sequence as queries and another sequence as keys and values. In this way, we can learn temporal dependencies of sets from both two sequences. Formally, considering the item $\mathbf{x}_i^I$ in $\mathbf{X}^I$, we can compute the correlation between the item $\mathbf{x}_i^I$ and the set $\mathbf{x}_j^S$ as:

$$\mu_{ij}^h = \frac{\exp\left(\left(\widetilde{\mathbf{W}}_h^Q \mathbf{x}_i^I\right)^\top \widetilde{\mathbf{W}}_h^K \mathbf{x}_j^S / \sqrt{d/H}\right)}{\sum_{k=1}^{m} \exp\left(\left(\widetilde{\mathbf{W}}_h^Q \mathbf{x}_i^I\right)^\top \widetilde{\mathbf{W}}_h^K \mathbf{x}_k^S / \sqrt{d/H}\right)}, \tag{15}$$

where $\widetilde{\mathbf{W}}_h^Q, \widetilde{\mathbf{W}}_h^K \in \mathbb{R}^{d/H \times d}$. We can get the hidden state of the item $\mathbf{x}_i^I$ by calculating the weighted sum of the set representations in $\mathbf{X}^S$, followed by a point-wise feed-forward network:

$$\mathbf{z}_i^{I,S} = \text{LayerNorm}\left(\mathbf{x}_i^I + \widetilde{\mathbf{W}}^O \Big\|_{h=1}^{H} \left\{\sum_{j=1}^{m} \mu_{ij}^h \widetilde{\mathbf{W}}_h^V \mathbf{x}_j^S\right\}\right), \tag{16}$$

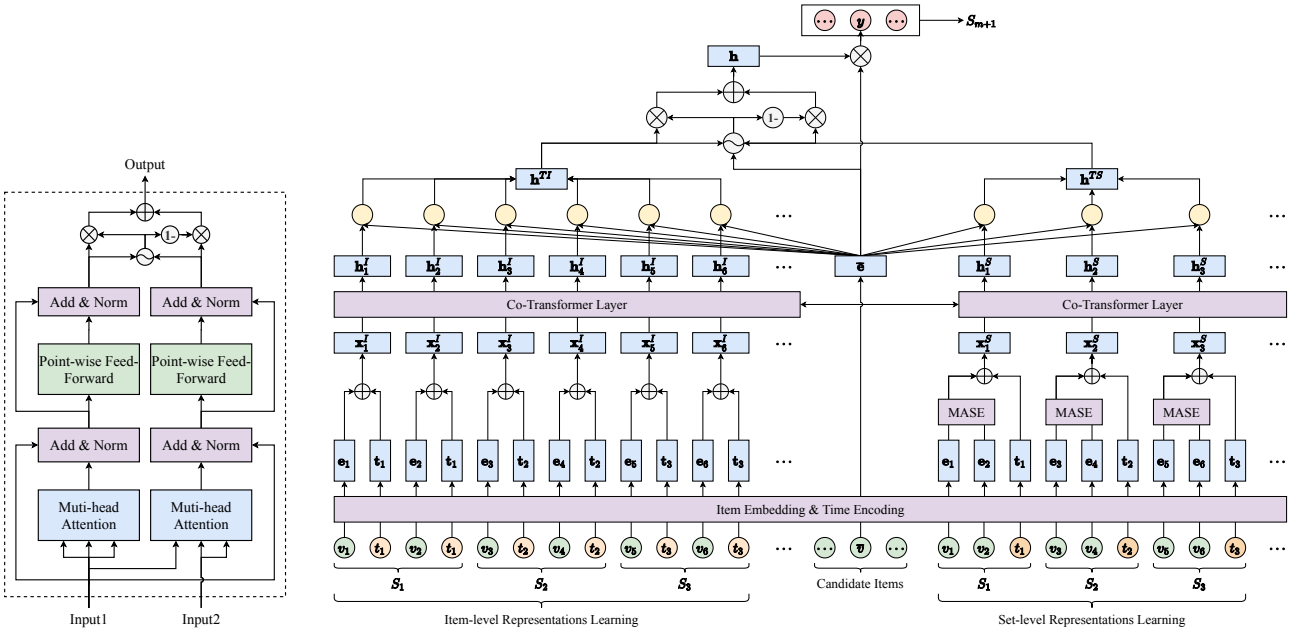
$$\mathbf{h}_i^{I,S} = \text{LayerNorm}\left(\mathbf{z}_i^{I,S} + \text{FNN}\left(\mathbf{z}_i^{I,S}\right)\right), \tag{17}$$

where $\widetilde{\mathbf{W}}_h^V \in \mathbb{R}^{d/H \times d}$, $\widetilde{\mathbf{W}}^O \in \mathbb{R}^{d \times d}$. Then we can compute the final hidden state of the item $\mathbf{x}_i^I$ by fusing $\mathbf{h}_i^I$ and $\mathbf{h}_i^{I,S}$ via a gated neural layer, which is formulated as follows:

$$\mathbf{g}_i^I = \text{Sigmoid}\left(\mathbf{W}_c^I \mathbf{h}_i^I + \mathbf{W}_c^{I,S} \mathbf{h}_i^{I,S} + b_c^I\right), \tag{18}$$

$$\tilde{\mathbf{h}}_i^I = \mathbf{g}_i^I \odot \mathbf{h}_i^I + \left(1 - \mathbf{g}_i^I\right) \odot \mathbf{h}_i^{I,S}, \tag{19}$$

where $\mathbf{W}_c^I, \mathbf{W}_c^{I,S} \in \mathbb{R}^{d \times d}$, and $\odot$ is element-wise multiplication. In contrast, we can also take the set sequence $\mathbf{X}^S$ as queries and the item sequence $\mathbf{X}^I$ as keys and values. Then we can calculate the hidden state for each set from $\mathbf{X}^S$ in the same way.



(a) Co-Transformer (b) DSNTSP

Figure 4: Dual sequential network for temporal sets prediction.

4.3.2 Item-level and Set-level Representations Fusion. For each candidate item $\overline{v}$, we can compute the item-level representation $\mathbf{h}^{TI}$ and the set-level representations $\mathbf{h}^{TS}$ respectively. In order to fuse these two representations, we design a gated neural module that takes $\overline{\mathbf{e}}$, $\mathbf{h}^{TI}$ and $\mathbf{h}^{TS}$ as inputs. We first compute a gate vector $\mathbf{g}^f$ to decide contribution percentages of item-level representations and set-level representations for the candidate item $\overline{v}$, and then use the gate vector to compute the final sequence representation $\mathbf{h}$, which is formulated as follows:

$$\mathbf{g}^f = \text{Sigmoid}\left(\mathbf{W}_f^I \mathbf{h}^{TI} + \mathbf{W}_f^S \mathbf{h}^{TS} + \mathbf{W}_f^c \overline{\mathbf{e}} + b_f\right), \tag{20}$$

$$\mathbf{h} = \mathbf{g}^f \odot \mathbf{h}^{TI} + \left(1 - \mathbf{g}^f\right) \odot \mathbf{h}^{TS}, \tag{21}$$

where $\mathbf{g}^f, \mathbf{h} \in \mathbb{R}^d$, $\mathbf{W}_f^I, \mathbf{W}_f^S, \mathbf{W}_f^c \in \mathbb{R}^{d \times d}$. Finally, the sequence representation $\mathbf{h}$ can be used to compute the probability of the user interacting with the candidate item $\overline{v}$ in the future.

4.4 Model Learning

The problem of temporal sets prediction can be seen as a multi-label classification, and one item is regarded as a label. One way to solve this problem is to transform the multi-label classification into multiple binary classification problem. That is to say, we train one binary classifier for each item independently. Therefore, we use the binary cross-entropy loss as our objective function. Formally, for each user, we define the objective function as:

$$L = -\sum_{\overline{v} \in S_{m+1}} \log f(TS, \overline{v}) - \sum_{\overline{v} \notin S_{m+1}} \log(1 - f(TS, \overline{v})), \tag{22}$$

where $f(TS, \overline{v})$ is the output of our network representing the predicted probability for the user interacting with the item $\overline{v}$ according to the set sequence TS. Finally, we can minimize the objective function by performing Adam optimizer [12].

5 EXPERIMENTS

This section evaluates the effectiveness of the proposed method. We first introduce the used data sets, evaluation metrics, and compared baselines; then present the performance comparisons of our method with both classical and the state-of-the-art methods. Ablation study is conducted to verify the effectiveness of the components in our method, the result of set embedding is also visualized to discuss the interpretability of our method.

5.1 Experiment Setup

5.1.1 Data Sets. We conduct experiments on four real-world datasets. The statistics of the four datasets are summarized in the Table 1.

TaFeng[1]: The TaFeng dataset is a public dataset that contains 4 months (from November 2000 to February 2001) of shopping transactions from a Chinese grocery store. We remove users that purchased less than 10 items and items purchased less than 10 times. We treat all the items bought in the same order as a set.

TaoBao[2]: The TaoBao dataset is a public online e-commerce dataset provided by Ant Financial Services. It contains the transactions about which items are purchased or clicked by each user in each order with the timestamp. We select the transactions from August 1, 2015 to September 15, 2015. The users that purchased less than 10 items are removed, so are the items purchased less than 30 times. We treat all the items purchased in the same day as a set.

[1] https://www.kaggle.com/chiranjivdas09/ta-feng-grocerydataset
[2] https://tianchi.aliyun.com/dataset/dataDetail?dataId=53

Table 1: Statistics of the datasets

Dataset	#items	#sets	#users	Ave. #items per set	Ave. #sets per user
TaFeng	11208	80093	9684	6.1582	8.2706
TaoBao	14872	99926	14714	2.2472	6.7912
JingDong	22735	84273	16302	1.3723	5.1695
Dunnhumby	15730	79389	2306	10.3802	34.4271

Jingdong[3]: The JingDong dataset contains user action records from February 1, 2018 to April 15, 2018, including browsing, purchasing, following, commenting and adding to shopping carts. We only select purchasing actions and filter out users that purchased less than 3 items and items purchased less than 3 times. We treat all the items purchased in the same day as a set.

Dunnhumby[4]: The Dunnhumby dataset is a public dataset provided by dunnhumby, which is a global customer data science company. It contains the transactions about which items are bought by each user in each order with the timestamp. We select 300 days transaction records and filter out users that purchased less than 3 items and items purchased less than 3 times. We treat all the items bought in the same order as a set.

For each dataset, we can get a series of set sequences. Given the user-set interactions $TS = \{(S_1, t_1), \cdots, (S_m, t_m)\}$ for a user, we use the first k interactions $\{(S_1, t_1), \cdots, (S_k, t_k)\}$ to predict the $(k+1)^{th}$ interaction (S_{k+1}, t_{k+1}) in the training set, where $k = 1, \cdots, m-3$. We use the first $m-2$ interactions to predict the $(m-1)^{th}$ interaction in the validation set and use the first $m-1$ interactions to predict the last one in the test set.

5.1.2 Evaluation Metrics. To evaluate the effectiveness of different methods, we use *Recall* and *Normalized Discounted Cumulative Gain(NDCG)* metrics.

Recall represents the ability of coverage in the user's ground truth. For each user u, recall is calculated as:

$$\text{Recall@K}(u) = \frac{|P_u \cap T_u|}{|T_u|}, \tag{23}$$

where P_u is the predicted top K items for user u and T_u is the ground truth of user u. We use the average recall of all users as the measurement.

Normalized Discounted Cumulative Gain is a measurement of ranking quality. For each user u, NDCG is calculated as:

$$\text{NDCG@K}(u) = \frac{\sum_{k=1}^{K} p_i / \log_2(k+1)}{\sum_{k=1}^{\min(K,|T_u|)} 1/\left(\log_2(k+1)\right)}. \tag{24}$$

The average NDCG of all users is used as the measurement.

5.1.3 Baseline Methods. To verify the effectiveness of our method, we compare it with the following representative baselines:

POP: it ranks items according to their popularity. For any user, the subsequent set is a combination of the most frequently purchased items.

PERSON POP: it ranks items based on their personal popularity. For each user, the subsequent set is a combination of the user's historical frequently purchased items.

[3]https://jdata.jd.com/html/detail.html?id=8

[4]https://www.dunnhumby.com/careers/engineering/sourcefiles

MF [13]: it is a collaborative filter method, which applies matrix factorization over the user-item matrix. It does not consider the temporal dependencies and co-occurrence of items.

FPMC [25]: it is a hybrid model combining markov chain and matrix factorization for next basket recommendation. Both temporal dependencies between items and user's general interests are taken into account for prediction.

DREAM [36]: it is a RNN-based method for next basket recommendation, where the representation of a basket is aggregated by item's embedding via a pooling layer.

DIN [39]: it is a DNN-based method used for click-though rate prediction. It uses attention mechanism to learn the representation of user's historical behaviors for the target items. To speedup, we replace its local activation unit with dot-product attention, and compute score for each item by dot product.

SETS2SETS [11]: it is the state-of-art method in temporal sets prediction. It uses average pooling operation to get the representations for sets. A repeated-element-specified element-element interaction component is designed to further improve the performance.

5.1.4 Implementation Details. We use PyTorch [23] to implement our model and deploy it on TITAN X GPU with 12G memory. Adam optimizer [12] with learning rate 0.0001 is used to update parameters. The batch size is set to 64. The dimension of embedding is set to 256. The number of multi-head self-attention layers is set to 4, and the number of heads in the multi-head self-attention layer is set to 8. The number of heads in our set embedding module is set to 8. The temperature parameters in the item-oriented attention layers are set to 0.8, 1.0, 4.0, 16.0 for TaFeng, TaoBao, JingDong and Dunnhumby datasets respectively.

5.2 Performance Comparison

We report experiment results of different methods in Table 2 and Figure 5. As we can see, our DSNTSP model achieves better performance than other comparative methods in most cases. Besides, there are some findings in these comparative experiments.

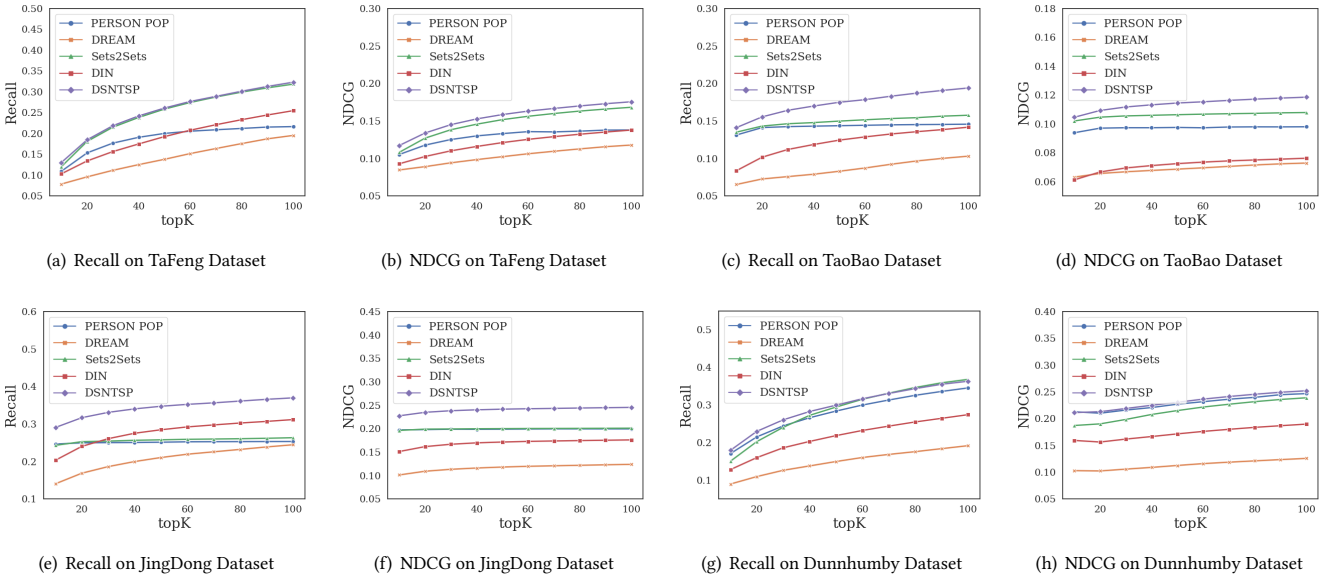
Firstly, FPMC and DREAM achieve worse performance for both Recall and NDCG on these four datasets compared to most other baseline methods. They even perform worse than POP on TaFeng dataset. FPMC only considers the last set interacted by each user, while DREAM considers all sets in the user historical behaviors, but it only captures temporal dependencies of the set sequences at the set-level. The results show that it is difficult to achieve good results if we only learn the set-level representations of the set sequences.

Secondly, DIN and PERSON POP perform better than FPMC and DREAM, suggesting that the frequencies of items appearing in the past sets of a given user are informative. Sets2Sets and DSNTSP perform even better than DIN and PERSON POP, which cannot take into account temporal dependencies in the user behaviors.

Thirdly, Sets2Sets is the state-of-art method for temporal sets prediction and it achieves better performance compared to other baseline methods in most cases. The reason why Sets2Sets can achieve such good results is that it not only considers the temporal dependencies of the set sequences at the set-level, but also models repeated-element-specified element-element relation in the set sequences. However, Sets2Sets performs worse than our method

Table 2: Experiment results of different methods

Models	TaFeng				TaoBao				JingDong				Dunnhumby			
	K=50		K=100		K=50		K=100		K=50		K=100		K=50		K=100	
	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
POP	0.1496	0.1084	0.2141	0.1258	0.0887	0.0696	0.1083	0.0736	0.0840	0.0407	0.1241	0.0479	0.1522	0.1056	0.1935	0.1195
PERSON POP	0.2001	0.1331	0.2165	0.1379	0.1437	0.0975	0.1457	0.0979	0.2514	0.1988	0.2532	0.1995	0.2838	0.2270	0.3454	0.2467
MF	0.2000	0.0911	0.2547	0.1063	0.1474	0.0741	0.1591	0.0765	0.2519	0.1686	0.2714	0.1719	0.1336	0.0705	0.2061	0.0938
FPMC	0.1421	0.0887	0.1852	0.1003	0.1008	0.0677	0.1199	0.0715	0.2195	0.1559	0.2319	0.1581	0.1828	0.1449	0.2216	0.1580
DREAM	0.1379	0.1024	0.1953	0.1179	0.0829	0.0686	0.1031	0.0728	0.2105	0.1181	0.2445	0.1238	0.1496	0.1125	0.1917	0.1257
DIN	0.1925	0.1212	0.2549	0.1379	0.1245	0.0724	0.1417	0.0762	0.2846	0.1715	0.3118	0.1760	0.2185	0.1715	0.2743	0.1898
SETS2SETS	0.2586	0.1518	0.3185	0.1683	0.1498	0.1064	0.1576	0.1078	0.2577	0.2001	0.2634	0.2011	0.2945	0.2151	**0.3682**	0.2389
DSNTSP	**0.2611**	**0.1585**	**0.3232**	**0.1756**	**0.1741**	**0.1141**	**0.1931**	**0.1181**	**0.3473**	**0.2416**	**0.3698**	**0.2455**	**0.2995**	**0.2305**	0.3633	**0.2521**
Improvement	0.97%	4.41%	1.48%	4.34%	16.22%	7.24%	21.37%	9.55%	22.03%	20.74%	40.39%	22.08%	1.70%	7.16%	-1.33%	2.19%



(a) Recall on TaFeng Dataset (b) NDCG on TaFeng Dataset (c) Recall on TaoBao Dataset (d) NDCG on TaoBao Dataset

(e) Recall on JingDong Dataset (f) NDCG on JingDong Dataset (g) Recall on Dunnhumby Dataset (h) NDCG on Dunnhumby Dataset

Figure 5: Results comparison with K changing from 10 to 100 with step size 10.

because it doesn't consider the item-level temporal dependencies of the set sequences.

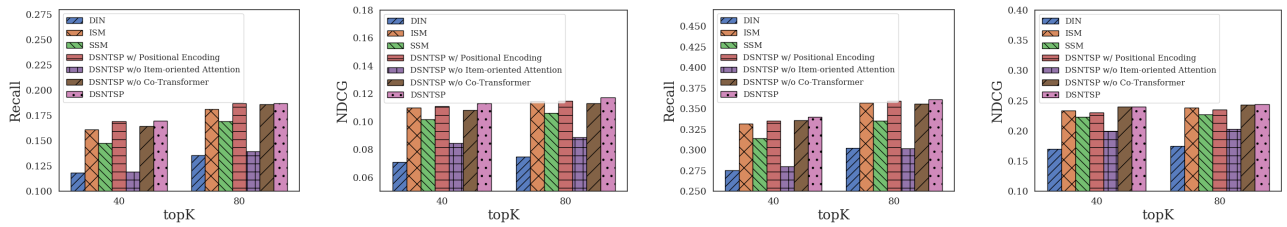
Fourthly, our proposed DSNTSP achieves significant improvements for both Recall and NDCG metrics on TaoBao and JingDong datasets compared to other baseline methods. We believe the reason is that in the e-commerce scenario, people tend to purchase different items each time, which is not very common in the retail stores. Sets2Sets only consider the frequency for each item appearing in the user historical behaviors, making it tend to recommend repeated items for customers. Compared to Sets2Sets, our method can learn both the item-level and set-level temporal dependencies of the set sequences, which makes our method able to achieve better performance than Sets2Sets.

5.3 Ablation Study

This section conducts ablation studies on TaoBao and JingDong datasets, Figure 6 presents the results.

5.3.1 Effect of the Multi-head Self-Attention Layers. The performance comparison of ISM and DIN could reveal the impact of multi-head self-attention layers because DIN can be seen as a variant of the ISM that removes the multi-head self-attention layers. From the Figure 6, we can see that ISM achieves significant improvements compared to the DIN on both TaoBao and JingDong datasets. It indicates that the multi-head self-attention layers make our model available to capture the complex temporal dependencies of set sequences, which will help to improve the prediction accuracy.

5.3.2 Effect of the Time Encoding. To investigate the effect of the time encoding, we compare the performance between DSNTSP with time encoding (DSNTSP) and DSNTSP with positional encoding (DSNTSP w/ Positional Encoding) on both TaoBao and JingDong Datasets. From the Figure 6, we can see that DSNTSP with time encoding achieves better performance than DSNTSP with positional encoding on both TaoBao and JingDong dataset. It indicates that



(a) Recall on TaoBao Dataset (b) NDCG on TaoBao Dataset (c) Recall on JingDong Dataset (d) NDCG on JingDong Dataset

Figure 6: Ablation study of DSNTSP on TaoBao and JingDong data sets.

time encoding employed in DSNTSP could help our model make full use of the temporal information in the set sequences.

5.3.3 Effect of the Item-oriented Attention Layer. We also evaluate the necessity of the item-oriented attention layer in our model. We develop a variant of the DSNTSP that computes sequence representations by aggregating the hidden states of the sequences via average pooling operations. From the Figure 6, we can see that our DSNTSP far outperforms the DSNTSP without the item-oriented attention layer on both TaoBao and JingDong datasets. It indicates that using one fixed-length vector will be a bottleneck to express dynamic evolving patterns in the set sequences. Our item-oriented attention layer can compute the sequence representation for each target item by making each target item choose the most relevant parts in the set sequence to use, which will help to improve the model performance.

5.3.4 Effect of the Fusion of Item-level and Set-level Representations. In order to investigate the effect of the fusion of item-level and set-level representations, we compare our DSNTSP with ISM and SSM on both TaoBao and JingDong datasets. From the Figure 6, we can see that our DSNTSP outperforms both ISM and SSM for all evaluation metrics on both TaoBao and JingDong datasets. It shows that learning both the item-level representations and set-level representations can make our model better solve the temporal sets prediction problem.

5.3.5 Effect of the Co-Transformer. In order to investigate the influence of our co-transformer module, we compare the performance of DSNTSP with a variant without co-transformer module. From the Figure 6, we can see that our DSNTSP achieves better performance than DSNTSP without co-transformer. It indicates that our co-transformer module can improve the performance of our model by learning the temporal dependencies of set sequences from both item-level and set-level.

5.4 Visualization of Set Embedding

In this subsection, we investigate how effective our multi-head attention based set embedding method is. We conduct a visualization experiment to verify the effectiveness of our set embedding method. We apply our set embedding method over a set containing four types of items sampled from TaoBao dataset. The types of items in the set include A, B, C and D. Then we visualize the attention weights of the different heads over these items, which is shown in Figure 7. We can see that different heads focus on different types of items in the set. Head 1 mainly concentrates on the items of type A, while head 2 pays more attention on the items of type D. Head 3 assigns higher weights to the items of type C and head 4 gives more attention to items of type B. The results show that different heads in our set embedding method usually focus on different aspects of the set, making our model available to get more reasonable vectorized representations for sets.

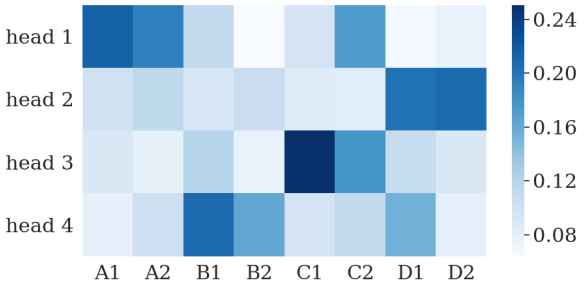


Figure 7: Visualization of attention weights from head 1 to head 4 in the multi-head attention based set embedding module with four heads on TaoBao data set.

In summary, this section conducts experiments on four real-world data sets. Four classical and three state-of-the-art methods have been implemented to provide baseline performance. We use both *Recall* and *NDCG* to get a comprehensive comparison of our method with the baselines. Finally, the proposed method DSNTSP achieves the best prediction performance in most cases. Ablation study demonstrates that both item-level and set-level representations are useful for temporal sets prediction, the co-transformer and gated neural fusing module also contributes to the improvement of prediction accuracy. Additionally, visualization experiment suggests that our method could achieve a semantic representation of sets. All the above results and observations support the previous theoretical analysis.

6 CONCLUSION

This paper provides a novel prediction method for temporal sets, which consists of four key modules. Item-level and set-level representation learning modules could capture the correlations of

items and sets separately, and then achieve both item-level and set-level representations of set sequence. The co-transformer module is able to learn the multiple temporal dependencies of items and sets. The gated neural module fuses all the multi-level correlations and multiple temporal dependencies comprehensively to predict the subsequent set. Experiments have been conducted on four real-world data sets, results demonstrate the superiority of our method over both classical and the state-of-the-art methods. Ablation study validates the effectiveness of the components in our method. In summary, this work studies a multi-level representation learning method for set sequence, which enables us to discover multi-level correlations and multiple temporal dependencies of items and sets. Therefore, our method could obtain higher prediction performance than most of the existing methods. In future, we may study time-aware temporal sets prediction problem by extending the temporal sets prediction method proposed in this paper.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (51778033, 51822802, 51991395, 71901011, U1811463), the Science and Technology Major Project of Beijing (Z191100002519012) and the National Key R & D Program of China (2018YFB2101003).

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.
[2] Austin R. Benson, Ravi Kumar, and Andrew Tomkins. 2018. Sequences of Sets. In *KDD*. ACM, 1148–1157.
[3] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David A. Sontag, and Yan Liu. 2016. Recurrent Neural Networks for Multivariate Time Series with Missing Values. *CoRR* abs/1606.01865 (2016).
[4] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior Sequence Transformer for E-commerce Recommendation in Alibaba. *CoRR* abs/1905.06874 (2019).
[5] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*. ACL, 1724–1734.
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*. Association for Computational Linguistics, 4171–4186.
[7] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep Session Interest Network for Click-Through Rate Prediction. In *IJCAI*. ijcai.org, 2301–2307.
[8] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *ICASSP*. IEEE, 6645–6649.
[9] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR (Poster)*.
[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[11] Haoji Hu and Xiangnan He. 2019. Sets2Sets: Learning from Sequential Sets with Neural Networks. In *KDD*. ACM, 1491–1499.
[12] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*.
[13] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42, 8 (2009), 30–37.
[14] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *ICML (Proceedings of Machine Learning Research)*, Vol. 97. PMLR, 3744–3753.
[15] Liangyue Li, How Jing, Hanghang Tong, Jaewon Yang, Qi He, and Bee-Chung Chen. 2017. NEMO: Next Career Move Prediction with Contextual Embedding. In *WWW (Companion Volume)*. ACM, 505–513.
[16] Zachary Chase Lipton. 2015. A Critical Review of Recurrent Neural Networks for Sequence Learning. *CoRR* abs/1506.00019 (2015).
[17] Chuanren Liu, Kai Zhang, Hui Xiong, Geoff Jiang, and Qiang Yang. 2014. Temporal skeletonization on sequential data: patterns, categorization, and visualization. In *KDD*. ACM, 1336–1345.
[18] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*. The Association for Computational Linguistics, 1412–1421.
[19] Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. 2019. SDM: Sequential Deep Matching Model for Online Large-scale Recommender System. In *CIKM*. ACM, 2635–2643.
[20] Changping Meng, Jiasen Yang, Bruno Ribeiro, and Jennifer Neville. 2019. HATS: A Hierarchical Sequence-Attention Framework for Inductive Set-of-Sets Embeddings. In *KDD*. ACM, 783–792.
[21] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent Models of Visual Attention. In *NIPS*. 2204–2212.
[22] Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak A. Rao, and Bruno Ribeiro. 2019. Janossy Pooling: Learning Deep Permutation-Invariant Functions for Variable-Size Inputs. In *ICLR (Poster)*. OpenReview.net.
[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*. 8024–8035.
[24] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *RecSys*. ACM, 130–137.
[25] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *WWW*. ACM, 811–820.
[26] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. DiSAN: Directional Self-Attention Network for RNN/CNN-Free Language Understanding. In *AAAI*. AAAI Press, 5446–5455.
[27] Konstantinos Skianis, Giannis Nikolentzos, Stratis Limnios, and Michalis Vazirgiannis. 2019. Rep the Set: Neural Networks for Learning Set Representations. *CoRR* abs/1904.01962 (2019).
[28] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *CIKM*. ACM, 1441–1450.
[29] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2019. ERNIE 2.0: A Continual Pre-training Framework for Language Understanding. *CoRR* abs/1907.12412 (2019).
[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 5998–6008.
[31] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order Matters: Sequence to sequence for sets. In *ICLR (Poster)*.
[32] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *CVPR*. IEEE Computer Society, 3156–3164.
[33] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *CoRR* abs/1906.08237 (2019).
[34] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical Attention Networks for Document Classification. In *HLT-NAACL*. The Association for Computational Linguistics, 1480–1489.
[35] Zijun Yao, Yanjie Fu, Bin Liu, Yanchi Liu, and Hui Xiong. 2016. POI Recommendation: A Temporal Matching between POI Popularity and User Regularity. In *ICDM*. IEEE Computer Society, 549–558.
[36] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A Dynamic Recurrent Model for Next Basket Recommendation. In *SIGIR*. ACM, 729–732.
[37] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. 2017. Deep Sets. In *NIPS*. 3391–3401.
[38] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2018. ATRank: An Attention-Based User Behavior Modeling Framework for Recommendation. In *AAAI*. AAAI Press, 4564–4571.
[39] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *KDD*. ACM, 1059–1068.