Deep Structural Point Process for Learning Temporal Interaction Networks

Jiangxia Cao[1,2], Xixun Lin[1,2] (✉), Xin Cong[1,2], Shu Guo[3], Hengzhu Tang[1,2], Tingwen Liu[1,2] (✉), and Bin Wang[4]

[1] Institute of Information Engineering, Chinese Academy of Sciences
[2] School of Cyber Security, University of Chinese Academy of Sciences
[3] National Computer Network Emergency Response Technical Team/Coordination Center of China
[4] Xiaomi AI Lab, Xiaomi Inc.
{caojiangxia,linxixun,congxin,tanghengzhu,liutingwen}@iie.ac.cn, guoshu@cert.org.cn, wangbin11@xiaomi.com

Abstract. This work investigates the problem of learning temporal interaction networks. A temporal interaction network consists of a series of chronological interactions between users and items. Previous methods tackle this problem by using different variants of recurrent neural networks to model interaction sequences, which fail to consider the structural information of temporal interaction networks and inevitably lead to sub-optimal results. To this end, we propose a novel **D**eep **S**tructural **P**oint **P**rocess termed as **DSPP** for learning temporal interaction networks. DSPP simultaneously incorporates the *topological structure* and *long-range dependency structure* into the intensity function to enhance model expressiveness. To be specific, by using the topological structure as a strong prior, we first design a topological fusion encoder to obtain node embeddings. An attentive shift encoder is then developed to learn the long-range dependency structure between users and items in continuous time. The proposed two modules enable our model to capture the user-item correlation and dynamic influence in temporal interaction networks. DSPP is evaluated on three real-world datasets for both tasks of item prediction and time prediction. Extensive experiments demonstrate that our model achieves consistent and significant improvements over state-of-the-art baselines.

Keywords: Temporal Interaction Networks · Temporal Point Process · Graph Neural Networks

1 Introduction

Temporal interaction networks are useful resources to reflect the relationships between users and items over time, which have been successfully applied in many real-world domains such as electronic commerce [3], online education [18] and social media [12]. A temporal interaction network naturally keeps a graph data structure with temporal characteristics, where each edge represents a user-item interaction marked with a concrete timestamp.

Representation learning on temporal interaction networks has gradually become a hot topic in the research of machine learning [24]. A key challenge of modeling temporal interaction networks is how to capture the evolution of user interests and item features effectively. Because users may interact with various items sequentially and their interests may shift in a period of time. Similarly, item features are also ever-changing and largely influenced by user behaviours. Recent works have been proposed to tackle this challenge by generating the dynamic embeddings of users and items [21, 7, 16, 17]. Although these methods achieve promising results to some extent, they still suffer from the following two significant problems.

1) **Topological structure missing**. Most previous methods regard learning temporal interaction networks as a coarse-grained sequential prediction problem and ignore the topological structure information. In fact, instead of only treating a temporal interaction network as multiple interaction sequences, we can discover user similarity and item similarity from the view of the topological structure. Nevertheless, due to the bipartite nature of temporal interaction networks, each node is not the same type as its adjacent nodes, so that we have to develop a flexible method to capture such a meaningful topology. 2) **Long-range dependency structure missing**. Most current methods are built upon the variants of recurrent neural networks (RNNs) to learn interaction sequences. Hence, they typically pay more attention to short-term effects and miss the dependency structure in long-range historical information [25, 8]. But learning the long-range dependency structure in temporal interaction networks is also critical, since it can better model the long-standing user preference and intrinsic item properties.

In this paper, we propose the **D**eep **S**tructural **P**oint **P**rocess termed as **DSPP** to solve above problems. Following the framework of Temporal Point Process (TPP) [29], we devise a novel intensity function which combines the topological structure and the long-range dependency structure to capture the dynamic influence between users and items. Specifically, we first design a topological fusion encoder (TFE) to learn the topological structure. TFE includes a two-steps layer to encourage each node to aggregate homogeneous node features. To overcome the long-range dependency issue, we then develop an attentive shift encoder (ASE) to recognize the complex dependency between each historical interaction and the new-coming interaction. Finally, we incorporate the learned embeddings from TFE and ASE into our intensity function to make time prediction and item prediction. The main contributions of our work are summarized as follows:

- We propose the novel DSPP to learn temporal interaction networks within the TPP paradigm, with the goal of solving two above structural missing problems simultaneously.
- DSPP includes the well-designed TFE and ASE modules. TFE utilizes the topological structure to generate steady embeddings, and ASE exploits the

long-range dependency structure to learn dynamic embeddings. Furthermore, these two types of embeddings can be seamlessly incorporated into our intensity function to achieve future prediction.

– Extensive experiments are conducted on three public standard datasets. Empirical results show that the proposed method achieves consistent and significant improvements over state-of-the-art baselines[5].

2 Related Work

Previous studies for learning temporal interaction networks can be roughly divided into the following three branches: random walk based method, RNN based methods and TPP based method.

– Random walk based method. Nguyen *et al.* propose CTDNE [21] which models user and item dynamic embeddings by the random walk heuristics with temporal constraints. CTDNE first samples some time increasing interaction sequences, and then learns context node embeddings via the skip-gram algorithm [20]. However, it ignores the useful time information of the sampled sequences, e.g., a user clicks an item frequently may indicate that the user pays more attention to this item at the moment.
– RNN based methods. LSTM [11], RRN [27], LatentCross [2] and Time-LSTM [30] are pioneering works in this branch. For example, RRN provides a unified framework for combining the static matrix factorization features with the dynamic embeddings based on LSTM. Moreover, it provides a behavioral trajectory layer to project user and item embeddings over time. LatentCross is an extension of the architecture of GRU [5], which incorporates multiple types of context information. Time-LSTM develops the time gates for LSTM for modeling the interaction time information. Furthermore, JODIE [16] and DGCF [17] are the state-of-the-art methods for learning temporal interaction networks via the coupled variants of RNNs. JODIE defines two-steps embedding update operation and an embedding projection function to predict the target item embedding for each user directly. DGCF extends JODIE by considering the 1-hop neighboring information of temporal interaction networks.
– TPP based method. DeepCoevolve [7] is a promising work that applies TPPs to learn temporal interaction networks. It uses a multi-dimensional intensity function to capture the dynamic influence between users and items. However, DeepCoevolve maintains the same embeddings of user and item until it involves a new interaction, which is not consistent with real-world facts [16]. Furthermore, it uses a linear intensity function to describe the dynamic influence, leading to the limited model expressiveness.

[5] The source code is available from `https://github.com/cjx96/DSPP`.

3 Background

3.1 Temporal Interaction Network

A series chronological interactions can be represented as a temporal interaction network. Formally, a temporal interaction network on the time window $[0, T)$ can be described as $\mathcal{G}(T) = (\mathcal{U}, \mathcal{V}, \mathcal{E})$, where $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{E}$ denote the user set, item set and interaction set, respectively. Each element $(u_i, v_j, t) \in \mathcal{E}$ is an interaction, describing that the user $u_i \in \mathcal{U}$ conducts an action with the item $v_j \in \mathcal{V}$ at the concrete timestamp t.

3.2 Temporal Point Process

TPPs are one of branches of stochastic processes for modeling the observed random discrete events, e.g. user-item interactions over time. Using conditional intensity function $\lambda(t)$ is a convenient way to describe TPPs. Given a interaction sequence that only has time information $\mathcal{T} := \{t_i\}_{i=1}^{n}$ and an infinitesimal time interval $\mathrm{d}t$, where $t_i \in \mathbb{R}^+$ and $0 < t_1 < t_2 ... < t_n$, $\lambda(t)\mathrm{d}t$ is the conditional probability of happening an interaction in the infinitesimal interval $[t, t + \mathrm{d}t)$ based on $\mathcal{T}$. It can be also interpreted heuristically in the following way:

$$\lambda(t)\mathrm{d}t := \mathbb{P}\{\text{an interaction occurs in } [t, t + \mathrm{d}t)|\mathcal{T}\} = \mathbb{E}[N([t, t + \mathrm{d}t))|\mathcal{T}],$$

where $N([t, t+\mathrm{d}t))$ is used to count the number of interactions happened in the time interval $[t, t + \mathrm{d}t)$. A general assumption made here is that there is either zero or one interaction happened in this infinitesimal interval, i.e., $N([t, t + \mathrm{d}t)) \in \{0, 1\}$. Furthermore, given a future timestamp $t^+ > t_n$ and $\mathcal{T}$, we can formulate the conditional probability of no interactions happened during $[t_n, t^+)$ as $S(t^+) = \exp\left(-\int_{t_n}^{t^+} \lambda(t)\mathrm{d}t\right)$. Therefore, the conditional probability density of the next interaction happened at the future timestamp t^+ can be defined as: $f(t^+) = S(t^+)\lambda(t^+)$, which means that no interaction happens in time interval $[t_n, t^+)$ and an interaction happens in infinitesimal interval $[t^+, t^+ + \mathrm{d}t)$.

Multivariate Hawkes Process (MHP) is one of the most important TPPs for modeling interaction sequences [9]. We denote $\mathcal{S}_{u_i}(T) = (\mathcal{V}, \mathcal{H}_{u_i})$ as an interaction sequence of user u_i on the time window $[0, T]$, where $\mathcal{H}_{u_i}$ is the interaction sequence of user u_i. The h-th interaction of $\mathcal{H}_{u_i}$ is denoted as (v^h, t^h), which describes that the user $u_i \in \mathcal{U}$ has interacted with the item $v^h \in \mathcal{V}$ at $t^h \leq T$. The intensity function of an arbitrary item v_j in $\mathcal{S}_{u_i}(T)$ is defined as:

$$\lambda_{v_j}(t) = \mu_{v_j} + \sum\nolimits_{t^h < t} \alpha_{(v_j, v^h)} \kappa(t - t^h),$$

where μ_{v_j} (a.k.a base intensity) is a positive parameter which is independent of the interaction sequence $\mathcal{S}_{u_i}(T)$, $\alpha_{(v_j, v^h)}$ is also a positive parameter that estimates the influence between item pair (v_j, v^h) and the $\kappa(t-t^h)$ is a triggering kernel function. The intensity function explicit models dynamic influence among interaction sequence. However, most existing approaches ignore to model the

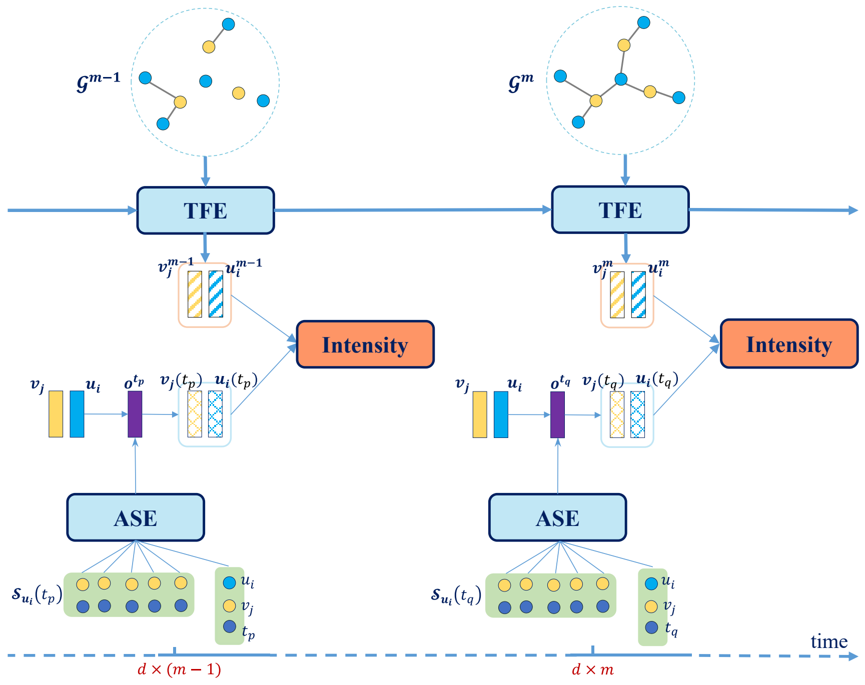

Fig. 1. A simple overview of DSPP. "TFE", "ASE", "Intensity" mean topological fusion encoder, attentive shift encoder and intensity function, respectively. $\mathcal{G}^{m-1}$ and $\mathcal{G}^{m}$ are two network snapshots. Noticeably, $\mathcal{G}^{m-1} \in \mathcal{G}^{m}$. (u_i, v_j, t_p), (u_i, v_j, t_q) denote two interactions, where $t_p \in [d \times (m-1), d \times m)$ and $t_q \in [d \times m, d \times (m+1))$. $\mathcal{S}_{u_i}(t_p)$ and $\mathcal{S}_{u_i}(t_q)$ are two interaction sequences.

topological structure between a series of interaction sequences. To fill this gap, DSPP includes a novel TFE module which provides a strong structure prior to enhance model expressiveness.

4 Proposed Model

4.1 Overview

For a temporal interaction network $\mathcal{G}(T) = (\mathcal{U}, \mathcal{V}, \mathcal{E})$, the adjacency matrix would change over time, because the emerge of a new interaction would introduce a new edge in the temporal interaction network, causing the huge occupation of memory. To sidestep this problem, we exploit an ordered snapshot sequence $\{\mathcal{G}^m\}_{m=0}^{M-1}$ with the same time interval $d = \frac{T}{M}$ to simulate the temporal interaction network $\mathcal{G}(T)$. Each snapshot $\mathcal{G}^m$ equals to $\mathcal{G}(d \times m)$, and M is a hype-parameter to control the number of snapshot.

Fig.1 shows the overview of our model. In DSPP, the TFE module aims to learn **steady embeddings** which represent the stable intentions of users and items in the corresponding time period $[d \times m, d \times (m+1))$ from $\mathcal{G}^m$. Here we denote the steady embeddings of an arbitrary user u_i and an item v_j as $\boldsymbol{u}_i^m$ and $\boldsymbol{v}_j^m$, respectively. The second module ASE aims to learn **dynamic embeddings** of users and items for describing their dynamic intentions at timestamp t.

4.2 Embedding Layer

The embedding layer is used to initialize node embeddings (all users and items) and time embedding. It consists of two parts: the node embedding layer and the time embedding layer.

Node Embedding Layer The node embedding layer aims to embed users and items into a low-dimensional vector space. Formally, given a user u_i or an item v_j, we can obtain its D-dimensional representation ($\boldsymbol{u}_i \in \mathbb{R}^D$ or $\boldsymbol{v}_j \in \mathbb{R}^D$) from an initialization embedding matrix with a simple lookup operation, where D denotes the dimension number of embeddings.

Time Embedding Layer Position embedding [13, 22] is widely used to recognize the ordered information of sequences. However, the continuous time information of interaction sequence cannot be well reflected by the discrete position embedding. Therefore, we design a time embedding layer that encodes the discrete ordered information and continuous time information simultaneously. Concretely, given a future timestamp $t^+ \geq t$, user u_i and the h-th interaction (v^h, t^h) of its interaction sequence $\mathcal{S}_{u_i}(t)$ (detailed in Sec. 3.2), the h-th interaction time embedding $\boldsymbol{p}_{t^h}(t^+)$ of future timestamp t^+ can be formulated as follows:

$$[\boldsymbol{p}_{t^h}(t^+)]_j = \begin{cases} \cos(\omega_j(t^+ - t^h) + h/10000^{\frac{j-1}{D}}), \text{if } j \text{ is odd}, \\ \sin(\omega_j(t^+ - t^h) + h/10000^{\frac{j}{D}}), \text{if } j \text{ is even}, \end{cases}$$

where $[\boldsymbol{p}_{t^h}(t^+)]_j$ is the j-th element of the given vector $\boldsymbol{p}_{t^h}(t^+)$, and ω_j is a parameter to scale the time interval $t^+ - t^h$ in the j-th dimension.

4.3 Topological Fusion Encoder

In this section, we propose a novel topological fusion encoder (TFE) to learn the topological structure. The existing graph encoders [23, 6] learn node embeddings by aggregating the features of neighboring nodes directly. However, in temporal interaction networks, it would easily lead to improper feature aggregations due to the fact that the user neighbors are items. In this paper, we introduce a topological aggregation layer (TAL) into TFE to alleviate this issue.

Topological Aggregation Layer Different with homogeneous graphs, the distance between a user (item) and other users (items) is always an even number in temporal interaction network, e.g. 2, 4 and 6. This fact indicates that our encoder should aggregate homogeneous information through the even-number-hop. Based on this topological structure, we design a novel topological aggregation layer (TAL) as shown in Fig.2. Concretely, given an interaction (u_i, v_j, t), we firstly compute its corresponding snapshot identifier $m = \lfloor \frac{t}{d} \rfloor$, where $\lfloor \cdot \rfloor$ denotes the floor function and d is the time interval of ordered snapshot sequence. Then, to generate the user representation $\boldsymbol{u}_i^k$ in the k-th TAL, we calculate its intermediate representation $\widehat{\boldsymbol{v}}_c^k$ as follows:

$$\widehat{\boldsymbol{v}}_c^k = \delta\Big(\widehat{W}_u^k \ \mathrm{MEAN}\big(\{\boldsymbol{u}_q^{k-1} : u_q \in \mathcal{N}_m(v_c)\}\big)\Big), \text{where } v_c \in \mathcal{N}_m(u_i), \tag{1}$$

where δ is the ReLU activity function, $\widehat{W}_u^k$ is a parameter matrix, and $\mathcal{N}_m(v_c)$ denotes the set of 1-hop neighbors (user-type) of item v_c in $\mathcal{G}^m$. Therefore, $\widehat{\boldsymbol{v}}_c^k$ can be consider as a user-type representation since it only aggregates the user-type features. After obtaining the intermediate representation $\widehat{\boldsymbol{v}}_c^k$, we leverage the attention mechanism [25] to learn different weights among the neighboring intermediate representations for user u_i. The final embedding $\boldsymbol{u}_i^k$ can be formulated as:

$$\begin{aligned} e_c &= \delta\big((\overline{W}_u^k \boldsymbol{u}_i^{k-1})^\top \widehat{\boldsymbol{v}}_c^k\big), \text{where } v_c \in \mathcal{N}_m(u_i), \\ \alpha_c &= \frac{\exp{(e_c)}}{\sum_{v_q \in \mathcal{N}_m(u_i)} \exp{(e_q)}}, \\ \overline{\boldsymbol{u}}_i^k &= \delta\big(\sum_{v_c \in \mathcal{N}_m(u_i)} \alpha_c \widehat{\boldsymbol{v}}_c^k\big), \\ \boldsymbol{u}_i^k &= W_u^k \ \big[\overline{\boldsymbol{u}}_i^k | \boldsymbol{u}_i^{k-1}\big], \end{aligned} \tag{2}$$

where the $\overline{W}_u^k$ and W_u^k are parameter matrices, $(\cdot)^\top$ is the transpose operation and $[\cdot|\cdot]$ is concatenation operation. Analogously, we can employ the same learning procedure to update $\boldsymbol{v}_j^{k-1}$. In TFE, we stack K TALs and denote the final outputs of $\boldsymbol{u}_i^K$ and $\boldsymbol{v}_j^K$ as our steady embeddings $\boldsymbol{u}_i^m$ and $\boldsymbol{v}_j^m$ for user u_i and item v_j in $\mathcal{G}^m$, respectively.

Temporal Fusion Layer The proposed TAL can effectively deal with a single network snapshot, but it cannot capture the structural variations across the ordered snapshot sequence $\{\mathcal{G}^0, \mathcal{G}^1, ..., \mathcal{G}^{M-1}\}$. To mitigate this problem, after obtaining user and item embeddings (e.g. $\boldsymbol{u}_i^m$ and $\boldsymbol{v}_j^m$) for each discrete snapshot $\mathcal{G}^m$, we introduce a temporal fusion layer to encode these dynamical changes in the ordered snapshot sequence:

$$\boldsymbol{u}_i^m = f_u(\boldsymbol{u}_i^{m-1}, \boldsymbol{u}_i^m), \quad \boldsymbol{v}_j^m = f_v(\boldsymbol{v}_j^{m-1}, \boldsymbol{v}_j^m), \tag{3}$$

where f_u and f_v are temporal modeling functions. There are many alternative methods that can be used for concrete implementations. In our model, we choose two separate GRUs [5] to model f_u and f_v, respectively.

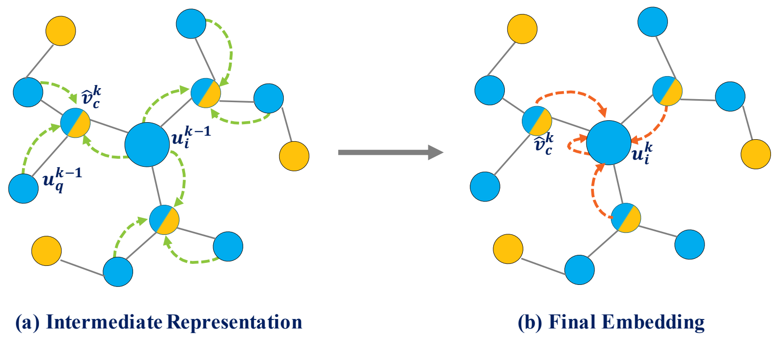

Fig. 2. Illustration of topological aggregation layer (TAL). Blue and yellow color nodes denote users and items, respectively. Nodes with two colors denote the intermediate representations, and gray lines denote that users have interacted with items. The subgraphs (a) and (b) show the learning procedures of $\boldsymbol{u}_i^{k-1}$ in k-th TAL. The green dotted lines (Eq.(1)) and orange dotted lines (Eq.(2)) describe how to derive the embedding $\boldsymbol{u}_i^k$ by considering the topological structure of temporal interaction network.

4.4 Attentive Shift Encoder

In this section, we develop an attentive shift encoder (ASE) for temporal interaction networks for capturing the long-range dependency structure. Previous works employ different RNN variants which tend to forget the long history information, leading to the problem of long-range dependency structure missing. In contrast, our ASE module can explicitly learn the dependencies between each historical interaction and the new-coming interaction via the attention mechanism.

Attentive Interaction Layer Considering a new-coming interaction (u_i, v_j, t) that user u_i has an interaction with item v_j at the timestamp t, we can use it to generate the dynamic embeddings of users and items and compute the correlation among historical interactions in $\mathcal{S}_{u_i}(t)$. The concrete implementation is given as:

$$e_h = \big(W_Q[\boldsymbol{u}_i|(\boldsymbol{v}_j + \boldsymbol{p}_{t^{|\mathcal{H}_{u_i}|}}(t))]\big)^\top W_K[\boldsymbol{u}_i|(\boldsymbol{v}^h + \boldsymbol{p}_{t^h}(t))], \text{where } (v^h, t^h) \in \mathcal{S}_{u_i}(t)$$
$$\alpha_h = \frac{\exp(e_h)}{\sum_{(v^c,t^c)\in\mathcal{S}_{u_i}(t)} \exp(e_c)}, \tag{4}$$
$$\boldsymbol{o}^t = \delta\big(\sum_{(v^h,t^h)\in\mathcal{S}_{u_i}(t)} \alpha_h W_V \boldsymbol{v}^h\big),$$

where $|\mathcal{H}_{u_i}|$ is the number of interaction sequence $\mathcal{S}_{u_i}(t)$, and $\boldsymbol{o}^t$ is the new-coming interaction feature. W_Q, W_K and W_V are the query, key and value parameter matrices, respectively. Afterwards, we generate the embeddings of user u_i and item v_j at timestamp t via the following operations:

$$\boldsymbol{u}_i(t) = g_u(\boldsymbol{u}_i, \boldsymbol{o}^t), \quad \boldsymbol{v}_j(t) = g_v(\boldsymbol{v}_j, \boldsymbol{o}^t), \tag{5}$$

where g_u and g_v are embedding generation functions. In our model, we also use two separate GRUs for their implementations.

Temporal Shift Layer Intuitively, the embeddings of user and item should be changed over time. For example, electronic products will gradually reduce their prices over time, and users may have different intentions when they returned to the E-commerce platform again. Hence, maintaining the same embeddings in a period cannot reflect the reality for the future prediction [2]. In our work, we devise a temporal shift layer to achieve dynamic embeddings over time. Specifically, after obtaining the embeddings of user u_i and item v_j at timestamp t, i.e., $\boldsymbol{u}_i(t)$ and $\boldsymbol{v}_j(t)$ in Eq.(5), their dynamic embeddings at future timestamp $t^+ \geq t$ can be calculated as follows:

$$\boldsymbol{u}_i(t^+) = (1 + \Delta \boldsymbol{w}_{u_i}) * \boldsymbol{u}_i(t), \quad \boldsymbol{v}_j(t^+) = \big(1 + \Delta \boldsymbol{w}_{v_j}\big) * \boldsymbol{v}_j(t), \tag{6}$$

where $\Delta = t^+ - t$ is the shift time interval, $*$ is the element-wise product, $\boldsymbol{w}_{u_i}$ and $\boldsymbol{w}_{v_j}$ are corresponding learnable shift vectors of user u_i and item v_j, respectively. We assume that the user or item embedding can shift in continuous space with its own trajectory, so each user or item has a specific shift vector.

4.5 Model Training

To explicitly capture dynamic influence between users and items, we devise a novel intensity function which is generated via the steady embeddings and dynamic embeddings.

Intensity Function We model all possible interactions for all users with items via a multi-dimensional intensity function, where each user-item pair holds one dimension. Formally, based on the learned user and item embeddings, the intensity function of user-item pair (u_i, v_j) is defined as follows:

$$\lambda_{(u_i,v_j)}(t) = \sigma\Big(\underbrace{(\boldsymbol{u}_i^{\lfloor \frac{t}{d} \rfloor})^\top \, \boldsymbol{v}_j^{\lfloor \frac{t}{d} \rfloor}}_{\substack{\text{base intensity} \\ \text{(TFE)}}} + \underbrace{(\boldsymbol{u}_i(t))^\top \, \boldsymbol{v}_j(t)}_{\substack{\text{dynamic change} \\ \text{(ASE)}}}\Big), \tag{7}$$

where $\lfloor \frac{t}{d} \rfloor$ denotes the corresponding network snapshot identifier and σ is the softplus function for ensuring that the intensity function is positive and smooth. Our intensity function is similar with MHP (detailed in Sec. 3.2): 1) The former term $(\boldsymbol{u}_i^{\lfloor \frac{t}{d} \rfloor})^\top \, \boldsymbol{v}_j^{\lfloor \frac{t}{d} \rfloor}$ is provided by TFE, which uses the topological structure of temporal interaction network as a strong prior to generate the base intensity. 2) The latter term $(\boldsymbol{u}_i(t))^\top \, \boldsymbol{v}_j(t)$ is obtained by ASE, which describes the dynamic changes for this user-item pair.

Algorithm 1 The training procedure of DSPP.

```
Input: The training temporal interaction network $\mathcal{G}(T_{tr})$, the ordered snapshot sequence $\{\mathcal{G}^0, \mathcal{G}^1, ..., \mathcal{G}^{M-1}\}$, the time interval $d$, the user set $\mathcal{U}$, the item set $\mathcal{V}$, sampling number $N$.
1: Initialize model parameters.
2: while not convergence do
3:    Enumerate a batch of consecutive interactions from $\mathcal{G}(T_{tr})$ as $B$.
4:    $\nabla \leftarrow 0$   \\ Happened interactions.
5:    $\varLambda \leftarrow 0$   \\ Non-happened interactions.
6:    for each interaction $(u_i, v_j, t) \in B$ do
7:       $m \leftarrow \lfloor \frac{t}{d} \rfloor$ \\ Calculate the snapshot identifier.
8:       Calculate steady embedding $\boldsymbol{u}_i^m$ and $\boldsymbol{v}_j^m$ via TFE based on $\mathcal{G}^m$.
9:       Calculate dynamic embedding $\boldsymbol{u}_i(t)$ and $\boldsymbol{v}_j(t)$ by ASE.
10:      $\nabla \leftarrow \nabla + \log(\lambda_{(u_i,v_j)}(t))$
11:      if $u_i$ has next interaction at future timestamp $t^+$ with item $v_c$ then
12:         $\nabla \leftarrow \nabla + \log(\lambda_{(u_i,v_c)}(t^+))$
13:         Uniformly sample a timestamp set $t^s = \{t_k^s\}_{k=1}^N \leftarrow \text{Uniform}(t, t^+, N)$.
14:         Sample the negative item set $\varUpsilon$ \\ Negative sampling.
15:         for $k \in \{2, ..., N\}$ do
16:            $\varLambda \leftarrow \varLambda + (t_k^s - t_{k-1}^s)\lambda(t_k^s)$ \\ Monte Carlo estimation.
17:         end for
18:      end if
19:   end for
20:   $\mathcal{L} \leftarrow \nabla - \varLambda$
21:   Update the model parameters by Adam optimizer.
22: end while
```

Objective Function Based on the proposed intensity function, we can train our model by maximizing the log-likelihood of these happened interactions during time window $[0, T)$:

$$\mathcal{L} = \sum_{(u_i,v_j,t)\in\mathcal{G}(T)} \log(\lambda_{(u_i,v_j)}(t)) - \underbrace{\int_0^T \lambda(t)\mathrm{d}t}_{\text{non-happened interactions}}, \tag{8}$$

$$\lambda(t) = \sum\nolimits_{u_i\in\mathcal{U}} \sum\nolimits_{v_j\in\mathcal{V}} \lambda_{(u_i,v_j)}(t). \tag{9}$$

Maximizing the likelihood function $\mathcal{L}$ can be interpreted intuitively in the following way: 1) The first term ensures that all happened interactions probabilities are maximized. 2) The second term penalizes the sum of the log-probabilities of infinite non-happened interactions, because the probability of no interaction happens during $[t, t + \mathrm{d}t)$ is $1 - \lambda(t)\mathrm{d}t$, and its log form is $-\lambda(t)\mathrm{d}t$ [19].

Prediction Tasks Beneficial from TPP framework, DSPP can naturally tackle the following two tasks:

– Item prediction: Given user u_i and a future time t^+, *what is the item that this user will interact at time t^+?* To answer this question, we rank all items and recommend the one that has the maximum intensity:

$$\operatorname{argmax}_{v_j} \frac{\lambda_{(u_i,v_j)}(t^+)}{\sum_{v_c \in \mathcal{V}} \lambda_{(u_i,v_c)}(t^+)}, \tag{10}$$

– Time prediction: Given the user u_i, item v_j and timestamp t_n, *how long will this user interact with this item again after timestamp t_n?* To answer this question, we estimate the following time expectation:

$$\Delta = \int_{t_n}^{\infty} (t - t_n) f_{(u_i,v_j)}(t) \mathrm{d}t, \tag{11}$$

where $f_{(u_i,v_j)}(t) = S_{(u_i,v_j)}(t)\lambda_{(u_i,v_j)}(t)$ is the conditional density (details in Section 3.2) and Δ is the expectation interaction time.

4.6 Model Analysis

Differences with Sequential Recommendation Sequential recommendation methods [10, 4, 13, 28] also focus on modeling sequential user preferences. Compared with them, DSPP has the following fundamental differences:

– In the task level, DSPP concentrates on modeling the dynamic evolution of users and items in continuous time. DSPP can not only predict the next item, but also explicitly estimates the time expectation of a given user-item interaction. In contrast, sequential recommendation aims to model interaction sequences in the discrete manner. Thus, most of them ignore the timestamp information and cannot model the time distribution.
– In the model level, DSPP simultaneously captures the topological structure and the long-range dependency structure via our TFE and ASE modules, but sequential recommendation methods usually ignore the topology information in temporal interaction networks.

Time Complexity To accelerate the training process of DSPP, we adopt *t-batch* algorithm [16] to organize data for paralleling training. Moreover, we apply Monte Carlo Algorithm [1] with the negative sampling trick [7] to estimate our objective function Eq.(8). Hence, the main operations of DSPP fall into the proposed TFE and ASE modules. The computational complexity of TFE is $\mathcal{O}(K|\mathcal{E}|D)$, and the ASE is $\mathcal{O}(HBD)$, where K is the number of TAL, B is the batch size, and H is a hype-parameter to control the maximum length of historical interactions. In general, our model keeps an efficient training speed. Empirically, in the same running environment, JODIE [16], DGCF [17] and DSPP would cost about 5.1 mins, 17.7 mins, and 8.15 mins per epoch on the Reddit dataset, respectively. The pseudo code of the training procedure is shown in Algorithm 1.

Table 1. Statistics of three datasets.

Datasets	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{V}\rvert$	Interactions	Action Repetition
Reddit	10,000	1,000	672,447	79%
Wikipedia	8,227	1,000	157,474	61%
Last.FM	1,000	1,000	1,293,103	8.6%

5 Experiments

5.1 Datasets

To make a fair comparison, we evaluate DSPP on three pre-processed benchmark datasets [16], i.e., Reddit[6], Wikipedia[7] and Last.FM[8]. The concrete statistics of users, items, interactions and action repetition are listed in Table 1. Noticeably, these datasets are largely different in terms of action repetition rate, which can verify whether DSPP is able to capture the dynamic influence in various action repetition scenarios accurately.

5.2 Experiment Setting

Data Preprocessing: As used in JODIE [16] and DGCF [17], for each dataset, we first sort all interactions by chronological order. Then, we use the first 80% interactions to train, the next 10% interactions to valid, and the remaining 10% interactions for the test. In contrast with JODIE and DGCF, we generate a snapshot sequence $\{\mathcal{G}^m\}_{m=0}^{M-1}$. In our setting, the validation snapshots cover the training data, and the test snapshots also contain all training and validation data.
Evaluation Metrics: To evaluate our model performance, for each interaction, we first generate corresponding user and item steady and dynamic embeddings. Then, we rank all items by Eq.(10) and predict the future time by Eq.(11). Afterward, we evaluate the item prediction task with the following metrics: Mean Reciprocal Rank (MRR) and Recall@10. higher values for both metrics are better. For the time prediction task, we use Root Mean Square Error (RMSE) to measure model performance, and a lower value for RMSE is preferred.
Baselines: We compare DSPP with the following baselines.

- Random walk model: CTDNE [21].
- Recurrent network models: LSTM [11], RRN [27], LatentCross [2], Time-LSTM [30], JODIE [16] and DGCF [17].
- Temporal point process model: DeepCoevolve [7].

Implementation Details: In our experiments, we use the official implementations[9] of DeepCoevolve. Except from it, we directly report the experimental

[6] `http://snap.stanford.edu/jodie/reddit.csv`
[7] `http://snap.stanford.edu/jodie/wikipedia.csv`
[8] `http://snap.stanford.edu/jodie/lastfm.csv`
[9] `https://hanjun-dai.github.io/supp/torch_coevolve.tar.gz`

Table 2. Performance (%) comparison of item prediction.

Model	Last.FM		Wikipedia		Reddit	
	Recall@10	MRR	Recall@10	MRR	Recall@10	MRR
CTDNE	1.0	1.0	5.6	3.5	25.7	16.5
LSTM	12.7	8.1	45.9	33.2	57.3	36.7
Time-LSTM	14.6	8.8	35.3	25.1	60.1	39.8
RRN	19.9	9.3	62.8	53.0	75.1	60.5
LatentCross	22.7	14.8	48.1	42.4	58.8	42.1
DeepCoevolve	33.6	21.3	60.6	48.5	78.7	65.4
JODIE	38.7	23.9	82.1	74.6	85.1	72.4
DGCF	45.6	32.1	85.2	78.6	85.6	72.6
DSPP	**47.1**	**34.3**	**90.5**	**82.1**	**86.7**	**74.5**

results in the original papers [16, 17]. DSPP follows the same hyper-parameter setting with baselines: the embedding dimension D is fixed as 128, the batch size B is fixed as 128, the learning rate is fixed as 0.001, the model weight decay is fixed as 0.00001, the sampling number for Monte Carlo estimate is fixed as 64, the number of negative sampling is fixed as 10, the number of TAL K is fixed as 2, the Attention is stacked 8 layers, the GRUs are 1 layer, the number of snapshots M is selected from $\{128, 256, 512, 1024\}$ and the maximum length of interaction sequence H is chosen from $\{20, 40, 60, 80\}$. The Adam [14] optimizer is used to update all model parameters.

5.3 Item Prediction

For item prediction, Table 2 shows the comparison results on the three datasets according to Recall@10 and MRR. From the experimental results, we have the following observations:

– DSPP consistently yields the best performances on all datasets for both metrics. Compared with state-of-the-art baselines, the most obvious effects are that DSPP achieves the 6.8% improvement in terms of MRR on Last.FM, the 6.2% improvement in terms of Recall@10 on Wikipedia and the 2.6% improvement in terms of MRR on Reddit. It reveals that incorporating the topological structure and long-range dependency structure can bring good robustness in different action repetition scenarios.
– DSPP outperforms DeepCoevolve significantly. This phenomenon demonstrates that DSPP has a more powerful intensity function that can better capture the dynamic influence between users and items.
– DSPP and DGCF are superior to other baselines on Last.FM, which indicates that it is critical to model the topological structure information for learning temporal interaction networks.

Table 3. Performance (hour^2) comparison of time prediction.

Model	Last.FM	Wikipedia	Reddit
DeepCoevolve	9.62	10.94	11.07
DSPP	7.78	8.71	9.06

Table 4. Performance (%) comparison of different model variants.

Model	Recall@10	MRR
DSPP	47.1	34.3
Remove TFE	40.2	25.3
Replace TAL with GCN	44.5	32.4
Replace TAL with GAT	45.2	32.7

5.4 Time Prediction

Table 3 shows the prediction performances of DeepCoevolve and DSPP. From it, we can observe that DSPP achieves more accurately time prediction. Specifically, our model achieves the 19.1% improvement on Last.FM, the 20.3% improvement on Wikipedia and the 18.1% improvement on Reddit. We suppose that the improvements owe to the following reasons: 1) DeepCoevolve uses a linear intensity function to model dynamic influence over time, which would reduce the model flexibility of intensity function. 2) DeepCoevolve remains the same user and item embeddings until it involves a new-coming interaction, so it limits model expressiveness. In contrast, our intensity function can learn the nonlinear dynamic influence, since the ASE module can provide time-aware dynamic embeddings.

5.5 Discussion of Model Variants

To investigate the effectiveness of our model components, we implement several variants of DSPP and conduct the experiment on Last.FM dataset for the task of item prediction. The experimental results are reported in Table 4. According to it, we can draw the following conclusions:

– Remove TFE. To verify whether our proposed TFE module is useful to enhance the expressiveness of the intensity function, we first remove it and only remain the second term of Eq.(7) as our intensity function. Then, we directly predict the item that is most likely to interact with each user via Eq.(10). As shown in the results, both metrics Recall@10 and MRR sharply drop 14.6% and 26.2%, respectively. It demonstrates that modeling the topological structure of temporal interaction networks can provide a powerful structural prior for enhancing the expressiveness of intensity function.
– Remove TFE. This variant can be also viewed as a non-graph based model, since it does not exploit the topological structure, and the remaining temporal attention shift encoder only provides the long-range dependency structure to model intensity function. Compared with DeepCoevolve, this variant

yields 19.6% and 18.7% improvements on Recall@10 and MRR respectively. This observation shows that our proposed temporal attention shift encoder can further enhance the intensity function.

– Replace TAL with GCN/GAT. To verify whether our proposed TAL is superior to other graph encoders for capturing the topological structure of temporal interaction networks. We replace TAL by GCN [15] and GAT [26]. For the GCN variant, both Recall@10 and MRR drop 5.8%. For GAT variant, Recall@10 and MRR drop 4.0% and 4.6%, respectively. So, We suppose that our proposed TAL can better capture the information of the same type entity.

6 Conclusion

In this paper, we present the deep structural point process for learning temporal interaction networks. Our model includes two proposed modules, i.e., topological fusion encoder and attentive shift encoder to learn the topological structure and the long-range dependency structure in temporal interaction networks, respectively. On top of that, a novel intensity function, which combines the learned steady and dynamic embeddings, is introduced to enhance the model expressiveness. Empirically, we demonstrate the superior performance of our model on various datasets for both tasks of item prediction and time prediction.

Acknowledgement

This work is supported in part by the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No. XDC02040400) and the Youth Innovation Promotion Association of CAS (Grant No. 2021153).

References

1. Aalen, O., Borgan, O., Gjessing, H.: Survival and event history analysis: a process point of view. Springer Science & Business Media (2008)
2. Beutel, A., Covington, P., Jain, S., Xu, C., Li, J., Gatto, V., Chi, E.H.: Latent cross: Making use of context in recurrent recommender systems. In: WSDM (2018)
3. Cao, J., Lin, X., Guo, S., Liu, L., Liu, T., Wang, B.: Bipartite graph embedding via mutual information maximization. In: WSDM (2021)
4. Chen, X., Xu, H., Zhang, Y., Tang, J., Cao, Y., Qin, Z., Zha, H.: Sequential recommendation with user memory networks. In: WSDM (2018)
5. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. ArXiv (2014)
6. Da, X., Chuanwei, R., Evren, K., Sushant, K., Kannan, A.: Inductive representation learning on temporal graphs. In: ICLR (2020)
7. Dai, H., Wang, Y., Trivedi, R., Song, L.: Deep coevolutionary network: Embedding user and item features for recommendation. ArXiv (2016)

8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. ArXiv (2018)
9. Hawkes, A.G.: Spectra of some self-exciting and mutually exciting point processes. Biometrika (1971)
10. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. ICLR (2016)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation (1997)
12. Iba, T., Nemoto, K., Peters, B., Gloor, P.A.: Analyzing the creative editing behavior of wikipedia editors: Through dynamic social network analysis. Procedia-Social and Behavioral Sciences (2010)
13. Kang, W.C., McAuley, J.: Self-attentive sequential recommendation. In: ICDM (2018)
14. Kingma, P.D., Ba, L.J.: Adam: A method for stochastic optimization. In: ICLR (2015)
15. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
16. Kumar, S., Zhang, X., Leskovec, J.: Predicting dynamic embedding trajectory in temporal interaction networks. In: KDD (2019)
17. Li, X., Zhang, M., Wu, S., Liu, Z., Wang, L., Yu, P.S.: Dynamic graph collaborative filtering. In: ICDM (2020)
18. Liyanagunawardena, T.R., Adams, A.A., Williams, S.A.: Moocs: A systematic study of the published literature 2008-2012. International Review of Research in Open and Distributed Learning (2013)
19. Mei, H., Eisner, J.M.: The neural hawkes process: A neurally self-modulating multivariate point process. In: NeurIPS (2017)
20. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. ArXiv (2013)
21. Nguyen, G.H., Lee, J.B., Rossi, R.A., Ahmed, N.K., Koh, E., Kim, S.: Continuous-time dynamic network embeddings. In: WWW (2018)
22. Sankar, A., Wu, Y., Gou, L., Zhang, W., Yang, H.: Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In: WSDM (2020)
23. Seo, Y., Defferrard, M., Vandergheynst, P., Bresson, X.: Structured sequence modeling with graph convolutional recurrent networks. In: ICONIP (2018)
24. Skarding, J., Gabrys, B., Musial, K.: Foundations and modelling of dynamic networks using dynamic graph neural networks: A survey. ArXiv (2020)
25. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, u., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)
26. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
27. Wu, C.Y., Ahmed, A., Beutel, A., Smola, A.J., Jing, H.: Recurrent recommender networks. In: WSDM (2017)
28. Xu, C., Zhao, P., Liu, Y., Xu, J., S. Sheng, V.S.S., Cui, Z., Zhou, X., Xiong, H.: Recurrent convolutional neural network for sequential recommendation. In: WWW (2019)
29. Yan, J., Xu, H., Li, L.: Modeling and applications for temporal point processes. In: KDD (2019)
30. Zhu, Y., Li, H., Liao, Y., Wang, B., Guan, Z., Liu, H., Cai, D.: What to do next: Modeling user behaviors by time-lstm. In: IJCAI (2018)