Interaction Event Network Modeling based on Temporal Point Process

Hang Dong[1], Kaibo Wang[1*]

[1]Department of Industrial Engineering, Tsinghua University, Beijing, China

[*]E-mail: kbwang@tsinghua.edu.cn

Abstract

Interaction event networks, which consist of interaction events among a set of individuals, exist in many areas from social, biological to financial applications. The individuals on networks interact with each other for several possible reasons, such as periodic contact or reply to former interactions. Regarding these interaction events as expectations based on previous interactions is crucial for understanding the underlying network and the corresponding dynamics. Usually, any change on individuals of the network will reflect on the pattern of their interaction events. However, the causes and expressed patterns for interaction events on networks have not been properly considered in network models. This paper proposes a dynamic model for interaction event networks based on the temporal point process, which aims to incorporate the impact from historical interaction events on later interaction events considering both network structure and node connections. A network representation learning method is developed to learn the interaction event processes. The proposed interaction event network model also provides a convenient representation of the rate of interaction events for any pair of sender-receiver nodes on the network and therefore facilitates monitoring such event networks by summarizing these pairwise rates. Both simulation experiments and experiments on real-world data validate the effectiveness of the proposed model and the corresponding network representation learning algorithm.

Keywords: temporal point process, network monitoring, network representation learning, interaction event, network model

1 Introduction

Network data widely exist nowadays, such as biological networks, financial networks, and academic networks. The most commonly seen example is social networks. On such networks, individuals interact with each other through interaction events, such as mention, comment, sending messages, or forwarding content to others. These interaction events are often collected and recorded for a number of social networks, which we refer to as interaction event networks (**Lerner** et al. 2013). To fully understand the dynamics of these networks, it is crucial to incorporate the triggering motivations for these interaction events into the network model. In reality, these interaction events can be motivated by a number of different reasons, such as periodic contact or reply to a former interaction event. However, these dynamic interactions are mostly aggregated as simple edges over nodes of individuals on networks (**Newman** et al. 2002) in previous studies. Such a simplification operation completely ignores the rich information in the occurring orders, the time intervals between related events as well as the motivation for each interaction event. It is therefore important to develop methods to characterize the occurring process without losing these types of information for interaction events in network models.

Figure 1 shows an example of an interaction event network: an email network inside an organization, with corresponding email records in Table 1. In this example, 7 individuals from A to G send emails on the network at several time stamps from t_1 to t_6. Each email records the sender, receiver, and the sent time of the email. In such a network, routine information of the organization usually spread over through fixed sub-networks, and therefore these interaction events can well reflect the running status of the underlying organization.

Intuitively, for a specific sender, the interaction event of sending an email is probably caused by a former received email. For example, in Figure 1, the email sent from A to F at time t_6 is probably a reply to the email A received from F at time t_4. Moreover, the behavior of sending an email can also be a custom, such as the emails sent from B to E at time t_1 and t_3 in Figure 1. The above observations indicate that considering the influence of related historical interaction events on each new interaction event is desirable for modeling interaction event networks.

To effectively incorporate the influence from historical interaction events on later ones and understand the dynamics on networks, we propose a temporal point process network model for interaction event networks, which explicitly models the rate of interaction events on the

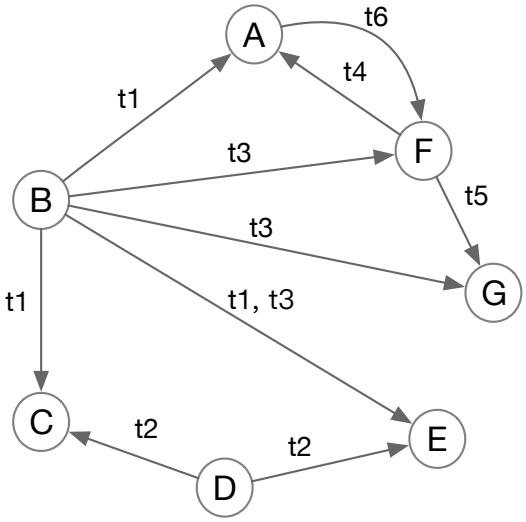


Figure 1: Example of an interaction event network.

Table 1: Email network in Figure 1.

Sender	Receiver	Time
B	A	t_1
B	C	t_1
B	E	t_1
D	C	t_2
D	E	t_2
B	E	t_3
B	F	t_3
B	G	t_3
F	A	t_4
F	G	t_5
A	G	t_6
.....	.....	.....

network with the information from related historical events and related nodes. A temporal point process describes how a series of events happen, and therefore is used in this work to model the rate of interaction events between a sender-receiver pair of nodes on the network. We also inherit the assumption of latent space network models (**Hoff** et al. 2002) that each node on the network can be projected into a latent space where the node information can be well preserved as the position in that latent space. To learn the model parameters and the node representation vectors (latent positions) of the network, we propose a network representation learning algorithm that is effective and efficient for the task. On one hand, the proposed model can effectively preserve the structural information of the network by node representation vectors; on the other hand, the event dynamics on the network can be well explained by the pairwise rate of interaction events over all the node pairs on the network. The proposed model also facilitates the network monitoring task by providing the rate of interaction events between each possible sender-receiver node pair. Simulation experiments and experiments on real-world data show that the proposed model can well characterize the influence from past events on later ones, and the proposed network representation learning algorithm can preserve sufficient node information for downstream tasks.

The remainder of the paper is organized as follows. Section 2 reviews research works in related fields, including temporal point process models, network representation learning, and network monitoring. In Section 3, we introduce the temporal point process network (TPPN) model which incorporates the pairwise rate of sender-receiver node interaction events into a latent space network model. Section 4 describes the network representation learning algorithm for learning the node representation vectors on such networks. In Section 5, both simulation experiments and experiments on real-world datasets validate the effectiveness of the proposed model and its corresponding learning algorithm. Finally, we conclude this work and propose future research directions in Section 6.

2 Literature Review

Rich works have been done in related fields, including in temporal point process models, network representation learning, and network monitoring. In this section, we review relevant works in the above fields and point out opportunities for this research to contribute.

2.1 Temporal Point Process Models

A temporal point process is a stochastic process that characterizes the rate of events occurring along the timeline. The key component of such a process is the conditional intensity function (CIF) characterizing this rate for each type of event. Due to the prevalence of such event data, the temporal point process has a lot of applications in the real world for event sequence modeling (**Daley** et al. 2008). These events can be seismic events, purchase events, device failure, communication activities, and many others.

According to the form of conditional intensity functions, the temporal point process has different types, such as the homogeneous Poisson process, the non-homogeneous Poisson process, and the Hawkes process. Among them, the Hawkes process (**Hawkes** 1971) is a self-exciting point process, where historical events have a positive exciting influence on later events. This characteristic fits well in a lot of realistic scenarios in many different fields, including financial trades, information systems, and social networks. Therefore, a lot of variants of the Hawkes process have been proposed for different applications.

For example, Hawkes process has been used in social network modeling (**Li** et al. 2014), ATM failure prediction (**Xiao** et al. 2017), information system analytics (**Yan** et al. 2015) and patient flow prediction (**Xu** et al. 2017). One key difference between the temporal point process and traditional time series is that a temporal point process preserves all the specific time when each event happens, compared to the underlying hypothesis of uniform sampling in time for time series. By preserving the exact timestamp when the event occurs, this kind of temporal point process can well capture the time intervals between events in the model.

For event sequences on networks, there are also several works modeling dynamic networks with temporal point process. For example, **Perry** et al. (2013) proposed to use temporal point process to characterize the repetitive directed interaction events on networks; **Linderman** et al. (2014) developed a probabilistic model combining mutually-exciting point processes with random graph models; **Hall** et al. (2016) proposed an online learning framework of multivariate Hawkes process model to track the network structure of social networks as it evolves; **Mei** et al. (2017) relaxed the positive influence assumption of the Hawkes process, and constructed a neurally self-modulating multivariate point process which can be learned through a continuous-time LSTM neural network; **Junuthula** et al. (2019) combined the stochastic block model with Hawkes process as a block point process to incorporate the community structure in social

network; **Zuo** et al. (2018) integrated the neighborhood formulation process as Hawkes process into network embedding so as to capture the influence of historical neighbors on the current neighbors. However, these works either neglected the structural positions of nodes on the network or did not consider interaction events on the network.

2.2 Network Representation Learning

Network representation learning is becoming popular for its powerful capability in processing large networks efficiently in the recent decade. From the model perspective, network representation learning is based on a latent space assumption, assuming each node has a latent position in the latent space where the distance between nodes in that space should correspond to the dissimilarity of these nodes in the actual application scenario.

To analyze networks more effectively, proper representation forms of networks are required. However, simply using the adjacency matrix tends to neglect the complex and high-order relationships on the network, such as paths and frequent sub-structures. Network representation learning helps to embed the network information into a latent space, where traditional machine learning algorithms based on vectors can be adopted conveniently for subsequent analyzing tasks, such as node classification and link prediction.

Early network representation learning originated from spectral clustering (**Brand** et al. 2003) and manifold learning (**Tenenbaum** et al. 2000; **Roweis** et al. 2000), conducting eigen-decomposition on the adjacency matrix of the network or preserving distances with neighbors to find a low-dimensional representation of the network. Recently, by virtue of the word2vec (**Mikolov** et al. 2013) and skip-gram model in natural language processing, many methods such as DeepWalk (**Perozzi** et al. 2014), LINE (**Tang** et al. 2015) and node2vec (**Grover** et al. 2016) have been proposed to learn node representation from a static network structure. As **Qiu** et al. (2018) and **Liu** et al. (2019) showed, these methods based on static network structure can be treated as a matrix decomposition task, which is equal to optimizing two objectives: making embedded vectors of similar nodes as close as possible in the latent space and making embedded vectors of different nodes as far as possible in the latent space. Based on the learned node representation vectors, many tasks related to network analytics can be conducted, including node classification (**Bhagat** et al. 2011), link prediction (**Liben-Nowell** et al. 2007), clustering (**Newman** 2006), visualization (**Maaten** et al. 2008) and so on.

There are still some main challenges for network representation learning, including preserving network structure and context information, dealing with data sparsity and scalability to large-scale networks (**Zhang** et al. 2017). Recently, more and more works focus on network representation learning from different perspectives, such as heterogeneous network embedding (**Chang** et al. 2015), network representation learning based on deep neural networks (**Li** et al. 2017), representation learning on directed networks(**Ou** et al. 2016) and so on.

For dynamic networks, a stream of works based on connectivity, spanning tree (**Holme** et al. 2012), and graph streams (**McGregor** 2014) have been proposed. Usually, dynamic network models create a series of static network slices according to a fixed period (**Kumar** et al. 2006). For example, **Nguyen** et al. (2018) learns node embedding for dynamic networks, where time is only used for identifying the order of edges, neglecting the specific time difference between two events. Generally, the dynamics on networks have not been fully considered as events with their specific timestamps in dynamic network models. In our work, we try to accommodate the specific time of events in the model and explicitly characterize the influence of historical events on later events in the network.

2.3 Network Monitoring

Network monitoring aims to monitor a network system and raise an alarm once the network goes out of control. It is also referred to as anomaly detection and treated as a change-point problem (**Antoch** et al. 1993). **Savage** et al. (2014) reviewed anomaly detection works on online social networks and classified these works by the target network is static or dynamic, whether there are node labels in the network. Another review of social network monitoring by **Woodall** et al. (2017) showed the relationships between network monitoring and engineering statistics or public health surveillance.

Generally, one stream of network monitoring methods is based on community structure (**Jun** et al. 2009), including monitoring based on variants of the stochastic block model (**Wilson** et al. 2016; **Dong** et al. 2020). These methods monitor the pairwise model parameters of different communities to reflect the overall status of the network. Another stream of methods monitors networks through representative network metrics. For example, **Priebe** et al. (2005) used scan statistics to detect anomaly events in email networks. **Cheng** et al. (2013) not only monitored scan statistics but also used cross-correlations between scanned network metrics in the moving

window to detect changes on the network. Furthermore, traditional monitoring methods in the field of statistical process control are also used for network monitoring, conducting exponentially weighted moving average (EWMA) or cumulative sum (CUSUM) control chart based on network metrics such as average betweenness and average closeness (**Noorossana** et al. 2018). **Neil** et al. (2013) used a scan statistic for both time windows and sub-graphs to detect anomalous subgraphs in computer networks. **Azarnoush** et al. (2016) proposed a network monitoring method based on the likelihood function combining the existence of edges with node attributes. However, these methods do not explicitly consider the dynamics of events on the network in the monitoring scheme.

There is still a great need for effective models and monitoring methods on networks incorporating the dynamics of interaction events, especially explainable motivations for these interaction events. In the following section, we will introduce a network event model that can well characterize the continuous dynamics of interaction events on the network, with an effective network representation learning method that can learn the structural information as well as the dynamics of interaction events on the network.

3 Network Event Model Based on Temporal Point Process

To effectively characterize the dynamics of interaction event networks, we propose a model for events on networks based on the temporal point process. The major contribution of this work is as follows: First, we develop a network model incorporating the influence of historical events on later events for interaction event networks, which is largely ignored in previous works; Second, the proposed model can be easily learned with the proposed network representation learning technique, which also provides convenient representations of nodes preserving structural information of the whole network and can perform well for classic network tasks such as node classification and link prediction; Last but not least, the proposed model can well restore the rate of all the possible events on the network and thus facilitate monitoring for such interaction event networks.

Figure **??** shows an example of an interaction event network. In Figure **??**, each node can be treated as an email account in an email network, and each edge is an event sending an email

from a sender to a receiver, with the specific sending time recorded. The emails sent at time t_k and t_{k+1} are shown in red arrows. Assume that we are concerned about the email-sending behavior of account F to A, B to E, and B to F, it is convenient to express the rate of the sending behavior of these pairs along the timeline in a functional curve as is shown in Figure **??**. This is the main motivation of the proposed model. From the perspective of formulating the event rate, it is straightforward to consider the influence of historical events effective on later events. In the above email case, as well as in other communication or purchase applications, each individual's interaction behavior is highly dependent on the others around in the network, thus the rate of interaction events should also be influenced by past events related to the current individual's network structure. Our work is based on this intuition and uses a temporal point process to characterize the influence of past events. Parameters of this temporal point process are learned by network representation learning.

For a given series of interaction events $\mathcal{H} = (s_i, r_i, t_i)_{i=1,2,\ldots}$, we aim to model the dynamics of these interaction events by explicitly express the rate of events through temporal point processes. A one-dimensional temporal point process is a random measure that maps each Borel set on $\mathbb{R}^+$ into a positive integer. Intuitively, we use $N\{(a, b)\}$ to represent the number of points (events) during the time period (a, b), where $N(t)$ is a counting process.

The above one-dimensional case is easily extended to a multi-dimensional case. Specifically, a multivariate temporal process can be represented by several counting processes $N = \{N_c(t)\}_{c\in\mathcal{C}}$, where $N_c(t)$ is the number of type-c events happened until time t. For each process $N_c(t)$, we note the immediate occurrence rate as $\lambda_c(t)$, which is the conditional intensity function:

$$\lambda_c(t) = \frac{\mathbb{E}[dN_c(t)|\mathcal{H}_t]}{dt}, \quad \mathcal{H}_t = \{(t_i, c_i)|t_i < t, c_i \in \mathcal{C}\}, \tag{1}$$

where $\mathcal{H}_t$ represents all the historical observations before time t. Under rather mild conditions, a temporal point process can be determined by a specific conditional intensity function uniquely.

For the interaction events on a network, we record the event history as a triplet $\mathcal{H} = \{(s_k, r_k, t_k)\}$, where s_k and r_k represent the sender and receiver of the event k, and t_k is the time of the event. In our proposed model, the conditional intensity function of the temporal

point process from one node u to another node v on the network is:

$$\lambda_{uv}(t) := g\left(\mu_{uv} + \sum_{k=1}^{K}\left[I_{r_k=u}\left(f_1(\mathbf{h}_{s_k}, \mathbf{h}_u, t-t_k)\right) + I_{s_k=u, r_k=v}\left(f_2(\mathbf{h}_u, \mathbf{h}_v, t-t_k)\right)\right]\right) \tag{2}$$

where μ_{uv} represents the base rate, I is the indicator function and only when the content meets the condition does this term equals to 1 and otherwise 0; $\mathbf{h}.$ is a D-dimensional representative vector of the specific node in the latent space. Assume that there are K interaction events up to time t, and for each event k, it has the sender s_k, receiver r_l. f_1, f_2 are mapping functions of $\mathbb{R}^D \times \mathbb{R}^D \times \mathbb{R} \to \mathbb{R}$, and g is an $\mathbb{R} \to \mathbb{R}$ function.

The first term in the summation part in (2) represents the influence from all the historical events when node u has appeared as a receiver on the current u-to-v event. In the example shown in Figure 2, to evaluate the conditional intensity of u-to-v event at time t_3, then the first term in (2) represents the influence from the past two events at t_0 and t_2 when node u was a receiver node.

The second term in the summation part in (2) represents the influence from all the historical events that have the same pattern, i.e. have the same sender node and receiver node as the current sender-receiver pair. In Figure 2, this term corresponds to the influence of the event occurred at time t_1 which is also an event from u to v.

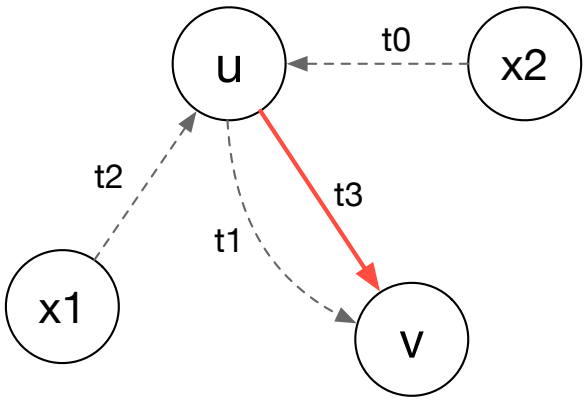


Figure 2: Illustration of the conditional intensity function of u-to-v event at time t_3.

By decomposing the conditional intensity function into external transitional influence and repeated pattern influence (corresponding to the first term and second term described above), this conditional intensity function is able to characterize the event occurrence process of interactions between all the node pairs on the network.

More details of the model form in (2) are as follows. The base rate μ_{uv} shows the affinity between the sender node and the receiver node in the latent space, so we use a negative squared Euclidean distance in this base rate:

$$\mu_{uv} = -\|\mathbf{h}_u - \mathbf{h}_v\|^2 \tag{3}$$

where $\mathbf{h}_u$ and $\mathbf{h}_v$ represent the representation vector of node u and node v in the latent space respectively. Functions f_1, f_2 and g are:

$$f_1(\mathbf{h}_{s_k}, \mathbf{h}_u, t - t_k) = f_2(\mathbf{h}_{s_k}, \mathbf{h}_u, t - t_k) = \alpha_{s_k,u}\kappa(t - t_k) \tag{4}$$

$$g(\cdot) = \exp(\cdot) \tag{5}$$

And $\kappa(t - t_k)$ is the influence from past events on the current time which decays with time:

$$\kappa(t - t_k) = \exp(-\delta_{s_k}(t - t_k)) \tag{6}$$

where δ_{s_k} is a decay parameter that depends on the sender node, indicating that in each historical interaction event, different sender nodes have different influence on later events. $\alpha_{s_k,u}$ is a coefficient that depends on the distance between the sender and receiver in historical events, so we also adopt the negative squared Euclidean distance as the metric:

$$\alpha_{s_k,u} = -\|\mathbf{h}_{s_k} - \mathbf{h}_u\|^2 \tag{7}$$

For all the historical events until time T, we can thus calculate the conditional intensity function for any sender-receiver node pair at any time $0 \leq t \leq T$ using (2). In this way, all the interaction events on the whole network are incorporated in this model. In the next section, the network representation learning method using historical events on the network to learn the model parameters will be introduced.

4 Model Inference and Event Network Monitoring

In our proposed model, the key parameters that need to be estimated include $\{\mathbf{h}_i, i = 1, ..., N\}$ and $\{\delta_i, i = 1, ..., N\}$. In this section, we introduce a network representation learning method

to inference these parameters.

For a specific sender of an event u, the probability that the receiver of the event is v is:

$$\mathbb{P}(v|u, \mathcal{H}(t)) = \frac{\lambda_{uv}}{\sum_{v'} \lambda_{uv'}(t)}. \tag{8}$$

Based on (2), we can obtain the conditional intensity function of all the observations of interaction events on the network up to time T, and the log likelihood function if these events are:

$$\log \mathcal{L} = \sum_{u \in V} \sum_{v \in \mathcal{H}(\mathcal{T})} \log \mathbb{P}\left(v|u, \mathcal{H}(T)\right). \tag{9}$$

To learn high-quality vector representations of nodes on the network, we adopt the negative sampling method to approximately optimize the log-likelihood function (**Mikolov** et al. 2013). Negative sampling can help to avoid the the in huge amount of calculation brought by summing over all nodes in (9). In this setting, the objective function corresponds to a sender node u and a receiver node v can be calculated as:

$$\log \sigma(\tilde{\lambda}_{uv}(t)) + \sum_{m=1}^{M} \mathbb{E}_{u^k \sim \mathbb{P}_n(u)}[-\log \sigma(\tilde{\lambda}_{u^k v})(t)] \tag{10}$$

where M is the number of negative samples for node v, which is set to follow a distribution $\mathbb{P}_n(u)$, such as $\mathbb{P}_n(u) \sim d(u)^{3/4}$ where $d(u)$ is the degree of node u. $\sigma(x)$ is the sigmoid function:

$$\sigma(x) = 1/(1 + \exp(-x)) \tag{11}$$

Additionally, the number of historical events considered in the calculation will have an impact on the computation load of the conditional intensity function $\lambda_{uv}(t)$, and the impact from far early historical events is so trivial that can be neglected. Therefore, in the model inference procedure, the maximum number of related historical events h is fixed, which means we only preserve the most recent h valid related historical events in the optimization of the objective function.

Classical optimization strategies can be easily adapted to optimize the objective function in (10), such as stochastic gradient descent (SGD) and Adam (**Kingma** et al. 2014). Without loss

of generality, we use the classic SGD optimizer, which can also be replaced by other off-the-shelf optimizers. The complete algorithm is shown in Algorithm 1.

Algorithm 1 TTPN Algorithm

Require: Network $G = (V, E)$; Set of events $\mathcal{H}$; Sampling size$w$; Fixed length of historical events$h$;
Ensure: Node representations $\mathbf{H} = \{\mathbf{h}_v\}_{v \in V}$, node attribute $\{\delta_v\}_{v \in V}$;
1: Initialization: Let $t = 0$; Randomly initialize $\mathbf{H}$ ad $\boldsymbol{\delta} = [\delta_1, ..., \delta_N]$;
2: **for** i=1,...,w **do**
3: Sample in $\mathcal{H}$ and obtain node u, node v, time t and related event history $\mathcal{H}_{uv}$;
4: Generate number of negative samples according to $\mathbb{P}_n(u)$ and conduct negative sampling;
5: Calculate the value of the objective function according to (10);
6: **end for**
7: **for** b=1,2,,,B **do**
8: Conduct SGD to optimize the objective function;
9: **end for**
10: **if** SGD result does not converge **then**
11: Back to step 7;
12: **else**
13: return $\mathbf{H} = \{\mathbf{h}_v\}_{v \in V}$ and $\{\delta_v\}_{v \in V}$;
14: **end if**

Based on the network representation learning results, we can establish the conditional intensity function between each sender-receiver node pair on the network:

$$\lambda_{uv}(t), \quad u, v = 1, ..., N, u \neq v \tag{12}$$

Given all the conditional intensity over the network, we can monitor either some specific local structures or the whole network. As a demonstration of model usage, we introduce two straight-forward methods for global monitoring of the whole network. In these methods, the summation and the maximum value for all the sender-receiver node pairs on the network are monitored separately:t

$$s_1(t) = \sum_{u=1}^{N} \sum_{v=1}^{N} \lambda_{uv}(t), \ u \neq v \tag{13}$$

$$s_2(t) = \max_{u,v} \lambda_{uv}(t), \ u \neq v \tag{14}$$

The above two types of monitoring strategies emphasize different aspects of the network: the summation method tends to characterize the overall status of the network, while the maximum value emphasizes more on the event intensity on active nodes. We will demonstrate this

difference through extensive experiments.

5 Experiments

5.1 Simulation Experiments

In simulation experiments, we generate interaction events over a network according to our proposed model form and make inference on model parameters based on our proposed network representation learning algorithm. We aim to serve the following three purposes in this experiment: First, we check the effectiveness of the estimated conditional intensity function by predicting the sender or receiver for the next event; Second, we evaluate the accuracy of estimated CIFs by comparing the estimated values with true values; Third, we show the potential for network monitoring with the proposed model and monitoring strategy.

The hardware environment of experiments in this work is an Azure NC6 instance virtual machine, with a CPU of Intel Xeon E5-2690v3, 2.60GHz, and 56GB memory of 8 cores.

The generation process of the simulated interaction events on the network is described in Algorithm 2. We generate events and train the model with the first 90%, test the performance of the model with the rest 10%.

Algorithm 2 TTPN Simulation Algorithm

Require: Number of nodes N; Dimension of representation vectors d; Number of events E; Start time t_0; Number of communities K;

Ensure: A series of interaction events on the network, each event includes a sender s, a receiver r and the time t;

1: Initialization: For each node in each community, randomly set the node representation to be a vector of length d;
2: **for** $i, j = 1, ..., N, i \neq j$ **do**
3: According to the thinning algorithm (**Ogata1981**), generate the proposed time τ_{ij} for the first event between each node pair;
4: **end for**
5: **for** k = 1,2,...,E **do**
6: Let $(s_k, r_k) = \text{argmin}\{\tau_{ij}\}, t_k = \min\{\tau_{ij}\}$;
7: Append (s_k, r_k, t_k) to the network event sequence;
8: Update related proposed time for next event $\tau_{s_k r_k}$ and $\tau_{r_k j}$, where $j = 1, ..., N, j \neq r_k$;
9: **end for**
10: **return** $\{(s_1, r_1, t_1), (s_2, r_2, t_2), ..., (s_E, r_E, t_E)\}$

5.1.1 Next Event Prediction

A direct application of the learned conditional intensity function is to predict the next interaction event for a specific node, including predict the receiver of the next event when the specific node is the sender, and predict the sender of the next event when the specific node is the receiver. These two tasks are also useful in real-world applications, such as predict to whom a specific user will next send an email.

For a specific node i as a sender in the network, the prediction of the next receiver is

$$\arg\max_{j \neq i} \mathbb{P}(j|i, \mathcal{H}(t)) \tag{15}$$

Combining with (8), this prediction is

$$\arg\max_{j \neq i} \mathbb{P}(j|i, \mathcal{H}(t)) = \arg\max_{j \neq i} \lambda_{ij}(t) \tag{16}$$

Similarly, we can predict the sender of the next sender for a specific node as receiver:

$$\arg\max_{j \neq i} \mathbb{P}(j|i, \mathcal{H}(t)) = \arg\max_{j \neq i} \lambda_{ji}(t) \tag{17}$$

With the simulation method described in Algorithm 2, we simulate 20,000 interaction events on a network of 30 nodes. To mimic the phenomenon that individuals in a network tend to form communities (**Newman** 2006), we divide these nodes evenly into K communities, $K = 1, 2, 3$. The dimension of representation vectors for each node is set to 32, with values $0.02 \times n + 0.5 \times q$, $q = 1, 2, ..., K$. The first 18,000 events are used as the training set and the rest 2,000 events are used as the testing set. The evaluation metric is the proportion of successful prediction of the top r results: if the true sender or receiver of the next event falls in the top r candidates of the prediction, then this prediction is treated as a success.

We compare the above evaluation metric of the proposed network representation method with the following methods:

- HTNE (**Zuo** et al. 2018), its model formulation is similar to our proposed model, but it focuses on the neighborhood formation mechanism instead of transmission of influence.

its conditional intensity function is

$$\hat{\lambda}_{v|u}(t) = \mu_{u,v} + \sum_{t_h<t} \alpha_{h,v}\kappa(t - t_h) \tag{18}$$

- DeepWalk (**Perozzi** et al. 2014), it is a network representation learning method for static networks, which conducts random walks on the network to get contexts and use *word2vec* (**Mikolov** et al. 2013) to learn node representation vectors.

- node2vec(**Grover** et al. 2016), it is also a network representation learning method for static networks. It is similar to DeepWalk, but it better balances the depth-first search and breadth-first search in the path sampling step.

In the last two methods for static network representation learning, the prediction is conducted only on the distance between learned node representation vectors, also choose the top r candidates to evaluate the performance. Moreover, the true node representation vectors are also evaluated (called True method in the experiment) as the upper bound of a successful prediction rate in this experiment.

The experiment result of the next receiver prediction when the current node is the sender is shown in Table 2. From this result, we can find that the True method performs best as expected, and our proposed TPPN method outperforms the other three methods when $r = 2, 3, 5$, and performs close to the True method. Moreover, the difference between TPPN and True becomes larger as the number of communities increases. Static network representation learning methods perform better when the number of communities is larger, but still not as good as those methods based on temporal point process.

The experiment result of the next sender prediction when the current node is the receiver is shown in Table 3. It is shown that in the next-sender prediction task, neither the real node representation vectors nor these network representation learning methods perform as well as the task of the next-receiver task. This systematic difference comes from the form of the conditional intensity function, which incorporates the influence of the sender of an event on the receiver more than the other way round. Moreover, the proposed TPPN method performs better than the rest network representation learning methods and performs even better than the True method when $r = 5$. Therefore, the proposed TPPN gives a satisfactory performance in the next event prediction task.

Community Number K	Method	r=2	r=3	r=5
1	True	0.2220	0.3145	0.4945
	TPPN	0.2040	0.3035	0.4815
	HTNE	0.1455	0.2130	0.3850
	DeepWalk	0.0980	0.1620	0.3015
	node2vec	0.0580	0.0815	0.1535
2	True	0.2215	0.3285	0.5375
	TPPN	0.2015	0.2870	0.5020
	HTNE	0.1530	0.2225	0.3285
	DeepWalk	0.1360	0.2070	0.3395
	node2vec	0.1755	0.2655	0.4195
3	True	0.2855	0.4220	0.6555
	TPPN	0.2410	0.3590	0.5790
	HTNE	0.2465	0.3590	0.5745
	DeepWalk	0.2040	0.3065	0.5550
	node2vec	0.2210	0.3510	0.5790

Table 2: Experiment result of next receiver prediction when the current node is the sender.

Community Number K	Method	r=2	r=3	r=5
1	True	0.1970	0.3145	0.3320
	TPPN	0.1840	0.2685	0.4545
	HTNE	0.1810	0.2665	0.4475
	DeepWalk	0.0905	0.1480	0.2905
	node2vec	0.0570	0.0775	0.1495
2	True	0.2105	0.3285	0.4055
	TPPN	0.1970	0.3075	0.4660
	HTNE	0.1970	0.3060	0.4535
	DeepWalk	0.1290	0.2035	0.3525
	node2vec	0.1595	0.2415	0.4090
3	True	0.2640	0.4220	0.5065
	TPPN	0.2325	0.3510	0.5775
	HTNE	0.2220	0.3325	0.5540
	DeepWalk	0.2105	0.3335	0.5555
	node2vec	0.2120	0.3305	0.5665

Table 3: Experiment result of next sender prediction when the current node is the receiver.

5.1.2 Evaluation of Conditional Intensity Function

The key component of the proposed network event model is the conditional intensity function for each possible sender-receiver pair on the network. Ideally, the network representation learning algorithm should output node representation vectors that can well restore the true conditional intensity function that generates the events on the network. To evaluate the restoration performance of the proposed network representation learning method, we compare the restored conditional intensity function with the proposed method with the true conditional intensity on the network in this simulation experiment.

We simulate 20,000 events using Algorithm 2, with the end time, noted as T, and conduct the network representation learning algorithm on the first 18,000 events. We consider the case where the number of communities K is 1, 2, 3 respectively, and the true node representation vectors are set to be of $d = 32$ dimension, values are set to $0.02 \times n + 0.5 \times q$, where $n = 1, 2, ..., d,\ q = 1, 2, ..., K$. Figure 3, 4, and 5 show the visualization result of the first two principal components from true node representation vectors and learned node representation vectors on the network. We can see from these figures that the proposed network representation learning method can cluster the nodes from the same community together, thus is able to conduct node clustering tasks through network representation learning in real applications on networks.

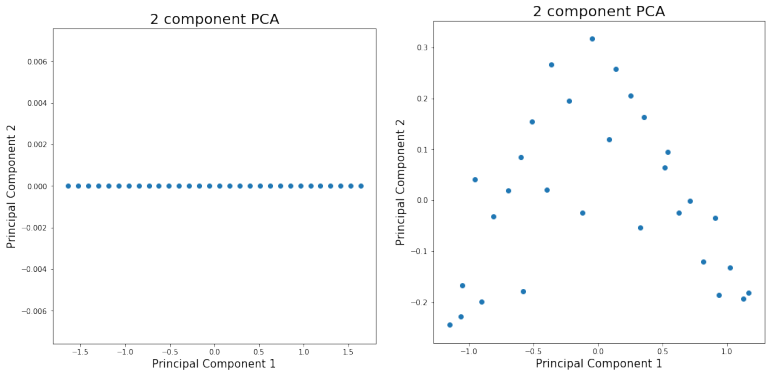


Figure 3: When $K = 1$, the principal components of the true node vectors and learned representation vectors.

Furthermore, the conditional intensity function in (2) reflects the occurring rate of the next interaction event from node u to node v, and can be compared with the true conditional intensity function to evaluate the performance of the learning algorithm. Figure 6 and Figure 7 show the learned matrix of conditional intensity functions, true matrix of conditional intensity function and the difference between these two for each node pair when the number of communities is

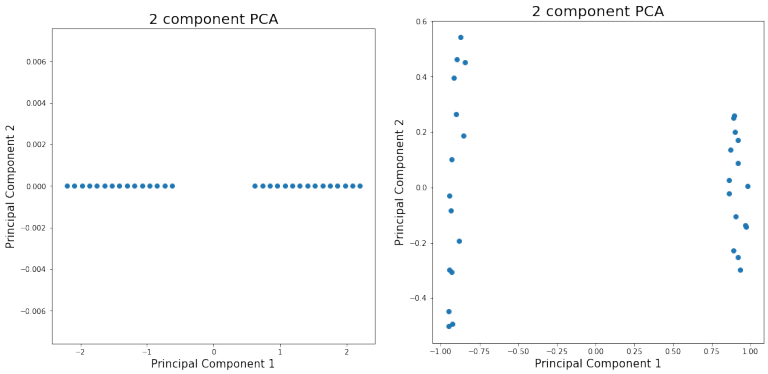


Figure 4: When $K = 2$, the principal components of the true node vectors and learned representation vectors.

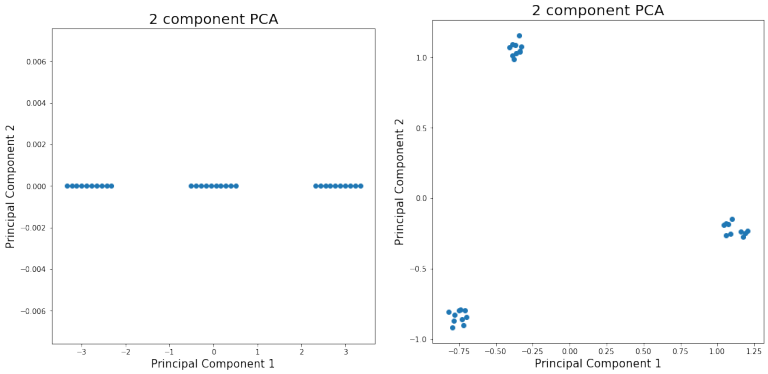


Figure 5: When $K = 3$, the principal components of the true node vectors and learned representation vectors.

$K = 1$ and $K = 2$ at time $t = 0.5T$. Generally, the restoration performance of the proposed method performs well for the conditional intensity function. When $K = 1$, the performance looks worse on the intensity between marginal nodes than that between the central nodes; when $K = 2$, the intensity between nodes inside the same community is restored better than that between nodes across communities.

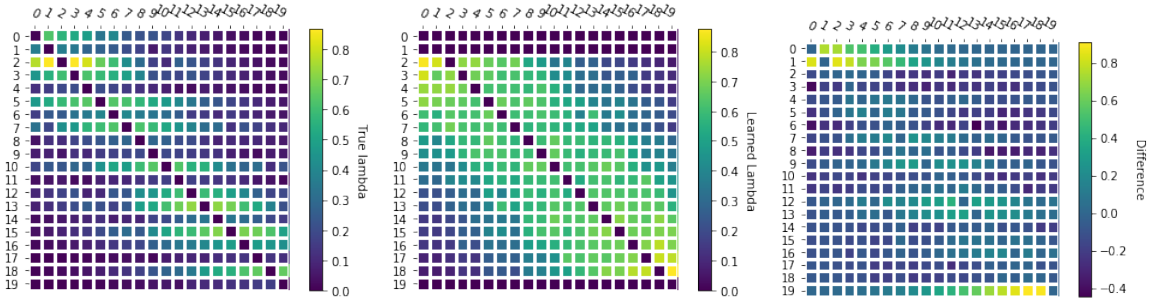


Figure 6: When $K = 1$, the comparison between the learned conditional intensity function and the true conditional intensity function.

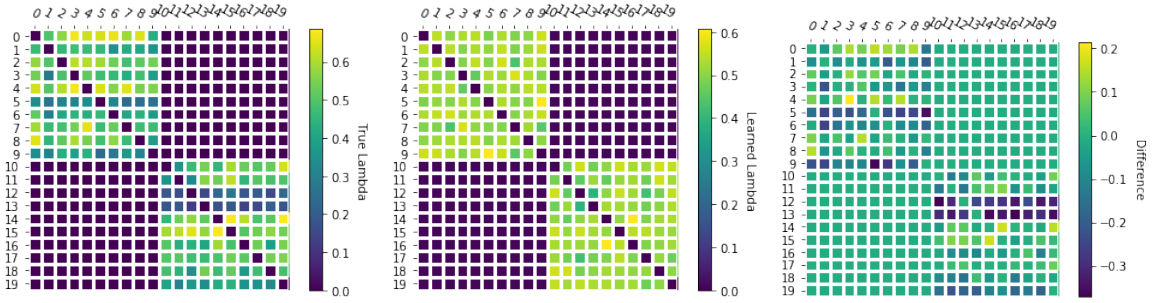


Figure 7: When $K = 2$, the comparison between the learned conditional intensity function and the true conditional intensity function.

For the case with $K = 1$, we manually add a shift over the network at time T, adding an extra value following $N(0.02, 0.02)$ normal distribution for each dimension of the last five nodes. According to the shifted node representation vectors, we further simulate events until the latest event occurs after $t = 2T$. Figure 8 and Figure 9 show when $t = 1.1T, t = 1.5T, t = 1.9T$, the matrix of conditional intensity function learned with the unchanged events and that calculated from the unchanged true node representation vectors. It is shown that when the network changes, the learned conditional intensity function will depart largely from that of the unchanged network, which motivates the idea of monitoring networks through these conditional intensity functions.

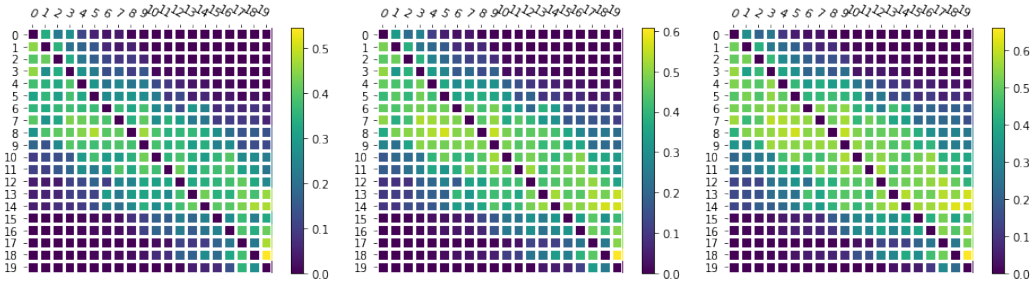


Figure 8: The learned conditional intensity function matrix on changed network, at $t = 1.1T, t = 1.5T, t = 1.9T$ respectively.

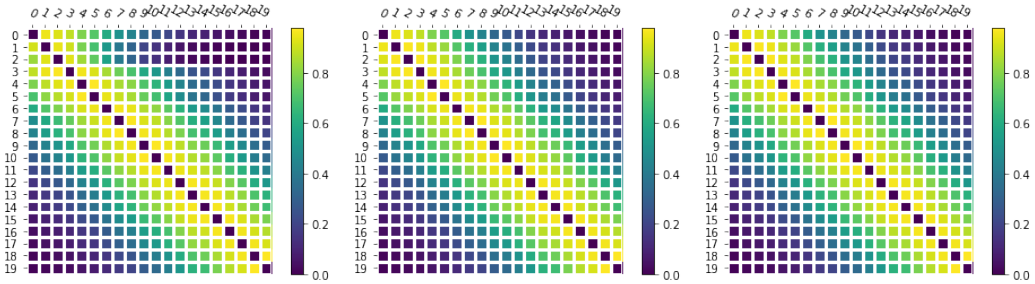


Figure 9: The true conditional intensity function matrix on changed network, at $t = 1.1T, t = 1.5T, t = 1.9T$ respectively.

5.1.3 Restoration Error of Conditional Intensity Function

To further evaluate the restoration performance of the conditional intensity function by the learned representation vectors of our proposed method, we also conduct simulation experiments on this task. In this experiment, we generate networks with $N = 30$ nodes, 20,000 events and number of communities $K = 1, 2, 3$ as described in Subsection 5.1.2. Among them, 90% of the events are used for learning the node representation vectors, and we evenly sample 200 time points and calculate the conditional intensity function for each sender-receiver node pair in the network.

To evaluate the restoration performance of the conditional intensity function, we calculate the root mean square error (RMSE) of $\hat{\lambda}$ for all the nodes in the network compared with the true λ calculated by the true network representation vectors:

$$\text{RMSE}(\hat{\lambda}(t), \lambda(t)) = \sqrt{\frac{\sum_{u,v=1,..,N,u\neq v} \left(\hat{\lambda}_{uv}(t) - \lambda_{uv}(t)\right)^2}{N^2 - N}} \tag{19}$$

We compare the performance of the proposed TPPN method with the HTNE method which

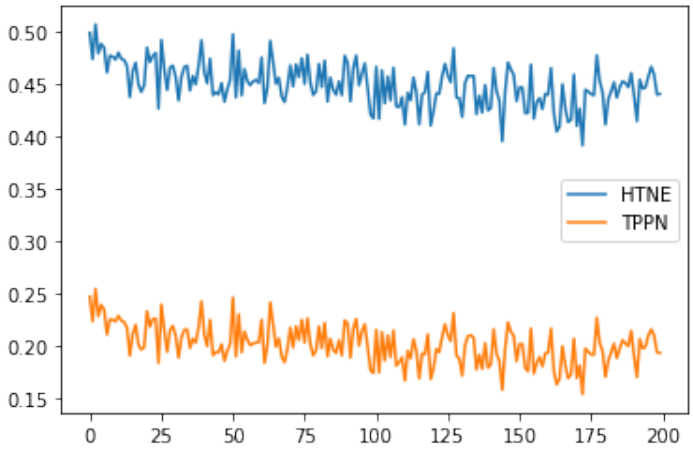
can also deliver the conditional intensity function as an intermediate result. The RMSE of CIF restoration at 200 time points for these two methods is shown in Figure 13. It is shown that as the time goes by, the RMSE of TPPN slowly decreases, but increases in the last 10% time points, because the events during this period are not included in the data for learning the node representation vectors. Moreover, the result of the HTNE method shows similar trends as the TPPN method, but the RMSE is much larger than that of the TPPN method. Therefore, it is useful to use the proposed TPPN method to learn node representation vectors on the network for downstream tasks when the true node representation vectors are unknown.

5.1.4 Network Monitoring

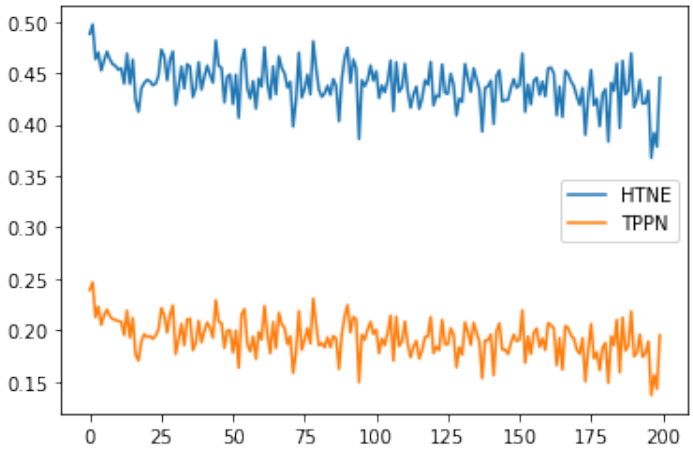
The proposed model provides a convenient indicator of the activity status over the network, which can be used to conduct the network monitoring task. As a demonstration, here we consider using the summation and the maximum value of the pairwise conditional intensity functions on the network, corresponding to the global rate of interaction events and the maximum local rate of interaction events: $\sum_{ij} \lambda_{ij}(t)$ and $\max_{ij} \lambda_{ij}(t)$, $i, j = 1, ..., N$.

In the experiment, we generate 10,000 interaction events on the network with 30 nodes, and the true node representation vectors follow that described in Subsection 5.1.2. Based on the 10,000 events, we add a shift of $N(0.02, 0.02)$ for the last 5 nodes in each representation dimension and generate 10,000 new events based on the shifted network. The proposed network representation learning algorithm is conducted with the first 10,000 events on the unshifted network, and the conditional intensity function matrix is calculated every 10 events for all the 20,000 events across the unshifted and the shifted network.

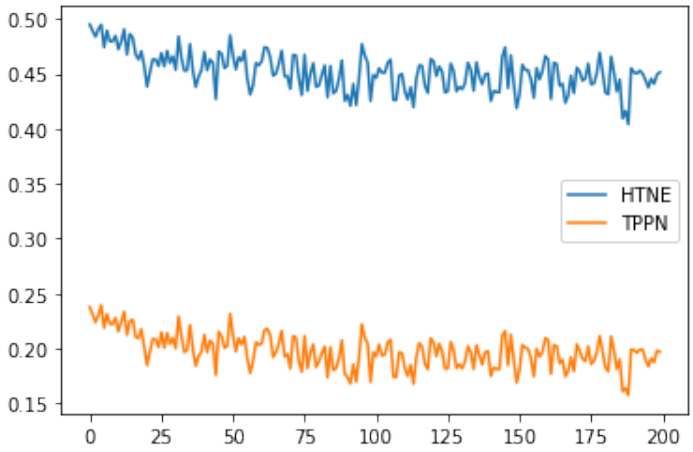
Figure 11 and Figure 12 show the result of the monitoring chart for monitoring the network as described. Before and after the shift on network occurs, there shows obvious change in $\sum_{ij} \lambda_{ij}$ and $\max_{ij} \lambda_{ij}$. Moreover, the summation of CIFs can better detect the change on several nodes, and the maximum value of CIFs can better reflect the rate of interaction events between sender-receiver pairs among a small number of nodes. Notably, using the learned CIF matrix instead of the true CIF matrix for monitoring shows a more obvious signal upon the shift of the network. In real application scenarios, the true node representation vectors are mostly unknown, and thus the learned node representation vectors are well suited for network monitoring in practice.



(a) When $K = 1$, the RMSE of λ restoration with TPPN and HTNE.

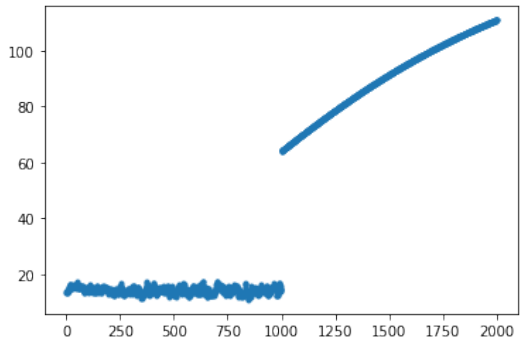


(b) When $K = 2$, the RMSE of λ restoration with TPPN and HTNE.

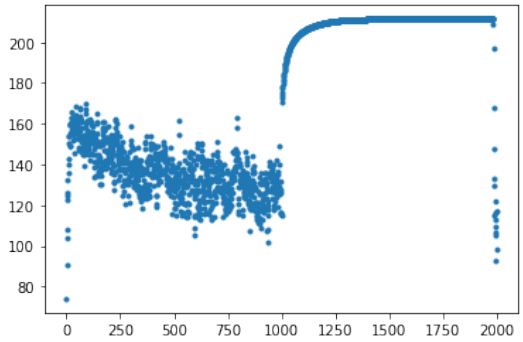


(c) When $K = 3$, the RMSE of λ restoration with TPPN and HTNE.

Figure 10: The RMSE of CIF Restoration with the proposed TPPN method and HTNE at 200 time points.

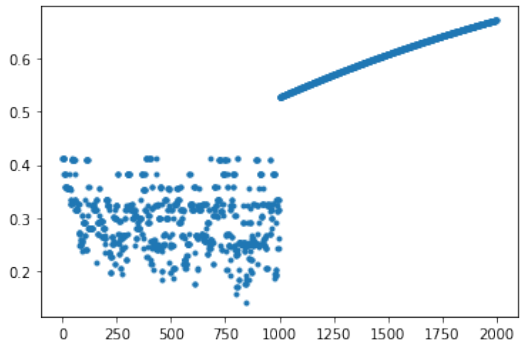


(a) Monitoring the network with learned $\sum \lambda_{ij}$.

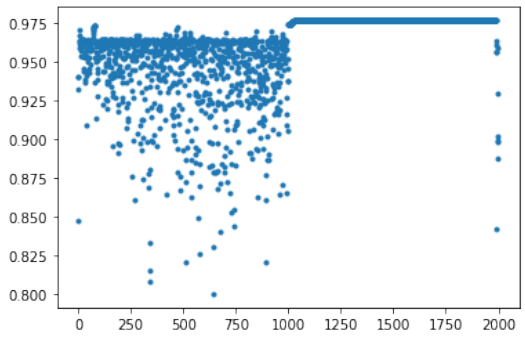


(b) Monitoring the network with true $\sum \lambda_{ij}$.

Figure 11: Monitoring the network with $\sum \lambda_{ij}$.



(a) Monitoring the network with learned $\max \lambda_{ij}$.



(b) Monitoring the network with $\max \lambda_{ij}$.

Figure 12: Monitoring the network with $\max \lambda_{ij}$.

5.2 Experiment on Real-world Data

The effectiveness of the proposed model and network representation learning method is validated on two real-world datasets: the DBLP co-author network and the Enron email network. Each node in the DBLP co-author network represents an author, and each event is a co-author publication. There are 28,085 nodes classified into 10 classes according to the author's research direction and 236,894 events. The Enron email dataset contains 184 email addresses, where each email address as a node, and 38,121 emails among these email addresses, where each email is an event. Two tasks are conducted in our experiment: node classification and link prediction. These two tasks are two classic tasks for the evaluation of network representation learning methods. We also show a case study of network monitoring on the Enron email network.

5.2.1 Node Classification

Node classification is a classic task on networks and has a variety of real applications. The purpose of this task is to identify the classes of unknown nodes according to the network structure and/or their attributes. The proposed TPPN method is compared with DeepWalk and HTNE on the DBLP co-author network dataset. The evaluation metrics are Macro-F1 and Micro-F1, which are both integration of precision and recallt[1]. Macro-F1 treats different classes equally without emphasis on sample size, and Micro-F1 balance the importance of different classes according to the sample sizes for each class. Usually, these two metrics are both evaluated for multi-class classification tasks.

We split the data into a training set and a testing set, and vary the proportion of the training data from 10% to 90% for this experiment. The experiment result of the three methods on the node classification task is shown in Table 4. According to the result, the proposed TPPN method achieves a better score than the other two methods in both Macro-F1 and Micro-F1. The HTNE method is better than the static network representation learning method DeepWalk for the incorporation of the formation process of neighbors on the network. Moreover, the Micro-F1 scores for these methods are generally higher than the Macro-F1 scores, which indicates that the network representation learning methods can well take care of the imbalance of different node classes.

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html`

Method	DeepWalk		HTNE		TPPN	
Metric	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
10%	0.6226	0.6309	0.6530	0.6182	**0.6576**	**0.6712**
20%	0.6434	0.6487	0.6371	0.6633	**0.6722**	**0.6806**
30%	0.6497	0.6519	0.6466	0.6683	**0.6773**	**0.6842**
40%	0.6519	0.6530	0.6503	0.6711	**0.6827**	**0.6877**
50%	0.6529	0.6549	0.6533	0.6719	**0.6827**	**0.6870**
60%	0.6515	0.6522	0.6581	0.6742	**0.6819**	**0.6859**
70%	0.6482	0.6501	0.6514	0.6708	**0.6779**	**0.6847**
80%	0.6491	0.6423	0.6491	0.6705	**0.6730**	**0.6835**
90%	0.6199	0.6401	0.6402	0.6671	**0.6788**	**0.6867**

Table 4: Node classification result on the DBLP dataset.

5.2.2 Link Prediction

Link prediction (**Lü** et al. 2011) is a task that identifies whether there is a link between two specific nodes on the network. We adopt the Enron email dataset to evaluate this task. The methods compared include DeepWalk, node2vec, LINE (**Tang** et al. 2015), HTNE, and the proposed TPPN method. For each of these methods, we use 70% of all the emails as positive samples, and randomly sample pairs of nodes without links in 1:1 ratio as negative samples. A support vector machine model is used to train a classifier for link identification, and the performance of the model is tested on the rest 30% links as well as the negative samples in a 1:1 ratio. We record the accuracy and Macro-F1 of the link prediction task for the compared methods. Table 5 shows the link prediction performance of the experiment. It is shown that the proposed TPPN method outperforms the other methods in both accuracy and Macro-F1.

Metric	DeepWalk	node2vec	LINE	HTNE	TTPN
Accuracy	0.7086	0.7944	0.8587	0.8227	**0.8753**
Macro-F1	0.6830	0.7884	0.8587	0.8227	**0.8751**

Table 5: Link prediction result on the Enron email dataset.

5.3 Monitoring the Enron Email Network

As is discussed in Section 4, the estimated CIFs for all the possible node pairs on the network can be used to establish a monitoring scheme for interaction event networks. In the following,

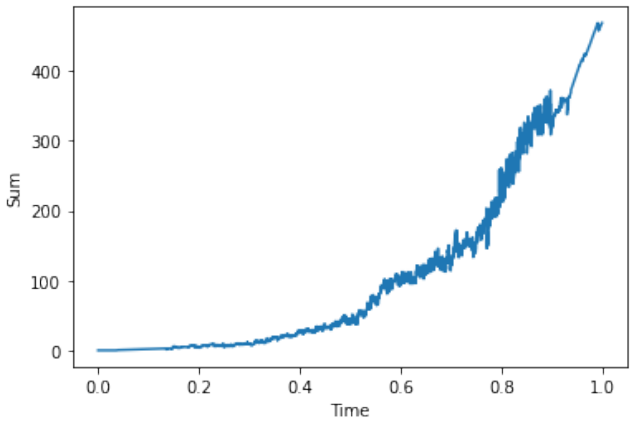
we conduct a case study on the Enron email network with this monitoring scheme.

The Enron email corpus contains all the email communications among the staff in Enron Corporation from 1998 to 2002. A scandal was publicized in October 2001, which eventually led to the bankruptcy of Enron Corporation. This issue was investigated by the Federal Regulatory Commission of the United States of America, who released this email dataset. The dynamics of the email network in Enron should show the emergence of illegal activities and the downward health status of the company.

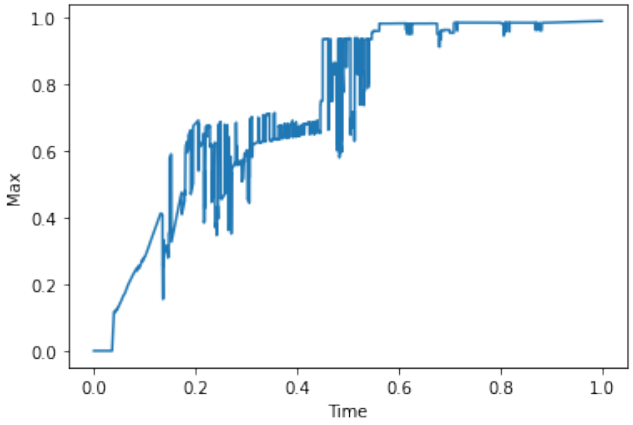
We apply the inference algorithm on this email dataset including 38,121 emails among 184 unique email addresses and obtain the pairwise CIFs for all the nodes. The email network was monitored with $\sum_{i \neq j} \lambda_{ij}$ and $\max_{i \neq j} \lambda_{ij}$, and these two monitoring metrics are calculated for each time stamp when a new email is sent. The monitoring results are shown in Figure 12. We normalize the period into the range of 0 to 1. It is shown in Figure 13a that with the fraud activities of the company, the emails are sent more and more frequently over the whole network. After the time when the scandal was publicized, the intensity of activities on the network sharply increases and reaches a peak after that. Moreover, the max intensity of interaction events over the network shown in Figure 13b rises quickly in the beginning, reaches the peak gradually, and remains stable after that. This experiment shows the different features for the two proposed monitoring metrics: the sum of CIFs emphasizes more on the global status over the whole network, and the maximum value of CIFs can well characterize the activities between the most intense pair of nodes.

6 Conclusion

In this paper, we propose a network model for interaction events based on the temporal point process, which explicitly models the influence of historical events on later events. The rate of interaction events for a sender-receiver interaction node pair is characterized by the conditional intensity function of a temporal point process, which is composed of external transitional influence and repeated pattern influence and the representation vectors for related nodes. We also propose a network representation learning algorithm to learn the node representation vectors on such networks. Based on the pairwise rate of interaction events on the network, network monitoring strategies are also introduced based on a summarization operation on these pairwise



(a) Monitoring the Enron network with learned $\sum \lambda_{ij}$.



(b) Monitoring the Enron network with learned $\max \lambda_{ij}$.

Figure 13: Monitoring the Enron Email Network with the proposed model.

rates of the network. Through simulation experiments and experiments on real-world data, we demonstrate the effectiveness of the proposed model and representation learning algorithm.

Network representation learning is a popular research topic nowadays. Researchers find different approaches to preserve the network structure and other useful information with node representation vectors on the network. Incorporating the dynamics of networks in network representation learning remains a challenging problem. This paper explores a possible approach to model the dynamics of networks by model the interaction event sequences. There are also several directions for future research. First, richer information on the network can be further utilized for better network representation learning, including node attributes, edge attributes, and higher-order dynamics on the network. Second, more efficient tools for network monitoring can be developed based on the rates of interaction events targeting specific types of changes. Furthermore, the idea of pre-training models and transfer learning can also be adopted for this field, such as initializing the node representation vectors with a pre-trained model.

References

1. **Lerner** J, **Bussmann** M, **Snijders** T, **Brandes** U. Modeling frequency and type of interaction in event networks. *Corvinus Journal of Sociology and Social Policy* 2013;**4**:3–32.

2. **Newman** M, **Watts** D, **Strogatz** S. Random graph models of social networks. *P Natl Acad Sci USA* 2002;**99**:2566–72.

3. **Hoff** PD, **Raftery** AE, **Handcock** MS. Latent Space Approaches to Social Network Analysis. *Journal of the American Statistical Association* 2002;**97**:1090–8.

4. **Daley** DJ, **Vere-Jones** D. An introduction to the theory of point processes. Vol. II. Second. Probability and its Applications (New York). General theory and structure. New York: Springer, 2008.

5. **Hawkes** AG. Spectra of Some Self-Exciting and Mutually Exciting Point Processes. *Biometrika* 1971;**58**:83–90.

6. **Li** L, **Zha** H. Learning Parametric Models for Social Infectivity in Multi-Dimensional Hawkes Processes. AAAI'14 2014:101–7.

7. **Xiao** S, **Yan** J, **Yang** X, **Zha** H, **Chu** SM. Modeling the Intensity Function of Point Process Via Recurrent Neural Networks. In: *AAAI*. 2017:1597–603.

8. **Yan** J, **Zhang** C, **Zha** H, et al. On Machine Learning Towards Predictive Sales Pipeline Analytics. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. Austin, Texas: AAAI Press, 2015:1945–51.

9. **Xu** H, **Wu** W, **Nemati** S, **Zha** H. Patient flow prediction via discriminative learning of mutually-correcting processes. *IEEE transactions on Knowledge and Data Engineering* 2017;**29**:157–71.

10. **Perry** PO, **Wolfe** PJ. Point process modelling for directed interaction networks. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 2013;**75**:821–49.

11. **Linderman** S, **Adams** R. Discovering latent network structure in point process data. In: *International Conference on Machine Learning*. 2014:1413–21.

12. **Hall** EC, **Willett** RM. Tracking dynamic point processes on networks. *IEEE Transactions on Information Theory* 2016;**62**:4327–46.

13. **Mei** H, **Eisner** JM. The neural hawkes process: A neurally self-modulating multivariate point process. In: *Advances in Neural Information Processing Systems*. 2017:6754–64.

14. **Junuthula** RR, **Haghdan** M, **Xu** KS, **Devabhaktuni** VK. The Block Point Process Model for Continuous-Time Event-Based Dynamic Networks. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'19. 2019.

15. **Zuo** Y, **Liu** G, **Lin** H, **Guo** J, **Hu** X, **Wu** J. Embedding Temporal Network via Neighborhood Formation. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD 18. London, United Kingdom, 2018:2857–66.

16. **Brand** M, **Huang** K. A unifying theorem for spectral embedding and clustering. In: *AISTATS*. 2003.

17. **Tenenbaum** JB, **Silva** V de, **Langford** JC. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science* 2000;**290**:2319.

18. **Roweis** ST, **Saul** LK. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science* 2000;**290**:2323–6.

19. **Mikolov** T, **Sutskever** I, **Chen** K, **Corrado** G, **Dean** J. Distributed Representations of Words and Phrases and Their Compositionality. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., 2013:3111–9.

20. **Perozzi** B, **Al-Rfou** R, **Skiena** S. Deepwalk: Online learning of social representations. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014:701–10.

21. **Tang** J, **Qu** M, **Wang** M, **Zhang** M, **Yan** J, **Mei** Q. LINE: Large-Scale Information Network Embedding. In: *Proceedings of the 24th International Conference on World Wide Web*. WWW '15. Florence, Italy: International World Wide Web Conferences Steering Committee, 2015:1067–77. DOI: 10.1145/2736277.2741093.

22. **Grover** A, **Leskovec** J. Node2vec: Scalable Feature Learning for Networks. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016:855–64. DOI: 10.1145/2939672.2939754.

23. **Qiu** J, **Dong** Y, **Ma** H, **Li** J, **Wang** K, **Tang** J. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and Node2vec. In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. WSDM '18. Marina Del Rey, CA, USA: Association for Computing Machinery, 2018:459–67. DOI: 10.1145/3159652.3159706.

24. **Liu** X, **Murata** T, **Kim** KS, **Kotarasu** C, **Zhuang** C. A General View for Network Embedding as Matrix Factorization. In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. WSDM '19. Melbourne VIC, Australia: Association for Computing Machinery, 2019:375–83. DOI: 10.1145/3289600.3291029.

25. **Bhagat** S, **Cormode** G, **Muthukrishnan** S. Node Classification in Social Networks. In: *Social Network Data Analytics*. Ed. by **Aggarwal** CC. Boston, MA: Springer US, 2011:115–48. DOI: 10.1007/978-1-4419-8462-3_5.

26. **Liben-Nowell** D, **Kleinberg** J. The Link-Prediction Problem for Social Networks. *J. Am. Soc. Inf. Sci. Technol.* 2007;**58**:1019–31.

27. **Newman** MEJ. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences* 2006;**103**:8577–82.

28. **Maaten** L van der, **Hinton** G. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 2008;**9**:2579–605.

29. **Zhang** D, **Yin** J, **Zhu** X, **Zhang** C. Network Representation Learning: A Survey. *IEEE Transactions on Big Data* 2017;**6**:3–28.

30. **Chang** S, **Han** W, **Tang** J, **Qi** GJ, **Aggarwal** CC, **Huang** TS. Heterogeneous Network Embedding via Deep Architectures. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '15. Sydney, NSW, Australia: Association for Computing Machinery, 2015:119–28. DOI: 10.1145/2783258.2783296.

31. **Li** H, **Wang** H, **Yang** Z, **Odagaki** M. Variation Autoencoder Based Network Representation Learning for Classification. In: *Proceedings of ACL 2017, Student Research Workshop*. Vancouver, Canada: Association for Computational Linguistics, 2017:56–61.

32. **Ou** M, **Cui** P, **Pei** J, **Zhang** Z, **Zhu** W. Asymmetric Transitivity Preserving Graph Embedding. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016:1105–14. DOI: 10.1145/2939672.2939751.

33. **Holme** P, **Saramäki** J. Temporal networks. *Physics Reports* 2012;**519**. Temporal Networks:97–125.

34. **McGregor** A. Graph Stream Algorithms: A Survey. *SIGMOD Rec.* 2014;**43**:9–20.

35. **Kumar** R, **Novak** J, **Tomkins** A. Structure and Evolution of Online Social Networks. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '06. Philadelphia, PA, USA: Association for Computing Machinery, 2006:611–7. DOI: 10.1145/1150402.1150476.

36. **Nguyen** GH, **Lee** JB, **Rossi** RA, **Ahmed** NK, **Koh** E, **Kim** S. Continuous-Time Dynamic Network Embeddings. In: *Companion Proceedings of the The Web Conference 2018*. WWW '18. Lyon, France: International World Wide Web Conferences Steering Committee, 2018:969–76. DOI: 10.1145/3184558.3191526.

37. **Antoch** J, **Hušková** M. Change Point Problem. In: *Computational Aspects of Model Choice*. Ed. by **Antoch** J. Heidelberg: Physica-Verlag HD, 1993:11–37.

38. **Savage** D, **Zhang** X, **Yu** X, **Chou** P, **Wang** Q. Anomaly detection in online social networks. *Social Networks* 2014;**39**:62–70.

39. **Woodall** WH, **Zhao** MJ, **Paynabar** K, **Sparks** R, **Wilson** JD. An overview and perspective on social network monitoring. *IISE Transactions* 2017;**49**:354–65.

40. **Jun** C, **Shun-zheng** Y. Network Monitoring Based on Community Structure Change. In: *2009 International Symposium on Intelligent Ubiquitous Computing and Education, IUCE*. IEEE, 2009:337–40.

41. **Wilson** JD, **Stevens** NT, **Woodall** WH. Modeling and detecting change in temporal networks via a dynamic degree corrected stochastic block model. *arXiv.org* 2016:arXiv:1605.04049.

42. **Dong** H, **Chen** N, **Wang** K. Modeling and Change Detection for Count-Weighted Multilayer Networks. *Technometrics* 2020;**62**:184–95.

43. **Priebe** CE, **Conroy** JM, **Marchette** DJ, **Park** Y. Scan Statistics on Enron Graphs. *Computational & Mathematical Organization Theory* 2005;**11**:229–47.

44. **Cheng** A, **Dickinson** P. Using scan-statistical correlations for network change analysis. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2013:1–13.

45. **Noorossana** R, **Hosseini** SS, **Heydarzade** A. An overview of dynamic anomaly detection in social networks via control charts. *Quality and Reliability Engineering International* 2018;**34**:641–8.

46. **Neil** J, **Hash** C, **Brugh** A, **Fisk** M, **Storlie** CB. Scan statistics for the online detection of locally anomalous subgraphs. *Technometrics* 2013.

47. **Azarnoush** B, **Paynabar** K, **Bekki** J. Monitoring Temporal Homogeneity in Attributed Network Streams. *Journal of Quality Technology* 2016;**48**:28.

48. **Kingma** DP, **Ba** J. Adam: A Method for Stochastic Optimization. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. 2014.

49. **Lü** L, **Zhou** T. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications* 2011;**390**:1150–70.