IISE Transactions

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/uiie21

Interaction event network modeling based on temporal point process

Hang Dong & Kaibo Wang

To cite this article: Hang Dong & Kaibo Wang (2022) Interaction event network modeling based on temporal point process, IISE Transactions, 54:7, 630-642, DOI: 10.1080/24725854.2021.1906468

To link to this article: https://doi.org/10.1080/24725854.2021.1906468

IISE TRANSACTIONS
2022, VOL. 54, NO. 7, 630–642
https://doi.org/10.1080/24725854.2021.1906468

Interaction event network modeling based on temporal point process

Hang Dong and Kaibo Wang

Department of Industrial Engineering, Tsinghua University, Beijing, China

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 article 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.

ARTICLE HISTORY
Received 6 August 2020
Accepted 6 February 2021

KEYWORDS
Temporal point process; network monitoring; network representation learning; interaction event; network model

1. Introduction

Network data widely exist nowadays, examples include 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, in previous studies these dynamic interactions are mostly aggregated as simple edges over nodes of individuals on networks (Newman *et al.*, 2002). 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, seven 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 closely 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 previously 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 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

CONTACT Kaibo Wang kbwang@tsinghua.edu.cn
Supplemental data for this article is available online at https://doi.org/10.1080/24725854.2021.1906468.

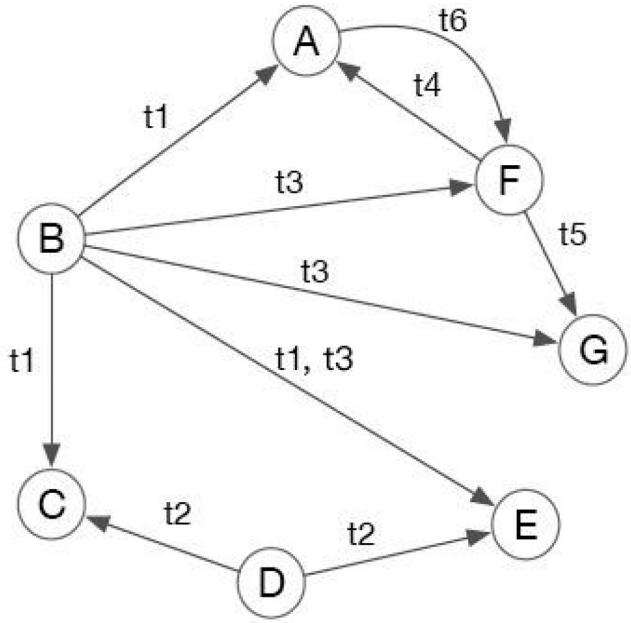


Figure 1. Example of an interaction event network.

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
……	……	……

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 this article 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

Numerous 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 and Vere-Jones, 2008). These events can be seismic events (Ogata and Vere-Jones, 1984), purchase events (Embrechts *et al.*, 2011), device failures (Xiao *et al.*, 2017), communication activities (Lerner *et al.*, 2013), and many others.

Based on the form of the CIF, 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, numerous variants of the Hawkes process have been proposed for different applications.

For example, a Hawkes process has been used in social network modeling (Li and Zha, 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 the specific times when each event happens, compared with 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.

Due to the prevalence of large online social networks, a number of works have adopted temporal point process for modeling event sequences on social networks (Farajtabar, 2018). For example, Perry and Wolfe (2013) proposed to use a temporal point process to characterize the repetitive

Table 2 Related works on temporal point processes.

Work	*Network*	*Application*	*Process*	*Solution*
Yan *et al.* (2015)	None	Business	Hawkes	Traditional
Xiao *et al.* (2017)	None	ATM Events	Time Series RNN	RNN
Perry and Wolfe (2013)	Static	Email Network	Cox	Traditional
Xu *et al.* (2017)	None	Patient Flow	Mutually-correcting	Traditional
Lerner *et al.* (2013)	Static	Political Network	ad-hoc	Traditional
Li and Zha (2014)	Static	Social Network	Hawkes	Traditional
Farajtabar *et al.* (2015)	Static	Social Network	ad-hoc	Traditional
Linderman and Adams (2014)	Static	Financial & Criminal	Hawkes	Traditional
Junuthula *et al.* (2019)	Static	Social Network	Hawkes	Traditional
Zuo *et al.* (2018)	Static	Social Network	ad-hoc	NN
Hall and Willett (2016)	Dynamic	Blog Network	Hawkes	Traditional
Mei and Eisner (2017)	Dynamic	Blog & Social	ad-hoc	RNN
Trivedi *et al.* (2019)	Dynamic	Social Network	ad-hoc	NN

directed interaction events on networks; Linderman and Adams (2014) developed a probabilistic model combining mutually-exciting point processes with random graph models; Farajtabar *et al.* (2015) modeled the information diffusion process as a multivariate Hawkes process and a network evolution process as a combination of survival and Hawkes processes; Hall and Willett (2016) proposed an online learning framework of a multivariate Hawkes process model to track the network structure of a social network as it evolves; Mei and Eisner (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 long short-term memory neural network; Junuthula *et al.* (2019) combined the stochastic block model with a Hawkes process as a block point process to incorporate the community structure in a social network. However, these works did not adopt representation learning methods to incorporate network structure information. Zuo *et al.* (2018) integrated the neighborhood formulation process as a Hawkes process into network embedding, so as to capture the influence of historical neighbors on the current neighbors; Trivedi *et al.* (2019) assumed that a latent mediation process bridges the topological evolution and activities, and proposed a two-time-scale deep temporal point process model to capture this characteristic. Although these works tried to adopt temporal point processes with representation learning on networks, they did not focus on modeling and monitoring the triggering mechanism of interaction events over the network. In our work, we explicitly model this mechanism and mainly decompose it into the transitive influence and repeated pattern influence from historical events.

The works mentioned above are listed in Table 2 according to the following characteristics: the network is static (assume no unknown added nodes) or dynamic, the application area, the underlying point process, and whether they used a neural network to solve the model (we classify them as traditional if not). In this article, we focus on a similar setting to that of Zuo *et al.* (2018), but use a different temporal point process to characterize the transitive and repeated pattern influence from historical interaction events to later ones.

2.2. Network representation learning

Network representation learning has become popular in recent decades, due to its powerful capability to efficiently process large networks (Hamilton *et al.*, 2017). 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, appropriate 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 and Huang, 2003) and manifold learning (Roweis and Saul, 2000; Tenenbaum *et al.*, 2000), conducting eigendecomposition 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 models in natural language processing, many methods such as DeepWalk (Perozzi *et al.*, 2014), LINE (Tang *et al.*, 2015) and node2vec (Grover and Leskovec, 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, which are based on a static network structure, can be treated as a matrix decomposition task, which is equivalent 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 and Kleinberg, 2007), clustering (Newman, 2006), visualization (Van Der Maaten and Hinton, 2008), and so on.

There are still some significant 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 have focused 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 and Saramki, 2012), and graph streams (McGregor, 2014) have been proposed. Usually, dynamic network models create a series of static network slices based on a fixed period (Kumar *et al.*, 2006). For example, Du *et al.* (2018) and Nguyen *et al.* (2018) learn node embedding for dynamic networks, where time is only used for identifying the order of edges, neglecting the specific time difference between two events. For more related works on dynamic network representation learning, readers can refer to Xie *et al.* (2020). 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 and Hušková, 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, and whether there are node labels in the network. Jeske *et al.* (2018) illustrated statistical tools for network surveillance applications in the context of network security, network reliability, and social networks. 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 and Shun-zheng, 2009), including monitoring based on variants of the stochastic block model (Wilson *et al.*, 2019; 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 and Dickinson (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 (Sparks and Wilson, 2019). Neil *et al.* (2013) used a scan statistic for both time windows and sub-graphs to detect anomalous sub-graphs 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.

For monitoring dynamic networks, there have been several works (Noorossana *et al.*, 2018). For example, Bian *et al.* (2019) proposed to directly model changes between two consecutive static networks by capturing the topological difference of the network. However, current works are mostly conducted on a number of static network snapshots and neglect the time interval information between events.

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 contributions of this work are 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 1 shows an example of an interaction event network. In Figure 1, 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 accounts 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 1. This is the main motivation of the proposed model. From the perspective of formulating the event rate, it is straightforward to consider the effective influence of historical events 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

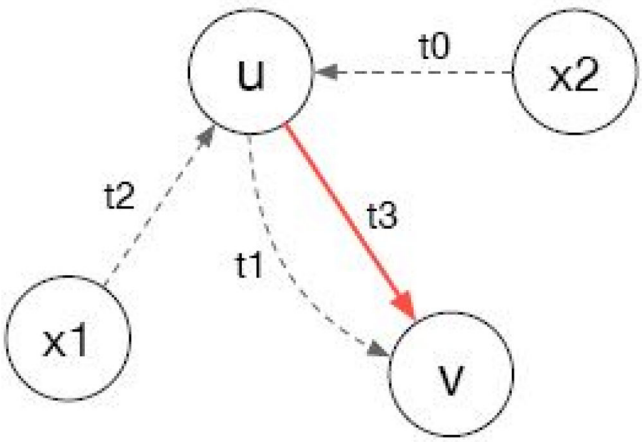


Figure 2. Illustration of the CIF of u-to-v event at time t_3.

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 expressing 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 that happen until time t. For each process $N_c(t)$, we note the immediate occurrence rate as $\lambda_c(t)$, which is the CIF:

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

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

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, CIF of the temporal point process from one node u to another node v on the network is:

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

where μ_{uv} represents the base rate term, I is the indicator function and only when the content meets the condition does this term become equal to one and otherwise zero; $\mathbf{h}$. 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_k. 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.

By decomposing the CIF into external transitional influence and repeated pattern influence (corresponding to the first term and second term described above), this CIF 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 term 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 term:

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

where $\mathbf{h}_u$ and $\mathbf{h}_v$ represent the representation vector of node u and node v in the latent space, respectively. This base rate term will be transformed into base rate through function g. 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 a 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 CIF for any sender–receiver node pair at any time $0 \le t \le T$ 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 $\{\mathbf{h}_i, i = 1, \ldots, N\}$ and $\{\delta_i, i = 1, \ldots, N\}$. In

this section, we introduce a network representation learning method to infer 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 CIF of all the observations of interaction events on the network up to time T, and the log likelihood function of these events is:

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

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 huge number of calculations created 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:

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

where $\tilde{\lambda} = g^{-1}(\lambda)$ is the CIF before the final transformation; 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 CIF $\lambda_{uv}(t)$, and the impact from far early historical events is so trivial that it 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.

Classic optimization strategies can be easily adapted to optimize the objective function in (10), such as Stochastic Gradient Descent (SGD) and Adam (Kingma and Ba, 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 $\boldsymbol{\delta}=[\delta_1,\ldots,\delta_N]$;
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 CIF between each sender–receiver node pair on the network:

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

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:

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

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

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, whereas 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 based on our proposed model form and make inferences on model parameters based on our proposed network representation learning algorithm. We aim to serve the following two purposes in this experiment: First, we check the effectiveness of the estimated CIF 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.

The hardware environment for the 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 eight 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% of data, and test the performance of the model with the remaining 10%.

Table 3. 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	0.2220	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

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 with centroids $\mathbf{c}_k \in \mathbb{R}^d, k = 1, .., K$; Deviation inside community ϵ;

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 k, starting with the community centroid, set the node representation to be a vector deviated from the last node by ϵ;
2: **for** $i, j = 1, ..., N, i \neq j$ **do**
3: According to the thinning algorithm (Ogata, 1981), 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 CIF 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:

Table 4. Experimental result of next sender prediction when the current node is the receiver.

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

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

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 remaining 2000 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:

1. 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 CIF is
$$\hat{\lambda}_{v|u}(t) = \mu_{u,v} + \sum_{t_h < t} \alpha_{h,v} \kappa(t - t_h). \tag{18}$$
2. 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 uses word2vec (Mikolov *et al.*, 2013) to learn node representation vectors.
3. node2vec(Grover and Leskovec, 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.

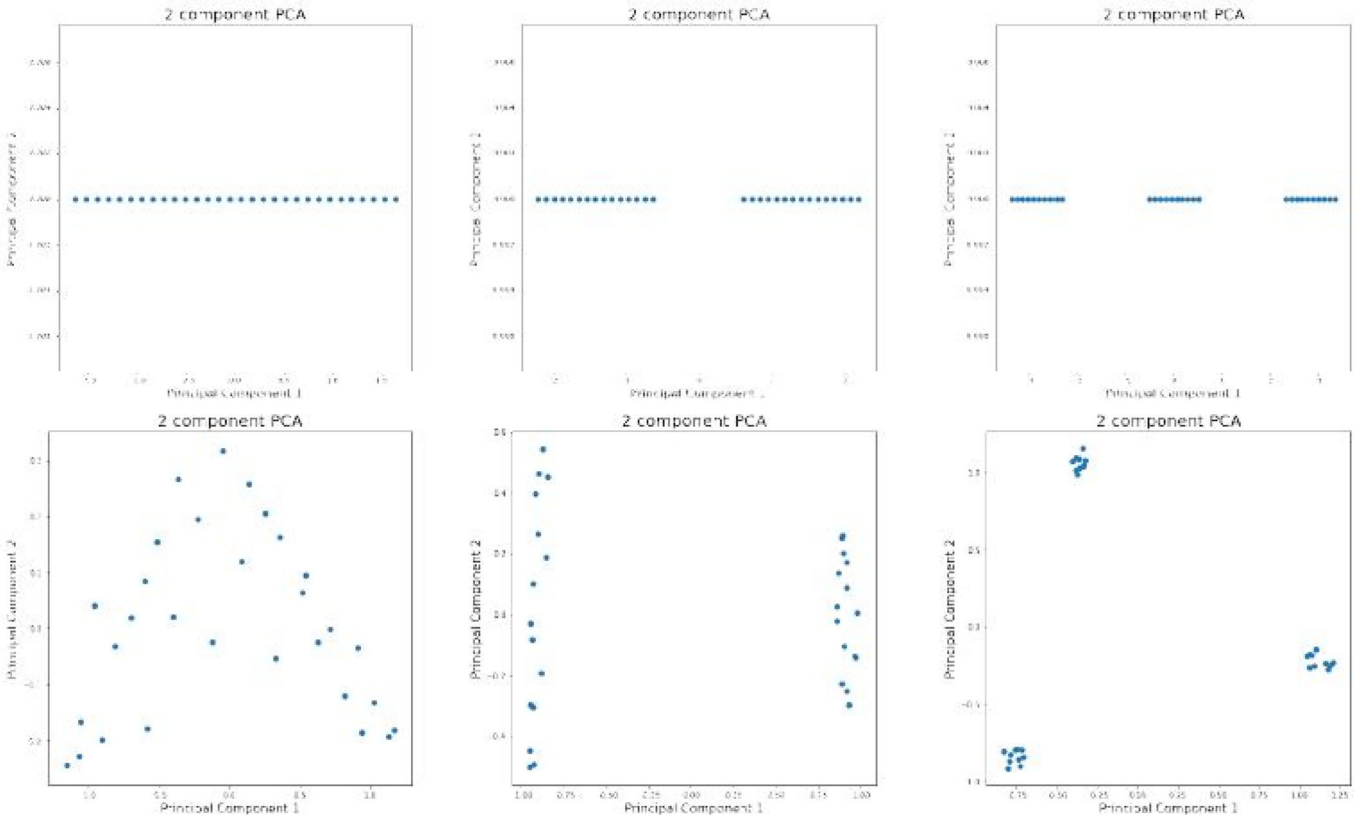


Figure 3. When $K = 1$, 2, 3 respectively, the principal components of the true node vectors and learned representation vectors.

The experiment result of the next receiver prediction when the current node is the sender is shown in Table 3. 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 it is still not as good as those methods based on temporal point process.

The experimental result of the next sender prediction when the current node is the receiver is shown in Table 4. 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 CIF, 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 remaining 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.

5.1.2. Evaluation of CIF

The key component of the proposed network event model is the CIF 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 CIF that generates the events on the network. To evaluate the restoration performance of the proposed network representation learning method, we compare the restored CIF 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 $d = 32$ dimensions, values are set to $0.02 \times n + 0.5 \times q$, where $n = 1, 2, \ldots, d, q = 1, 2, \ldots, K$. Figure 3 shows 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.

Furthermore, the CIF in (2) reflects the occurence rate of the next interaction event from node u to node v, and can be compared with the true CIF to evaluate the performance of the learning algorithm. Figure 4 and Figure 5 show the learned matrix of CIFs, true matrix of the CIF and the difference between these two for each node pair when the number of communities is $K = 1$ and $K = 2$ at time $t = 0.5T$. Generally, the restoration performance of the proposed method performs well for the CIF. When $K = 1$, the

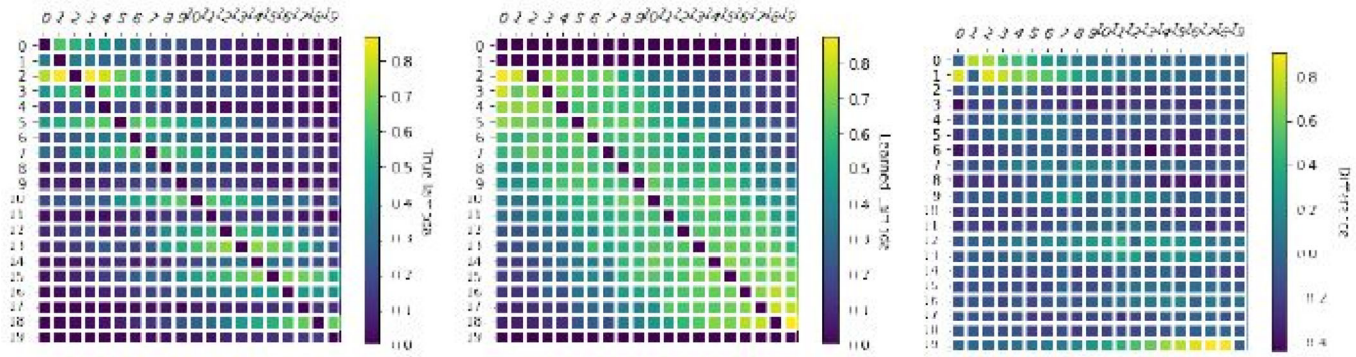


Figure 4. When $K = 1$, the comparison between the learned conditional intensity function and the true CIF.

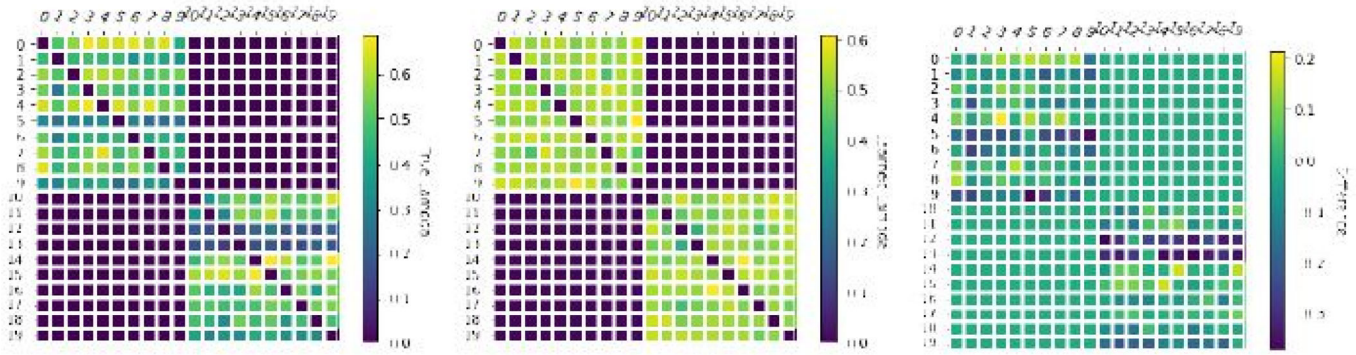


Figure 5. When $K = 2$, the comparison between the learned CIF and the true CIF.

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.

For the case with $K = 1$, we manually add a shift over the network at time T, adding an extra value following a $N(0.02, 0.02)$ 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 6 and Figure 7 show when $t = 1.1T, t = 1.5T, t = 1.9T$, the matrix of the CIF 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 CIF will significantly deviate from that of the unchanged network, which motivates the idea of monitoring networks through these CIFs.

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 acts 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 based on 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 recall[1]. Macro-F1 treats different classes equally without emphasis on sample size, and Micro-F1 balances 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%, and set the dimension of learned representation vectors as $d = 128$ for this experiment, as do as most network embedding works. The experimental results for the three methods on the node classification task are shown in Table 5. 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

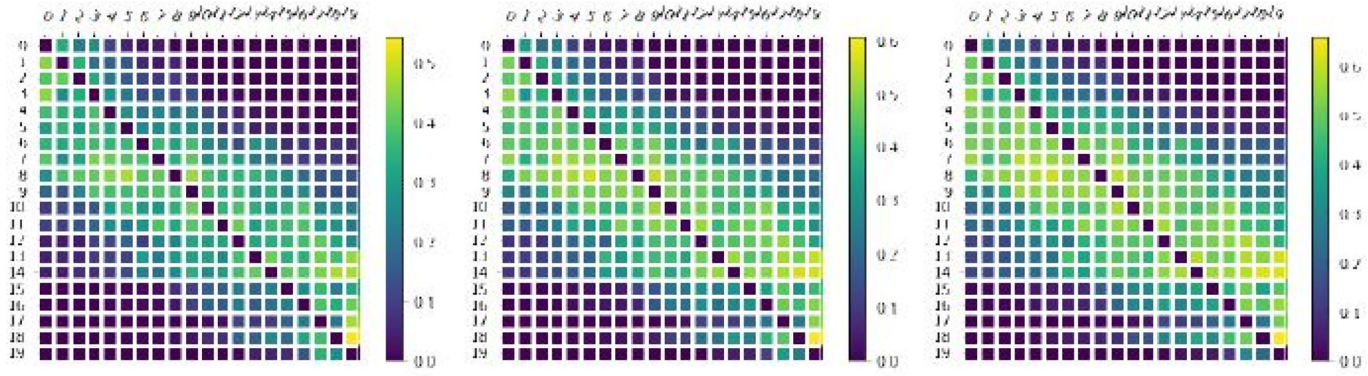


Figure 6. The learned CIF matrix on changed network, at $t = 1.1T, t = 1.5T, t = 1.9T$, respectively.

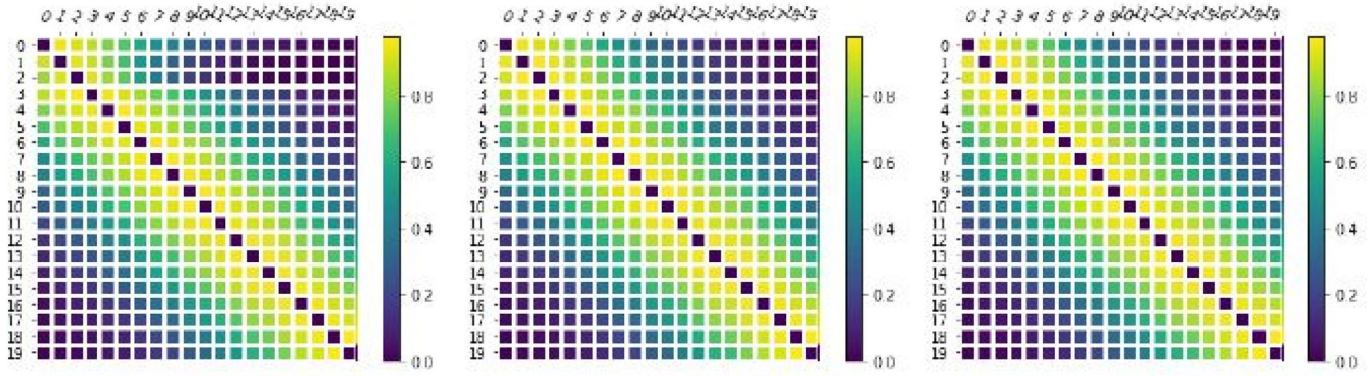


Figure 7. The true CIF matrix on changed network, at $t = 1.1T, t = 1.5T, t = 1.9T$, respectively.

Table 5. Node classification result on the DBLP dataset.

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**

learning methods can well take care of the imbalance of different node classes.

5.2.2. Link prediction

Link prediction (Lü and Zhou, 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 a 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 remaining 30% of links as well as the negative samples in a 1:1 ratio. The learned dimension is also set as $d = 128$. We record the accuracy and Macro-F1 of the link prediction task for the compared methods. Table 6 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.

Table 6. Link prediction result on the Enron email dataset.

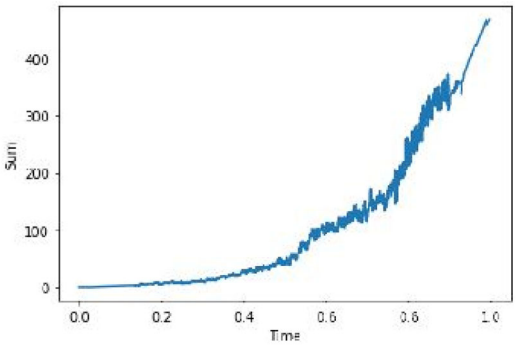
Metric	*DeepWalk*	*node2vec*	*LINE*	*HTNE*	*TTPN*
Accuracy	0.7086	0.7944	0.8587	0.8227	**0.8753**
Macro-F1	0.6830	0.7884	0.8587	0.8227	**0.8751**

5.3. Monitoring the Enron email network

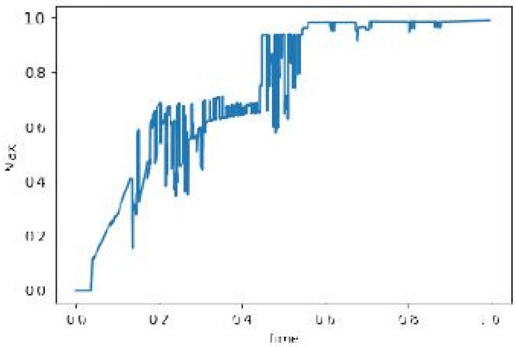
As is discussed in Section 4, the estimated CIFs for all the possible node pairs on the network can be used to establish a monitoring scheme for interaction event networks. In the following, we conduct a case study on the Enron email network with this monitoring scheme.

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 to this email dataset including 38,121 emails among 184 unique email addresses



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



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

Figure 8. Monitoring the Enron email network with the proposed model.

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 8. We normalize the period into the range of zero to one. It is shown in Figure 8(a) 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. Moreover, the max intensity of interaction events over the network shown in Figure 8(b) rises quickly in the beginning, reaches a peak gradually, and subsequently remains stable. 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 article, 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 CIF 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 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 article explores a possible approach to model the dynamics of networks by modeling 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.

Acknowledgments

The authors greatly thank the editor and three anonymous referees for their helpful comments and suggestions, which have helped us improve this work greatly.

Funding

This work was supported by the Key Program of the National Natural Science Foundation of China (Grant No. 71932006 and 71731008).

Note

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

Notes on contributorss

Hang Dong received her PhD degree in Management Science and Engineering from Department of Industrial Engineering, Tsinghua University in 2020. After that, she joined Microsoft Research Aisa as a researcher. Her research interests include statistical network modeling, prediction based scheduling and reinforcement learning for service intelligence.

Kaibo Wang is a professor in Department of Industrial Engineering, Tsinghua University. He received his BS and MS degrees in Mechatronics from Xi'an Jiaotong University, Xi'an, China, and his PhD in Industrial Engineering and Engineering Management from Hong Kong University of Science and Technology, Hong Kong. His research focuses on statistical quality control and data-driven system modeling, monitoring, diagnosis, and control, with a special emphasis on the integration of engineering knowledge and statistical theories for solving problems from the real industry.

ORCID

Kaibo Wang http://orcid.org/0000-0001-9888-4323

References

Antoch, J. and Hušková, M. (1993) Change point problem, in *Computational Aspects of Model Choice*, Physica-Verlag, Heidelberg, Germany pp. 11–37.

Azarnoush, B., Paynabar, K. and Bekki, J. (2016) Monitoring temporal homogeneity in attributed network streams. *Journal of Quality Technology*, **48**(1), 28–43.

Bhagat, S., Cormode, G. and Muthukrishnan, S. (2011) Node classification in social networks, in *Social Network Data Analytics*, Springer, Boston, MA, pp.115–148.

Bian, R., Koh, Y.S., Dobbie, G. and Divoli, A. (2019) Network embedding and change modeling in dynamic heterogeneous networks, in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Association for Computing Machinery, New York, NY, pp. 861–864.

Brand, M. and Huang, K. (2003) A unifying theorem for spectral embedding and clustering, in *Proceedings of the 9th International Conference on Artificial Intelligence and Statistics*, Society for Artificial Intelligence and Statistics, Key West, FL.

Chang, S., Han, W., Tang, J., Qi, G.-J., Aggarwal, C.C. and Huang, T.S. (2015) Heterogeneous network embedding via deep architectures, in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery Sydney, NSW, pp. 119–128.

Cheng, A. and Dickinson, P. (2013) Using scan-statistical correlations for network change analysis, in *Trends and Applications in Knowledge Discovery and Data Mining*, Berlin, Heidelberg, Germany, pp. 1–13.

Daley, D.J. and Vere-Jones, D. (2008) *An Introduction to the Theory of Point Processes*. Volume *II: General Theory and Structure*, Springer, New York, NY.

Dong, H., Chen, N. and Wang, K. (2020) Modeling and change detection for count-weighted multilayer networks. *Technometrics*, **62**(2), 184–195.

Du, L., Wang, Y., Song, G., Lu, Z. and Wang, J. (2018) Dynamic network embedding: An extended approach for skip-gram based network embedding, in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, AAAI Press, Stockholm, pp. 2086–2092.

Embrechts, P., Liniger, T. and Lin, L. (2011) Multivariate Hawkes processes: An application to financial data. *Journal of Applied Probability*, **48**(A), 367–378.

Farajtabar, M. (2018) Point process modeling and optimization of social networks. Ph.D. thesis, Georgia Institute of Technology, Atlanta, GA.

Farajtabar, M., Wang, Y., Gomez-Rodriguez, M., Li, S., Zha, H. and Song, L. (2015) COEVOLVE: A joint point process model for information diffusion and network co-evolution, in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, MIT Press, Montreal, Quebec, pp. 1954–1962.

Grover, A. and Leskovec, J. (2016) Node2vec: Scalable feature learning for networks, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, San Francisco, CA, pp. 855–864.

Hall, E.C. and Willett, R.M. (2016) Tracking dynamic point processes on networks. *IEEE Transactions on Information Theory*, **62**(7), 4327–4346.

Hamilton, W.L., Ying, R. and Leskovec, J. (2017) Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, **40**(3), 52–74.

Hawkes, A.G. (1971) Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, **58**(1), 83–90.

Hoff, P.D., Raftery, A.E. and Handcock, M.S. (2002) Latent space approaches to social network analysis. *Journal of the American Statistical Association*, **97**(460), 1090–1098.

Holme, P. and Saramki, J. (2012) Temporal networks. *Physics Reports*, **519**(3), 97–125.

Jeske, D.R., Stevens, N.T., Tartakovsky, A.G. and Wilson, J.D. (2018) Statistical methods for network surveillance. *Applied Stochastic Models in Business and Industry*, **34**(4), 425–445.

Jun, C. and Shun-zheng, Y. (2009) Network monitoring based on community structure change, in *2009 International Symposium on Intelligent Ubiquitous Computing and Education*, IEEE Press, Piscataway, NJ, pp. 337–340.

Junuthula, R.R., Haghdan, M., Xu, K.S. and Devabhaktuni, V.K. (2019) The block point process model for continuous-time event-based dynamic networks, in *The World Wide Web Conference (WWW '19)*. Association for Computing Machinery, New York, NY, pp. 829–839.

Kingma, D.P. and Ba, J. (2014) Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kumar, R., Novak, J. and Tomkins, A. (2006) Structure and evolution of online social networks, in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 611–617.

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

Li, H., Wang, H., Yang, Z. and Odagaki, M. (2017) Variation autoencoder based network representation learning for classification, in *Proceedings of ACL 2017, Student Research Workshop*, Association for Computational Linguistics, Vancouver, pp. 56–61.

Li, L. and Zha, H. (2014) Learning parametric models for social infectivity in multi-dimensional Hawkes processes, in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI Press, Quèbec City, Quèbec, Canada pp. 101–107.

Liben-Nowell, D. and Kleinberg, J. (2007) The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, **58**(7), 1019–1031.

Linderman, S. and Adams, R. (2014) Discovering latent network structure in point process data, in *Proceedings of the 31st International Conference on Machine Learning*, JMLR.org, Beijing, pp. 1413–1421.

Liu, X., Murata, T., Kim, K.-S., Kotarasu, C. and Zhuang, C. (2019) A general view for network embedding as matrix factorization, in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, Association for Computing Machinery, Melbourne, VIC, pp. 375–383.

Lü, L. and Zhou, T. (2011) Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, **390**(6), 1150–1170.

McGregor, A. (2014) Graph stream algorithms: A survey. *ACM SIGMOD Record*, **43**(1), 9–20.

Mei, H. and Eisner, J.M. (2017) The neural Hawkes process: A neurally self-modulating multivariate point process, in *Proceedings of the 31th International Conference on Neural Information Processing Systems*, Curran Associates Inc., Long Beach, CA, pp. 6754–6764.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J. (2013) Distributed representations of words and phrases and their compositionality, in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, Curran Associates Inc., Lake Tahoe, NV, pp. 3111–3119.

Neil, J., Hash, C., Brugh, A., Fisk, M. and Storlie, C.B. (2013) Scan statistics for the online detection of locally anomalous subgraphs. *Technometrics*, **55**(4), 403–414.

Newman, M.E.J. (2006) Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, **103**(23), 8577–8582.

Newman, M.E.J., Watts, D.J. and Strogatz, S.H. (2002) Random graph models of social networks. *Proceedings of the National Academy of Sciences*, **99**(suppl 1), 2566–2572.

Nguyen, G.H., Lee, J.B., Rossi, R.A., Ahmed, N.K., Koh, E. and Kim, S. (2018) Continuous-time dynamic network embeddings, in *Companion Proceedings of the The Web Conference 2018*, Association for Computing Machinery, Lyon, pp. 969–976.

Noorossana, R., Hosseini, S.S. and Heydarzade, A. (2018) An overview of dynamic anomaly detection in social networks via control charts. *Quality and Reliability Engineering International*, **34**(4), 641–648.

Ogata, Y. (1981) IOn Lewis' simulation method for point processes. *IEEE Transactions on Information Theory*, **27**(1), 23–31.

Ogata, Y. and Vere-Jones, D. (1984). Inference for earthquake models: A self-correcting model. *Stochastic Processes and their Applications*, **17**(2), 337–347.

Ou, M., Cui, P., Pei, J., Zhang, Z. and Zhu, W. (2016) Asymmetric transitivity preserving graph embedding, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, San Francisco, CA, pp. 1105–1114.

Perozzi, B., Al-Rfou, R. and Skiena S. (2014) Deepwalk: Online learning of social representations, in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, New York, NY, pp. 701–710.

Perry, P.O. and Wolfe, P.J. (2013) Point process modelling for directed interaction networks. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **75**(5), 821–849.

Priebe, C.E., Conroy, J.M., Marchette, D.J. and Park, Y. (2005). Scan statistics on Enron Graphs. *Computational & Mathematical Organization Theory*, **11**(3), 229–247.

Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K. and Tang, J. (2018) 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*, Association for Computing Machinery, Marina Del Rey, CA, pp. 459–467.

Roweis, S.T. and Saul, L.K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, **290**(5500), 2323–2326.

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

Sparks, R. and Wilson, J.D. (2019) Monitoring communication outbreaks among an unknown team of actors in dynamic networks. *Journal of Quality Technology*, **51**(4), 353–374.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J. and Mei, Q. (2015) LINE: Large-scale information network embedding in *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077.

Tenenbaum, J.B., de Silva, V. and Langford J.C. (2000) A global geometric framework for nonlinear dimensionality reduction. *Science*, **290**(5500), 2319.

Trivedi, R., Farajtabar, M., Biswal, P. and Zha, H. (2019) DyRep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*.

Van der Maaten, L. and Hinton, G. (2008) Visualizing data using t-SNE. *Journal of Machine Learning Research*, **9**(96), 2579–2605.

Wilson, J.D., Stevens, N.T. and Woodall, W.H. (2019) Modeling and detecting change in temporal networks via the degree corrected stochastic block model. *Quality and Reliability Engineering International*, **35**(5), 1363–1378.

Woodall, W.H., Zhao, M.J., Paynabar, K., Sparks, R. and Wilson, J.D. (2017) An overview and perspective on social network monitoring. *IISE Transactions*, **49**(3), 354–365.

Xiao, S., Yan, J., Yang, X., Zha, H. and Chu, S.M. (2017) Modeling the intensity function of point process via recurrent neural networks, in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 1597–1603.

Xie, Y., Li, C., Yu, B., Zhang, C. and Tang, Z. (2020) A survey on dynamic network embedding. *arXiv preprint arXiv:2006.08093*.

Xu, H., Wu, W., Nemati, S. and Zha, H. (2017) Patient flow prediction via discriminative learning of mutually-correcting processes. *IEEE Transactions on Knowledge and Data Engineering*, **29**(1), 157–171.

Yan, J., Zhang, C., Zha, H., Gong, M., Sun, C., Huang, J., Chu, S. and Yang, X. (2015) On machine learning towards predictive sales pipeline analytics, in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 1945–1951.

Zhang, D., Yin, J., Zhu, X. and Zhang, C. (2017) Network representation learning: A survey. *IEEE Transactions on Big Data*, **6**(1), 3–28.

Zuo, Y., Liu, G., Lin, H., Guo, J., Hu, X. and Wu, J. (2018) Embedding temporal network via neighborhood formation, in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2857–2866.