TREND: TempoRal Event and Node Dynamics for Graph Representation Learning

Zhihao Wen
Singapore Management University
Singapore
zhwen.2019@smu.edu.sg

Yuan Fang
Singapore Management University
Singapore
yfang@smu.edu.sg

ABSTRACT

Temporal graph representation learning has drawn significant attention for the prevalence of temporal graphs in the real world. However, most existing works resort to taking discrete snapshots of the temporal graph, or are not inductive to deal with new nodes, or do not model the exciting effects which is the ability of events to influence the occurrence of another event. In this work, We propose TREND, a novel framework for temporal graph representation learning, driven by TempoRal Event and Node Dynamics and built upon a Hawkes process-based graph neural network (GNN). TREND presents a few major advantages: (1) it is inductive due to its GNN architecture; (2) it captures the exciting effects between events by the adoption of the Hawkes process; (3) as our main novelty, it captures the individual and collective characteristics of events by integrating both event and node dynamics, driving a more precise modeling of the temporal process. Extensive experiments on four real-world datasets demonstrate the effectiveness of our proposed model.

CCS CONCEPTS

• **Computing methodologies → Learning latent representations**; • **Information systems → Data mining**.

KEYWORDS

Temporal graphs, Hawkes process, GNN, event and node dynamics

ACM Reference Format:
Zhihao Wen and Yuan Fang. 2022. TREND: TempoRal Event and Node Dynamics for Graph Representation Learning. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485447.3512164

1 INTRODUCTION

Graph-structured data widely exist in real-world scenarios, *e.g.*, social networks, citation networks, e-commerce networks, and the World Wide Web. To discover insights from these data, graph representation learning has emerged as a key enabler which can encode graph structures into a low-dimensional latent space. The state of the art has made important progress and many approaches are proposed, which can be mainly divided into two categories: network embedding [1] and graph neural networks (GNN) [41]. Network embedding approaches are often transductive, which directly learn node embedding vectors using various local structures, like random walks in DeepWalk [30] and node2vec [7], and 1st- and 2nd-order proximity in LINE [35]. In contrast, GNNs do not directly learn node embedding vectors. They instead learn an inductive aggregation function [10, 16, 37, 40, 44] which can be generalized to unseen nodes or even new graphs in the same feature space. Typical GNNs follow a message passing framework, where each node receives and aggregates messages (*i.e.*, node features or embeddings) from its neighboring nodes recursively in multiple layers. In other words, GNNs are capable of not only encoding graph structures, but also preserving node features.

Most of these graph representation methods focus on static graphs with structures frozen in time. However, real-world graphs often present complex dynamics that evolve continuously in time. For instance, in social networks, burst events often rapidly change the short-term social interaction pattern, while on an e-commerce user-item graph, long-term user preferences may drift as new generations of product emerge. More precisely, in a temporal graph [19], the temporal evolution arises from the chronological formation of links between nodes. As illustrated in Fig. 1, the toy graph evolving from time t_1 through t_3 can be described by a series of triple $\{(A, B, t_1), (B, C, t_1), (C, D, t_2), (C, E, t_2), (B, C, t_3), \ldots\}$, where each triple (i, j, t) denotes a link formed between nodes i and j at time t. Hence, the prediction of future links depends heavily on the dynamics embodied in the historical link formation [33]. In this paper, we investigate the important problem of *temporal graph representation learning*, in which we learn representations that evolve with time on a graph. In particular, we treat the formation of a link at a specific time as an *event*, and a graph evolves or grows continuously as more events are accumulated [12].

Prior work. In general, the requirement for temporal graph representation learning is that the learned representations must not only preserve graph structures and node features, but also reflect its topological evolution. However, this goal is non-trivial, and it is not until recently that several works on this problem have emerged. Among them, some [2, 6, 21, 27, 32, 45] discretize the temporal graph into a sequence of static graph snapshots to simplify the model. As a consequence, they cannot fully capture the continuously evolving dynamics, for the fine-grained link formation events "in-between" the snapshots are inevitably lost. For continuous-time methods, CTDNE [25] resort to temporal random walks that respect the chronological sequence of the edges; TGAT [42] employs a GNN framework with functional time encoding to map continuous time

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512164

and self-attention to aggregate temporal-topological neighborhood. However, these methods often fail to explicitly capture the *exciting effects* [47] between sequential events, particularly the influence of historical events on the current events. Nonetheless, such effects can be well captured by temporal point processes, most notably the *Hawkes process* [11, 23], which assumes that historical events prior to time t can excite the process in the sense that future events become more probable for some time after t. This property is desirable for modeling the graph-wide link formation process, in which each link formation is considered an *event* that can be excited by recent events. For example, in social networks, a celebrity who has attracted a large crowd of followers lately (*e.g.*, due to winning a prestigious award) is likely to attract more followers in the near future. However, for temporal graph representation learning, existing Hawkes process-based network embedding methods [22, 47] are inherently transductive. While DyRep [36] presents an inductive framework based on the temporal point process, it addresses a different problem setting of two-time scale with both association and communication events. In our paper, we focus on learning the dynamics of evolving topology, where each event represents the formation of a link.

Challenges and present work. To effectively model the events of link formation on a temporal graph, we propose a Hawkes process-based GNN framework to reap the benefits of both worlds. Previous methods do not employ Hawkes or similar point processes for modeling the exciting effects between events [42], or not use message-passing GNNs for preserving the structures and features of nodes in an inductive manner [22, 47], or neither [25]. More importantly, while the Hawkes process is well suited for modeling the graph-wide link formation process, prior methods fail to examine two open challenges on modeling the events (*i.e.*, link formation), as follows.

Challenge 1: *How do we capture the uniqueness of the events on an individual scale?* Different links are often formed out of different contexts and time periods, causing subtle differences among events. Taking the research collaboration network in Fig. 1 as an example, while links are all collaborations, each collaboration can be unique in its own way. For instance, the collaboration between researchers A and B, and that between F and G, could be formed due to different backgrounds and reasons (*e.g.*, they might simply be students of the same advisor who devised the solution together, or they possess complementary skills required in a large multi-disciplinary project). Multiple collaborations can also be formed at different times, such as between B and C at t_1 and t_3, for potentially different reasons. Conventional methods on temporal graphs train one model to fit all events, where different events tend to pull the model in many opposing directions. The resulting model would be overly diffuse with its center of mass around the most frequent patterns among the events, whilst neglecting many long-tailed patterns covering their individual characteristics. Hence, in this paper, motivated by hypernetworks [8, 29, 39], we learn an *event prior*, which only encodes the general knowledge of link formation. This event prior can be further specialized in an event-wise manner to fit the individual characteristics of events.

Challenge 2: *How do we govern the occurrence of events on a collective scale?* While events exhibit individual characteristics, they are not formed in isolation and related events often manifest collective characteristics. Events sharing a common node can be "constrained as a collection" due to the common influence from their shared node. That is, the collection of events of each node should match the arrival rate of the node, which we call node dynamics. Of course, for two different nodes, each would have its own event collection, and each collection should match different arrival rates of the two nodes. As shown in Fig. 1, researchers A and C have different tendency to form a collaboration with others at time t_3, with C being more active in seeking collaborations. Moreover, as the graph evolves from time t_1 to t_3, researcher C's tendency in collaborating with others also evolves to become higher (*e.g.*, due to C's growing reputation). In other words, the events stemming from a common node are collectively governed by the dynamics of the node as a function of time. Hence, we formulate the notion of *node dynamics* to model the collective characteristics of the events from the same node. Intuitively, integrating the node dynamics provides a regularizing mechanism beyond individual events, to ensure that the events from a node, as a collection, conform to the continuous evolution of the node. Despite the importance of node dynamics, it has not been explored in temporal graph representation learning.

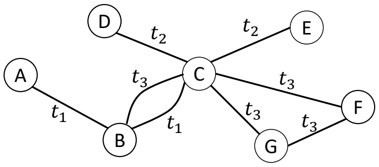


Figure 1: Toy temporal graph for research collaborations that evolves through time $t_1, t_2, t_3, \cdots$. Each node is a researcher and each link is a collaboration between researchers formed at a specific time.

Contributions. We propose TREND, a novel framework for temporal graph representation learning driven by TempoRal Event and Node Dynamics. TREND is built upon a Hawkes process-based GNN, and presents a few major advantages. Firstly, owning to its GNN architecture, it is inductive in nature, *i.e.*, able to handle new nodes at test time. Secondly, owning to the adoption of the Hawkes process, it maps the graph-wide link formation process to capture a holistic view of the temporal evolution. Thirdly, TREND integrates both the event and node dynamics to respectively capture the individual and collective characteristics of events, which drives a more precise modeling of the link formation process.

In summary, our work encompasses the following contributions. (1) For the first time in temporal graph representation learning, we recognize the importance of modeling the events at an individual and collective scale, and formulate them as event and node dynamics. (2) We propose a novel framework called TREND with both event and node dynamics to more precisely model events under a Hawkes process-based GNN. On one hand, the event dynamics learns an adaptable event prior to capture the uniqueness of events individually. On the other hand, the node dynamics regularizes the events at the node level to govern their occurrences collectively. (3) We conduct extensive experiments on four real-world datasets, which demonstrate the advantages of TREND.

2 RELATED WORK

Recently, a large body of graph representation learning methods has been proposed, including network embedding [1] and graph neural networks [41]. To address real-world scenarios in which graphs continuously evolve in time, there have been some efforts in temporal graph representation learning. Intuitively, a temporal graph can be modeled as a series of snapshots. The general idea is to learn node representations for each graph snapshot, and then capture both the graph structures in each snapshot and the sequential effect across the snapshots. The specific techniques vary in different works, such as matrix perturbation theory [20, 46], skip-grams [2] and triadic closure process [45]. To effectively capture the sequential effect, recurrent neural networks (RNNs) have been a popular tool [5, 9, 17, 34], which leverage the chronological sequence of representations across all snapshots. From a different perspective, instead of using RNNs to generate node representations, EvolveGCN [27] uses RNN to evolve GCN parameters. Besides, rather than directly learning the representation, DynGEM [6] incrementally builds the representations of a snapshot from those of the previous snapshot.

However, snapshots are approximations which discretize a continuously evolving graph, inevitably suffering from a fair degree of information loss in the temporal dynamics. To overcome this problem, another line of work aims to model the continuous process of temporal graph evolution, usually by treating each event (typically defined as the formation of a link that can occur continuously in time) as an individual training instance. Among them, some employ temporal random walks to capture the continuous-time network dynamics, including CTDNE [25] based on time-respect random walks, and CAW-N [38] based on causal anonymous walks. Apart from random walks, GNN-based models have also emerged to deal with continuous time, e.g., TGAT [42]. While these methods can deal with a continuously evolving graph, they fail to explicitly model the exciting effects between sequential events holistically on the entire graph. In view of this, several network embedding methods [13, 22, 47] incorporate temporal point processes such as Hawkes process into their models, being capable of modeling the graph-wide formation process. Moreover, DyREP [36] is an inductive GNN-based model that also exploits the temporal point processes.

Note that, among existing methods for temporal graph representation learning, those employing an embedding lookup for node representations are usually transductive [2, 22, 25, 34, 45, 47] and thus unable to directly make predictions on new nodes at a future time. In contrast, GNN-based methods [27, 36, 42] are naturally inductive, able to extend to new nodes in the same feature space. However, among them only DyREP [36] leverages temporal point processes, but it is designed to capture association and communication events, which differs from our problem setting to specifically deal with the link formation process. Furthermore, none of existing methods integrates both event- and node-dynamics to capture the individual and collective characteristics of events, respectively.

3 PRELIMINARIES

In this section, we first present the problem of temporal graph representation learning, and then introduce a brief background on the Hawkes process.

3.1 Temporal Graph Representation Learning

A *temporal graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathbf{X})$ is defined on a set of nodes $\mathcal{V}$, a set of edges $\mathcal{E}$, a time domain $\mathcal{T}$ and an input feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_0}$. Each node has a d_0-dimensional input feature vector corresponding to one row in $\mathbf{X}$. An *event* is a triple (i, j, t) denoting the formation of an edge $(i, j) \in \mathcal{E}$ (also called a link) between node $i \in \mathcal{V}$ and node $j \in \mathcal{V}$ at time $t \in \mathcal{T}$. Alternatively, a temporal graph can be defined as a chronological series of events $\mathcal{I} = \{(i, j, t)_m : m = 1, 2, \dots, |\mathcal{E}|\}$. Note that two nodes may form a link more than once at different times. Thus, there may be two events (i, j, t_1) and (i, j, t_2) such that $t_1 \neq t_2$. Besides, in this work, we only consider the growth of temporal graph, and make deletions of node and edge as future work.

We study the problem of inductive temporal graph representation learning. Specifically, given $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathbf{X})$, we aim to learn a parametric model $\Phi(\cdot; \theta)$ with parameters θ, such that Φ maps any node in the same feature space of $\mathbf{X}$ at any time to a representation vector. That is, $\Phi : \mathcal{V}' \times \mathcal{T} \to \mathbb{R}^d$, where $\mathcal{V} \subset \mathcal{V}'$. The set difference $\mathcal{V}' \setminus \mathcal{V}$ consists of new nodes in the same feature space of $\mathbf{X}$ that may appear at a future time. Such a model Φ is apparently inductive given the ability to handle new nodes.

3.2 Hawkes Process

A Hawkes process [11] is a stochastic process that can be understood as counting the number of events up to time t. Its behavior is typically modeled by a *conditional intensity* function $\lambda(t)$, the rate of event occurring at time t given the past events. A common formulation of the conditional intensity [22, 47] is given by

$$\lambda(t) = \mu(t) + \int_{-\infty}^{t} \kappa(t - s) dN(s), \tag{1}$$

where $\mu(t)$ is the base intensity at time t, κ is a kernel function to model the time decay effect of historical events on the current event (usually in the shape of an exponential function), and $N(t)$ is the number of events occurred until t. Since the Hawkes process is able to model the exciting effects between events to capture the influence of historical events holistically, it is well suited for modeling the graph-wide link formation process in a temporal graph.

4 PROPOSED APPROACH

In this section, we present a novel framework for temporal graph representation learning called TREND.

4.1 Overview of TREND

Building upon a Hawkes process-based GNN, the proposed TREND is able to model the graph-wide link formation process in an inductive manner. More importantly, it integrates event and node dynamics into the model to fully capture the individual and collective characteristics of events.

The overall framework of TREND is shown in Fig. 2. First of all, in Fig. 2(a), an input temporal graph undergoes a temporal GNN aggregation in multiple layers, whose output representations serve as the input for modeling event and node dynamics. The GNN layer aggregates both the self-information and historical neighbors' information, which are building blocks to materialize the conditional intensity in the Hawkes process. Next, we model event dynamics

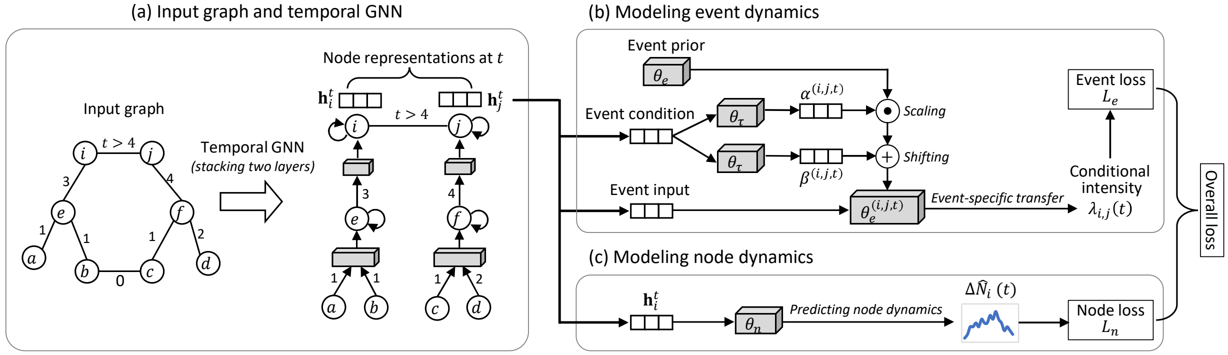


Figure 2: Overall framework of TREND, which integrates event and node dynamics in a Hawkes process-based GNN.

to capture the individual characteristics of events, as shown in Fig. 2(b). We perform an event-conditioned, learnable transformation to adapt the event prior to the input event, resulting in an event-specific transfer function to generate the conditional intensity in the Hawkes process. Moreover, we model node dynamics to capture the collective characteristics of events at the node level, as shown in Fig. 2(c). We build an estimator to predict the node dynamics across nodes and times, which governs the behavior of events occurring on the same node. At last, we integrate the event and node losses to jointly optimize event and node dynamics.

4.2 Hawkes process-based GNN

We first introduce a Hawkes process-based GNN framework, which is to be further integrated with event and node dynamics later.

Hawkes process on temporal graph. In the context of temporal graph, the Hawkes process is able to model the graph-wide link formation process. Specifically, whether nodes i and j form a link at t, can be quantified by the conditional intensity of the event,

$$\lambda_{i,j}(t) = \mu_{i,j}(t) + \sum_{(i,j',t')\in\mathcal{H}_i(t)} \gamma_{j'}(t')\kappa(t-t') \\ + \sum_{(i',j,t')\in\mathcal{H}_j(t)} \gamma_{i'}(t')\kappa(t-t'). \quad (2)$$

In particular, $\mu_{i,j}(t)$ is the base rate of the event that i and j form a link at time t, which is not influenced by historical events on i or j. $\mathcal{H}_i(t)$ is the set of historical events on i w.r.t. time t, *i.e.*, $\mathcal{H}_i(t) = \{(i,j',t') \in \mathcal{I} : t' < t\}$, and we call j' a historical neighbor of i. $\gamma_{j'}(t')$ represents the amount of excitement induced by a historical neighbor j' at t' on the current event. Note that we are treating each link as undirected, and thus the current event is influenced by historical neighbors of both nodes i and j. In the case of directed link, we can modify Eq. (2) by keeping only one of the two summation terms. $\kappa(\cdot)$ is a kernel function to capture the time decay effect w.r.t. t, defined in the form of an exponential function as $\kappa(t-t') = \exp(-\delta(t-t'))$, where $\delta > 0$ is a learnable scalar to control the rate of decay.

Next, temporal graph representations are used to materialize the conditional intensity in Eq. (2). Given the temporal representations of nodes i, j at time t, denoted $\mathbf{h}_i^t, \mathbf{h}_j^t$ respectively, the conditional intensity can be generated from a *transfer function* f [23, 36], *i.e.*,

$$\lambda_{i,j}(t) = f(\mathbf{h}_i^t, \mathbf{h}_j^t), \quad (3)$$

which should meet the following criteria. (1) The input representations $\mathbf{h}_i^t, \mathbf{h}_j^t$ should be derived from not only their inherent self-information, but also their historical neighbors' information. While the self-information is the basis of the base intensity $\mu_{i,j}(t)$, historical neighbors are crucial to model the excitement induced by historical events. (2) The output of the transfer function f must be positive, since it represents an intensity. We will discuss the choice of the transfer function in Sect. 4.3.

Temporal GNN layer. We materialize the temporal representations in Eq. (3) using GNNs, owning to their inductive nature and superior performance. Based on the message passing scheme, each node receives, aggregates and maps messages (*e.g.*, features or embeddings) from its neighboring nodes recursively, in multiple layers. Here we present a temporal formulation of GNN in consideration of the representational criteria listed above, so that the learned temporal representations can be used to materialize the conditional intensity. Formally, let $\mathbf{h}_i^{t,l} \in \mathbb{R}^{d_l}$ be the d_l-dimensional embedding vector of node i at time t in the l-th layer, which is computed by

$$\mathbf{h}_i^{t,l} = \sigma\Big(\underbrace{\mathbf{h}_i^{t,l-1}\mathbf{W}_{\text{self}}^l}_{\substack{\text{self-information}\\ \text{(for base intensity)}}} + \underbrace{\sum_{(i,j',t')\in\mathcal{H}_i(t)} \mathbf{h}_{j'}^{t',l-1}\mathbf{W}_{\text{hist}}^l\tilde{\kappa}_i(t-t')\Big)}_{\substack{\text{historical neighbors' information}\\ \text{(for excitement by historical events)}}}, \quad (4)$$

where σ is an activation function (*e.g.*, ReLU), $\mathbf{W}_{\text{self}}^l \in \mathbb{R}^{d_{l-1}\times d_l}$ is a learnable weight matrix to map the embedding of node i itself from the previous layer, $\mathbf{W}_{\text{hist}}^l \in \mathbb{R}^{d_{l-1}\times d_l}$ is another learnable weight matrix to map the embeddings of historical neighbors, and $\tilde{\kappa}_i(t-t')$ captures the time decay effect based on the time kernel with softmax, which is given by $\tilde{\kappa}_i(t-t') = \frac{\kappa(t-t')}{\sum_{(i,j'',t'')\in\mathcal{H}_i(t)}\kappa(t-t'')}$.

In other words, the temporal representation of a node is derived by receiving and aggregating messages of itself and the historical neighbors from the previous layer. The self-information is responsible for capturing the base intensity, while the historical neighbors' information is responsible for capturing the excitement induced by historical events. To enhance the representational capacity, we stack multiple temporal GNN layers. In the first layer, the node message can be initialized by the input node features $\mathbf{X}$; in the last layer, the output temporal representation is denoted as $\mathbf{h}_i^t \in \mathbb{R}^d$ for node i at time t. The collection of parameters of all the layers is $\theta_g = \{\mathbf{W}_{\text{self}}^l, \mathbf{W}_{\text{hist}}^l : l = 1, 2, \ldots\}$.

Connection to conditional intensity. A well chosen transfer function f, taking the temporal representations as input, is equivalent to the conditional intensity of the Hawkes process in Eq. (2). We formally show the connection in Appendix A.

4.3 Modeling Event Dynamics

The key to materialize the conditional intensity is to fit a transfer function f on top of the temporal GNN layers. Previous studies on Hawkes process employ the softplus function or its variant [23, 36] as the transfer function. To ensure that f is well-fit to the conditional intensity, our first proposal is to instantiate f as a learnable function. More specifically, we use a fully connected layer (FCL). That is,

$$\lambda_{i,j}(t) = f(\mathbf{h}_i^t, \mathbf{h}_j^t) = \text{FCL}_e((\mathbf{h}_i^t - \mathbf{h}_j^t)^{\circ 2}; \theta_e), \tag{5}$$

where θ_e denotes the parameters of the fully connected layer FCL_e. Note that the input to FCL_e can be in various forms, such as the concatenation of $\mathbf{h}_i^t$ and $\mathbf{h}_j^t$, or the element-wise square (denoted by $^\circ 2$) of the difference between them. We use the latter in our formulation, which tends to achieve better empirical performance. A potential reason is that the differential representation is a good predictor of whether an event occurs between the two nodes. Lastly, FCL_e employs a sigmoid activation, to ensure the transfer function is positive.

Meanwhile, we recognize that each event can be unique in its own way, as different links are often formed out of different contexts and time periods. To precisely capture the uniqueness of events on an individual scale (Challenge 1), a global model in Eq. (5)—our first proposal—becomes inadequate. To be more specific, in a conventional one-model-fits-all approach, given the diversity in events, the learned model tends to converge around the most frequent patterns among events, while leaving long-tailed patterns that reflect the individual characteristics of events uncovered. On the other hand, training a large number of models for different kinds of events can easily cause overfiting and scalability issues, not to mention that it is difficult to categorize events in the first place. Inspired by meta-learning, particularly the line of work on *hypernetworks* [8, 29], we address the dilemma by learning an *event prior*, which can be quickly adapted to a unique model for each event, without the need to train a large number of models.

Event prior and adaptation. In our first proposal in Eq. (5), we learn a global model for all events, *i.e.*, the same θ_e parameterizes a global FCL_e as the transfer function for all events. To deal with the diversity of events, we propose to learn an event prior θ_e that aims to encode the general knowledge of link formation, such that it can be quickly specialized to fit the individual characteristics of each event. In other words, θ_e does not directly parameterize FCL_e used as the transfer function; instead, it will be adapted to each event via a learnable transformation model first, and the adapted parameters will instantiate an event-specific FCL_e as the transfer function for each event. This approach is a form of hypernetwork [8], in which a secondary neural network is used to generate the parameters of the primary network. This means the parameters of the primary network can flexibly adapt to its input, as opposed to conventional models whose parameters are frozen once training is completed. In our context, the primary network is FCL_e for the transfer function, and the secondary network is the learnable transformation model.

Particularly, during the adaptation, the event prior θ_e will transform into event (i, j, t)-specific parameters $\theta_e^{(i,j,t)}$ as follows.

$$\theta_e^{(i,j,t)} = \tau(\theta_e, \mathbf{h}_i^t \| \mathbf{h}_j^t; \theta_\tau), \tag{6}$$

which (1) is parameterized by θ_τ; (2) is conditioned on event-specific temporal representations of nodes i, j, namely, $\mathbf{h}_i^t \| \mathbf{h}_j^t$ where $\|$ is the concatenation operator; (3) outputs adapted parameters $\theta_e^{(i,j,t)}$ by transforming the event prior θ_e conditioned on $\mathbf{h}_i^t \| \mathbf{h}_j^t$. The transformed $\theta_e^{(i,j,t)}$ will further parameterize FCL_e as the transfer function, and materialize the conditional intensity below.

$$\lambda_{i,j}(t) = \text{FCL}_e((\mathbf{h}_i^t - \mathbf{h}_j^t)^{\circ 2}; \theta_e^{(i,j,t)}). \tag{7}$$

In the following, we will materialize the transformation model τ and its parameters θ_τ in detail.

Learnable transformation. We consider Feature-wise Linear Modulation (FiLM) [29], which employs affine transformations including scaling and shifting on the event prior, conditioned on event-specific temporal representations. Compared with gating [43] which can only adjust the parameters in a diminishing way, FiLM is more flexible in adjusting the parameters and can be conditioned on arbitrary input. Specifically, we employ fully connected layers to generate the scaling operator $\alpha^{(i,j,t)}$ and shifting operator $\beta^{(i,j,t)}$, conditioned on the event-specific input $\mathbf{h}_i^t \| \mathbf{h}_j^t$, as follows.

$$\alpha^{(i,j,t)} = \sigma\big((\mathbf{h}_i^t \| \mathbf{h}_j^t)\mathbf{W}_\alpha + \mathbf{b}_\alpha\big), \tag{8}$$

$$\beta^{(i,j,t)} = \sigma\big((\mathbf{h}_i^t \| \mathbf{h}_j^t)\mathbf{W}_\beta + \mathbf{b}_\beta\big), \tag{9}$$

where $\mathbf{W}_\alpha, \mathbf{W}_\beta \in \mathbb{R}^{2d \times d_{\theta_e}}$ and $\mathbf{b}_\alpha, \mathbf{b}_\beta \in \mathbb{R}^{d_{\theta_e}}$ are learnable weight matrices and bias vectors of the fully connected layers, in which d is the dimension of node representation and d_{θ_e} is the total number of parameters in the event prior θ_e. The output $\alpha^{(i,j,t)}, \beta^{(i,j,t)} \in \mathbb{R}^{d_{\theta_e}}$ are both d_{θ_e}-dimensional vectors, which represent the scaling and shifting operations of the transformation model τ. They are used to transform the event prior into event (i, j, t)-specific parameters by element-wise scaling and shifting, given by

$$\theta_e^{(i,j,t)} = \tau(\theta_e, \mathbf{h}_i^t \| \mathbf{h}_j^t; \theta_\tau) = (\alpha^{(i,j,t)} + \mathbf{1}) \odot \theta_e + \beta^{(i,j,t)}, \tag{10}$$

where $\odot$ stands for element-wise multiplication, and $\mathbf{1}$ is a vector of ones to ensure that the scaling factors are centered around one. Note that θ_e contains all the weights and biases of FCL_e, and we flatten it into a d_{θ_e}-dimensional vector.

In summary, the learnable transformation model τ is parameterized by $\theta_\tau = \{\mathbf{W}_\alpha, \mathbf{b}_\alpha, \mathbf{W}_\beta, \mathbf{b}_\beta\}$, *i.e.*, the collection of parameters of the fully connected layers that generate the scaling and shifting operators. Furthermore, τ is also a function of the event condition $\mathbf{h}_i^t \| \mathbf{h}_j^t$, for $\alpha^{(i,j,t)}$ and $\beta^{(i,j,t)}$ are functions of the event condition.

Event loss. Given an event $(i, j, t) \in \mathcal{I}$ that has occurred on the graph, we expect a higher conditional intensity $\lambda_{i,j}(t)$. On the contrary, given an event $(i, j, t) \notin \mathcal{I}$ that does not happen, we expect a lower conditional intensity. Thus, we formulate the event loss based on negative log-likelihood, the optimization of which encourages the conditional intensity of an event to match its occurrence or non-occurrence. Given any event $(i, j, t) \in \mathcal{I}$ that has occurred, its loss is defined as

$$L_e(i, j, t) = -\log(\lambda_{i,j}(t)) - Q \cdot \mathbb{E}_{k \sim P_n} \log(1 - \lambda_{i,k}(t)), \quad (11)$$

where we sample a negative node k according to the distribution P_n, so that $(i, k, t) \notin \mathcal{I}$ does not occur, and Q is the number of negative samples for each positive event. As a common practice, P_n is defined on the node degrees, namely, $P_n(v) \propto \deg(v)^{\frac{3}{4}}$ where $\deg(v)$ is the degree of node v.

4.4 Modeling Node Dynamics

Different from the event dynamics that captures the individual characteristics of events, node dynamics aims to govern the collective characteristics of events. While events can be individually different, they do not occur in isolation. Particularly, links are formed to connect nodes, which means their behaviors are collectively bounded by their common nodes. Thus, we propose to govern the collective characteristics of nodes at the node level, to capture the "event tendency" of nodes (Challenge 2)—different nodes have varying tendency to form new links with others, and even the same node would manifest different tendency at different times.

Estimator of node dynamics. More specifically, the node dynamics or the event tendency of a node at time t can be quantified by the number of new events occurring on the node at t, denoted $\Delta N_i(t)$. We build an estimator for node dynamics with a fully connected layer, trying to fit the number of new events on a given node:

$$\Delta \hat{N}_i(t) = \mathrm{FCL}_n(\mathbf{h}_i^t; \theta_n), \quad (12)$$

where the input is the temporal representation $\mathbf{h}_i^t$, the output $\Delta \hat{N}_i(t)$ is the predicted number of new events occurring on node i at time t, and θ_n contains the parameters of FCL_n.

Node loss. To ensure that the occurrence of events are consistent with the node dynamics evolving continuously on a temporal graph, we formulate a node loss such that the estimator $\Delta \hat{N}_i(t)$ can accurately reflect the groundtruth dynamics $\Delta N_i(t)$ across all nodes and times. In particular, we adopt the following smooth L_1 loss [4]:

$$L_n(i, t) = \begin{cases} 0.5(\Delta \hat{N}_i(t) - \Delta N_i(t))^2, & |\Delta \hat{N}_i(t) - \Delta N_i(t)| < 1 \\ |\Delta \hat{N}_i(t) - \Delta N_i(t)| - 0.5. & \text{otherwise} \end{cases} \quad (13)$$

The smooth L_1 loss can be viewed as a combination of both L_1 loss and L_2 loss. It is less sensitive to outliers than the L_2 loss when the input is large, and it suffers from less oscillations than the L_1 loss when the input is small. In our scenario, there exist some nodes with a very large number of new links at certain times (*e.g.*, due to burst topics on social networks). To prevent the models from being overly skewed to these nodes, and to simultaneously cater to nodes with only a few links, the smooth L_1 loss is an ideal choice.

4.5 Overall Model: TREND

Finally, we integrate both event and node dynamics into a Hawkes process-based GNN model, resulting in our proposed model TREND. Consider the set of training events $\mathcal{I}^{\mathrm{tr}} = \{(i, j, t) \in \mathcal{I} : t \leq t^{\mathrm{tr}}\}$, *i.e.*, all events on the graph up to time t^{tr}. (New events after time t^{tr} can be reserved for testing.) We optimize all parameters $\Theta = (\theta_g, \theta_e, \theta_\tau, \theta_n)$ jointly, including those of the temporal GNN layers θ_g, the event prior θ_e, the transformation model θ_τ and the estimator of node dynamics θ_n, based on the following loss:

$$\arg\min_{\Theta} \sum_{(i,j,t) \in \mathcal{I}^{\mathrm{tr}}} L_e + \eta_1 L_n + \eta_2(\|\alpha^{(i,j,t)}\|_2^2 + \|\beta^{(i,j,t)}\|_2^2), \quad (14)$$

where (1) $\eta_1 > 0$ is a hyper-parameter controlling the contribution of node dynamics to our model TREND; (2) the L_2 regularization on $\alpha^{(i,j,t)}$ and $\beta^{(i,j,t)}$ constrains the scaling and shifting operators, as it is preferred that the scaling is close to 1 and the shifting is close to zero, in order to avoid overfitting to individual events; (3) $\eta_2 > 0$ is a hyperparameter controlling the effect of the L_2 regularizor.

For implementation, we perform optimization over batches of training events using a gradient-based optimizer. The overall training procedure of TREND is outlined in Appendix B. It can be seen that the training time complexity is $O(K|\mathcal{I}^{\mathrm{tr}}|h^l Q))$, where K is the number of epochs, $|\mathcal{I}^{\mathrm{tr}}|$ is the number of training events, h is the number of historical neighbors in temporal GNN aggregation, l is the number of temporal GNN layers, and Q is the number of negative samples per training event. Note that Q and l are small constants (typically 5 or less), and h can also be a constant when employing a commonly used neighborhood sampling approach [10]. Hence, the complexity can be regarded as linear in the number of events or temporal edges on the graph.

Table 1: Statistics of datasets.

Dataset	CollegeMsg	cit-HepTh	Wikipedia	Taobao
# Events	59,835	51,315	157,474	4,294,000
# Nodes	1,899	7,577	8,227	1,818,851
# Node features	–	128	172	128
Multi-edge?	Yes	No	Yes	Yes
New nodes in testing	22.79%	100%	7.26%	23.46%

5 EXPERIMENTS

We conduct extensive experiments to evaluate TREND, with comparison to state-of-the-art baselines and in-depth model analysis.

5.1 Experimental Setup

Datasets. Four public temporal networks are used in our experiments, as summarized in Tab. 1. Note that "new nodes in testing" refers to the ratio of testing events containing at least one new node not seen during training. (1) **CollegeMsg** [26]: an online social network in which an event is a user sending another user a private message. (2) **cit-HepTh** [18]: a citation graph about high energy physics theory in which an event is a paper citation. (3) **Wikipedia** [17]: a Wikipedia graph in which an event is a user editing a page. (4) **Taobao** [3]: an e-commerce platform in which an event is a user purchasing an item. More dataset details are presented in Appendix C.

Prediction tasks. We adopt *temporal link prediction* as our main task. We evaluate a model by predicting future links based on historical links [33]. Given a temporal graph, we split the events into training and testing. Specifically, the set of training events $\mathcal{I}^{\mathrm{tr}} = \{(i, j, t) \in \mathcal{I} : t \leq t^{\mathrm{tr}}\}$ consists of all events up to time

Table 2: Performance of temporal link prediction by TREND and the baselines, in percent, with 95% confidence intervals.

In each column, the best result is **bolded** and the runner-up is underlined. Improvement by TREND is calculated relative to the best baseline. "-" indicates no result obtained due to out of memory issue; * indicates that our model significantly outperforms the best baseline based on two-tail t-test ($p < 0.05$).

	CollegeMsg		cit-HepTh		Wikipedia		Taobao	
	Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1
DeepWalk	66.54±5.36	67.86±5.86	51.55±0.90	50.39±0.98	65.12±0.94	64.25±1.32	53.59±0.18	56.67±0.12
Node2vec	65.82±4.12	69.10±3.50	65.68±1.90	66.13±2.15	75.52±0.58	75.61±0.52	52.74±0.33	54.86±0.32
VGAE	65.82±5.68	68.73±4.49	66.79±2.58	67.27±2.84	66.35±1.48	68.04±1.18	55.97±0.22	59.80±0.16
GAE	62.54±5.11	66.97±3.22	69.52±1.10	70.28±1.33	68.70±1.34	69.74±1.43	58.13±0.15	61.40±0.07
GraphSAGE	58.91±3.67	60.45±4.22	70.72±1.96	71.27±2.41	72.32±1.25	73.39±1.25	60.74±0.18	61.61±0.20
CTDNE	62.55±3.67	65.56±2.34	49.42±1.86	44.23±3.92	60.99±1.26	62.71±1.49	51.64±0.32	43.99±0.38
EvolveGCN	63.27±4.42	65.44±4.72	61.57±1.53	62.42±1.54	71.20±0.88	73.43±0.51	-	-
GraphSAGE+T	69.09±4.91	69.41±5.45	67.80±1.27	69.12±1.12	57.93±0.53	63.41±0.91	67.05±0.23	67.69±0.17
TGAT	58.18±4.78	57.23±7.57	78.02±1.93	78.52±1.61	76.45±0.91	76.99±1.16	70.07±0.59	71.31±0.18
HTNE	73.82±5.36	74.24±5.36	66.70±1.80	67.47±1.16	77.88±1.56	78.09±1.40	59.03±0.17	60.34±0.19
MMDNE	73.82±5.36	74.10±3.70	66.28±3.87	66.70±3.39	79.76±0.89	79.87±0.95	58.24±0.10	59.04±0.16
TREND	**74.55**±1.95	**75.64**±2.09	**80.37***±2.08	**81.13***±1.92	**83.75***±1.19	**83.86***±1.24	**78.56***±0.17	**80.67***±0.15
(improv.)	(+0.99%)	(+1.89%)	(+3.01%)	(+3.32%)	(+5.00%)	(+4.99%)	(+12.11%)	(+13.12%)

t^{tr}. The remaining events after time t^{tr}, denoted by the set $\mathcal{I}^{\text{te}} = \mathcal{I} \setminus \mathcal{I}^{\text{tr}}$, is reserved for testing. Given a candidate triple (i, j, t) for some $t > t^{\text{tr}}$, the objective is to predict whether a link between nodes i and j is formed at the given future time t, *i.e.*, if $(i, j, t) \in \mathcal{I}^{\text{te}}$. Note that our model can perform temporal link prediction between all nodes, including new nodes not seen during training, due to its inductive nature. Specifically, in testing, we first generate temporal node representations based on the trained model, and feed them to a downstream logistic regression classifier to predict if a candidate triple is positive or negative. The classifier is trained using a 80%/20% train/test split on the testing events, and repeated for five different splits. More details are given in Appendix D.

We further adopt a secondary task of *temporal node dynamics prediction.* While the training and testing events follow the same setup of the main task, we aim to predict the number of new neighbors of a node i at a specific future time $t > t^{\text{tr}}$. Similarly, the first step in testing is to generate temporal node representations based on the trained model, which are then fed into a downstream linear regression model. To train the regression model, we randomly split the nodes of the testing events into 80%/20% train/test split.

Settings of TREND. For the temporal GNN, we employ two layers with a ReLU activation. The hidden layer dimension is set to 16 on all datasets. The output dimension is set to 32 on CollegeMsg, 16 on cit-HepTh and 128 on Wikipedia and Taobao, based on the size and complexity of the graph. The transfer function FCL_e employs a sigmoid activation, the estimator of node dynamics FCL_n uses a ReLU activation, and $\alpha^{(i,j,t)}, \beta^{(i,j,t)}$ both employ a LeakyReLU. The number of negative samples per positive event is set to $Q = 1$. For the final loss function in Eq. (14), the coefficient of node loss is set to $\eta_1 = 0.1$ on Taobao and $\eta_1 = 0.01$ on other datasets, whereas the coefficient of L_2 regularizer is set to $\eta_2 = 0.001$ on CollegeMsg and cit-HepTh, $\eta_2 = 0.01$ on Taobao, and $\eta_2 = 1$ on Wikipedia. Note that we will present an analysis on the impact of the hyperparameters Q, η_1 and η_2 in Sect. 5.4. Lastly, we use the Adam optimizer with the learning rate 0.001.

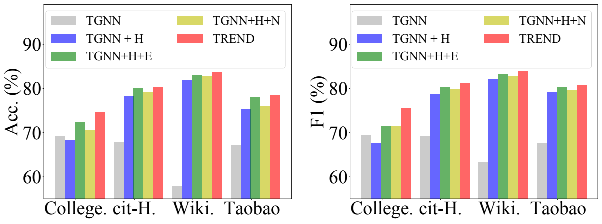
Baselines. We compare TREND with a competitive suit of baselines from three categories. (1) *Static approaches*: DeepWalk [30], Node2vec [7], VGAE [15], GAE [15] and GraphSAGE [10]. They train a model or node embedding vectors on the static graph formed from the training events, without considering any temporal information. (2) *Temporal approaches*: CTDNE [25], EvolveGCN [27], GraphSAGE+T [10] and TGAT [42]. They train a model or node embedding vectors on the temporal graph formed from the training events. Note that GraphSAGE+T is a temporal extension of GraphSAGE implemented by us, in which the time decay effect is incorporated into the aggregation function. (3) *Hawkes process-based approaches*: HTNE [47] and MMDNE [22]. They similarly train node embedding vectors on the temporal graph formed from the training events. However, they leverage the node representations to model the conditional intensity of events based on the Hawkes process. More baseline descriptions are in Appendix E.

5.2 Temporal Link Prediction

In Tab. 2, we compare the performance of TREND with the baselines on the main task. In general, our method performs the best among all methods, demonstrating the benefits of event and node dynamics. We make two further observations.

First, among static methods, we can see that GNN-based methods (VGAE, GAE and GraphSAGE) tend to perform better, as they are inductive in nature, and their message passing scheme is able to integrate both node features and graph structures. On the other hand, DeepWalk and Node2vec are transductive, which cannot directly extend to new nodes in testing. In our experiments, the embedding vector of new nodes are randomly initialized for tranductive methods, and thus their performance can be poor when dealing with new nodes. One exception is on the CollegeMsg dataset, where there is no node features and one-hot encoding of node IDs are used instead. In this case, GNN-based methods lose the inductive capability and do not outperform transductive methods.

Second, temporal approaches are generally superior to static approaches, showing the importance of temporal information. Among



(a) Accuracy

(b) F1

Figure 3: Effect of main components.

the three GNN-based approaches (EvolveGCN, GraphSAGE+T and TGAT), EvolveGCN often performs the worst. The reason is that EvolveGCN is based on discrete snapshots, which inevitably suffers from some loss in the temporal evolution. Moreover, the Hawkes process-based approaches (HTNE and MMDNE) achieve strong performance, demonstrating that the Hawkess process is ideal for modeling the temporal evolution on graphs. Unfortunately, they are transductive and thus do not outperform GraphSAGE-T and TGAT on cit-HepTh and Taobao where there are a large proportion of new nodes in testing. Besides, we can see that TREND performs much better on Taobao than on other datasets. A potential reason is that Taobao is the biggest graph having more "diversity" in events, such that the adaptation of event prior becomes more crucial and can lead to larger performance gain.

5.3 Ablation Study

To understand the contribution of each component in TREND, we study the following ablated models on the task of temporal link prediction. (1) *TGNN*, which only stacks two temporal GNN layers and optimizes the inner product of node pairs; (2) *TGNN+H*, which adds the global transfer function for the Hawkess process in Eq. (5) to *TGNN*; (3) *TGNN+H+E* and *TGNN+H+N*, which further model the event and node dynamics on top of *TGNN+H*, respectively. Note that *TGNN+H+E* uses the event-specific transfer function in Eq. (7).

As shown in Fig. 3, the performance generally increases when we gradually add more components to *TGNN*. This shows that every component is useful for modeling temporal graphs. Note that *TGNN+H+E* typically outperforms *TGNN+H+N*, since the event dynamics directly deals with individual events while the node dynamics only works at the node level. Nevertheless, when integrating both event and node dynamics, the full model TREND achieves the best performance, showing that it is important to jointly model both event and node dynamics.

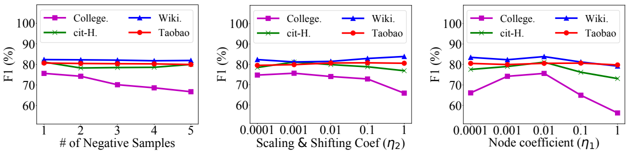
5.4 Hyperparameter Study

Here we present a sensitivity analysis of the hyperparameters.

Negative sampling size. As shown in Fig. 4(a), generally speaking, the performance of TREND does not improve with more negative samples. On all datasets, it is robust to choose just one negative sample for each positive event for efficiency.

Regularization on scaling and shifting. To prevent overfitting, the event-conditioned transformations are regularized to prevent excessive scaling or shifting. The regularization is controlled by the coefficient η_2, and we study its effect in Fig. 4(b). The performance is quite stable over different values of η_2, although smaller values in the range [0.0001, 0.01] tend to perform better. Worse performance can be observed on larger values, which implies very little scaling and shifting similar to removing the event-conditioned transformation.

Coefficient for node loss. We vary η_1, which controls the weight of the node loss, and study its impact in Fig. 4(c). We observe that the performance is suboptimal if η_1 is too small or large, and the performance of TREND is generally robust when η_1 is round 0.01. This shows that a well balanced node and event loss can improve the stability and performance.



(a) Negative samples

(b) Regularization

(c) Node coefficient

Figure 4: Hyperparameter sensitivity.

Table 3: Performance of node dynamics prediction.

Model	CollegeMsg	cit-HepTh	Wikipedia	Taobao
CTDNE	10.0636	3.0173	7.3265	0.5789
EvolveGCN	3.1964	2.5610	6.8651	-
GraphSAGE+T	21.9444	<u>2.2421</u>	<u>5.9231</u>	<u>0.5505</u>
TGAT	<u>2.6903</u>	2.8094	7.7737	0.5550
HTNE	12.3587	3.2781	6.8860	0.5749
MMDNE	8.0555	2.7456	6.9552	0.5643
TREND	**2.3549**	**2.2066**	**5.9140**	**0.5491**

5.5 Temporal Node Dynamics Prediction

Finally, we evaluate the task of temporal node dynamics prediction. We report the mean absolute error (MAE) between the predicted value $\Delta\hat{N}_i(t)$ and the groundtruth $\Delta N_i(t)$ in Tab. 3. The results show that TREND consistently achieves the smallest MAE on all four datasets, which demonstrate its versatility beyond temporal link prediction, and that the estimator of node dynamics works well as intended.

6 CONCLUSION

In this paper, we studied the problem of temporal graph representation learning. Specifically, we proposed TREND, a novel framework for temporal graph representation learning, driven by event and node dynamics on a Hawkes process-based GNN. TREND is inductive and able to capture a holistic view of the link formation process. More importantly, it integrates both the event and node dynamics to respectively capture the individual and collective characteristics of events, for a more precise modeling of the temporal evolution. Finally, we conducted extensive experiments on four real-world datasets and demonstrated the superior performance of TREND.

ACKNOWLEDGMENTS

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funds (Grant No. A20H6b0151).

REFERENCES

[1] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1616–1637.

[2] Lun Du, Yun Wang, Guojie Song, Zhicong Lu, and Junshan Wang. 2018. Dynamic network embedding: an extended approach for skip-gram based network embedding. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Vol. 2018. 2086–2092.

[3] Zhengxiao Du, Xiaowei Wang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Sequential scenario-specific meta learner for online recommendation. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2895–2904.

[4] Ross Girshick. 2015. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*. 1440–1448.

[5] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. 2020. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems* 187 (2020), 104816.

[6] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. 2018. DynGEM: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273* (2018).

[7] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 855–864.

[8] David Ha, Andrew Dai, and Quoc V Le. 2017. Hypernetworks. In *Proceedings of the International Conference on Learning Representations*.

[9] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. 2019. Variational Graph Recurrent Neural Networks. In *Proceedings of the International Conference on Neural Information Processing Systems*. 10701–10711.

[10] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the International Conference on Neural Information Processing Systems*. 1025–1035.

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

[12] Petter Holme and Jari Saramäki. 2012. Temporal networks. *Physics reports* 519, 3 (2012), 97–125.

[13] Yugang Ji, Tianrui Jia, Yuan Fang, and Chuan Shi. 2021. Dynamic Heterogeneous Graph Embedding via Heterogeneous Hawkes Process. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, Part I*. 388–403.

[14] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proceedings of the International Conference on Learning Representations*.

[15] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[16] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the International Conference on Learning Representations*.

[17] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1269–1278.

[18] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 177–187.

[19] Aming Li, Sean P Cornelius, Y-Y Liu, Long Wang, and A-L Barabási. 2017. The fundamental advantages of temporal networks. *Science* 358, 6366 (2017), 1042–1046.

[20] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed network embedding for learning in a dynamic environment. In *Proceedings of the ACM Conference on Information and Knowledge Management*. 387–396.

[21] Taisong Li, Jiawei Zhang, S Yu Philip, Yan Zhang, and Yonghong Yan. 2018. Deep dynamic network embedding for link prediction. *IEEE Access* 6 (2018), 29219–29230.

[22] Yuanfu Lu, Xiao Wang, Chuan Shi, Philip S Yu, and Yanfang Ye. 2019. Temporal network embedding with micro-and macro-dynamics. In *Proceedings of the ACM International Conference on Information and Knowledge Management*. 469–478.

[23] Hongyuan Mei and Jason M Eisner. 2017. The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 6754–6764.

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the International Conference on Neural Information Processing Systems*. 3111–3119.

[25] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference*. 969–976.

[26] Pietro Panzarasa, Tore Opsahl, and Kathleen M Carley. 2009. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *Journal of the American Society for Information Science and Technology* 60, 5 (2009), 911–932.

[27] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 5363–5370.

[28] James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. *Linguistic inquiry and word count: LIWC 2001*. Mahway: Lawrence Erlbaum Associates.

[29] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[30] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[31] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the International Conference on Machine Learning*. 1278–1286.

[32] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. DySAT: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the ACM International Conference on Web Search and Data Mining*. 519–527.

[33] Purnamrita Sarkar, Deepayan Chakrabarti, and Michael I Jordan. 2012. Nonparametric Link Prediction in Dynamic Networks. In *Proceedings of the International Conference on Machine Learning*. 1897–1904.

[34] Uriel Singer, Ido Guy, and Kira Radinsky. 2019. Node Embedding over Temporal Graphs. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 4605–4612.

[35] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *Proceedings of the International Conference on World Wide Web*. 1067–1077.

[36] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning representations over dynamic graphs. In *Proceedings of the International Conference on Learning Representations*.

[37] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proceedings of the International Conference on Learning Representations*.

[38] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. 2021. Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks. In *Proceedings of the International Conference on Learning Representations*.

[39] Zhihao Wen, Yuan Fang, and Zemin Liu. 2021. Meta-Inductive Node Classification across Graphs. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1219–1228.

[40] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *Proceedings of the International Conference on Machine Learning*. 6861–6871.

[41] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2020), 4–24.

[42] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. In *Proceedings of the International Conference on Learning Representations*.

[43] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine learning*. PMLR, 2048–2057.

[44] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *Proceedings of the International Conference on Learning Representations*.

[45] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. 2018. Dynamic network embedding by modeling triadic closure process. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[46] Dingyuan Zhu, Peng Cui, Ziwei Zhang, Jian Pei, and Wenwu Zhu. 2018. High-order proximity preserved embedding for dynamic networks. *IEEE Transactions on Knowledge and Data Engineering* 30, 11 (2018), 2134–2144.

[47] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. 2018. Embedding temporal network via neighborhood formation. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2857–2866.

APPENDICES

A CONNECTION BETWEEN TRANSFER FUNCTION AND CONDITIONAL INTENSITY

In the following, we show that a well chosen transfer function f, taking the temporal representations as input, is equivalent to the conditional intensity of the Hawkes process in Eq. (2). Suppose the temporal representations are generated through stacking l temporal GNN layers. First, let us define the base intensity as a function of the self-information:

$$\mu_{i,j}(t) = f_\mu(\mathbf{h}_i^{t,l-1}\mathbf{W}_{\text{self}}^l, \mathbf{h}_j^{t,l-1}\mathbf{W}_{\text{self}}^l). \tag{15}$$

Next, we define the amount of excitement induced by a historical neighbor as a function of the historical neighbors' information:

$$\gamma_{j'}(t') = f_\gamma(\mathbf{h}_{j'}^{t',l-1}\mathbf{W}_{\text{hist}}^l), \quad \gamma_{i'}(t') \quad = f_\gamma(\mathbf{h}_{i'}^{t',l-1}\mathbf{W}_{\text{hist}}^l). \tag{16}$$

Given these building blocks, we rewrite the conditional intensity in Eq. (2) as

$$\begin{aligned}\lambda_{i,j}(t) = f_\lambda\Big(&\mathbf{h}_i^{t,l-1}\mathbf{W}_{\text{self}}^l, \{\mathbf{h}_{j'}^{t',l-1}\mathbf{W}_{\text{hist}}^l : (i,j',t') \in \mathcal{H}_i(t)\},\\ &\mathbf{h}_j^{t,l-1}\mathbf{W}_{\text{self}}^l, \{\mathbf{h}_{i'}^{t',l-1}\mathbf{W}_{\text{hist}}^l : (i',j,t') \in \mathcal{H}_j(t)\}\Big),\end{aligned} \tag{17}$$

where f_λ is a composite function of f_μ, f_γ and the summation. By choosing the right transfer function f, we further rewrite f_λ as the composition of f and the temporal GNN layer f_g given in Eq. (4), *i.e.*, $f_\lambda = f \circ f_g$. Subsequently, the conditional intensity is given by

$$\begin{aligned}\lambda_{i,j}(t) &= (f \circ f_g)\Big(\mathbf{h}_i^{t,l-1}\mathbf{W}_{\text{self}}^l, \{\mathbf{h}_{j'}^{t',l-1}\mathbf{W}_{\text{hist}}^l : (i,j',t') \in \mathcal{H}_i(t)\},\\ &\qquad \mathbf{h}_j^{t,l-1}\mathbf{W}_{\text{self}}^l, \{\mathbf{h}_{i'}^{t',l-1}\mathbf{W}_{\text{hist}}^l : (i',j,t') \in \mathcal{H}_j(t)\}\Big)\\ &= f(\mathbf{h}_i^t, \mathbf{h}_j^t).\end{aligned} \tag{18}$$

Thus, a well-fit transfer function f, such as a neural network, can approximate the conditional intensity.

B PSEUDOCODE

We outline the training procedure of TREND in Algorithm 1.

Algorithm 1 TRAINING PROCEDURE OF TREND

```
Input: Training graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathrm{X})$, training events $\mathcal{I}^{\text{tr}}$.
Output: Temporal GNN $\theta_g$, event prior $\theta_e$, transformation model $\theta_\tau$, estimator of node dynamics $\theta_n$.
$\theta_g, \theta_e, \theta_\tau, \theta_n \leftarrow$ parameters initialization;
while not converged do
    sample a batch of temporal events $(i, j, t)$ from $\mathcal{I}^{\text{tr}}$;
    for each event $(i, j, t)$ in the batch do
        calculate node representations $\mathbf{h}_i^t, \mathbf{h}_j^t$for nodes $i,j$;    ▹ Eq. (4)
        $\theta_e^{(i,j,t)} \leftarrow$ event-conditioned adaptation on $\theta_e$;    ▹ Eq. (10)
        calculate event intensity $\lambda_{i,j}(t)$;    ▹ Eq. (7)
        calculate the overall loss;    ▹ Eqs. (11), (13), (14)
    end for
    $\theta_g, \theta_e, \theta_\tau, \theta_n \leftarrow$ backpropagation of overall loss    ▹ Eq. (14)
end while
return $\theta_g, \theta_e, \theta_\tau, \theta_n$.
```

C ADDITIONAL DESCRIPTION OF DATASETS

We include more details of the datasets below.

- CollegeMsg [26] is an online social network where private messages were sent and received at the University of California, Irvine. If user i sent a private message to user j at time t, there is a temporal edge (i, j, t). Since the nodes have no feature, we use the one-hot encoding of the node ID as node features.
- cit-HepTh [18] is a citation graph about high energy physics theory from the e-print arXiv, in the period from January 1993 to April 2003. A temporal edge (i, j, t) here means a paper i cites paper j at time t. We use word2vec [24] to convert the text of paper abstract (*i.e.*, the raw node features) into node embedding as the node feature.
- Wikipedia [17] is a graph in which temporal edges are interactions induced by users' editing on the Wikipedia pages in one month. User edits consist of textual features, which are converted into 172-dimensional LIWC [28] feature vectors. The edit vectors of each user are added and normalized to serve as the node feature.
- Taobao [3] is a quite large online purchase network on the e-commerce platform taobao.com. If a user i purchased an item j at time t, there is a temporal edge (i, j, t). Node features are preprocessed embeddings of textual features.

D DETAILS OF TASK SETUP

We describe more details of our main task, namely, temporal link prediction. For each temporal graph, node representations are learnt on the graph consisting of events before time t^{tr}, and we try to predict events on or after t^{tr}. In our experiments, we use all the events before the last time step for training, and test on the events at the last time step. For instance, on the graph cit-HepTh, we train the model only using events before the 78th time step, and we predict the links formed at the 78th time step. At test time, a logistic regression classifier is trained for the downstream task of temporal link prediction. While links formed at the last time step are our positive examples, we further randomly sample an equal number of negative examples (*i.e.*, node pairs which do not form a link at the last time step). We define the feature vector of a candidate triple (i, j, t) as $|\mathbf{h}_i^t - \mathbf{h}_j^t|$ [22].

E ADDITIONAL DESCRIPTION OF BASELINES

We include more details of the baselines below.

(1) *Static approaches*, in which models or node embedding vectors are trained on the static graph formed before the testing time, regardless of the temporal information.

- DeepWalk [30]: a static network embedding method, which regards the random walk sequences as sentences and leverages skip-grams [24] to learn node embeddings.
- Node2vec [7]: another static network embedding method, which generalizes DeepWalk with biased random walks.
- VGAE [15]: based on variational auto-encoder (VAE) [14, 31], it is a classical GNN-based link prediction model, using a GCN [16] encoder and an inner product decoder.
- GAE [15]: a non-probabilistic variant of the VGAE model.
- GraphSAGE [10]: a GNN model on static graphs, which supports inductive representation learning on large graphs.

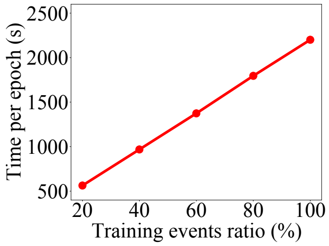


Figure 5: Time complexity.

(2) *Temporal approaches*, which train models or node embedding vectors on the temporal graph formed before the testing time.

- CTDNE [25]: based on random walks, it is a network embedding method which learns time-respecting embedding from continuous-time dynamic networks.
- EvolveGCN [27]: using RNN to evolve GCN parameters to capture the dynamic information of sequences of static graph snapshots.
- GraphSAGE+T: our implementation based on GraphSAGE. Specifically, when aggregating neighbors' information, it will consider the time decay effect, *i.e.*, earlier neighbors will get smaller weights while more recent neighbors will get larger weights during aggregation.
- TGAT [42]: it uses the self-attention mechanism to aggregate temporal-topological neighborhood features. Besides, based on Bochner's theorem, it encodes the event time as part of the node embedding vector.

(3) *Hawkes process-based approaches*, which train node embedding vectors using the temporal graph formed before the testing time, based on Hawkes process.

- HTNE [47]: a network embedding method which integrates the Hawkes process into network embedding so as to capture the influence of historical neighbors on the current neighbors.
- MMDNE [22]: a network embedding method with micro- and macro-dynamics. Specifically, the micro-dynamics describe the link formation process, while the macro-dynamics refer to the evolution pattern of the network scale.

For DeepWalk, Node2vec and CTDNE, we set their random walk sampling parameters, such as number of walks, walk length and window size according to their recommended settings, respectively. For all network embedding methods, the node embedding dimension is 128, which tends to perform well empirically. For all GNN-based methods, we set the number of layers, the node embedding dimensions and the learning rates to the same with our model TREND. For HTNE, MMDNE and EvolveGCN, we set their historical window size to 5 (*i.e.*, only use the 5 most recent neighbors), given the empirical performance and efficiency considerations. For efficiency reasons, we perform random neighborhood sampling on GraphSAGE and GraphSAGE+T, setting the sample size to 10 on CollegeMsg and cit-HepTh, 20 on Wikipedia, and 5 on Taobao (which is the most sparse graph), the same with TREND; we use the same sample size on TGAT, but sample the more recent neighbors with higher probability following its original design. For EvolveGCN, we use the EvolveGCN-O version. For TGAT, we set the number of attention heads to 3. Other hyperparameters are chosen empirically, following guidance from literature.

F SCALABILITY STUDY

On Taobao (with a total of more than 4 million events), we extract different ratios (20%–100%) of training events to form 5 subgraphs, and record the training time per epoch on each subgraph. In Fig. 5, the training time grows linearly in the number of training events, which is consistent with our complexity analysis, and implies that the proposed model is scalable.