Deep Representation Learning for Forecasting Recursive and Multi-Relational Events in Temporal Networks

Tony Gracious, Ambedkar Dukkipati

Department of Computer Science and Automation, Indian Institute of Science, Bangalore - 560012, INDIA
{tonygracious, ambedkar}@iisc.ac.in

Abstract

Understanding relations arising out of interactions among entities can be very difficult, and predicting them is even more challenging. This problem has many applications in various fields, such as financial networks and e-commerce. These relations can involve much more complexities than just involving more than two entities. One such scenario is evolving recursive relations between multiple entities, and so far, this is still an open problem. This work addresses the problem of forecasting higher-order interaction events that can be multi-relational and recursive. We pose the problem in the framework of representation learning of temporal hypergraphs that can capture complex relationships involving multiple entities. The proposed model, *Relational Recursive Hyperedge Temporal Point Process* (RRHyperTPP) uses an encoder that learns a dynamic node representation based on the historical interaction patterns and then a hyperedge link prediction-based decoder to model the occurrence of interaction events. These learned representations are then used for downstream tasks involving forecasting the type and time of interactions. The main challenge in learning from hyperedge events is that the number of possible hyperedges grows exponentially with the number of nodes in the network. This will make the computation of negative log-likelihood of the temporal point process expensive, as the calculation of survival function requires a summation over all possible hyperedges. In our work, we develop a noise contrastive estimation method to learn the parameters of our model, and we have experimentally shown that our models perform better than previous state-of-the-art methods for interaction forecasting.

Code & Datasets — https://tinyurl.com/RRHyperTPP
Extended version — https://arxiv.org/pdf/2404.17943

1 Introduction

With the advancement of the field of geometric deep learning and the availability of more and more datasets that capture complex interactions among entities, there is a new interest in learning representations from higher-order relations in networks (Bronstein et al. 2017, 2021). Until now the focus had been only on hypergraphs, where relations are represented as hyperedges that represent interactions of groups of entities/nodes (Ghoshdastidar and Dukkipati 2017a,b; Lee and Shin 2023). With the availability of richer datasets that show more complex interaction events, learning problems must consider scenarios beyond hypergraphs by incorporating a wide variety of information associated with these interactions. These ideas have been explored in recent works of topology-aware deep learning, where higher-order network structures such as simplicial complexes, cell complexes, and multi-scale topological information are used for improving graph neural networks (Hajij et al. 2023; Horn et al. 2022). Recent works have shown that n-ary facts or multi-relational hyperedges are the natural data representations for Knowledge graphs, and models based on those perform better than that use pairwise multi-relation models (Wen et al. 2016; Fatemi et al. 2020).

Most of these works ignore the temporal aspects of the network, where entities interact and evolve (S. Gupta and Dukkipati 2019). This has been addressed using dynamic network models based on temporal point processes (TPP) and link prediction. The recent works model interactions as instantaneous edges (Kumar, Zhang, and Leskovec 2019; Dai et al. 2016; Trivedi et al. 2019; Cao et al. 2021)/hyperedges (Gracious and Dukkipati 2023) forming between a pair of nodes or a group of nodes, and a generative model is parameterized to model the formation of edges given the history.

However, real-world interactions can be much more complex than just a group of entities interacting. These interactions can exhibit group structure within themselves. For example, a directed hyperedge involves interaction between two groups of entities in the source and destination. Here, one can represent the source and destination as hyperedges, which act as nodes to the interaction hyperedge. In our work, we generalize this recursive group structure property of real-world interactions to incorporate a variable number of groupings. Examples of these types of interactions can be seen in political events, where source and target hyperedges function as nodes within larger hyperedges. Figure 1 illustrates a real-world event where `Putin` delivers a speech at the `EU`. In this context, each source and target hyperedge consists of nodes that are themselves hyperedges, representing the actor, sector, and country. The source and target hyperedges are connected by the relation `SpokedIn`, indicating the action of the source hyperedge when giving a speech, while the relation `InvSpokedIn` represents the inverse re-

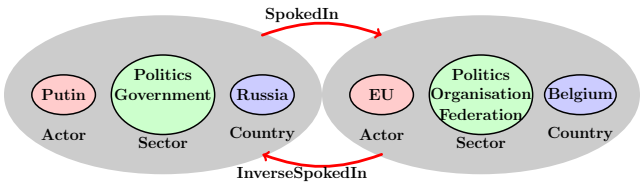

Figure 1: Real world events represented as Relational Recursive Hyperedges. Here, there are source and target hyperedges with three relations inside them and relation types between them to indicate the nature of the interaction.

lation, with the target acting as the source and the source as the target. Similar observations can be made in citation networks, where authors of papers form a hyperedge, with each author's position within the paper represented as nodes, indicating a smaller hyperedge of authors in the same position.

In this paper, we use a multi-relational recursive hyperedges structure to incorporate these complex higher-order interactions. Here, hyperedges can act as nodes in other hyperedges (like CCed or receiver groups in email interactions) and contain relation types indicating their role in the interactions. Then, a TPP is defined with these hyperedges as event types with conditional intensity function parameterized using representations/embeddings of the nodes and relations involved in the interactions. Furthermore, the previous approaches in TPP based interaction forecasting use negative sampling to approximate the loss associated with survival function, which is intractable to calculate due to the huge number of possible event types in the TPP. This can introduce biases in training, especially in these complex interactions. To address these challenges, we propose the model *Relational Recursive Hyperedge Temporal Point Process* (RRHyperTPP). The following are the contributions of this work. 1)We propose a contrastive learning strategy for higher-order interaction for hyperedge events in networks. This provides a technique for training the model without maximum likelihood estimation, thereby avoiding the computationally expensive survival function calculation. We establish that the proposed model performs better than the previous models that use negative sampling; 2) We proposed a new deep learning architecture for learning node representation from higher-order interaction. This involves two stages: interaction update and temporal drift. Interaction update is used to revise the node representation when an event involving it occurs by using the features of the interaction. Temporal drift is used to model the evolution of node representation during the interevent period using time projection based on JODIE (Kumar, Zhang, and Leskovec 2019) or Neural ODE (Chen et al. 2018) or time embeddings based on Fourier time features (Xu et al. 2020); 3) Finally, we propose a new method for hyperedge link prediction for multi-relational recursive hypergraphs; 4) Extensive experiments are done to show the advantage of the proposed method over existing state-of-the-art models.

2 Related Works

The recent works on modeling continuous time dynamic networks are of two kinds: those that are trained using TPP loss and those that use link prediction loss. The models in the first category parameterize the probability density function of a Neural Temporal Point Process with edges as the mark associated with events (Shchur et al. 2021). These involve models DeepCoevolve (Dai et al. 2016), DyReP (Trivedi et al. 2019), DSPP (Cao et al. 2021), and GNPP (Xia, Li, and Li 2022) that uses neural networks to parameterize the conditional intensity function of a TPP based on the nodes involved in the edges. These works mostly focus on interaction forecasting on bipartite and homogeneous networks. More recently, GHNN (Han et al. 2020) defined TPPs on Temporal Knowledge Graphs where entities of different kinds and interactions are associated with relation type. However, all these models are developed for pair-wise interaction forecasting and cannot accurately model the higher-order interactions. Recent work (Gracious and Dukkipati 2023) modeled higher-order interactions as hyperedge event formation in a network. Here, they used recurrent neural network architecture to capture the evolution of nodes with each interaction and used fourier-based time features to model the dynamic of node representation during intervention time. These node representations are then used by a hyperedge-based link predictor to define the conditional intensity function of the TPP. Although this model can capture interactions between any number of nodes, it does not consider the internal groupings of nodes or their relationships with other nodes, such as in email exchanges or co-authorship networks.

The works in the second category use graph representation learning to learn continuous-time representations for the nodes that are then used to predict links using a binary scoring function. This involves models like JODIE (Kumar, Zhang, and Leskovec 2019), TGAT (Xu et al. 2020), and TGN (Rossi et al. 2020) which uses combination of temporal graph attention and recurrent neural networks to learn dynamic node representation. Recently, CAt-Walk (Behrouz et al. 2023) have shown to predict higher-order interaction in the form of hyperedges. Here, they used causal random walks, which maintain the temporal order of interactions to create features that are then used by a set of prediction-based architectures to forecast hyperedges as binary classification problems at the time of interest. However, all these models can only predict the presence of an interaction at a given time and cannot answer temporal queries like when a group of entities will interact or the time of the next interaction.

3 Preliminaries and Problem Statement

Tables 4 and 5 in Appendix A summarize important notations used in this work.

Hypergraph. A Hypergraph is denoted by the tuple $\mathcal{G} = (\mathcal{V}, \mathcal{H})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$ is the set of nodes, $\mathcal{H}$ is the set of hyperedges with $\mathcal{H} \subset \mathcal{C}(\mathcal{V})$ and $\mathcal{C}(\mathcal{V})$ is the powerset of $\mathcal{V}$.

Recursive Hypergraph. A Recursive Hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{H})$ is a generalization of Hypergraph, where hyperedges can contain nodes and other hyperedges. Let, $\mathcal{H}^\ell = \mathcal{C}(\cup_{i=0}^{\ell-1}\mathcal{H}^i)$, if $\ell \geq 1$, where $\mathcal{H}^0 = \mathcal{C}(\mathcal{V})$. A recursive hypergraph is of depth ℓ if $\mathcal{H} \subset \mathcal{H}^\ell$. This work focuses on recursive hypergraphs of depth one and two.

Temporal Multi-relational Recursive Hypergraphs. A Temporal Multi-relational Recursive hypergraph is denoted by tuple $\mathcal{G}(t) = (\mathcal{V}, \mathcal{H}, \mathcal{R}, \mathcal{E}(t))$, where $\mathcal{E}(t) = \{(e_1, t_1), \ldots, (e_n, t_n)\}$ is the ordered set of historical events till time t with $e_n \in \mathcal{H}$ and n is number events occurred and $\mathcal{R}$ is the set of relation. Here, each hyperedge $h \in \mathcal{H}$ is formed of a set of recursive hyperedges, $h = h^\ell = \{(h_1^{\ell-1}, r_1^{\ell-1}), \ldots, (h_{k^{\ell-1}}^{\ell-1}, r_{k^{\ell-1}}^{\ell-1})\}$ of depth ℓ. Here, $r_i^{\ell-1} \in \mathcal{R}$ is the relation of hyperedge $h_i^{\ell-1}$ with respect to hyperedge h^ℓ, and $h_i^{\ell-1} \in \mathcal{H}^{\ell-1}$. Here, $k^{\ell-1}$ is the number of relational groups, or depth $\ell - 1$ hyperedges in hyperedge h, and $\mathcal{E}(t_a, t_b)$ is the ordered set of events observed during the interval $[t_a, t_b]$.

Using the above definition, we can represent a *Directed Temporal Hypergraph* by using relations indicating the source and destination groups, here $|\mathcal{R}| = 2$. Furthermore, including the node type attribute, we can extend this definition to a *Bipartite Temporal Hypergraph* where the nodes set is a union of two types, $\mathcal{V} = \mathcal{U}^r \cup \mathcal{U}^\ell$. Here, $\mathcal{U}^r$ and $\mathcal{U}^\ell$ are the two sets of nodes, and all interactions are between them. To use this framework for *Temporal Knowledge Graph* (TKGs), where interactions are represented as a triplet of the subject entity v^s, object entity v^o, and relation r between them, $h = (v^s, r, v^o))$. Here, $v^s, v^o \in \mathcal{V}$. This can be framed as a Multi-relational Recursive Hypergraph with $h = \{(h_0^0, r_1^0), (h_1^0, r_2^0)\}$. Here, $r_1^0 = r, r_2^0 = r^{-1}$ is the inverse relation, $h_0^0 = \{v^s\}$, and $h_1^0 = \{v^o\}$.

Problem. Given $\mathcal{E}(t) = \{(e_1, t_1), \ldots, (e_n, t_n)\}$ historical interactions until time $t > t_n$, we aim to forecast the next interaction (t_{n+1}, e_{n+1}), where $t_{n+1} > t > t_n$. Here, we want to estimate the time t_{n+1} and the type of interaction e_{n+1}.

This work mainly focuses on higher-order interactions in communications networks and real-world events stored in source and target entities. The higher-order interaction in communications networks is represented using a recursive hyperedge graph of depth one $h = \{(h_i^0, r_i^0))\}_{i=0}^{k^0}$. Here, $\{r_i^0\}_{i=0}^3$ relations are used to represent the sender, recipients, and carbon-copied recipients' addresses in case of email exchange and $\{r_i^0\}_{i=0}^4$ represents the sender, receiver, retweeter, and retweeted nodes in Twitter network. The higher-order interactions in real-world events are represented using depth two hyperedges $h = \{(h_0^1, r_0^1), (h_1^1, r_1^1)\}$ and $h_{0/1}^1 = \{(h_i^0, r_i^0)\}_{i=0}^3$. Here, h_0^1, h_1^1 are the two entities involved in the interactions, r_0^1 and r_1^1 are the inverse relation pair indicating the type of interactions, $\{r_i^0\}_{i=0}^3$ are relations indicating the actor, sector, country of the entity. Figure 1 shows an example of this type of higher-order interaction.

4 RRHyperTPP Model

The probability of the occurrence of hyperedge event h at time t is defined as follows, $\mathrm{P}_h(t) = \lambda_h(t)\mathrm{S}_h(t_h^p, t)$. Here, $\lambda_h(t)$ is the conditional intensity function of the temporal point process, $\mathrm{S}_h(t_h^p, t)$ is the survival function that denotes the probability that the event does not occur during the interval $[t_h^p, t]$, $\mathrm{S}_h(t_h^p, t) = \exp\left(-\int_{t_h^p}^t \lambda_h(\tau)\mathrm{d}\tau\right)$. In this, $t_h^p < t$ is the time of the last occurrence of hyperedge h before time t, and it is initialized to zero for all the hyperedges at the beginning. The complete likelihood for observing $\mathcal{E}(T) = \{(e_1, t_2), \ldots, (e_N, t_N)\}$ in the interval $[0, T]$ can be written as, $\mathrm{P}(\mathcal{E}(T)) = \prod_{n=1}^N \mathrm{P}_{e_n}(t_n) \prod_{h\in\mathcal{H}} \mathrm{S}_h(t_h^\ell, T)$. Here, t_h^ℓ is the last occurrence of hyperedge h with $t^\ell = 0$ for $h \in \mathcal{H}$ that is not observed in the training data.

$$\begin{aligned}\mathrm{P}(\mathcal{E}(T)) &= \prod_{n=1}^N \lambda_{e_n}(t_n)\mathrm{S}_{e_n}(t_{e_n}^p, t_n) \prod_{h\in\mathcal{H}} \mathrm{S}_h(t_h^\ell, T), \\ &= \prod_{n=1}^N \lambda_{e_n}(t_n) \prod_{h\in\mathcal{H}} \mathrm{S}_h(0, T). \end{aligned} \tag{1}$$

For learning the parameters of conditional intensity $\lambda_h(t)$, we calculate the loss by taking a negative log-likelihood as shown below,

$$\mathcal{NLL} = -\sum_{n=1}^N \log \lambda_{e_n}(t_n) + \sum_{h\in\mathcal{H}} \int_0^T \lambda_h(t)\mathrm{d}t. \tag{2}$$

However, doing maximum likelihood estimation (MLE) by minimization of the above loss function is computationally expensive due to the summation over all possible hyperedges in the survival function, which have an exponential number of possibilities, and also due to the integral over the entire duration of observation $\int_0^T(.)$. So, to avoid these difficulties in computing the likelihood, we use the noise contrastive estimation technique of learning multivariate temporal point process (Mei, Wan, and Eisner 2020) as explained in Section 4.1.

Furthermore, defining separate conditional intensity functions for each hyperedge will lead to difficulty in learning, as the number of parameters will increase linearly to the number of hyperedges, which is exponential on the order of the number of nodes, as explained earlier. It leads to overfitting while training and poor generalization in the test set. Hence, we define the conditional intensity function as a positive function of dynamic embeddings nodes involved in it, $\lambda_h(t) = f(h; \mathbf{V}(t), \mathbf{R})$. Here, $\mathbf{V}(t)$ is the dynamic node embedding of nodes at time t, and $\mathbf{R}$ is the relation embeddings. In this model, the number of parameters will be linear to the number of nodes, as hyperedges that share nodes will share the respective parameters. Sections 5 and 6 describe the architecture used to implement dynamic node representation and conditional intensity function, respectively. Figure 5 in Appendix B contains the block diagram of the model.

4.1 Learning and Inference

An alternate approach to MLE for learning parameters is to use noise contrastive estimation (NCE). To achieve this, we

simulate N_q noise streams $\{\mathcal{E}^i(T)\}_{i=1}^{N_q}$ from the noise distribution $\mathrm{Q}(T)$ in addition to observed data $\mathcal{E}(T)$. The complete loss function for the noise contrastive estimation can be written as follows,

$$\mathcal{L}_{NCE} = -\mathbf{E}_{\mathcal{E}(T)\sim \mathrm{P}(.),\{\mathcal{E}^i(T)\}_{i=1}^{N^q}\sim \mathrm{Q}(.)}\Big[\sum_{\mathcal{E}(t,t+\mathrm{d}t)=\{(h,t)\}} \log\frac{\lambda_h(t)}{\underline{\lambda_h}(t)} + \sum_{i=1}^{N^q}\sum_{\mathcal{E}^i(t,t+\mathrm{d}t)=\{(h,t)\}} \log\frac{\lambda_h^q(t)}{\underline{\lambda_h}(t)}\Big]. \quad (3)$$

Here, $\underline{\lambda_h}(t) = \lambda_h(t) + \lambda_h^q(t)N^q$. In the above equation, we contrast the samples from the true distribution, the observed data, to the N^q independently sampled noise stream. Appendix E contains the details of the derivation.

Additionally, to direct the gradients toward better model parameters, we add a classification-based noise contrastive loss at the time of the event as follows,

$$\mathcal{L}^s_{NCE} = \sum_{(e_n,t_n)\in\mathcal{E}(T)} \log\frac{\lambda_{e_n}(t)}{\lambda_{e_n}(t) + \sum_{h\in\mathcal{H}^{neg}_{e_n}}\lambda_h(t)}. \quad (4)$$

Here, $\mathcal{H}^{neg}_{e_n}$ are the negative samples generated by negative sampling for the hyperedge e_n. Then, the combined loss can be written as follows,

$$\mathrm{Loss} = \mathcal{L}_{NCE} + \alpha\mathcal{L}^s_{NCE}. \quad (5)$$

Here, α is a hyperparameter.

Algorithm 1: Training procedure of RRHyperTPP

Input: $\mathcal{G}(T)$.
while not convergence **do**
  $\mathcal{H}^{neg} \sim \mathrm{CorruptionModel}(\mathcal{H})$.
  $\mathcal{H}^c = \mathcal{H} \cup \mathcal{H}^{neg}$.
  Set $t = t_0 = 0$, $n = 1$, $\mathcal{L}_{NCE} = 0$.
  for $n \le N$ **do**
    while $t < t_n$ **do**
      Sample $\lambda^q(t) \sim \frac{1}{Nw}\sum_{i=1}^{N}\mathcal{K}(\lambda^q - \frac{1}{t_i - t_{i-1}}; w)$.
      Sample $\Delta t \sim \mathrm{exponenetial}(N^q\lambda^q(t))$.
      Sample $h \sim \mathrm{Uniform}(\mathcal{H}^c)$.
      Next event time $t = t + \Delta t$.
      $\mathrm{Loss} = \mathrm{Loss} - \log\frac{\lambda_h^q(t)}{\underline{\lambda_h}(t)}$
    end while
    $\mathcal{H}^{\mathrm{neg}}_{e_n} \sim \mathrm{CorruptionModel}(e_n)$
    $\mathrm{Loss} = \mathrm{Loss} - \log\frac{\lambda_{e_n}(t)}{\underline{\lambda_{e_n}}(t)}$
    $\mathrm{Loss} = \mathrm{Loss} - \alpha\log\frac{\lambda_{e_n}(t)}{\lambda_{e_n}(t) + \sum_{h\in\mathcal{H}^{\mathrm{neg}}_{e_n}}\lambda_h(t)}$
    $n = n + 1$.
    If $n \mod B = 0$, Update the model parameters by AdamW Optimizer and set $\mathrm{Loss} = 0$.
  end for
end while

4.2 Noise Distribution

Here, we propose a method for simulating noise sequences by modeling the noise distribution $\mathrm{Q}(.)$. We model the rate of events λ^q using a stochastic process independent of the event type and time to reduce the computational complexity. The probability density function of this process is learned from the observed event times in the training data $\mathcal{E}(T)$ using kernel density estimation (Silverman 1986), $\lambda^q \sim \frac{1}{Nw}\sum_{i=1}^{N}\mathcal{K}\left(\lambda^q - \frac{1}{t_i - t_{i-1}}; w\right)$. Here, $\mathcal{K}$ is the kernel, and w is the bandwidth of the kernel. We use the Gaussian kernel, and bandwidth is learned from the dataset. This will allow for a closed-form sampling of noise event times. Then, we generate the type of event by uniformly sampling from a set of candidate hyperedges $\mathcal{H}^c$. Then $\lambda_h^q = \frac{1}{|\mathcal{H}^c|}\lambda^q$ and $\mathrm{Q}(\mathcal{E}^i(t, t+\mathrm{d}t) = \{(h,t)\}) = \frac{1}{|\mathcal{H}^c|}\lambda^q\exp\left(-\lambda^q(t - t_n)\right)$. The set of candidate hyperedges, denoted as $\mathcal{H}^c$, is formed by combining the true hyperedges observed in the training data with the negative hyperedges generated through the corruption of observed hyperedges. For depth one hyperedge datasets, we expand the hyperedge as a tuple of node and relation pairs in ascending order. Then, we learn the categorical distribution of each position condition on all the previous relations, along with an end state. For each positive hyperedge, we first sample the relations till the end state is observed. Subsequently, we randomly populate each position in the negative hyperedge with nodes. This is achieved by populating each position with nodes from the true hyperedge or by randomly selecting nodes, ensuring no repetition if they share the same relation value. The process ensures that at most half of the nodes in the negative hyperedges are from the true hyperedge, with the remainder filled randomly, thus maintaining diversity in the negative samples. For depth two hyperedge datasets, we randomly corrupt any one of the depths one hyperedge within them by the above procedure to generate negative hyperedges. Here, for each observed hyperedge $h \in \mathcal{H}$, we generate $N^e = |\mathcal{H}_h^{\mathrm{neg}}|$ negative hyperedges. For generating N^q noise streams, we multiply the noise distribution intensity function by N^q, $\lambda^q(t)N^q$, and simulated samples. This scaling will not affect NCE loss Equation 3, as all the noise streams have the same intensity values.

Algorithm 1 summarizes the entire training procedure of the model. The $\mathrm{CorruptionModel}(.)$ is the negative hyperedge generation function, and $\mathrm{Uniform}(\mathcal{H}^c)$ uniformly samples a hyperedge from the candidate hyperedges, $\mathcal{H}^c$. In the following section, we explain the dynamic node representation learning used to parameterize the conditional intensity function of the model. In our implementation, we use the same negative samples to generate the noise stream, and in the supervised noise contrastive loss in Equation 4, $\mathcal{H}^n = \cup_{h\in\mathcal{H}}\mathcal{H}_h^{\mathrm{neg}}$.

5 Dynamic Node Representation

The dynamic node representation of the nodes $\mathbf{V} \in \mathbb{R}^{|\mathcal{V}|\times d}$ in the networks are learned using a continuous time recurrent-neural network model using two stages, (i) Node Update and (ii) Drift, for node evolution.

5.1 Node Update

This stage is used to update the representation of a node v when it is involved in an interaction $h = \{(h_1^{\ell-1}, r_1^{\ell-1}), \ldots, (h_{k^{\ell-1}}^{\ell-1}, r_{k^{\ell-1}}^{\ell-1})\}$ at time t. The update equation uses a recurrent neural network-based architecture that updates the node's previously stored embedding based on the features of interaction calculated as follows,

$$\mathbf{i}_v^h = \text{MLP}\left(\left[\mathbf{d}_v^h; \mathbf{v}(t^-); \mathbf{\Psi}(t - t_v^p)\right]\right)$$
$$\mathbf{v}(t^+) = \text{RNN}\left(\mathbf{v}(t_v^p), \mathbf{i}_v^h\right). \quad (6)$$

Here, $v \in h^0, (h^0, r^0) \in \ldots \in h$, MLP is a two-layer multi-layer perceptron. Appendix C explains the architecture of the MLP layer. Here, $\mathbf{d}_v^h \in \mathbb{R}^d$ is the dynamic embedding calculated as a function of embeddings of other nodes involved in the interaction, $\mathbf{d}_v^h = f_{dyn}(h; \mathbf{V}(t), \mathbf{R})$. The architecture of $f_{dyn}(\cdot)$ is shared with the hyperedge link predictor explained in Section 6. The second term $\mathbf{\Psi}(t - t_v^p) \in \mathbb{R}^d$ is the Fourier features for the duration since an event was observed on v, $(t - t_v^p)$. Here, t_v^p is the recent time node v is involved in an interaction, and Fourier time features (Xu et al. 2020) are defined as $\mathbf{\Psi}(t) = [\cos(\omega_1 t + \phi_1), \ldots, \cos(\omega_d t + \phi_d)]$, where $\{\omega_i\}_{i=1}^d$, and $\{\phi_i\}_{i=1}^d$ are learnable parameters. When v is involved in multiple positions in hyperedge h, an average of $\mathbf{i}_v^h$ is taken to update the node embedding.

5.2 Drift

This stage is used to model the evolution of the nodes during the inter-event period and avoid the staleness of node embeddings due to not observing any event for a long duration (Kazemi et al. 2020).

Time Projection. Here, we project the node embeddings from the node update stage based on the elapsed time since the last update as follows,

$$\mathbf{v}(t) = (1 + \mathbf{W}_t(t - t_v^p)) \circ \mathbf{v}(t_v^p). \quad (7)$$

Here, $\mathbf{W}_t \in \mathbb{R}^{d \times 1}$ is a learnable parameter, and $\circ$ denotes the Hadamard product. This version of the embedding projection is introduced in the work JODIE (Kumar, Zhang, and Leskovec 2019).

Time Embeddings. Here, we use Fourier time features to model the evolution of the nodes during the interevent time as follows,

$$\mathbf{v}(t) = \tanh(\mathbf{W}_s \mathbf{v}(t_v^p) + \mathbf{W}_t \mathbf{\Psi}(t - t_v^p)). \quad (8)$$

Here, $\mathbf{W}_s, \mathbf{W}_t \in \mathbb{R}^{d \times d}$ are learnable parameters. This form of drift stage is followed in previous work by Gracious and Dukkipati (2023).

Neural ODE. The node embedding evolution is modeled using a neural Ordinary Differential Equation (ODE) (Chen et al. 2018) as follows,

$$\mathbf{v}(t) = \mathbf{v}(t_v^p) + \int_{t_v^p}^{t} f_{\nabla \mathbf{v}}\left(\mathbf{v}(t), \mathbf{\Psi}(t - t_v^p)\right) \mathrm{d}t. \quad (9)$$

Here, $f_{\nabla \mathbf{v}}$ is the gradient of the node embedding $\mathbf{v}(t)$ implemented using a two layer multi-layer neural network. Appendix C explains the architecture of the MLP layer.

6 Hyperedge Link Prediction

Given the node embeddings and relation embeddings of a hyperedge, we use the query-key-value attention mechanism used in transformer models (Vaswani et al. 2017) for hyperedge link prediction. For this, we first expand hyperedges at depth 0 as a tuple of node and relation pair and create an expanded depth one hyperedge $\bar{h}^0 = \{(v, r^0)\}_{v \in h^0, (h^0, r^0) \in h^1}$. Here, the relation is repeated for each node in the hyperedges. Then we create embeddings for query, key, and value, $\{\mathbf{v}_i^{\text{q}}(t) = \mathbf{v}_i^{\text{v}}(t) = [\mathbf{v}_i(t); \mathbf{r}_i^0]\} \forall (v_i, r_i^0) \in \bar{h}^0$, then we calculate the dynamic hyperedge embedding (here, dynamic is used to indicate the hyperedge dependent embeddings $\mathbf{d}_*^*$'s) for each node as follows,

$$\mathbf{d}_v^{\bar{h}^0} = \text{MHAtt}(\{\mathbf{v}^{\text{q}}(t)\}, \{\mathbf{v}_j^{\text{k}}(t)\}_{j=1}^{\bar{k}^0}, \{\mathbf{v}_j^{\text{v}}(t)\}_{j=1}^{\bar{k}^0}). \quad (10)$$

Here, $\text{MHAtt}(\cdot)$ is the MultiHeadAttention architecture proposed by Vaswani et al. (2017). Based on these, we create the depth one hyperedge embeddings by averaging out all the above node embeddings $\mathbf{h}_i^1 = \sum_{v \in \bar{h}_i^0} \frac{\mathbf{d}_v^{\bar{h}_i^0}}{|\bar{h}_i^0|}$. Then, for the hyperedge at depths $2 \leq \ell \leq l$, we apply the following self-attention layer with query, key, and value as $\{\mathbf{v}_i^{\text{q}}(t) = \mathbf{v}_i^{\text{k}}(t) = [\mathbf{h}_i^{\ell-1}(t); \mathbf{r}_i^{\ell-1}]\}$, for all$(h_i^{\ell-1}, r_i^{\ell-1}) \in h^\ell$,

$$\mathbf{d}_{h_i^{\ell-1}}^{h^\ell} = \text{MHAtt}\left(\left\{\mathbf{v}_i^{\text{q}}(t)\right\}, \left\{\mathbf{v}_j^{\text{k}}(t)\right\}_{j=1}^{k^{\ell-1}}, \left\{\mathbf{v}_j^{\text{v}}(t)\right\}_{j=1}^{k^{\ell-1}}\right),$$

$$\mathbf{h}^\ell = \sum_{(h_i^{\ell-1}, r_i^{\ell-1}) \in h^\ell} \frac{\mathbf{d}_{h_i^{\ell-1}}^{h^\ell}}{|h^\ell|}. \quad (11)$$

Then conditional density is parameterized by the difference between the hyperedge embedding and dynamic hyperedge embedding of the $\ell - 1$th layer as follows,

$$o_i^h = \mathbf{W}_o(\mathbf{h}_i^{\ell-1} - \mathbf{d}_{h_i^{\ell-1}}^{h^\ell})^2 + b_o$$
$$\lambda_h(t) = \text{Softplus}\left(\sum_{(h_i^{\ell-1}, r_i^{\ell-1}) \in h} o_i^h\right). \quad (12)$$

The dynamic hyperedge embedding used in the interaction update is defined as follows:

$$\mathbf{d}_v^h = [\mathbf{d}_v^{h^0}; \mathbf{d}_{h^0}^{h^1}; \ldots; \mathbf{d}_{h^{\ell-1}}^{h^\ell}; \mathbf{h}^\ell],$$

here $v \in h^0, h^0 \in h^1, \ldots, h^{\ell-1} \in h$.

7 Experimental Settings and Results

Datasets. Table 1 shows the vital statistics of the datasets used in this work. These datasets are created from the works of Chodrow and Mellor (2019); Domenico and Altmann (2020); Omodei, De Domenico, and Arenas (2015); Boschee et al. (2015). More details of the datasets and preprocessing are provided in Appendix D.

Baselines. TGN (Rossi et al. 2020) a state-of-the-art pairwise interaction prediction model, HGDHE (Gracious and Dukkipati 2023) a state-of-the-art hyperedge interaction forecasting model, and we created the model

Datasets	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}(T)\rvert$	$\lvert\mathcal{R}\rvert$	T	depth (l)
Enron	98	10,355	3	10,443.0	1
Twitter	2,714	52,383	4	54,028	1
Boston	2,400	20,070	4	20,069	1
Obama	1,721	22,690	4	22,689	1
Pope	6,648	67,779	4	67,778	1
Cannes	672	9,078	4	9,077	1
ICEWS-India	1,066	86,609	400	364	2
ICEWS-Nigeria	894	69,433	360	715	2

Table 1: Temporal Multi-Relational Recursive Hypergraphs datasets and their vital statistics.

RRHyperTPP-N to show the advantage of our training strategy over negative sampling explored for training HGDHE. In RRHyperTPP-N, we use time embeddings for drift to make the model closer to the one proposed in their work.

Forecasting Tasks And Evaluation Metrics. We use two tasks, (i) Interaction type prediction and (ii) Interaction duration Prediction, to evaluate our models' performance. In the first task, we predict which type of hyperedge occurs at time t given the history. This can be estimated by finding the hyperedge with maximum conditional intensity function, $\hat{h} = \arg\max_h \lambda_h(t)$. We evaluate it using the Area Under the Curve (AUC) metric (Fawcett 2006) by using the $\mathrm{CorruptionModel}$ to create N^e false positive samples for each true positive sample. For the second task, we try to predict the time at which interaction h occurs from the last interaction duration of nodes in the hyperedge t_h^p. This is to be estimated using the condition probability density as $\hat{t}_i = \int_{t_h^p}^{\infty} t \mathrm{P}_h(t|\mathcal{E}(t_h^p))\mathrm{d}t$. Then, for evaluation, we use Mean Absolute Error (MAE) metric, $\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{t}_i - t_i|$ for all the samples in the test.

Parameter Settings. In all our experiments, we fixed the embedding dimension d, hidden and input dimension for the RNN at 64, batch size $B = 128$, number of noise streams $N^q = 20$, number of corrupted hyperedges per true hyperedge $N^e = 20$, and number heads of MHAtt to four. The training is done for 200 epochs, and the model parameters that gave the least validation loss are used for testing. For all the training, we use the AdamW optimizer (Reddi, Kale, and Kumar 2018; Loshchilov and Hutter 2019) of PyTorch (Kingma and Ba 2015) with learning rate set to 0.0005. The first 50% of interactions is used for training, the next 25% for validation, and the rest for testing.

7.1 Results

Table 2 shows the performance of our models RRHyperTPP against baseline models. Here, the models that use time projection, ODEs, and time embeddings for **Drift** stage are denoted by -j, -o, and -f at the end, respectively. Here, we can also observe that no single model outperforms the rest in all the tasks. However, theoretically, Neural ODEs should perform better than others as they have fewer assumptions when compared to Fourier and time projection-based methods. We have also done experiments on depth two hyperedges, and the results are shown in Table 3.

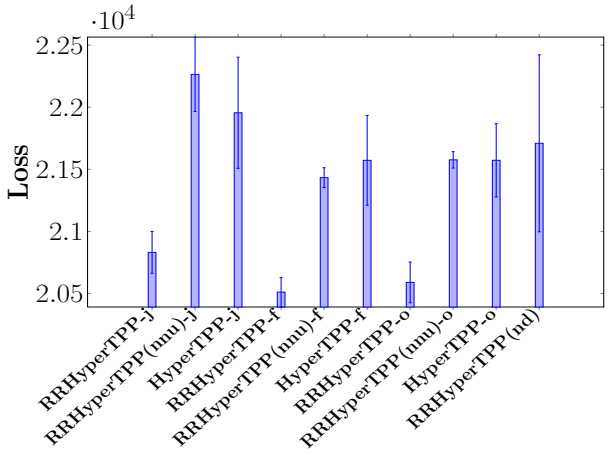

Figure 2: Loss for RRHyperTPP and its variants on Enron dataset. We can observe that the proposed RRHyperTPP models perform better than their variants.

Moreover, when comparing our models against TGN, a state-of-the-art pairwise interaction forecasting model, we can observe an increment of 2.9%, 2.3%, and 2.2% for RRHyperTPP-j, RRHyperTPP-f, and RRHyperTPP-o, respectively, in interaction type prediction task. Hence, one can conclude that recursive hyperedge-based models perform better than pairwise link prediction models. Similarly, we compared our model RRHyperTPP-f against the HGDHE state-of-the-art hyperedge forecasting model, as both use time embeddings for the drift stage. Here, we can observe a gain of 13.6% in AUC for interaction type prediction. We can observe similar improvements for other types of drift compared to the HGDHE model.

To show the advantage of our training strategy over negative sampling-based loglikelihood approximation followed in the previous work by Gracious and Dukkipati (2023), we compare models RRHyperTPP-f against RRHyperTPP-N. Here, both models use the same architecture, but the former uses noise-contrastive loss for training, and the latter uses negative sampling based on approximate negative log-likelihood. There is an average improvement of 8.2% in AUC for RRHyperTPP-f models compared to RRHyperTPP-N. However, there is an average increase of 15.3% in MAE for RRHyperTPP-f compared to RRHyperTPP-N. Hence, more exploration is needed to find a better noise generation strategy to achieve better performance simultaneously in both tasks. Further, RRHyperTPP-N has an improvement of 4.9% in AUC and 9.8% reduction in MAE over previous work HGDHE, which uses hyperedge to represent higher-order interaction. Hence, we can conclude that multi-relational recursive provides better representation for higher-order interactions than hyperedges.

Ablation Studies. To evaluate the efficiency of the dynamic node representation architecture described in Section 5, we conduct a series of ablation studies by systematically removing components of the model. We define the RRHy-

Methods	Enron		Twitter		Boston		Obama		Pope		Cannes	
	AUC	MAE	AUC	MAE	AUC	MAE	AUC	MAE	AUC	MAE	AUC	MAE
TGN[a]	85.8 ± 2.9	NA	97.0 ± 0.5	NA	89.3 ± 1.5	NA	93.0 ± 1.0	NA	92.5 ± 1.2	NA	87.6 ± 2.4	NA
HGDHE[b]	87.6 ± 0.8	3.6 ± 0.1	88.0 ± 1.1	26.26 ± 0.5	75.3 ± 1.3	49.16 ± 5.2	82.0 ± 0.1	29.8 ± 0.8	79.1 ± 0.3	33.6 ± 1.2	79.1 ± 1.0	27.9 ± 2.0
RRHyperTPP-N	91.0 ± 0.9	3.5 ± 0.0	94.2 ± 0.6	**21.9 ± 0.4**	80.8 ± 0.2	**39.9 ± 0.1**	85.2 ± 0.4	**27.6 ± 0.6**	82.5 ± 0.5	**31.9 ± 0.3**	81.9 ± 1.0	**25.4 ± 1.8**
RRHyperTPP-j	93.3 ± 0.7	**3.3 ± 0.2**	98.5 ± 0.1	30.6 ±1.8	**89.2 ± 0.1**	45.7 ± 0.4	**93.5 ± 0.5**	32.5 ± 0.4	94.2 ± 0.4	**36.2 ± 1.0**	92.1 ± 0.4	25.6 ± 1.5
RRHyperTPP-f	**93.4 ± 0.3**	3.5 ± 0.2	**98.4 ± 0.1**	32.3 ± 0.5	89.2 ± 0.2	45.1 ± 0.4	92.9 ± 0.3	31.8 ± 0.3	**93.9 ± 0.3**	37.6 ± 0.2	89.3 ± 0.4	25.2 ± 0.9
RRHyperTPP-o	93.5 ± 0.2	3.7 ± 0.1	98.3 ± 0.1	30.1 ± 0.9	88.0 ± 1.7	46.3 ± 3.1	92.3 ± 1.0	35.6 ± 2.0	**94.0 ± 0.7**	35.4 ± 1.2	**91.2 ± 0.7**	28.6 ± 0.6

Table 2: Performance on interaction type and duration prediction tasks on hyperedges of depth one. Here, interaction type prediction is evaluated using AUC in %, and interaction duration prediction is evaluated using MAE. The proposed model RRHyperTPP beats baseline models in almost all settings. Here, -j, -f, and -o indicate the drift stage used. Citations: [a] (Rossi et al. 2020), [b] (Gracious and Dukkipati 2023)

Methods	ICEWS-India		ICEWS-Nigeria	
	AUC	MAE	AUC	MAE
RRHyperTPP-j	**58.7 ± 0.7**	**0.99 ± 0.0**	61.0 ± 0.3	**0.98 ± 0.0**
RRHyperTPP-f	58.9 ± 0.2	0.75 ± 0.2	60.5 ± 0.5	0.81 ± 0.1
RRHyperTPP-o	58.2 ± 1.4	0.99 ± 0.0	**58.7 ± 0.3**	0.97 ± 0.0

Table 3: Performance on interaction type and duration prediction tasks on hyperedges of depth two.

perTPP(nnu) baseline by excluding the **Node Update** stage while retaining the **Drift** stage. This results in the following variants: RRHyperTPP(nnu)-j/f/o. Additionally, we create the RRHyperTPP(nd) variant by removing **Drift** stage while keeping **Node Update**. Figure 2 presents the performance of these models on the Enron dataset. The results indicate that the RRHyperTPP models achieve an average 5.1% reduction in loss compared to the RRHyperTPP(nnu) models that exclude the **Node Update** stage. This demonstrates that incorporating the **Node Update** stage enhances performance by enabling the dynamic node representation to evolve with each new interaction. Furthermore, the RRHyperTPP(nd) model, which excludes **Drift** stage, exhibits a 4.9% higher loss compared to the proposed model. This underscores the importance of the **Drift** stage in dynamic node representation, as it aids in modeling the evolution of node representations during interevent periods, as discussed in Section 5.2. Furthermore, to show the advantage of the recursive hyperedge link prediction in Section 6, we created baseline models HyperTPP-j/f/o to forecast hyperedges created from the nodes involved in the interactions, $h = \{v_i; v_i \in h^0, (h^0, r^0) \in \ldots \in h\}$. Here, we can observe a 4.8% reduction in Loss for RRHyperTPP models when compared to HyperTPP-based models in interaction-type prediction tasks. Similar trends are observed across other datasets, as detailed in Appendix G.

Performance gain due to supervised NCE. Figure 3 illustrates the performance improvement in AUC of RRHyperTPP-f with different weights of the supervised noise contrastive loss in Equation 5 for the interaction type prediction task. Here, it can be observed that when $\alpha = 1$, there is a 1.7% increase in AUC compared to $\alpha = 0$. Therefore, we infer that the supervised noise contrastive loss enhances performance. Additionally, it can be noted that when α is increased to two, there are no significant improvements, and a performance decrement is observed for the Cannes dataset. Consequently, we can conclude that increasing α will not yield better performance. Furthermore, we observed no significant improvement for the interaction duration prediction task by changing α.

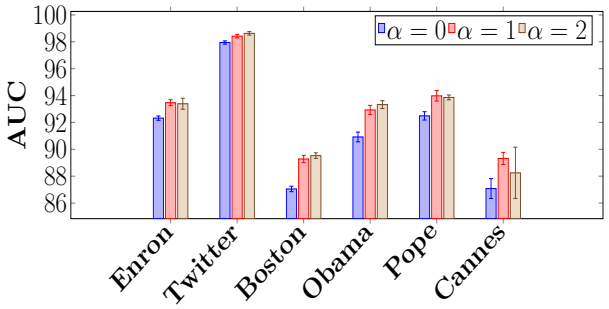

Figure 3: The performance gain obtained due to the addition of classification-based noise contrastive loss mentioned Equation 5. Here, we can observe that models trained with supervised noise contrastive loss $\alpha > 0$ have more AUC scores for interaction type prediction than models that do not ($\alpha = 0$).

8 Conclusion

In this work, we have established the significance of using the multi-relational recursive hyperedges for interaction modeling as it captures real-world events closely. To achieve this, we proposed a dynamic node representation learning technique and link predictor framework that captures complex relationships to forecast events. In addition, we also proposed a noise contrastive learning framework that avoids the integration calculation in the survival function and can train when the number of types of events is of exponential order. To evaluate the model, we curated six datasets of depth one and two datasets of depth two hyperedges. We observed that our model RRHyperTPP performs better than the previous state-of-the-art graph pairwise link prediction model TGN and hyperedge prediction model HGDHE.

Acknowledgements

The authors would like to thank the SERB, Department of Science and Technology, Government of India, for the generous funding towards this work through the IMPRINT Project: IMP/2019/000383.

References

Behrouz, A.; Hashemi, F.; Sadeghian, S.; and Seltzer, M. 2023. CAT-Walk: Inductive Hypergraph Learning via Set Walks. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Boschee, E.; Lautenschlager, J.; O'Brien, S.; Shellman, S.; Starz, J.; and Ward, M. 2015. ICEWS Coded Event Data.

Bronstein, M. M.; Bruna, J.; Cohen, T.; and Veličković, P. 2021. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*.

Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P. 2017. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42.

Cao, J.; Lin, X.; Cong, X.; Guo, S.; Tang, H.; Liu, T.; and Wang, B. 2021. Deep structural point process for learning temporal interaction networks. In *Machine Learning and Knowledge Discovery in Databases. Research Track*, 305–320.

Chen, R. T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, volume 31.

Chodrow, P.; and Mellor, A. 2019. Annotated hypergraphs: Models and applications. *Applied Network Science*, 5: 1–25.

Dai, H.; Wang, Y.; Trivedi, R.; and Song, L. 2016. Deep coevolutionary network: Embedding user and item features for recommendation. arXiv:1609.03675.

Domenico, M. D.; and Altmann, E. G. 2020. Unraveling the origin of social bursts in collective attention. *Scientific Reports*, 10: 4629.

Fatemi, B.; Taslakian, P.; Vazquez, D.; and Poole, D. 2020. Knowledge Hypergraphs: Prediction Beyond Binary Relations. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, 2191–2197.

Fawcett, T. 2006. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8): 861–874.

Ghoshdastidar, D.; and Dukkipati, A. 2017a. Consistency of Spectral Hypergraph Partitioning under Planted Partition Model. *The Annals of Statistics*, 45(1): 289–315.

Ghoshdastidar, D.; and Dukkipati, A. 2017b. Uniform Hypergraph Partitioning: Provable Tensor Methods and Sampling Techniques. *The Journal of Machine Learning Research*, 18(50): 1–41.

Gracious, T.; and Dukkipati, A. 2023. Dynamic representation learning with temporal point processes for higher-order interaction forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 7748–7756.

Hajij, M.; Zamzmi, G.; Papamarkou, T.; Miolane, N.; Guzmán-Sáenz, A.; Ramamurthy, K. N.; Birdal, T.; Dey, T. K.; Mukherjee, S.; Samaga, S. N.; et al. 2023. Topological deep learning: Going beyond graph data. arXiv:2206.00606.

Han, Z.; Ma, Y.; Wang, Y.; Günnemann, S.; and Tresp, V. 2020. Graph Hawkes Neural Network for Forecasting on Temporal Knowledge Graphs. In *Automated Knowledge Base Construction*.

Horn, M.; Brouwer, E. D.; Moor, M.; Moreau, Y.; Rieck, B.; and Borgwardt, K. 2022. Topological graph neural networks. In *International Conference on Learning Representations*.

Kazemi, S. M.; Goel, R.; Jain, K.; Kobyzev, I.; Sethi, A.; Forsyth, P.; and Poupart, P. 2020. Representation learning for dynamic graphs: A survey. *The Journal of Machine Learning Research*, 21(1): 2648–2720.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for stochastic optimization. In *International Conference on Learning Representations*.

Kumar, S.; Zhang, X.; and Leskovec, J. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Lee, D.; and Shin, K. 2023. I'm Me, We're Us, and I'm Us: Tri-directional Contrastive Learning on Hypergraphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

Mei, H.; Wan, T.; and Eisner, J. 2020. Noise-contrastive estimation for multivariate point processes. *Advances in Neural Information Processing Systems*, 33: 5204–5214.

Omodei, E.; De Domenico, M.; and Arenas, A. 2015. Characterizing interactions in online social networks during exceptional events. *Frontiers in Physics*, 3: 59.

Reddi, S. J.; Kale, S.; and Kumar, S. 2018. On the Convergence of Adam and Beyond. In *International Conference on Learning Representations*.

Rossi, E.; Chamberlain, B.; Frasca, F.; Eynard, D.; Monti, F.; and Bronstein, M. 2020. Temporal graph networks for deep learning on dynamic graphs. arXiv:2006.10637.

S. Gupta, G. S.; and Dukkipati, A. 2019. A generative model for dynamic networks with applications. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Shchur, O.; Türkmen, A. C.; Januschowski, T.; and Günnemann, S. 2021. Neural Temporal Point Processes: A Review. In *IJCAI 2021*.

Silverman, B. W. 1986. *Density estimation for statistics and data analysis*, volume 26. CRC press.

Trivedi, R.; Farajtabar, M.; Biswal, P.; and Zha, H. 2019. DyRep: Learning Representations over Dynamic Graphs. In *International Conference on Learning Representations*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Wen, J.; Li, J.; Mao, Y.; Chen, S.; and Zhang, R. 2016. On the representation and embedding of knowledge bases beyond binary relations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 1300–1307.

Xia, W.; Li, Y.; and Li, S. 2022. Graph neural point process for temporal interaction prediction. *IEEE Transactions on Knowledge and Data Engineering*, 35: 4867–4879.

Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2020. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*.