Representation Learning for Dynamic Hyperedges

Tony Gracious , **Ambedkar Dukkipati**
Department of Computer Science and Automation, Indian Institute of Science Bangalore
{tonygracious, ambedkar}@iisc.ac.in

Abstract

Recently there has been a massive interest in extracting information from interaction data. Traditionally this is done by modeling it as pair-wise interaction at a particular time in a dynamic network. However, real-world interactions are seldom pair-wise; they can involve more than two nodes. In literature, these types of group interactions are modeled by hyperedges/hyperlinks. The existing works for hyperedge modeling focused only on static networks, and they cannot model the temporal evolution of nodes as they interact with other nodes. Also, they cannot answer temporal queries like which type of interaction will occur next and when the interaction will occur. To address these limitations, in this paper, we develop a temporal point process model for hyperlink prediction. Our proposed model uses dynamic representation techniques for nodes to model the evolution and uses this representation in a neural point process framework to make inferences. We evaluate our models on five real-world interaction data and show that our dynamic model has significant performance gain over the static model. Further, we also demonstrate the advantages of our technique over the pair-wise interaction modeling technique.

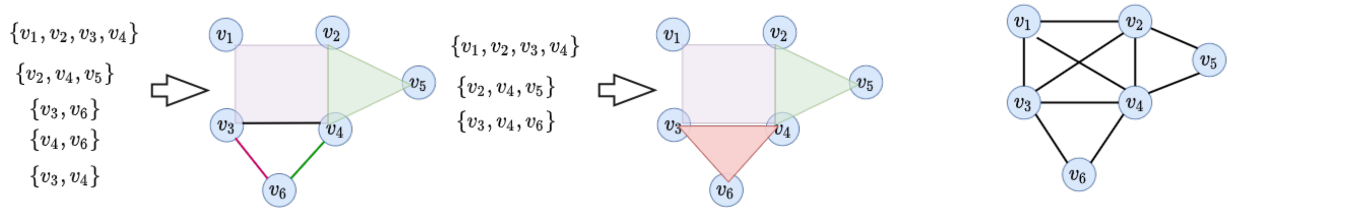
1 Introduction

Information about temporal interactions between entities helps in extracting meaningful information about them. For example, how a person interacts in social media can provide information about that person's preferences, and it can help in recommending items to that the person. Similarly, an e-commerce website can better guess the users' needs if it efficiently extracts information from users' consumption history. Previous techniques [Nguyen *et al.*, 2018; Kumar *et al.*, 2019] use representation/embedding learning in dynamic networks to model these interactions. They model interactions as an instantaneous link/edge between two nodes formed at the time of interaction.

Representation learning in dynamic networks involves learning a latent low-dimensional representation for the nodes. They achieve this by defining node embeddings as a function of time and making them similar for interacting nodes at the time of the interaction [Nguyen *et al.*, 2018; da Xu *et al.*, 2020; Kumar *et al.*, 2019]. Even though these approaches can infer the type of interaction that occurs at a particular time, they cannot infer the duration to wait for it to occur. For this, Temporal Point Processes (TPP) [Daley and Vere-Jones, 2003] have been introduced for modeling edge formation in dynamic networks. TPPs are stochastic processes that model localized events in time, and these events can be of multiple types. To model dynamic networks, one uses each edge as an event type, and a probability distribution is defined over the time of its formation. Here, the probability distribution is parameterized using an intensity function based on representations of nodes. These node representations are functions of time and past interaction events. The parameters of these functions are learned by minimizing the negative log-likelihood on the interactions in the training data.

However, most real-world interactions are more complex than just pairwise interactions. For example, a person can have multiple items in a single shopping order, a group of people can coauthor an article, mutual funds have stocks of various companies, and so on. A common technique that is used in this case is to approximate these multiway interactions with pairwise interactions. That amounts to approximating hypergraphs with graphs and that leads to enormous information loss. This is demonstrated in Figure 1, where two different kinds of interaction between nodes $\{v_1, v_2, v_3, v_4, v_5, v_5\}$ have the same pairwise interaction graph. Further, it is impossible to infer back the original interactions once they are projected into a pairwise graph. Hence, it is crucial to model higher-order interactions as hypergraph edges as hyperedges/hyperlinks involve multiple nodes. Recently, there have been many theoretical advances have taken place for analysing hypergraphs latent structure [Ghoshdastidar and Dukkipati, 2017a; Ghoshdastidar and Dukkipati, 2017b]. In representation learning, previous works [Bai *et al.*, 2021] on hypergraphs use graph neural network-based techniques, and they work only in supervised or semi-supervised settings. Recent works [Huang *et al.*, 2019; Zhang *et al.*, 2019] have introduced training on link prediction loss for representation learning.

So, to address the problem of predicting hyperedge formation events using the TTP framework, in this paper, we



(a) Hyperedges at time t

(b) Hyperedges at time t

(c) Projected pairwise graph for Hypergraphs in 1a and 1b

Figure 1: Higher-order interactions at time t are shown as hyperedges in Figures 1a and 1b. Here, hyperedges are represented by geometric shapes with their ends/corners showing the nodes and color showing their identity. We can see two different hypergraphs having the same projected graph in Figure 1c. So, we need a technique that predicts hyperedges without projecting them to a pairwise graph.

answer the following questions. (i) How to define the conditional intensity for hyperedges? (ii) How to find node representations based on hyperedge events?

To answer the first question of defining the conditional intensity function for TPP, one needs to consider that the interaction can have a variable number of nodes. Hence, the previous techniques developed for uniform hyperlink prediction [Huang *et al.*, 2019] cannot be applicable in this scenario. In our work, we use Hyper-SAGNN [Zhang *et al.*, 2019] based architecture for finding the intensity function of a TPP. This model takes in the node embeddings of the hyperedge as input and outputs the conditional intensity of an event.

Now to answer the second question that deals with finding good representations, we develop the following strategy. As nodes interact, they evolve, so one needs to have a representation that evolves with time. Earlier works on dynamic networks, node embeddings are updated based on the node embedding of the other node in the interaction. For example, consider edge event (v_a, v_b) happening at time t. To update node embeddings of v_a, we use the node embeddings of v_b and vice versa. However, hyperedge events have a variable number of nodes (Figure 1), so the techniques developed for pair-wise interaction are not directly applicable for higher-order interaction. We use a self-attention-based encoding for node update with parameters shared with the hyperlink prediction model 4.2 to solve this. Finally, we present model **Dynamic Hyperedge (DHE)** to model higher-order interactions as hyperedge events in a dynamic Hypergraph. The following are the main contributions of our work,

- A temporal point process framework for hyperedge modeling that can forecast type and time of interaction.
- A model for representation learning for higher-order interaction data.
- Extensive experiments on real-world datasets.
- Empirical results on performance gain obtained when we use hyperedges instead of pair-wise modeling.
- Empirical results on performance gain obtained when we use dynamic models instead of static models.

2 Related Works

2.1 Dynamic Networks

Earlier works in modeling temporal information into networks can be categorized as (i) discrete-time models, and (ii) continuous-time models. In discrete-time models, time is discretized into bins of equal size, and recurrent neural network based models are used for modeling temporal evolution [Zhou *et al.*, 2018; Goyal *et al.*, 2018; Gupta *et al.*, 2019; Goyal *et al.*, 2020; Gracious *et al.*, 2021]. Since discretization results in information loss and selecting bin size is a difficult task, the recent focus has been on continuous-time models. This involves works like JODIE [Kumar *et al.*, 2019] with time projection embeddings and TGAT [da Xu *et al.*, 2020] with functional time embeddings to learn representations of nodes by modeling the dynamic interactions. Unlike these, TPP based continuous-time models can predict both dynamic interaction and time of interaction. The works like DeepCoevolve [Dai *et al.*, 2016], DyRep [Trivedi *et al.*, 2019], and DSPP [Cao *et al.*, 2021] use neural network based TPP to model the dynamic interactions. However, the earlier works model higher-order interactios between nodes by decomposing them into pair-wise interactions. It has been shown in Hyper-SAGNN [Zhang *et al.*, 2019] that directly modeling the higher-order interaction will result in better performance than decomposing them into pair-wise interactions. Our work proposes a TPP based model for predicting higher-order interaction between nodes.

2.2 HyperGraph Link Prediction

Higher-order interaction between nodes can be modeled as link prediction in a hypergraph. Earlier works use matrix completion techniques to predict hyperedge. Coordinated Matrix Minimization (CMM) [Zhang *et al.*, 2018], infer the missing hyperedges in the network by modeling adjacency matrix using a non-negative matrix factorization based latent representation. For this, they use an expectation-maximization framework with maximization step for learning latent representation, and expectation step is used for inferring the missing hyperlinks. HGE [Yu *et al.*, 2018] uses tensor factorization techniques to infer missing hyperlinks. HGDL [Arya and Worring, 2018] uses geometric deep learning techniques to do matrix completion of the incidence matrix of a hypergraph.

Recent works mostly concentrate on neural network-based scoring functions as they perform better than matrix completion-based techniques and are easier to train. Hyperpath [Huang *et al.*, 2019] model's hyperedge as a tuple and uses a neural network-based scoring function to predict links. This method cannot model higher-order interactions as it expects the hyperedge size to be uniform for all edges. NHP [Yadati *et al.*, 2019] defines a hypergraph neural network for predicting links. They achieve this using a hyperedge-aware graph convolution network and define a novel scoring function for hyperlinks. However, this method requires node attribute information for performing link prediction. HyperSANN [Zhang *et al.*, 2019] uses a self-attention based architecture for predicting hyperlinks. It can learn node embeddings and predict non-uniform hyperlinks. Hence, in our work, we use modified HyperSANN's architecture to parameterize TPP to forecast hyperlinks when networks are evolving with time.

3 Background on Temporal Point Process

A TPP [Daley and Vere-Jones, 2003] is a continuous-time stochastic process that models discrete events in time t_i, $t_i \in \mathbb{R}^+$. It defines a conditional density function (CDF) for future event time t by observing historical event times till time t_n, $\mathcal{T}(t_n) = \{t_1, t_2, ..., t_n\}$. A convenient and interpretable way to parameterize CDF is by defining a conditional intensity function or instantaneous stochastic rate of events $\lambda(t)$ defined below,

$$\lambda(t)dt = p(\text{event in}[t, t+dt]|\mathcal{T}(t)). \tag{1}$$

That is $\lambda(t)dt$ is the probability of observing an event in interval $[t, t+dt]$ by observing history till t. Then we can write CDF of next event time as,

$$\begin{aligned} p(t|\mathcal{T}(t)) &= \lambda(t)\mathcal{S}(t), \\ \mathcal{S}(t) &= \exp\,(\int_{t_n}^{t} -\lambda(t)dt). \end{aligned} \tag{2}$$

Here, $\mathcal{S}(t)$ is the probability that no event happened during the interval $[t_n' t)$.

The choice of functional form of $\lambda(t)$ depends upon the nature of the problem we are trying to model. For example, Poisson process have constant intensity $\lambda(t) = \mu$. Hawkes process [Hawkes, 1971] is used when events have self exciting nature with $\lambda(t) = \mu + \alpha \sum_{t_i \in \mathcal{T}(t)} \mathcal{K}(t - t_i)$. Here, $\mu \geq 0$ is the base rate, $\mathcal{K}(t) \geq 0$ is the excitation kernel, and $\alpha \geq 0$ is the strength of excitation. Rayleigh process [Ghosh, 2009] with $\lambda(t) = \alpha t$, where rate of events increase with time and $\alpha > 0$ is the weight parameter. Neural Temporal Point process [Shchur *et al.*, 2021] if the intensity is the output of neural network $f(.)$ that takes history as input, $\lambda(t) = f(\mathcal{T}(t))$.

4 Dynamic Hypergraph Representation Learning

Given a set of nodes $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$, interaction event between these nodes is modeled as a hyperedge (h_i) with time of interaction as its edge attribute. Here, h_i is a subset of nodes in $\mathcal{V}$ and $\mathcal{H} = \{h_1, h_2, \ldots h_{|\mathcal{H}|}\}$ is set of all valid combination of nodes. Given the historical of events $\mathcal{E}(t_m) = \{(h_1, t_1), , \ldots, (h_m, t_m)\}$ till time t_m, we want to forecast future hyperedge h_i at time $t > t_m$.

4.1 Hyperedge Event Modelling

Given hyperedge $h = \{v_1, v_2, \ldots, v_k\}$, the probability of it occurring at time t can be modeled using temporal point process with conditional intensity $\lambda_h(t)$ as shown below,

$$\begin{aligned} p_h(t) &= \lambda_h(t) \exp\,(\int_{t_h^p}^{t} -\lambda_h(t)dt), \\ \text{where, } t_h^p &= \max_{v \in h} t_v^p. \end{aligned} \tag{3}$$

Here, t_v^p is the most recent interaction time of node in h. $\lambda_h(t)$ is parameterized by defining positive function over the embeddings of nodes in h as shown below,

$$\lambda_h(t) = f(v_1(t), v_2(t), \ldots, v_k(t)). \tag{4}$$

Here $f(.) \geq 0$ is defined by neural network for hyperedge events, and $v_i(t) \in \mathbb{R}^d$ is the node embeddings at time t for node v_i. We follow the same architecture hyperlink modeling technique Hyper-SAGNN [Zhang *et al.*, 2019].

Architecture. Given node embeddings, a self-attention mechanism similar to [Vaswani *et al.*, 2017; Veličković *et al.*, 2018] is used get importance weights for each node in the hyperedge as shown below,

$$\begin{aligned} e_{ij} &= (W_Q^T v_i(t))^T W_K^T v_j(t), \forall 1 \leq i, j \leq k, i \neq j, \\ \alpha_{ij} &= \frac{\exp\,(e_{ij})}{\sum\limits_{1 \leq \ell \leq k, i \neq \ell} \exp\,(e_{i\ell})}. \end{aligned} \tag{5}$$

These weights are used to calculate the dynamic embeddings for each node as shown below,

$$\vec{d_i} = \tanh\left(\sum_{1 \leq j \leq k, i \neq j} \alpha_{ij} W_V^T v_j(t)\right). \tag{6}$$

Here, W_Q, W_K, W_K are learnable weights. We also create a static embeddings $\vec{s_i} = W_s v_i(t)$. Then we calculate the Hadamard power of the difference between static and dynamic embedding pairs followed by a linear layer and average pooling to get final score p as shown below,

$$\begin{aligned} o_i &= W_o^T(\vec{d_i} - \vec{s_i})^2) + b_o, \\ \mathcal{P}^h &= \frac{1}{k}\sum_{i=1}^{k} \mathcal{P}_i^h = \frac{1}{k}\sum_{i=1}^{k} \log(1 + \exp\,(o_i^h)). \end{aligned} \tag{7}$$

In our model the value of $f(v_1(t), v_2(t), \ldots, v_k(t)) = \mathcal{P}^h$. We also created baseline model with piece-wise constant node embeddings with intensity defined by Rayleigh Process as shown below,

$$\lambda_h(t) = f(v_1(t_h^p), v_2(t_h^p), \ldots, v_k(t_h^p))(t - t_h^p). \tag{8}$$

4.2 Dynamic Node Representation

For each node in the network, we learn a low dimensional embedding $v(t) \in \mathbb{R}^d$ that changes with time. It is done through two stages, i) Temporal Drift and ii) Interaction Update.

Temporal Drift. This stage models the inter-event evolution of a node with time. For a node v with previous event time t_v^p, current embedding at time t is modelled as,

$$v(t) = \tanh(W_0 v(t_v^{p+}) + W_1 \Phi(t - t_v^p) + b_0). \quad (9)$$

Here, $W_0, W_1 \in \mathbb{R}^{d \times d}, b \in \mathbb{R}^d$ are learnable parameters, $v(t_v^{p+})$ is node embedding just after interaction update at time t_v^p, and $\Phi(t) \in \mathbb{R}^d$ is fourier time features [da Xu *et al.*, 2020; Cao *et al.*, 2021] defined as follows,

$$\Phi(t) = [\cos(\omega_1 t + \theta_1), \ldots, \cos(\omega_d t + \theta_d)]. \quad (10)$$

Here, $\{\omega_i\}_{i=1}^d$, and $\{\theta_i\}_{i=1}^d$ are learnable parameters.

Interaction Update. When a node v is involved in an interaction h, it is influenced by the nodes it interacts with in h. For extracting aggregated features of interaction embedding, we use the dynamic embedding $\vec{d_v}$ calculated at time t in 4.1 for node v. The entire update equation is as follows,

$$v(t^+) = \tanh(W_2 v(t_v^{p+}) + W_3 \Phi(t - t_v^p) + W_4 \vec{d_v} + b_1). \quad (11)$$

Here, $W_2, W_3, W_4 \in \mathbb{R}^{d \times d}$ and $b \in \mathbb{R}^d$ are learnable parameters.

5 Learning Procedure

5.1 Loss Function

Once intensity parameterization is fixed for temporal point process as in Equation 4 or 8. The likelihood for hyperedge events $\mathcal{E}(t) = \{(h_1, t_1), \ldots, (h_m, t_m)\}$ occurring in an interval $[0, T]$ can be modeled as,

$$p(\mathcal{E}(t)) = \prod_{i=1}^{m} p_{h_i}(t_i) \prod_{h \in \mathcal{H}} \mathcal{S}_h(t_h^l, T). \quad (12)$$

Here, $p_{h_i}(t_i)$ is the probability of hyperedge event h_i happening at time t_i as defined in Equation 3, $\mathcal{S}_h(t_h^l, T)$ is the probability that no event happened for hyperedge h for the interval $[t_h^l, T]$, and t_h^l is the last time occurrence for h ($t_h^l = 0$ when no event of h is observed). The loss for learning parameters can be found by taking the negative of log-likelihood as,

$$\mathcal{L} = -\sum_{i=1}^{m} \log(\lambda_h(t_i)) + \sum_{h \in \mathcal{H}} \int_0^T \lambda_h(t) dt. \quad (13)$$

Here, the first term corresponds to the sum of the negative log intensity of occurred events. The second term corresponds to the sum of intensities of all events. The following happens when we minimize the loss, the intensity rate of occurred events increases due to minimization of the first term. Similarly, intensity rates of events not occurred decrease due to minimization of the second term. However, directly implementing this equation is computationally inefficient as $|\mathcal{H}| \leq 2^{|\mathcal{V}|}$ is huge. Further, the integration in the second term does not always have close form expression. Next, we will give this model a computationally efficient mini-batch training procedure.

5.2 Mini-Batch Loss

Given a batch of consecutive events for an interval $[T_0 = t_1, T = t_M]$, $\mathcal{E}_M = \{(h_1, t_1), \ldots, (h_M, t_M)\}$, for each (h_i, t_i) we find the previous event time t_h^p then use Mont-Carlo integration to find the log survival term of $p_{h_i}(t_i)$. Then the negative loglikelihood for that event is,

$$\begin{aligned} \mathcal{T}^s &= \{t_j^s\}_{j=1}^N \leftarrow \text{Uniform}(t_{h_i}^{p^+}, t, N) \\ \mathcal{L}_{h_i} &= -\log(\lambda_{h_i}(t_i)) + \sum_{j=2}^{N} (t_j^s - t_{j-1}^s) \lambda_{h_i}(t_j^s). \end{aligned} \quad (14)$$

Here, $\mathcal{T}^s$ is the set of uniformly sampled time points from the interval $[t_{h_i}^{p^+}, t]$. Then to consider the interactions events $h \in \mathcal{H}$ that was not observed during the above period, we sample some negative hyperedges for each interaction event (h_i, t_i) as described as below,

1. Choose the size of negative hyperedge k based on a categorical distribution over hyperedge sizes observed in the training data. Here, parameters of the categorical distribution are learned from the training dataset.
2. Sample $\min(k//2, |h_i|)$ nodes from the hyperedge h_i and rest of the nodes from $\mathcal{V} - h_i$. This strategy will avoid the trivial negative samples.

Following the above steps, we sample $\mathcal{H}_i^n = \{h_1^n, \ldots, h_{\mathcal{B}}^n\}$ negative hyperedges, and for each of them, we calculate the negative log likelihood for events not happening using Monte-Carlo integration. Then Equation 14 becomes,

$$\mathcal{L}_{h_i} = -\log(\lambda_{h_i}(t_i)) + \sum_{h \in \mathcal{H}_i^n \cup \{h_i\}} \sum_{j=2}^{N} (t_j^s - t_{j-1}^s) \lambda_h(t_j^s). \quad (15)$$

Then final mini-batch loss is calculated by summing all $\mathcal{L}_{h_i}$ for the events (h_i, t_i) in $\mathcal{E}_M$.

6 Prediction Tasks

Using temporal point process models, we can predict both the next event type and the time of the event. The following are the equivalent tasks in our settings.

Interaction Type Prediction. The type of interaction that occurs at time t can be predicted by finding the h_i with the maximum intensity value at that time, as shown below,

$$\hat{h} = \arg\max_{h_i} \lambda_{h_i}(t). \quad (16)$$

Interaction Duration Prediction. For interaction h occurred at time t_h^p, to predict the duration for future interaction, we have to calculate the expected time t with respect the conditional distribution $p_h(t)$ in Equation 3,

$$\hat{t} = \int_{t_h^p}^{\infty} (t - t_h^p) p_h(t). \tag{17}$$

If $\lambda(t)$ is modeled using a Rayleigh process as in Equation 8, we calculate the $\hat{t}$ in close form as,

$$\hat{t} = \sqrt{\frac{\pi}{2 \exp f(v_1(t_h^p), v_2(t_h^p), \dots, v_k(t_h^p))}}. \tag{18}$$

Otherwise, we have to compute the integration by sampling.

7 Experiments

Datasets	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{H}\rvert$
email-Enron	143	10,883	1,542
email-Eu	998	234,760	25,791
congress-bills	1718	260,851	85,082
NDC-classes	1,161	49,724	1,222
NDC-substances	5,311	112,405	10,025

Table 1: Datasets used along with their vital statistics

7.1 Datasets

All the above datasets are taken from work [Benson *et al.*, 2018][1]. Table 1 shows the important statistics of these datasets.

Email Network [email-Enron [Klimt and Yang, 2004]; email-Eu [Paranjape *et al.*, 2017]]. This is a sequence of timestamped email interactions between employees at a company. The entities involved are the email addresses of the sender and receivers. The timestamps in this are recorded at a resolution of 1-millisecond and are scaled-down in all our experiments. The scaling factor is chosen as the median value of inter-event duration for both datasets.

congress-bills [Fowler, 2006]. Here, interactions are the legislative bill put forth in the House of Representatives and the Senate in the US. The entities involved are the US congresspersons who sponsor and co-sponsor the bill. The timestamps in this are the date when the bill is introduced.

Drug Networks [NDC-classes, NDC-substances]. Each interaction is a sequence of drugs, and timestamps are the date at which the drug was introduced in the market. In NDC classes, nodes involved in the interaction are class labels applied to the drugs. In NDC-substances, nodes involved in the interaction are substances that make up the drug.

7.2 Baselines

The following are the variants of our model DHE. In these, the first two use the Rayleigh process in Equation 8 for conditional intensity function. The next model uses the pairwise decomposition of hyperedges as in Figure 1, and the last model does not use the **Interation Update** stage of DHE.

[1]https://www.cs.cornell.edu/ arb/data/

Rayleigh Hyperedge (RHE). In this, we keep embeddings $v(t)$ fixed for every time instance. Then intensity is modeled as Rayleigh process as in Equation 8, and duration predictions are made using the close expression in Equation 18.

Rayliegh Dynamic Hyperedge (RDHE). Similar to RHE, but here we allow the node embeddings $v(t)$ to evolve when an interaction involving the node v occurs as per the **Interaction Update** stage but not through **Temporal Drift** stage. So, $v(t)$ is piece-wise continuous (node embeddings are constant during inter-event time) as we do not allow the model to evolve the node embedding during the interevent time. This model is based on the DeepCoevolve [Dai *et al.*, 2016] model used for sequential recommendation.

Dynamic Edge-Drift (DE-Drift). In this model, each hyperedge event is modeled as a concurrent pairwise edge events. For example, $h = \{v_1, v_2, v_3\}$ is modeled as concurrent edge events $\{(v_1, v_2), (v_2, v_3), (v_1, v_3)\}$. Here, node embeddings $v(t)$ are allowed to evolve during the interevent time using **Temporal Drift** stage. For predicting interaction at time t, we use the product of all conditional intensity functions of concurrent edge events (e.g. $\lambda_{v_1,v_2}(t)\lambda_{v_2,v_3}(t), \lambda_{v_1,v_3}(t)$ for interaction h). For predicting time of interaction, we calculate $\hat{t}$ in Equation 17 for each edge in the interaction and average them. In this model, conditional intensity is defined as $\lambda_{v_1(t),v_2(t)} = v_1^T(t)v_2(t)$.

Dynamic Edge (DE). This uses the same modeling technique as DE-Drift. In addition to that, it also uses the **Interaction Update** stage to evolve embeddings.

Dynamic Hyperedge-Drift (DHE-Drift). This uses hyperedge modeling explained in the Section 4.2, but do not use the **Interaction Update** stage.

7.3 Metrics of Evaluation

Mean Average Reciprocal Rank (MRR) . We use this for evaluating the performance of interaction prediction at time t. We rank the intensity of true hyperedge against candidate negative hyperedge in descending order of $\lambda_h(t)$ and then average them for all the test set as shown below,

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{r_i + 1}. \tag{19}$$

Here, a higher MRR value means better the models' performance.

Mean Average Error (MAE) . We use this for evaluating the performance of interaction duration prediction, as shown below,

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |\hat{t}_i - t_i^{true}|. \tag{20}$$

Here, lower the MAE value means better the models' performance.

Methods	email-Enron		email-Eu		congress-bills		NDC-classes		NDC-substances	
	MRR	MAE	MRR	MAE	MRR	MAE	MRR	MAE	MRR	MAE
RHE	24.88	103.45	46.89	1010.48	30.72	25.88	**91.26**	115.98	77.15	28.04
RDHE	22.43	69.55	22.56	919.07	49.61	1.95	79.53	102.25	49.79	20.85
DE-drift	46.87	78.15	42.17	917.02	42.96	12.37	67.36	103.45	72.66	33.57
DE	49.10	66.73	50.10	917.4	56.65	5.25	68.52	102.74	72.66	45.83
DHE-drift	49.80	110.54	55.19	916.15	74.13	**0.8**	89.58	101.7	70.28	19.52
DHE	**57.90**	**62.56**	**58.05**	**914.83**	**86.09**	**0.8**	91.18*	**100.33**	**78.17**	**19.02**

Table 2: Performance of dynamic hyperedge forecasting in tasks of interaction type and interaction duration prediction. Here, interaction type prediction is evaluated using MRR in %, and interaction duration prediction is evaluated using MAE. Proposed model DHE beats baseline models in almost in settings.

7.4 Experimental Settings and Results

Experimental Settings

For all experiments, we use the learning rate of 0.001, the embedding size d is fixed at 64, the batch size M is fixed as 128, and the negative sampling is fixed as $\mathcal{B} = 20$. For the Monte Carlo estimate of $\log$ of survival probability in Section 5.2, we use $N = 100$ for datasets email-Enron and email-EU datasets, $N = 50$ for NDC-classes, and NDC-substances datasets, and $N = 20$ for the congress-bill dataset. The choice is N is made by considering memory constraints. All models are implemented PyTorch [Paszke *et al.*, 2019], and all training is done using its Adam [Kingma and Ba, 2015] optimizer. For all datasets, we use the first 50% of interactions for training, the next 25% for validation, and the rest for testing.

Results

In Table 2, we can see our model Dynamic Hyperedge (DHE) performs better than baselines in almost all settings. Further, it outperforms RHE that use static and RDHE that piece-wise constant node embeddings significantly. It is because continuous-time models have more expressiveness than other models. However, this increment of the performance came at the cost of computation because for models RHE and RDHE, we can find a closed-form solution for loss calculation as used in DeepCoevovle [Dai *et al.*, 2016], and duration estimation $\hat{t}$ can be done using Equation 18. We can also observe that the most significant gains are for the email-Enron dataset. This is because email-Enron is a smaller dataset, and dynamic node embeddings can be efficiently trained for this compared to other datasets with longer chains of interactions.

Further, we can observe the advantage of using hyperedge for modeling higher-order interactions when comparing models DHE-drift to DE-drift and DHE to DE. Between DHE-drift and DE-drift, there is an improvement of MRR metric in interaction type prediction tasks for all datasets except for NDC-substances. For the time duration prediction task, we can observe MAE reduction for all datasets except for the email-Enron dataset. DHE model outperforms DE in all tasks for all datasets uniformly. There is an average performance increase of 25% in MRR for the interaction type prediction task and 1.41% MAE reduction for the duration prediction task. Even though pair-wise edges can give decent performance if there are concurrent hyperedges with common nodes we cannot identify the hyperedge among them using pair-wise edge modeling as explained in Figure 1. Next, we will look at the advantage of node updating techniques used for dynamic node representations.

To infer **Interaction Update** stage has resulted in performance improvement, we will compare models that have this stage in their update equation to models that do not. Firstly, we can observe that there is an average increase of 10% in the MRR metric for interaction type prediction task and a reduction of 16% in the MAE metric for DHE in comparison to DHE-drift. Similarly, between RHE and RDHE, we can see that RDHE has a performance gain in interaction duration prediction, and it is uniformly for all the datasets. Even-though RHE performs better in MRR in some datasets, we can see MRR for RDHE is much better in congress-bills datasets, and email-Eu MRR is comparable. In addition to that DE performs uniformly better than DE-drift in almost all datasets in the interaction prediction task.

Similarly, the advantage of using **Temporal Drift** stage can be observed by comparing the performance of RHE to DHE-drift and RDHE to DHE. There is an average 49% improvement of MRR in interaction type prediction task for DHE-drift compared to RHE. Similarly, DHE uniformly outperforms RDHE for all tasks in all datasets. There is an average of 92% improvement in the MRR for the interaction type prediction task and a 33% decrease in the MAE error for the duration prediction task.

8 Conclusion and Future Work

This work proposes a model for forecasting higher-order interaction between nodes in a network as temporal hyperedge formation events. It is done by using a temporal point process based on the dynamic representation of nodes in the network. For learning dynamic node representation, we present a recurrent neural network formulation for dynamic node representation that updates the node embedding based on the nodes involved in the hyperedge event. On top of that, an efficient negative sampling based training strategy for this model is presented in this work. Further, we demonstrate the superior performance of our model in both interaction type prediction and interaction duration prediction tasks in five real-world datasets.

Future works include extending this idea for modeling multi-relational higher orders interaction [Fatemi *et al.*, 2020]. We also like to reduce the training time by using t-batch [Kumar *et al.*, 2019] based approach. This will allow us to do batch based node embedding training that can use parallel training techniques, unlike the sequential training we are currently using. Further, incorporating multi-hop information into node representation using hypergraph neural network-based techniques will result in better predictive performance.

References

[Arya and Worring, 2018] Devanshu Arya and Marcel Worring. Exploiting relational information in social networks using geometric deep learning on hypergraphs. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, page 117–125, 2018.

[Bai *et al.*, 2021] Song Bai, Feihu Zhang, and Philip HS Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637, 2021.

[Benson *et al.*, 2018] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 2018.

[Cao *et al.*, 2021] Jiangxia Cao, Xixun Lin, Xin Cong, Shu Guo, Hengzhu Tang, Tingwen Liu, and Bin Wang. Deep structural point process for learning temporal interaction networks. In Nuria Oliver, Fernando Pérez-Cruz, Stefan Kramer, Jesse Read, and Jose A. Lozano, editors, *Machine Learning and Knowledge Discovery in Databases. Research Track*, pages 305–320, 2021.

[da Xu *et al.*, 2020] da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations (ICLR)*, 2020.

[Dai *et al.*, 2016] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. Deep coevolutionary network: Embedding user and item features for recommendation. *arXiv*, 2016.

[Daley and Vere-Jones, 2003] D. J. Daley and D. Vere-Jones. *An introduction to the theory of point processes. Vol. I.* Springer-Verlag, 2003.

[Fatemi *et al.*, 2020] Bahare Fatemi, Perouz Taslakian, David Vazquez, and David Poole. Knowledge hypergraphs: Prediction beyond binary relations. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 2191–2197, 7 2020.

[Fowler, 2006] James H Fowler. Legislative cosponsorship networks in the us house and senate. *Social Networks*, 28(4):454–465, 2006.

[Ghosh, 2009] Jayanta K. Ghosh. Survival and event history analysis: A process point of view by odd o. aalen, Ørnulf borgan, håkon k. gjessing. *International Statistical Review*, 77(3):463–464, 2009.

[Ghoshdastidar and Dukkipati, 2017a] D. Ghoshdastidar and A. Dukkipati. Consistency of spectral hypergraph partitioning under planted partition model. *The Annals of Statistics*, 45(1):289–315, 2017.

[Ghoshdastidar and Dukkipati, 2017b] D. Ghoshdastidar and A. Dukkipati. Uniform hypergraph partitioning: Provable tensor methods and sampling techniques. *The Journal of Machine Learning Research*, 18(50):1–41, 2017.

[Goyal *et al.*, 2018] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. Dyngem: Deep embedding method for dynamic graphs. *CoRR*, abs/1805.11273, 2018.

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

[Gracious *et al.*, 2021] Tony Gracious, Shubham Gupta, Arun Kanthali, Rui M Castro, and Ambedkar Dukkipati. Neural latent space model for dynamic networks and temporal knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4054–4062, 2021.

[Gupta *et al.*, 2019] Shubham Gupta, Gaurav Sharma, and Ambedkar Dukkipati. A generative model for dynamic networks with applications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7842–7849, 2019.

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

[Huang *et al.*, 2019] Jie Huang, Xin Liu, and Yangqiu Song. Hyper-path-based representation learning for hypernetworks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 449–458, 2019.

[Kingma and Ba, 2015] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[Klimt and Yang, 2004] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*, pages 217–226, 2004.

[Kumar *et al.*, 2019] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2019.

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

[Paranjape *et al.*, 2017] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. Motifs in temporal networks. In

Proceedings of the tenth ACM international conference on web search and data mining, pages 601–610, 2017.

[Paszke *et al.*, 2019] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[Shchur *et al.*, 2021] Oleksandr Shchur, Ali Caner Türkmen, Tim Januschowski, and Stephan Günnemann. Neural temporal point processes: A review. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4585–4593. International Joint Conferences on Artificial Intelligence Organization, 8 2021.

[Trivedi *et al.*, 2019] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*, 2019.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6000–6010, 2017.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[Yadati *et al.*, 2019] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[Yu *et al.*, 2018] Chia-An Yu, Ching-Lun Tai, Tak-Shing Chan, and Yi-Hsuan Yang. Modeling multi-way relations with hypergraph embedding. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1707–1710, 2018.

[Zhang *et al.*, 2018] Muhan Zhang, Zhicheng Cui, Shali Jiang, and Yixin Chen. Beyond link prediction: Predicting hyperlinks in adjacency space. In *AAAI*, 2018.

[Zhang *et al.*, 2019] Ruochi Zhang, Yuesong Zou, and Jian Ma. Hyper-sagnn: a self-attention based graph neural network for hypergraphs. In *International Conference on Learning Representations*, 2019.

[Zhou *et al.*, 2018] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.