ST-TPP: Learning Semi-Transductive Temporal Point Processes with Gromov-Wasserstein Barycentric Regularization

Qingmei Wang[1], Tianyu Huang[2], Yujie Long[3], Yuxin Wu[1], Fanmeng Wang[1], Xi Sun[4], Junchi Yan[2], Hongteng Xu[1*]

[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]School of Artificial Intelligence, Shanghai Jiao Tong University
[3]School of Computing and Artificial Intelligence, Fudan University
[4]AI Platform Department, MetaLight HK Limited

Abstract

The generative mechanisms behind real-world event sequences are often heterogeneous, leading to data that possesses inherent clustering structures. However, most existing temporal point processes (TPPs) treat different event sequences independently, without leveraging the clustering structures when predicting events. In this study, we design and learn a novel semi-transductive temporal point process (ST-TPP), which explicitly improves prediction performance by co-training sequence clusters. In particular, given a set of event sequences, our method learns a neural TPP together with cluster centers of the sequences. Besides maximizing the likelihood of the event sequences, we leverage a data-based kernel matrix and prior knowledge to regularize the sequence embeddings, leading to a Gromov-Wasserstein barycentric (GWB) regularizer. Based on the optimal transport plans associated with the GWB regularizer, we derive the cluster centers by the push-forward of the sequence embeddings. When a new sequence comes, the learned model first assigns a cluster center to the sequence and then jointly encodes the sequence and the cluster center to predict future events, leading to a semi-transductive prediction scheme. Experiments demonstrate that ST-TPP achieves competitive sequence clustering results and strong prediction performance.

Introduction

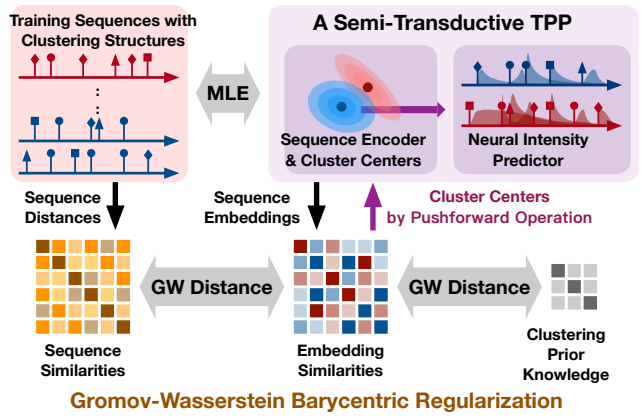
As a powerful statistical tool, temporal point process (TPP) is commonly used to model and predict event sequences in the continuous-time domain. In the past decades, many TPPs have been proposed, e.g., Hawkes process (Hawkes 1971; Liniger 2009), self-correcting process (Isham and Westcott 1979; Xu et al. 2016), and those neural TPPs (Du et al. 2016; Mei and Eisner 2017; Zuo et al. 2020; Wang et al. 2025; Gong et al. 2025). These models have achieved encouraging performance in various sequence modeling tasks, e.g., health data management (Enguehard et al. 2020), network analysis (Zhao et al. 2015; Kong et al. 2023), and financial engineering (Bacry, Mastromatteo, and Muzy 2015).

Despite their usefulness, the above TPPs seldom consider the clustering structures of event sequences. Real-world event sequences often yield various generative mechanisms and thus have clustering structures naturally, e.g., the admission behaviors of patients with different diseases, the job-hopping behaviors of employees in different industries, and so on. Ignoring such clustering information may lead to model misspecification and suboptimal model performance. Typically, we can define various distance metrics for event sequences (Iwayama, Hirata, and Aihara 2017; Xiao et al. 2017; Xu, Luo, and Zha 2021) and then cluster event sequences by spectral clustering (Ng, Jordan, and Weiss 2001). Unfortunately, this *transductive* strategy does not model the generative mechanisms of event sequences and thus cannot apply to event prediction. Recently, mixture models of TPPs (Luo et al. 2015; Xu and Zha 2017a; Zhang et al. 2022) are applied to learn multiple TPPs for the event sequences with clustering structures. However, the complexity of these mixture models is linear with the number of clusters, and they suffer a high risk of over-fitting because of learning multiple TPPs. To our knowledge, how to incorporate sequence clusters with event prediction is still a significant open problem in the community of TPP modeling.

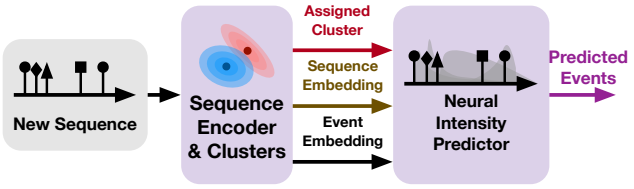
In this study, we propose a novel semi-transductive temporal point process (ST-TPP) and corresponding learning method with the help of optimal transport (OT) (Mémoli 2011a), leading to a learning paradigm of TPP enhancing event prediction with proper clustering information. As illustrated in Figure 1, ST-TPP consists of a sequence encoder and a neural intensity predictor, working for sequence embedding and event prediction, respectively. The sequence encoder is associated with the cluster centers of the training sequences' embeddings. In both training and testing phases, given a sequence, the sequence encoder obtains the event- and sequence-level embeddings and assigns a cluster center to the sequence. The predictor leverages the three kinds of information to predict future events for the sequence. This prediction paradigm leads to the so-called semi-transductive learning strategy (Alzubaidi et al. 2012; Ge et al. 2023), which combines the inductive embeddings of the target sequence with the transductive clustering information learned from the whole training set.

We learn the ST-TPP model in a regularized maximum likelihood estimation (MLE) framework. As shown in Fig-

*Corresponding author: hongtengxu@ruc.edu.cn



(a) Training



(b) Testing

Figure 1: The training and testing schemes of ST-TPP.

ure 1(a), the embeddings of training sequences are regularized to minimize the Gromov-Wasserstein barycenter (GWB) (Peyré, Cuturi, and Solomon 2016) of the kernel matrices deriving from the data and prior knowledge, respectively. Based on the OT plans associated with the GWB regularizer, the cluster centers can be obtained by the pushforward operation of the sequence embeddings (Peyré, Cuturi et al. 2019). Note that the proposed learning method is compatible with stochastic gradient descent, with low computational complexity.

ST-TPP provides a new solution to enhance event prediction with sequence clusters. Its semi-transductive prediction paradigm is applicable to most existing TPP models that involve event and sequence embedding, which may inspire new TPP modeling strategies. Experiments on both synthetic and real-world datasets demonstrate the effectiveness of ST-TPP. Compared with state-of-the-art TPP models, ST-TPP achieves competitive prediction accuracy and derive clustered sequence embeddings, whose model performance and interpretability are improved significantly.

Preliminaries and Related Work

TPP and Its MLE Framework Denote an event sequence with N events as $\boldsymbol{s} = \{(t_n, c_n)\}_{n=1}^{N}$, where (t_n, c_n) is the n-th event, $t_n \in [0, T]$ is its timestamp, and $c_n \in \mathcal{C} = \{1, ..., C\}$ is its event type. A multivariate TPP is denoted as $\boldsymbol{N}(\theta) = \{N_c(t;\theta)\}_{c\in\mathcal{C}, t\in[0,T]}$, where θ denotes the model parameters and $N_c(t;\theta)$ is a stochastic process counting the number of the type-c events till time t. The TPP is characterized by a multivariate intensity function, denoted as $\boldsymbol{\lambda}(t) = \{\lambda_c(t)\}_{c\in\mathcal{C}, t\in[0,T]}$, where

$$\lambda_c(t)\mathrm{d}t = \mathrm{d}\mathbb{E}[N_c(t;\theta)|\mathcal{H}_t^{\mathcal{C}}],\ \forall c \in \mathcal{C}. \tag{1}$$

Here, $\lambda_c(t)$ represents the expected instantaneous rate of the type-c event happening at time t given the historical events $\mathcal{H}_t^{\mathcal{C}} = \{(t_n, c_n) \in \boldsymbol{s} | t_n < t\}$.

Given M observed event sequences, i.e., $\mathcal{S} = \{\boldsymbol{s}_m\}_{m=1}^{M}$, we can learn the TPP in the following maximum likelihood estimation (MLE) framework (Liniger 2009):

$$\min_\theta -\sum\nolimits_{m=1}^{M} \log \mathcal{L}(\boldsymbol{s}_m;\theta). \tag{2}$$

Here, $\mathcal{L}(\boldsymbol{s}_m;\theta)$ is the likelihood of the m-th sequence $\boldsymbol{s}_m = \{(t_{n,m}, c_{n,m})\}_{n=1}^{N_m}$, i.e.,

$$\mathcal{L}(\boldsymbol{s}_m;\theta) = \prod_{n=1}^{N_m} \lambda_{c_{n,m}}(t_{n,m}) \exp\Big(-\sum_{c\in\mathcal{C}} \int_0^T \lambda_c(s)\mathrm{d}s\Big). \tag{3}$$

TPPs Considering Sequence-level Clusters For modeling event sequences with underlying clustering structures, a straightforward solution is learning a mixture model of multiple TPPs (Xu and Zha 2017a; Zhang et al. 2022), i.e., $\{w_k, \boldsymbol{N}(\theta_k)\}_{k=1}^{K}$, where K is the number of clusters, $\boldsymbol{N}(\theta_k)$ is the TPP generating the k-th cluster's event sequences, and w_k is the probability of the k-th cluster. The $\{\boldsymbol{N}(\theta_k)\}_{k=1}^{K}$ can be classic TPPs (like Hawkes processes) or neural ones. The MLE of this mixture model, i.e.,

$$\min_{\{w_k,\theta_k\}_{k=1}^{K}} -\sum\nolimits_{m=1}^{M} \log\Big(\sum\nolimits_{k=1}^{K} w_k \mathcal{L}(\boldsymbol{s}_m;\theta_k)\Big), \tag{4}$$

can be solved by the expectation-maximization (EM) algorithm (Xu and Zha 2017a). *Given a new sequence, the mixture model selects the TPP with the highest likelihood to predict its future events, in which the model essentially degrades to a single TPP and fails to fully leverage the learned clustering information.*

Optimal Transport-Based Sequence Clustering Given M event sequences with K clusters, the key step to cluster them is defining a distance metric for the event sequences. To achieve this aim, many valid distances have been proposed, including the counting-based distance in (Iwayama, Hirata, and Aihara 2017), the Wasserstein distance in (Xiao et al. 2017), and the hierarchical optimal transport (HOT) distance in (Xu, Luo, and Zha 2021). Denote the distance matrix of the sequences as $\boldsymbol{D} = [d(\boldsymbol{s}_m, \boldsymbol{s}_{m'})] \in \mathbb{R}^{M\times M}$. We can further define a kernel function based on the distance (e.g., a Gaussian kernel with a bandwidth σ), resulting in a kernel matrix $\widetilde{\boldsymbol{K}} = [\exp(-d^2(\boldsymbol{s}_m, \boldsymbol{s}_{m'})/(2h^2))] \in \mathbb{R}^{M\times M}$ to capture the pairwise similarities of the sequences. As shown in (Xu and Zha 2017b), this kernel matrix reflects the clustering structure of the event sequences.

Recently, the work in (Xu, Luo, and Carin 2019; Chowdhury and Needham 2021) demonstrates that computing the Gromov-Wasserstein (GW) distance (Mémoli 2011b) between $\widetilde{\boldsymbol{K}}$ and an identity matrix with size $K \times K$ (denoted as $\boldsymbol{I}_K$) leads to a generalized spectral clustering method.

Definition 1 (Gromov-Wasserstein distance)**.** *Denote $\mathcal{X}_{\mu,\kappa_1}$ and $\mathcal{Y}_{\nu,\kappa_2}$ as two metric-measure spaces, respectively, where*

μ, and ν are probability measures, and $\kappa_1 : \mathcal{X}^2 \mapsto \mathbb{R}_+$ and $\kappa_2 : \mathcal{Y}^2 \mapsto \mathbb{R}_+$ are two kernel functions. The p-order GW distance between the two spaces, denoted as $GW_p(\mathcal{X}_{\mu,\kappa_1}, \mathcal{Y}_{\nu,\kappa_2})$, is defined as

$$\inf_{\pi\in\Pi_{\mu,\nu}} \Big(\int_{\mathcal{X}^2\times\mathcal{Y}^2} r^p(x,x',y,y')\mathrm{d}\pi(x,y)\mathrm{d}\pi(x',y')\Big)^{\frac{1}{p}},$$

where $\pi \in \Pi_{\mu,\nu} = \{\pi \geq 0 | \int_{\mathcal{X}} \mathrm{d}\pi(x,y) = \nu(y), \int_{\mathcal{Y}} \mathrm{d}\pi(x,y) = \mu(x)\}$ is called transport plan between μ and ν. $r(x,x',y,y') = |\kappa_1(x,x') - \kappa_2(y,y')|$ is the distance between the kernel functions' values.

According to the definition and Proposition 1 in (Peyré, Cuturi, and Solomon 2016), the discrete 2-order GW distance between two kernel matrices (i.e., $GW_2(\boldsymbol{K}_1, \boldsymbol{K}_2)$, where $\boldsymbol{K}_1 \in \mathbb{R}^{M\times M}$ and $\boldsymbol{K}_2 \in \mathbb{R}^{L\times L}$) is

$$\begin{aligned}&\min_{\boldsymbol{T}\in\Pi_{M,L}} \Big(\sum\nolimits_{m,m',l,l'} [|\boldsymbol{K}_1(m,m') - \boldsymbol{K}_2(l,l')|^2]\Big)^{1/2} \\ &= \min_{\boldsymbol{T}\in\Pi_{M,L}} \langle \boldsymbol{C}(\boldsymbol{K}_1, \boldsymbol{K}_2, \boldsymbol{T}), \boldsymbol{T}\rangle^{1/2}.\end{aligned} \tag{5}$$

$\boldsymbol{C}(\boldsymbol{K}_1, \boldsymbol{K}_2, \boldsymbol{T}) = \frac{1}{M}(\boldsymbol{K}_1 \odot \boldsymbol{K}_1)\mathbf{1}_{M\times L} + \frac{1}{L}\mathbf{1}_{M\times L}(\boldsymbol{K}_2 \odot \boldsymbol{K}_2) - 2\boldsymbol{K}_1\boldsymbol{T}\boldsymbol{K}_2^\top$, $\odot$ is the Hadamard product, and $\Pi_{M,L} = \{\boldsymbol{T} \in \mathbb{R}_+^{M\times L} | \boldsymbol{T}\mathbf{1}_L = \frac{1}{M}\mathbf{1}_M, \boldsymbol{T}^\top\mathbf{1}_M = \frac{1}{L}\mathbf{1}_L\}$ is the feasible domain of the OT plan matrix $\boldsymbol{T}$.

Based on the work in (Xu, Luo, and Carin 2019; Chowdhury and Needham 2021), we can cluster event sequences by computing $GW_2(\widetilde{\boldsymbol{K}}, \boldsymbol{I}_K)$. The optimal transport plan, denoted as $\boldsymbol{T}^* \in \Pi_{M,K}$, corresponds to the joint distribution of the M sequences and K clusters and thus indicates the clustering results.[1] *However, as aforementioned, how to apply the clustering results in event prediction is still an open problem for TPP models.*

Proposed Method

Semi-Transductive Paradigm for Event Prediction

Different from existing TPPs, our ST-TPP model applies a semi-transductive paradigm for event prediction. As shown in Figure 1, ST-TPP consists of a sequence encoder and a neural intensity predictor, working for embedding event sequences and predicting events, respectively. In addition, K cluster centers of sequence-level embeddings are learned associated with the encoder, which stores the clustering information of training data. Specifically, the key modules of ST-TPP are shown below.

Sequence Encoder As shown in (Wang et al. 2023), most existing TPPs embed event sequences when calculating their intensity functions. Typically, given an event sequence $\boldsymbol{s}$ with N events, we obtain its *event-level embeddings* as

$$\{\boldsymbol{h}_n\}_{n=1}^N = g(\boldsymbol{s}), \tag{6}$$

where $\boldsymbol{h}_n \in \mathbb{R}^D$ is the D-dimensional embedding of the n-th event. g denotes the sequence encoder, which can be implemented as a continuous-time recurrent neural network (Du et al. 2016; Mei and Eisner 2017), a Transformer encoder (Zhang et al. 2020; Zuo et al. 2020), or any other sequential neural networks.

Given event-level embeddings, we can obtain a *sequence-level embedding* by mean-pooling, i.e.,

$$\boldsymbol{h}(t) = \text{MeanPooling}(\{\boldsymbol{h}_n\}_{t_n\leq t}) \in \mathbb{R}^D, \tag{7}$$

where $\boldsymbol{h}(t)$ denotes the sequence embedding till time t. Accordingly, $\boldsymbol{h}(t_N)$ is the embedding of the whole sequence.

In addition, the ST-TPP model contains K learnable cluster centers associated with the sequence encoder, denoted as $\boldsymbol{A} = [\boldsymbol{a}_1, ..., \boldsymbol{a}_K]^\top \in \mathbb{R}^{K\times D}$. Given an arbitrary sequence-level embedding $\boldsymbol{h}(t)$, we can assign the closest cluster center to it, i.e.,

$$\boldsymbol{a}(t) = \arg\min_{\boldsymbol{a}\in\{\boldsymbol{a}_k\}_{k=1}^K} \|\boldsymbol{h}(t) - \boldsymbol{a}_k\|_2. \tag{8}$$

The selected cluster center, denoted as $\boldsymbol{a}(t)$, works as the *cluster-level embedding*, which contains the structural information of the whole training dataset rather than the temporal information of an individual sequence.

Neural Intensity Predictor Suppose that we observe N events happening till time t, i.e., $\boldsymbol{s} = \{(t_n, c_n)\}_{n=1}^N$, and would like to predict a possible event happening at time t' ($t' > t$). The proposed neural intensity predictor leverages the embeddings at event, sequence, and cluster levels jointly to model the multivariate intensity function at time t', i.e.,

$$\boldsymbol{\lambda}(t') = f(\text{Concat}(\underbrace{\boldsymbol{h}_N, \boldsymbol{h}(t')}_{\text{inductive}}, \underbrace{\boldsymbol{a}(t')}_{\text{transductive}}, \underbrace{t' - t_N}_{\text{time interval}}), \tag{9}$$

where $\text{Concat}(\cdots)$ denotes the concatenation operation of vectors. f is a neural network modeling the intensity function, which is implemented as a multi-layer perceptron (MLP). $\boldsymbol{h}_N$ is the embedding of the last event, whose timestamp t_N is the closest timestamp to the target time t'. The time interval $t' - t$ captures the decay of the observed events' influence on the future event at time t'. Given $\boldsymbol{\lambda}(t')$, we predict event at time t' by $c^* = \arg\max_{c\in\mathcal{C}} \lambda_c(t')$.

As shown in (9), our model combines the embedding information of the target sequence with the clustering information from other sequences. The former is *inductive* because the event- and sequence-level embeddings are derived directly from the parametric sequence encoder, which is independent with other sequences. In contrast, the latter is *transductive* — the cluster-level embedding is selected from the cluster centers learned from the sequence-level embeddings of training sequences in a nonparametric way.

Our model is motivated by the following facts:

- The event-level embedding mainly captures the short-range event dependency. Especially when f is implemented by recurrent neural networks, the early events are likely to be forgotten with the increase of time, and they will contribute little to the event-level embedding $\boldsymbol{h}_N$.
- The sequence-level embedding aggregates event-level embeddings evenly, without considering the decay of early events' impacts. As a result, $\boldsymbol{h}(t')$ can capture long-range event dependency better than $\boldsymbol{h}_N$.
- The cluster-level embedding $\boldsymbol{a}(t')$ allows the target sequence to leverage the information of the similar sequences used in the training phase, it helps enhance the robustness of event prediction.

[1] The details of GW distance and generalized spectral clustering are shown in supplementary file.

Our model leverages the three kinds of embeddings jointly, which is expected to achieve better prediction accuracy.

Gromov-Wasserstein Barycentric Regularization

In the training phase, we learn ST-TPP (including the sequence encoder, the neural intensity predictor, and the cluster centers) in a regularized MLE framework, in which a GWB regularizer is applied to obtain the cluster centers.

Joint utilization of data and prior knowledge Specifically, given M training sequences $\{\boldsymbol{s}_m\}_{m=1}^M$, we can obtain their sequence-level embeddings, denoted as $\boldsymbol{H} = [\boldsymbol{h}^{(1)}(t_{N_1}), ..., \boldsymbol{h}^{(M)}(t_{N_M})]^\top \in \mathbb{R}^{M\times D}$. The sequence-level embeddings can be used to measure the similarity among the event sequences. Given arbitrary two sequence-level embeddings, we can apply a Gaussian kernel to measure their similarity, leading to the following kernel matrix, denoted as $\boldsymbol{K} = [\exp(-\|\boldsymbol{h}^{(m)} - \boldsymbol{h}^{(m')}\|_2^2/(2\sigma^2))] \in \mathbb{R}^{M\times M}$.

Inspired by the GW-based generalized spectral clustering methods (Xu, Luo, and Carin 2019; Chowdhury and Needham 2021), we regularize the kernel matrix $\boldsymbol{K}$ by

$$\min_{\boldsymbol{K}} \overbrace{\underbrace{GW_2^2(\boldsymbol{K}, \widetilde{\boldsymbol{K}})}_{\text{data-centric}} + \underbrace{GW_2^2(\boldsymbol{K}, \boldsymbol{I}_K)}_{\text{prior knowledge}}}^{\text{GWB}(\boldsymbol{K})}. \tag{10}$$

Here, the first term is data-centric, corresponding to the GW distance between the embedding-based kernel $\boldsymbol{K}$ and the sequence distance-based kernel $\widetilde{\boldsymbol{K}}$. Penalizing this term encourages $\boldsymbol{K}$ to inherit the similarities obtained from the pairwise distances of the event sequences. The second term in (10) computes the GW distance between $\boldsymbol{K}$ and $\boldsymbol{I}_K$, which follows the principle of the GW-based generalized spectral clustering. By penalizing this term, we impose the prior knowledge of clustering structure on $\boldsymbol{K}$, encouraging the sequence-level embeddings to be clustered into K classes. Minimizing these two GW distances jointly leads to the proposed GWB regularizer of $\boldsymbol{K}$, denoted as $\text{GWB}(\boldsymbol{K})$. The optimal $\boldsymbol{K}$ is the GW barycenter of $\widetilde{\boldsymbol{K}}$ and $\boldsymbol{I}_K$ (Peyré, Cuturi, and Solomon 2016).

- **Remark 1.** In the first term of (10), we apply GW distance rather than the mean-square-error (i.e., $\|\boldsymbol{K} - \widetilde{\boldsymbol{K}}\|_F^2$) to make the regularizer applicable in practice, especially in those large-scale applications. On one hand, given M sequences, we can select a subset of sequences and compute a small-sized $\widetilde{\boldsymbol{K}}$ in advance.[2] Without computing a kernel matrix with size $M \times M$, we can reduce the computational complexity significantly when M is large. On the other hand, when M is large, we need to train the ST-TPP model by stochastic gradient descent, and accordingly, compute a small-sized $\boldsymbol{K}$ for each batch of sequences. As a result, the rows and columns of $\widetilde{\boldsymbol{K}}$ are not aligned to those of $\boldsymbol{K}$, and the mean-square-error becomes inapplicable in such a situation.

[2]In this work, we apply the counting-based distance in (Iwayama, Hirata, and Aihara 2017) to compute the pairwise distance matrix of sequence and derive $\widetilde{\boldsymbol{K}}$ accordingly.

Algorithm 1: The computation of $GW_2(\boldsymbol{K}_1, \boldsymbol{K}_2)$

Require: Kernel matrices $\boldsymbol{K}_1 \in \mathbb{R}^{M\times M}$, $\boldsymbol{K}_2 \in \mathbb{R}^{L\times L}$, the weight of the proximal term λ.

Initialize $\boldsymbol{b} = \frac{1}{L}\mathbf{1}_L$ and $\boldsymbol{T} = \frac{1}{ML}\mathbf{1}_{M\times L}$
While Not converge **Do**
  $\boldsymbol{G} = \exp\left(-\frac{\boldsymbol{C}(\boldsymbol{K}_1, \boldsymbol{K}_2, \boldsymbol{T})}{\lambda}\right) \odot \boldsymbol{T}$
  While Not converge **Do**
    $\boldsymbol{a} = \mathbf{1}_M/(M\boldsymbol{G}\boldsymbol{b})$, and then $\boldsymbol{b} = \mathbf{1}_L/(L\boldsymbol{G}^\top \boldsymbol{a})$
  $\boldsymbol{T} = (\boldsymbol{a}\boldsymbol{b}^\top) \odot \boldsymbol{G}$
return The OT plan $\boldsymbol{T}^* \leftarrow \boldsymbol{T}$, and $GW_2(\boldsymbol{K}_1, \boldsymbol{K}_2)$

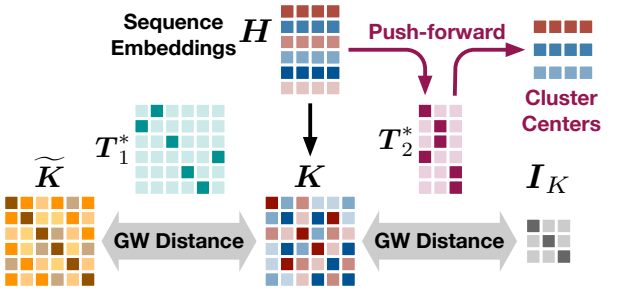


Figure 2: An illustration of the push-forward operation deriving cluster centers.

Deriving Cluster Centers via Push-forward We compute the GW distances in (10) by the proximal gradient algorithm (Peyré, Cuturi, and Solomon 2016; Xu, Luo, and Carin 2019), whose algorithmic scheme is shown in Algorithm 1. We denote the optimal transport matrices corresponding to $GW_2(\boldsymbol{K}, \widetilde{\boldsymbol{K}})$ and $GW_2(\boldsymbol{K}, \boldsymbol{I}_K)$ as $\boldsymbol{T}_1^*$ and $\boldsymbol{T}_2^*$. As illustrated in Figure 2, $\boldsymbol{T}_1^*$ and $\boldsymbol{T}_2^*$ align $\boldsymbol{K}$ to $\widetilde{\boldsymbol{K}}$ and $\boldsymbol{I}_K$, respectively. For the sequences constructing $\boldsymbol{K}$ and those constructing $\widetilde{\boldsymbol{K}}$, the high-valued elements in $\boldsymbol{T}_2^*$ indicate the correspondence between them. Similarly, for the sequences constructing $\boldsymbol{K}$, the high-valued elements in $\boldsymbol{T}_1^*$ indicate the clustering results of the sequences. As a result, we can derive cluster centers of sequence-level embeddings by the following push-forward operation (Courty et al. 2016):

$$\boldsymbol{A} = (M\boldsymbol{T}_2^*)^\top \boldsymbol{H}, \tag{11}$$

in which each cluster center is achieved by the weighted average of the sequence-level embeddings. The weight matrix $(M\boldsymbol{T}_2^*)^\top = [p(\boldsymbol{h}^{(m)}(T)|\boldsymbol{a}_k)]$ is a nonnegative and column-wisely normalized matrix. Its element $p(\boldsymbol{h}^{(m)}(T)|\boldsymbol{a}_k)$ can be explained as the conditional probability of the m-th sequence given the k-th cluster.

- **Remark 2:** Given M sequences, applying the push-forward operation to the whole sequence-level embedding matrix is time-consuming because it involves computing the GW distance with complexity $\mathcal{O}(M^2K)$. To achieve a trade-off between computational efficiency and model performance, we call the push-forward operation in (11) for each batch. Given the i-th batch of event sequences, whose batch size is B, we obtain the sequence-level embedding matrix $\boldsymbol{H}^{(i)}$ and the corresponding ker-

Algorithm 2: The learning algorithm solving (13)

Require: Event sequences $\mathcal{S} = \{\boldsymbol{s}_m\}_{m=1}^M$ in $[0, T]$, the hyperparameter τ, the predefined number of clusters K, the batch size B, the number of epochs J.

Initialize model parameters $\{g, f, \boldsymbol{A}\}$ randomly.
Sample L sequences randomly and compute $\widetilde{\boldsymbol{K}}$ based on the distance matrix of the sequences. $\mathcal{O}(L^2)$
for $j = 1, \cdots, J$ **do**
  – Update $\{g, f\}$**:**
  for each batch $\mathcal{B}$ **do**
    for each sequence $\boldsymbol{s} \in \mathcal{B}$ **do**
      Get event embeddings for each event via (6).
      Get sequence embeddings $\{\boldsymbol{h}(t_n)\}_{n=1}^N$ via (7).
      Get cluster embeddings $\{\boldsymbol{a}(t_n)\}_{n=1}^N$ via (8).
      Get the intensity of each event by (9).
    end for
    Get $-\sum_{\boldsymbol{s}\in\mathcal{B}} \log \mathcal{L}(\boldsymbol{s})$ based on (3). $\mathcal{O}(BN^2)$
    Get $\boldsymbol{K}$ based on $\boldsymbol{H} = [\boldsymbol{h}^{(b)}(t_{N_b})]$. $\mathcal{O}(B^2D)$
    Get GW distances and OT plans $\{\boldsymbol{T}_1^*, \boldsymbol{T}_2^*\}$ via Algorithm 1. $\mathcal{O}(BL^2 + B^2L + BK^2 + B^2K)$
    Get the loss $-\log \mathcal{L}(\boldsymbol{s}) + \tau\text{GWB}(\boldsymbol{K})$ and update $\{g, f\}$ by Adam (Kingma and Ba 2014).
  end for
  – Update $\boldsymbol{A}$**:**
  Initialize $\boldsymbol{A} = \boldsymbol{0}_{K\times D}$.
  for each batch $\mathcal{B}$ **do**
    Get $\boldsymbol{K}$ based on $\boldsymbol{H} = [\boldsymbol{h}^{(b)}(t_{N_b})]$. $\mathcal{O}(B^2D)$
    Compute $\boldsymbol{T}_2^*$ via Algorithm 1. $\mathcal{O}(BK^2 + B^2K)$
    $\boldsymbol{A} \leftarrow \boldsymbol{A} + (\frac{B}{I}\boldsymbol{T}_2^*)^\top \boldsymbol{H}$. $\mathcal{O}(KBD)$
  end for
end for
return The ST-TPP Model $\{g, f, \boldsymbol{A}\}$.

nel matrix $\boldsymbol{K}^{(i)}$. By computing $GW_2(\boldsymbol{K}^{(i)}, \boldsymbol{I}_K)$ and applying the push-forward operation, we derive the i-th OT plan, denoted as $\boldsymbol{T}_2^{*(i)}$, and the corresponding cluster centers $\boldsymbol{A}^{(i)}$. As a result, the average of all batch-based cluster centers leads to the learning result, i.e.,

$$\boldsymbol{A}^{(i)} = (B\boldsymbol{T}_2^{*(i)})^\top \boldsymbol{H}^{(i)}, \quad \boldsymbol{A} \approx 1/I \sum\nolimits_{i=1}^{I} \boldsymbol{A}^{(i)}, \quad (12)$$

where $I = M/B$ is the number of batches.

Learning Algorithm of ST-TPP

Combining the GWB regularizer in (10) with the MLE loss in (2), we obtain the learning problem of ST-TPP, i.e.,

$$\min_{\theta=\{g,f,\boldsymbol{A}\}} \Big(-\sum_{m=1}^{M} \log \mathcal{L}(\boldsymbol{s}_m; \theta) + \tau\, \text{GWB}(\boldsymbol{K}; \theta) \Big), \quad (13)$$

where the model parameters include $\{g, f, \boldsymbol{A}\}$. The hyperparameter $\tau > 0$ controls the significance of the GWB regularizer. This problem can be solved efficiently by iterative alternating optimization. Algorithm 2 shows the training process of ST-TPP, together with the computational complexity of key steps. In each epoch, we update $\{g, f\}$ and $\boldsymbol{A}$ by alternating optimization, and the updating of $\{g, f\}$ is achieved by stochastic gradient descent.

In practice, the number of clusters (i.e., K) is unknown in general. Therefore, given a set of real-world event sequences, we set it empirically in the range $[5, 10]$, which is significantly smaller than the batch size. In addition, to achieve a trade-off between efficiency and performance, we set the batch size B and the subset size L are significantly smaller than the dataset size M. In such a situation, we can find that the computational cost introduced by the GW distances is mild, which is about $\mathcal{O}(BL^2 + B^2L)$.

Dataset	#Event types C	#Sequences M	Max. #Events N
Syn. ($K = 2$)	5	8,000	50
Taobao	17	2000	64
SO	22	2200	100
Taxi	10	2000	38
Amazon	16	8300	94

Table 1: The basic of datasets

Experiments

We evaluate our method on several synthetic and real-world datasets, demonstrating its superiority compared to baselines. In addition, analytic studies are conducted to demonstrate the interpretability, robustness, and scalability of our method. All experiments are run on a server with two 3090 GPUs, and for each method we record its averaged performance and standard deviation in three trials.

Implementation Details

Datasets We conducted experiments using both synthetic and real-world data. In particular, the work in (Zhang et al. 2022) generates a set of event sequences by two TPPs (i.e., Hawkes process and In&Ex Process in which the event relation is either inhibition or excitation). Accordingly, we construct a synthetic dataset with two sequence clusters, in which each cluster corresponds to the sequences generated by a specific TPP model. For the real-world data, we consider four representative datasets, including **Taobao** (Xue et al. 2022), **StackOverflow (SO)** (Linderman and Adams 2014), **Taxi** (Whong 2014), and **Amazon** (Ni 2018). The statistics of the datasets can be found in Table 1.

Baselines and Implementation Details We consider representative methods as baselines, including i) learning a single TPP by MLE and ii) learning the state-of-the-art mixture model of TPPs (Zhang et al. 2022). To demonstrate the universality of our method, we consider various TPP models as backbones, including i) the ordinary differential equation-based **ODETPP** (Chen, Amos, and Nickel 2020), ii) the recurrent neural network-based **RMTPP** (Du et al. 2016), and iii) the Transformer-based models (**SAHP** (Zhang et al. 2020), **THP** (Zuo et al. 2020), and **AttNHP** (Yang, Mei, and Eisner 2021)). Applying our semi-tranductive (ST) paradigm to the TPPs leads to a series of ST-TPP models.

Method	Taobao		StackOverflow		Amazon		Taxi		Synthetic ($K = 2$)			
	ELL ↑	ACC ↑	ELL ↑	ACC ↑	ELL ↑	ACC ↑	ELL ↑	ACC ↑	ELL ↑	ACC ↑	NMI ↑	ARI ↑
RMTPP	-0.718$_{0.043}$	0.436$_{0.000}$	-2.756$_{0.009}$	0.425$_{0.003}$	-2.196$_{0.002}$	0.259$_{0.009}$	0.276$_{0.004}$	0.846$_{0.002}$	-0.529$_{0.000}$	0.282$_{0.003}$	0.689$_{0.021}$	0.776$_{0.026}$
ST-RMTPP	**-0.514**$_{0.006}$	**0.436**$_{0.000}$	**-1.597**$_{0.015}$	**0.437**$_{0.010}$	**1.531**$_{0.138}$	**0.311**$_{0.006}$	**0.457**$_{0.005}$	**0.913**$_{0.001}$	**-0.517**$_{0.009}$	**0.313**$_{0.007}$	**0.730**$_{0.008}$	**0.823**$_{0.006}$
SAHP	-0.734$_{0.116}$	0.408$_{0.224}$	-11.956$_{0.547}$	0.021$_{0.003}$	-2.521$_{0.234}$	0.204$_{0.040}$	-0.415$_{0.146}$	0.606$_{0.013}$	-0.547$_{0.014}$	0.268$_{0.002}$	0.651$_{0.024}$	0.755$_{0.022}$
ST-SAHP	**-0.650**$_{0.004}$	**0.437**$_{0.007}$	**-5.386**$_{0.322}$	**0.046**$_{0.002}$	**-1.872**$_{0.024}$	**0.260**$_{0.013}$	**-0.047**$_{0.041}$	**0.877**$_{0.012}$	**-0.503**$_{0.018}$	**0.298**$_{0.007}$	**0.738**$_{0.021}$	**0.830**$_{0.018}$
THP	-0.238$_{0.035}$	0.477$_{0.001}$	-5.543$_{0.017}$	0.380$_{0.002}$	-2.450$_{0.014}$	0.336$_{0.001}$	0.055$_{0.017}$	0.869$_{0.001}$	-0.519$_{0.003}$	0.290$_{0.003}$	0.643$_{0.014}$	0.777$_{0.023}$
ST-THP	**0.234**$_{0.001}$	**0.596**$_{0.002}$	**-2.343**$_{0.011}$	**0.460**$_{0.004}$	**-2.109**$_{0.010}$	**0.383**$_{0.002}$	**0.324**$_{0.003}$	**0.921**$_{0.002}$	**-0.483**$_{0.012}$	**0.316**$_{0.008}$	**0.716**$_{0.014}$	**0.810**$_{0.012}$
ODETPP	-0.448$_{0.015}$	0.436$_{0.000}$	-4.754$_{0.258}$	0.323$_{0.025}$	-2.341$_{0.243}$	0.339$_{0.006}$	0.931$_{0.161}$	0.866$_{0.003}$	0.036$_{0.012}$	0.248$_{0.001}$	0.633$_{0.011}$	0.716$_{0.022}$
ST-ODETPP	**0.795**$_{0.017}$	**0.507**$_{0.002}$	**0.466**$_{0.161}$	**0.371**$_{0.007}$	**-1.102**$_{0.083}$	**0.348**$_{0.006}$	**1.901**$_{0.063}$	**0.909**$_{0.003}$	**0.082**$_{0.021}$	**0.330**$_{0.003}$	**0.700**$_{0.009}$	**0.792**$_{0.018}$
AttNHP	-0.723$_{0.003}$	0.436$_{0.001}$	-6.292$_{0.008}$	0.365$_{0.001}$	-2.584$_{0.001}$	0.299$_{0.001}$	-0.066$_{0.033}$	0.684$_{0.028}$	-0.508$_{0.002}$	0.247$_{0.008}$	0.658$_{0.016}$	0.749$_{0.022}$
ST-AttNHP	**-0.327**$_{0.006}$	**0.501**$_{0.005}$	**-2.466**$_{0.013}$	**0.444**$_{0.001}$	**-0.489**$_{0.014}$	**0.341**$_{0.013}$	**0.155**$_{0.014}$	**0.886**$_{0.014}$	**-0.507**$_{0.001}$	**0.286**$_{0.001}$	**0.665**$_{0.012}$	**0.767**$_{0.011}$

Table 2: Comparison experiments. The standard deviation is shown as the subscript of each result.

For each ST-TPP model, it considers sequence- and cluster-level embeddings when predicting events.

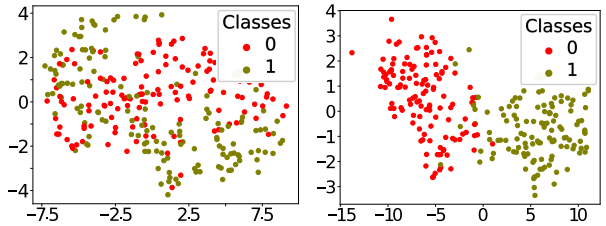
For our method, the bandwidth σ of kernel is a key hyper-parameter. We apply an adaptive method to determine the bandwidth: Given the distances among the event sequences, we empirically set σ based on the median of the distances. For the remaining hyperparameters, e.g., learning rate, batch size, epochs, and so on, we configure them based on the default settings in (Xue et al. 2024) for a fair comparison.

Evaluation Measurements Given a learned model, we use i) the log-likelihood per event (**ELL**) and ii) the prediction accuracy of event types (**ACC**) to evaluate its data fidelity and prediction power, respectively. The visualization of the sequence-level embeddings derived by the model is used to qualitatively evaluate its clustering capability. When the model is learned on synthetic datasets, whose cluster labels are known, we employ i) Adjusted Rand Index (**ARI**) and ii) Normalized Mutual Information (**NMI**) to evaluate the model's clustering performance quantitatively.

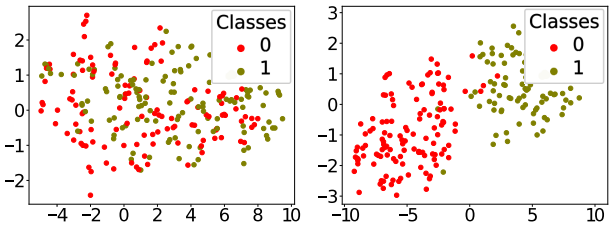
Event Prediction and Sequence Clustering

Numerical Comparisons Table 2 shows the superior performance of our method across different datasets and evaluation metrics. In particular, for each TPP model, learning it by our regularized MLE method and leveraging our semi-tranductive prediction paradigm always improve the event prediction accuracy. These results demonstrate the usefulness of sequence- and cluster-level information for event prediction. Besides the event prediction power, our method also improves the sequence clustering performance of the models, as shown in the results achieved on the synthetic data. It means that the proposed GWB regularizer helps learn structured sequence-level embeddings, which enhances the model interpretability.

Visualization Results To further demonstrate the rationality of our method, we show the t-SNE plots (Maaten and Hinton 2008) of the sequence-level embeddings learned by different methods in Figure 3. According to the visual effects, we can find that the embeddings learned by our method have more distinguishable clustering structures, i.e., the sequence-level embeddings corresponding to different clusters are separated while those within the same cluster are concentrated.



(a) THP v.s. ST-THP

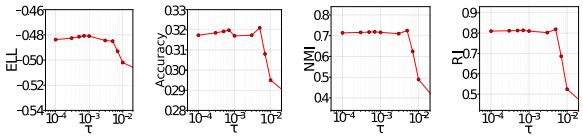


(b) RMTPP v.s. ST-RMTPP

Figure 3: The t-SNE plots of sequence-level embeddings achieved by different methods. In each subfigure, we visualize 256 sequence-level embeddings per cluster.

Analytic Experiments

The Impact of GWB Regularization The weight τ in (13) controls the significance of our GWB regularizer. In Figure 4, we set $\tau \in [1e-4, 1e-2]$ and show the evaluation measurements of our method under different settings. With the increase of τ, the GWB regularizer provides useful information to enhance the clustering structure of the sequence-level embeddings, leading to good NMI and RI results. At the same time, the model can maintain promising ELL and ACC. It means that the GWB regularizer helps improve model interpretability without doing harm to the model prediction power. However, ELL and ACC may drop



(a) ELL (b) ACC (c) NMI (d) RI

Figure 4: The impact of τ when learning ST-THP on the synthetic dataset.

Loss function	ELL ↑	ACC ↑	NMI ↑	RI ↑
MLE+GWB+CE	**-0.483**$_{0.012}$	**0.316**$_{0.008}$	**0.716**$_{0.014}$	**0.810**$_{0.012}$
MLE+GWB	-0.504$_{0.001}$	0.298$_{0.004}$	0.689$_{0.012}$	0.788$_{0.012}$
MLE+CE	-0.496$_{0.003}$	0.306$_{0.004}$	0.675$_{0.014}$	0.771$_{0.009}$
MLE	-0.516$_{0.004}$	0.296$_{0.003}$	0.673$_{0.016}$	0.772$_{0.023}$

Table 3: The impacts of different regularizers on learning THP (Zuo et al. 2020)

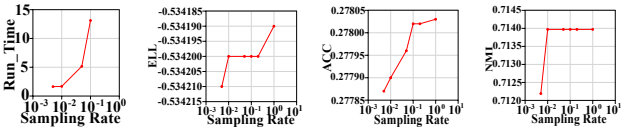
when setting $\tau > 5e-3$. In such a situation, the MLE term is overlooked, and the training data is not well-fitted. According to the results, our method performs robustly when setting τ in a reasonable range.

Note that, our GWB regularizer is compatible with other regularizers. Take THP (Zuo et al. 2020) as an example. Besides MLE, THP enhances its event prediction power by considering a cross-entropy (CE) loss between the normalized intensity functions and the event types in one-hot formats. When implementing our ST-THP, we add this CE-based regularizer into our learning problem as well. Table 3 shows that our GWB regularizer is compatible with the CE-based regularizer. In particular, the MLE in Table 3 means learning ST-TPP by the classic MLE, without the help of the regularizer, and the cluster centers are learned by the spectral clustering of sequence-level embeddings. Both GWB and CE-base regularizers benefit the model performance, and applying them jointly leads to the best learning result. In addition, the results in Table 3 imply that compared to the CE-based regularizer, our GWB regularizer focuses more on enhancing the clustering structure of sequence-level embeddings — the improvements on clustering metrics achieved by GWB regularizer are more significant than those on data fitting and event prediction.

The Impacts of The Embeddings at Different Levels As one of our main contributions, ST-TPP leverages the sequence- and cluster-level embeddings to enhance event prediction. Take The impacts of the embeddings at different levels are shown in Table 4. Both sequence- and cluster-level embeddings help improve model performance, and applying them jointly leads to the best result. These phenomena demonstrate the effectiveness of our semi-transductive prediction paradigm. On the one hand, the sequence-level embedding aggregates the historical event embeddings evenly. Ignoring the decay of events' influences over time, it helps capture the long-range dependency among events within the target sequence. On the other hand, the cluster-level embedding allows the target sequence to leverage the information of training sequences, achieving a new information sharing mechanism based on sequence clustering.

Levels	ELL ↑	ACC ↑	NMI ↑	RI ↑
Event+Seq.+Cluster	**-0.483**$_{0.012}$	**0.316**$_{0.008}$	**0.716**$_{0.014}$	**0.810**$_{0.012}$
Event+Cluster	-0.520$_{0.003}$	0.293$_{0.004}$	0.642$_{0.055}$	0.743$_{0.05}$
Event+Seq.	-0.488$_{0.005}$	0.309$_{0.002}$	0.646$_{0.046}$	0.750$_{0.042}$
Event	-0.511$_{0.005}$	0.291$_{0.002}$	0.642$_{0.048}$	0.742$_{0.032}$

Table 4: The impacts of different embeddings on learning ST-THP on synthetic data.



(a) Runtime (b) ELL (c) ACC (d) NMI

Figure 5: The training time and final model performance with respect to the sampling rate of $\widetilde{\boldsymbol{K}}$ when learning ST-THP on synthetic data.

The Impact of Kernel Approximation Sampling $\widetilde{\boldsymbol{K}}$ helps achieve a trade-off between the scalability and performance of our method. Take ST-THP as an example, we apply our method using different sampling rates. Given M training sequences, we set the sampled kernel size as $L = \text{round}(Mr)$, where $r \in \{0.005, 0.01, 0.05, 0.1\}$ is the sampling rate of the kernel matrix. Figure 5 visualizes the evaluation metrics under different settings. As shown in Figure 5(a), the runtime per epoch of our method quadratically increases with the kernel size due to the complexity of the GW distance. This result matches well with the computational complexity shown in Algorithm 2. Moreover, the learned model obtains promising and stable performance (i.e., ELL, ACC and NMI) even when a small kernel size is applied, demonstrating the rationality of using a small sampling rate in large-scale applications.

Conclusion

We introduced a novel ST-TPP model for event sequence modeling, which enhances event prediction with the help of the clustering information of training data. An MLE method with Gromov-Wasserstein barycentric regularization is proposed to learn the model, which is compatible with stochastic gradient descent and thus applicable to large-scale learning problems. Through extensive evaluations of synthetic and real-world datasets, our method exhibited superior performance compared to state-of-the-art methods. Overall, our work presents a new approach to improving TPPs, which may inspire new learning and prediction paradigms. In the future, we plan to apply our work in practical applications and develop more efficient algorithms to compute GW distance and accelerate the training of the model.

Acknowledgements

This work was supported by NSFC 92270110 and the Fundamental Research Funds for the Central Universities. We also acknowledge the support provided by the fund for building world-class universities (disciplines) of Renmin University of China and by the funds from Beijing Key Laboratory of Research on Large Models and Intelligent Governance, Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education, and from Intelligent Social Governance Interdisciplinary Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China.

References

Alzubaidi, M.; Balasubramanian, V.; Patel, A.; Panchanathan, S.; and Black Jr, J. A. 2012. A novel semi-transductive learning framework for efficient atypicality detection in chest radiographs. In *Medical Imaging 2012: Computer-Aided Diagnosis*, volume 8315, 947–955. SPIE.

Bacry, E.; Mastromatteo, I.; and Muzy, J.-F. 2015. Hawkes processes in finance. *Market Microstructure and Liquidity*, 1(01): 1550005.

Chen, R. T.; Amos, B.; and Nickel, M. 2020. Neural spatio-temporal point processes. *arXiv preprint arXiv:2011.04583*.

Chowdhury, S.; and Needham, T. 2021. Generalized spectral clustering via Gromov-Wasserstein learning. In *International Conference on Artificial Intelligence and Statistics*, 712–720. PMLR.

Courty, N.; Flamary, R.; Tuia, D.; and Rakotomamonjy, A. 2016. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9): 1853–1865.

Du, N.; Dai, H.; Trivedi, R.; Upadhyay, U.; Gomez-Rodriguez, M.; and Song, L. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1555–1564.

Enguehard, J.; Busbridge, D.; Bozson, A.; Woodcock, C.; and Hammerla, N. 2020. Neural temporal point processes for modelling electronic health records. In *Machine Learning for Health*, 85–113. PMLR.

Ge, C.; Wang, J.; Qi, Q.; Sun, H.; Xu, T.; and Liao, J. 2023. Semi-transductive learning for generalized zero-shot sketch-based image retrieval. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 7678–7686.

Gong, S.; Fu, Y.; Ran, F.; Kong, Q.; and Zhou, F. 2025. TPP-SD: Accelerating Transformer Point Process Sampling with Speculative Decoding. *arXiv preprint arXiv:2507.09252*.

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

Isham, V.; and Westcott, M. 1979. A self-correcting point process. *Stochastic processes and their applications*, 8(3): 335–347.

Iwayama, K.; Hirata, Y.; and Aihara, K. 2017. Definition of distance for nonlinear time series analysis of marked point process data. *Physics Letters A*, 381(4): 257–262.

Kingma, D. P.; and Ba, J. 2014. Adam: a method for stochastic optimization. In *International Conference on Learning Representations*.

Kong, Q.; Calderon, P.; Ram, R.; Boichak, O.; and Rizoiu, M.-A. 2023. Interval-censored transformer hawkes: Detecting information operations using the reaction of social systems. In *Proceedings of the ACM Web Conference 2023*, 1813–1821.

Linderman, S.; and Adams, R. 2014. Discovering latent network structure in point process data. In *International conference on machine learning*, 1413–1421. PMLR.

Liniger, T. J. 2009. *Multivariate hawkes processes*. Ph.D. thesis, ETH Zurich.

Luo, D.; Xu, H.; Zhen, Y.; Ning, X.; Zha, H.; Yang, X.; and Zhang, W. 2015. Multi-task multi-dimensional hawkes processes for modeling event sequences. In *Proceedings of the 24th International Conference on Artificial Intelligence*, 3685–3691.

Maaten, L. v. d.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov): 2579–2605.

Mei, H.; and Eisner, J. M. 2017. The neural hawkes process: A neurally self-modulating multivariate point process. *Advances in neural information processing systems*, 30.

Mémoli, F. 2011a. Gromov–Wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 11: 417–487.

Mémoli, F. 2011b. A spectral notion of Gromov–Wasserstein distance and related methods. *Applied and Computational Harmonic Analysis*, 30(3): 363–401.

Ng, A.; Jordan, M.; and Weiss, Y. 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14.

Ni, J. 2018. Amazon review data. *Online abrufbar unter: https://nijianmo. github. io/amazon/index. html*.

Peyré, G.; Cuturi, M.; and Solomon, J. 2016. Gromov-wasserstein averaging of kernel and distance matrices. In *International conference on machine learning*, 2664–2672. PMLR.

Peyré, G.; Cuturi, M.; et al. 2019. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6): 355–607.

Wang, Q.; Cheng, M.; Yuan, S.; and Xu, H. 2023. Hierarchical Contrastive Learning for Temporal Point Processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Wang, Q.; Wu, Y.; Long, Y.; Huang, J.; Ran, F.; Su, B.; and Xu, H. 2025. A Plug-and-Play Bregman ADMM Module for Inferring Event Branches in Temporal Point Processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 21216–21224.

Whong, C. 2014. FOILing NYC's taxi trip data. *FOILing NYCs Taxi Trip Data. Np*, 18: 14.

Xiao, S.; Farajtabar, M.; Ye, X.; Yan, J.; Song, L.; and Zha, H. 2017. Wasserstein learning of deep generative point process models. *Advances in neural information processing systems*, 30.

Xu, H.; Luo, D.; and Carin, L. 2019. Scalable gromov-wasserstein learning for graph partitioning and matching. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 3052–3062.

Xu, H.; Luo, D.; and Zha, H. 2021. Hawkes processes on graphons. *arXiv preprint arXiv:2102.02741*.

Xu, H.; Wu, W.; Nemati, S.; and Zha, H. 2016. Patient flow prediction via discriminative learning of mutually-correcting processes. *IEEE transactions on Knowledge and Data Engineering*, 29(1): 157–171.

Xu, H.; and Zha, H. 2017a. A dirichlet mixture model of hawkes processes for event sequence clustering. *Advances in neural information processing systems*, 30.

Xu, H.; and Zha, H. 2017b. THAP: A matlab toolkit for learning with Hawkes processes. *arXiv preprint arXiv:1708.09252*.

Xue, S.; Shi, X.; Chu, Z.; Wang, Y.; Hao, H.; Zhou, F.; Jiang, C.; Pan, C.; Zhang, J. Y.; Wen, Q.; Zhou, J.; and Mei, H. 2024. EasyTPP: Towards Open Benchmarking Temporal Point Processes. In *International Conference on Learning Representations (ICLR)*.

Xue, S.; Shi, X.; Zhang, J.; and Mei, H. 2022. Hypro: A hybridly normalized probabilistic model for long-horizon prediction of event sequences. *Advances in Neural Information Processing Systems*, 35: 34641–34650.

Yang, C.; Mei, H.; and Eisner, J. 2021. Transformer embeddings of irregularly spaced events and their participants. *arXiv preprint arXiv:2201.00044*.

Zhang, Q.; Lipani, A.; Kirnap, O.; and Yilmaz, E. 2020. Self-attentive Hawkes process. In *International conference on machine learning*, 11183–11193. PMLR.

Zhang, Y.; Yan, J.; Zhang, X.; Zhou, J.; and Yang, X. 2022. Learning mixture of neural temporal point processes for multi-dimensional event sequence clustering. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, Vienna, Austria*, 23–29.

Zhao, Q.; Erdogdu, M. A.; He, H. Y.; Rajaraman, A.; and Leskovec, J. 2015. Seismic: A self-exciting point process model for predicting tweet popularity. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 1513–1522.

Zuo, S.; Jiang, H.; Li, Z.; Zhao, T.; and Zha, H. 2020. Transformer hawkes process. In *International conference on machine learning*, 11692–11702. PMLR.