SMURF-THP: Score Matching-based UnceRtainty quantiFication for Transformer Hawkes Process

Zichong Li [1] **Yanbo Xu** [2] **Simiao Zuo** [2] **Haoming Jiang** [3] **Chao Zhang** [2] **Tuo Zhao** [2] **Hongyuan Zha** [4]

Abstract

Transformer Hawkes process models have shown to be successful in modeling event sequence data. However, most of the existing training methods rely on maximizing the likelihood of event sequences, which involves calculating some intractable integral. Moreover, the existing methods fail to provide uncertainty quantification for model predictions, e.g., confidence intervals for the predicted event's arrival time. To address these issues, we propose SMURF-THP, a score-based method for learning Transformer Hawkes process and quantifying prediction uncertainty. Specifically, SMURF-THP learns the score function of events' arrival time based on a score-matching objective that avoids the intractable computation. With such a learned score function, we can sample arrival time of events from the predictive distribution. This naturally allows for the quantification of uncertainty by computing confidence intervals over the generated samples. We conduct extensive experiments in both event type prediction and uncertainty quantification of arrival time. In all the experiments, SMURF-THP outperforms existing likelihood-based methods in confidence calibration while exhibiting comparable prediction accuracy.

1. Introduction

Sequences of discrete events in continuous time are routinely generated in many domains such as social media (Yang et al., 2011), financial transactions (Bacry et al., 2015), and personalized treatments (Wang et al., 2018). A common question to ask is: Given the observations of the past events, when and what type of event will happen next? Temporal point process such as the Hawkes process (Hawkes, 1971) uses an *intensity function* to model the arrival time of events. A conventional Hawkes process defines a parametric form of the intensity function, which assumes that the influence of past events on the current event decreases over time. Such a simplified assumption limits the model's expressivity and fails to capture complex dynamics in modern intricate datasets. Recently, neural Hawkes process models parameterize the intensity function using recurrent neural networks (Du et al., 2016; Mei & Eisner, 2017) and Transformer (Zuo et al., 2020; Zhang et al., 2020). These models have shown to be successful in capturing complicated event dependencies.

However, there are two major drawbacks with existing training methods for neural Hawkes process models. First, they mainly focus on maximizing the log-likelihood of the events' arrival time and types. To calculate such likelihood, we need to compute the integral of the intensity function over time, which is usually computationally challenging, and therefore numerical approximations are often applied. As a result, not only is the computation complex, but the estimation of the intensity function can still be inaccurate in these likelihood-based methods. Second, the arrival time of events often follows a long tail distribution, such that point estimates of the arrival time are unreliable and insufficient. Thus, confidence intervals that can quantify model uncertainty are necessary. However, calculating the confidence interval directly from density is computationally challenging. This is because we need to compute the coverage probability, which is essentially another integral of the intensity function (recall that the intensity is an infinitesimal probability).

To overcome the above two challenges, we propose SMURF-THP, a score-based method for learning Transformer Hawkes process models and quantifying the uncertainty of the models' predicted arrival time. The score-based objective in SMURF-THP, originally introduced by Hyvärinen & Dayan (2005), is to match the derivative of the log-likelihood (known as *score*) of the observed events' arrival time to the derivative of the log-density of the empirical (unknown) distribution. Note that calculating scores

[1]University of Science and Technology of China, Hefei, China [2]Georgia Institute of Technology, Atlanta, USA [3]Amazon.com Inc., Palo Alto, USA [4]The Chinese University of Hong Kong, Hongkong, China. Correspondence to: Zichong Li <lzc123@mail.ustc.edu.cn>, Tuo Zhao <tourzhao@gatech.edu>, Hongyuan Zha <zhahy@cuhk.edu.cn>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

no longer requires computing the intractable integral in the likelihood function. Moreover, SMURF-THP is essentially a score-based generative model: With the neural score function learnt by SMURF-THP, we can generate samples from the predicted distribution of the arrival time. This naturally allows for the quantification of uncertainty in predictions by computing confidence intervals over the generated samples.

Note that Transformer Hawkes process jointly models event types and arrival time. Because score matching methods can only be applied to handle the continuous arrival time but not the categorical event types, we need to decompose the joint likelihood into a marginal likelihood of the arrival time and a partial likelihood of the types. When training the Transformer Hawkes process models using SMURF-THP, the objective becomes a weighted sum of the score-matching loss (which replaces the marginal likelihood of the arrival time) and the negative partial log-likelihood of event types.

We demonstrate the effectiveness of SMURF-THP for learning Transformer Hawkes process models on four real-world datasets, each of which contains sequences of multiple types of discrete events in continuous time. We decompose the joint likelihood by conditioning event type on the arrival time and consider a Transformer Hawkes process where one single intensity function is shared across all event types. Our experimental results show that compared with likelihood-based methods, SMURF-THP demonstrates superior performance on uncertainty quantification of arrival time, while exhibiting comparable event-type prediction accuracy. For example, SMURF-THP achieves within 0.3% prediction accuracy compared with likelihood-based methods, while gaining 4-10% confidence calibration in predictions of arrival times.

The rest of the paper is organized as follows: Section 2 briefly reviews the background, Section 3 describes our proposed method, Section 4 presents the experimental results, and Section 5 discusses the related work and draws a brief conclusion.

2. Background

We briefly review the neural Hawkes process models, the Transformer model and the score matching method.

2.1. Neural Hawkes Process

For simplicity, we review neural Hawkes process with only one event type. And we will introduce Hawkes process with multiple event types (i.e., marked point process) in the next section. Let $\mathcal{S} = \{t_i\}_{i=1}^L$ denote an event sequence with length L, where each event is observed at time $t_i \in [0, t_{\max}]$. We further denote the history up to time t as $\mathcal{H}_t = \{t_j : t_j < t\}$, and the conditional intensity function as $\lambda(t \mid \mathcal{H}_t)$. Then, we have the conditional probability of the next event proceeding t_{i-1} as:

$$p_T(t \mid \mathcal{H}_{t_i}) = \lambda(t \mid \mathcal{H}_{t_i}) e^{-\int_{t_{i-1}}^{t} \lambda(\tau|\mathcal{H}_{t_i}) d\tau}. \quad (1)$$

The log-likelihood of the event sequence $\mathcal{S}$ is then

$$\ell_0(\mathcal{S}) = \sum_{i=1}^{L} \log \lambda(t_i|\mathcal{H}_{t_i}) - \int_0^{t_{\max}} \lambda(\tau|\mathcal{H}_{t_i}) d\tau. \quad (2)$$

Notice that exact computation of the integral in Eq. (1) is intractable, such that numerical approximations are required.

Neural Hawkes Process parameterizes the intensity function λ with deep neural networks. For example, Du et al. (2016); Mei & Eisner (2017) use variants of recurrent neural networks, and Zuo et al. (2020); Zhang et al. (2020) use the Transformer model (Vaswani et al., 2017). Other examples include employments of deep Fourier kernel (Zhu et al., 2021) and feed-forward neural networks (Omi et al., 2019).

2.2. Transformer Model

In this paper, we parameterize the intensity function λ in Eq. (1) using a Transformer model (Vaswani et al., 2017). The Transformer model has multiple Transformer layers, where each layer contains a self-attention mechanism and a feed-forward neural network. The self-attention mechanism assigns attention weights to every pair of events in a sequence. These weights signify the strength of the dependency between pairs of events, with smaller weights indicating weaker dependencies and larger weights suggesting stronger ones. Such mechanism efficiently models the dependencies between events irrespective of their position in the sequence, thereby capturing long-term effects. The feed-forward neural network in each layer further incorporates non-linearities to offer a larger model capacity for learning complex patterns. Studies have shown that the Transformer model outperforms recurrent neural networks in modeling event sequences (Zuo et al., 2020; Zhang et al., 2020).

2.3. Score Matching

Score matching (Hyvärinen, 2005) is originally designed for estimating non-normalized statistical methods on $\mathbb{R}^n$ without challenging computation of the normalization term. In practice, however, score matching is not scalable to high-dimensional data and deep neural networks due to the calculation of derivative of density. Denoising score matching (Vincent, 2011) is one of its variants that circumvent the derivative by adding perturbation to data points. Song et al. (2019) also enhance the scalability by projecting the score function onto random vectors. Yu et al. (2019) and Yu et al. (2022) further generalize the origin score matching to accommodate densities supported on a more general class of

domains. Early work (Sahani et al., 2016) has adopted score matching in modeling Poisson point process, but it fails to model complicated event dependencies induced in the modern event data due to the simplified assumption that the intensity function is independent of the historical events. A suitable sampling algorithm for score matching-based models is *Langevin Dynamics (LD)*, which can produce samples from a probability density using only its score function. Hsieh et al. (2018) propose Mirror Langevin Dynamics (MLD) as a variant of Langevin Dynamics that focuses on sampling from a constrained domain. We employs both LD and MLD for generating event samples in our experiments.

3. Method

3.1. Score Matching Objective of Hawkes Process

Let $\mathcal{S} = \{(t_i, k_i)\}_{i=1}^{L}$ denote an event sequence of length L, where each pair (t_i, k_i) corresponds to an event of type $k_i \in \{1, \cdots, M\}$ happened at time $t_i \in [0, t_{\max}]$. Also, we denote the history events up to time t as $\mathcal{H}_t = \{(t_j, k_j) : t_j < t\}$. Our goal is to learn $p_{T,K}(t, k \mid \mathcal{H}_t)$, the joint conditional probability of the event proceeding time t given the history $\mathcal{H}_t$.

To employ score matching, we decompose the joint conditional pdf $p_{T,K}$ by conditioning on the event time. By doing so, the partial likelihood of the discrete event types can be maximized by minimizing the cross entropy loss; and the marginal likelihood of the continuous event time can be substituted by a score-matching objective. Such a substitution avoids the intractable integral of the intensity. Specifically, we condition $p_{T,K}$ on the event time and we have:

$$p_{T,K}(t, k \mid \mathcal{H}_t) = p_T(t \mid \mathcal{H}_t) \cdot p_{K \mid T}(k \mid t, \mathcal{H}_t).$$

Correspondingly, we have the log-likelihood:

$$\ell(\mathcal{S}) = \sum_{i=1}^{L} \log p_T(t_i | \mathcal{H}_{t_i}) + \sum_{i=1}^{L} \log p_{K|T}(k_i | t_i, \mathcal{H}_{t_i}). \quad (3)$$

We can use a neural network to parameterize the intensity function and directly train the model using Eq. (3). However, such an approach inevitably faces computational challenges, i.e., exact computation of the intensity's integral is intractable. Therefore, we derive a score-matching objective to substitute the first term in Eq. (3).

In SMURF-THP, we use a Transformer model with parameters θ to parameterize the intensity function. More details are presented in Section 3.3. A sample's *score* is defined as the gradient of its log-density. Then, using Eq. (1), we can write the score of the i-th event given its history $\mathcal{H}_{t_i}$ and the model parameters θ as:

$$\begin{aligned}\psi(t_i \mid \mathcal{H}_{t_i}; \theta) &= \partial_t \log p_T(t_i \mid \mathcal{H}_{t_i}; \theta) \\ &= \partial_t \log \lambda(t_i \mid \mathcal{H}_{t_i}; \theta) - \lambda(t_i \mid \mathcal{H}_{t_i}; \theta).\end{aligned}$$

The original objective of score matching is to minimize the expected squared distance between the score of the model $\psi(\cdot; \theta)$ and the score of the ground truth $\psi^*(\cdot)$. However, minimizing such an objective is infeasible since it relies on the unknown score $\psi^*(\cdot)$. We can resolve this issue by following the general derivation in Hyvärinen (2005) and arrive at an empirical score-matching objective for Hawkes process with single type:

$$\widehat{J}(\theta) = \sum_{i=1}^{L} [\frac{1}{2} \psi(t_i \mid \mathcal{H}_{t_i}; \theta)^2 + \partial_t \psi(t_i \mid \mathcal{H}_{t_i}; \theta)], \quad (4)$$

where $\partial_t \psi(t_i \mid \mathcal{H}_{t_i}; \theta) = \partial_t^2 \log \lambda(t_i \mid \mathcal{H}_{t_i}; \theta) - \partial_t \lambda(t_i \mid \mathcal{H}_{t_i}; \theta)$. We state in the follow theorem that the score matching objective in Eq. (4) satisfies local consistency: minimizing $\widehat{J}(\theta)$ is as sufficient as maximizing the first term of Eq. (3) for estimating the model parameters.

Theorem 3.1. *Assume the event time in sequence $\mathcal{S}$ follows the model: $p_T^*(t \mid \mathcal{H}_t) = p_T(t \mid \mathcal{H}_t; \theta^*)$ for some θ^*, and that no other parameter gives a pdf that is equal [1] to $p_T(\cdot; \theta^*)$. Assume further that the optimization algorithm is able to find the global minimum and $p_T(t \mid \mathcal{H}_t; \theta)$ is positive for all t and θ. Then the score matching estimator obtained by minimizing Eq. (4) is consistent, i.e., it converges in probability towards θ^* when sample size approaches infinity.*

Proof. Let $\psi^*(\cdot)$ and $\psi(\cdot; \theta)$ be the associated score function of $p_T^*(\cdot)$ and $p_T(\cdot; \theta)$, respectively. The objective in Eq. (4) is an empirical estimator of the following objective:

$$J(\theta) = \frac{1}{2} \left\langle \sum_{i=1}^{L} (\psi(t_i \mid \mathcal{H}_{t_i}; \theta) - \psi^*(t_i \mid \mathcal{H}_{t_i}))^2 \right\rangle_{\{t_i\}_{i=1}^{L} \sim p_T^*}.$$

We first prove that $J(\theta) = 0 \Rightarrow \theta = \theta^*$. Since $p_T^*(\cdot)$ is positive, we can infer from $J(\theta') = 0$ that $\psi(\cdot; \theta')$ and $\psi^*(\cdot)$ are equal, which implies $\log p_T^*(\cdot) = \log p_T(\cdot; \theta') + c$ for some constant c. Because both p_T^* and p_T are pdf's, the constant c must be 0 and hence we have $p_T^* = p_T$. By assumption, θ^* is the only parameter that fulfills this equation, so necessarily $\theta' = \theta^*$.

Then according to the law of large numbers, $\widehat{J}$ converges to J as the sample size approaches infinity. Thus, the estimator converges to a point where J is globally minimized.

[1] Equality of pdf's are taken in the sense of equal almost everywhere with respect to the Lebesgue measure.

Considering $J(\theta) = 0 \Rightarrow \theta = \theta^*$, the minimum is unique and must be found at the true parameter θ^*. □

We highlight that by substituting the first term of Eq. (3) with the score-matching objective in Eq. (4), we no longer need to compute the intractable integral. For the second term in Eq. (3), we employ a neural network classifier $g(t_i, \mathcal{H}_{t_i}; \omega)$ parameterized by ω. The classifier takes the timestamp t_i and its history $\mathcal{H}_{t_i}$ as inputs, and the classifier outputs an event type distribution of the next event proceeding time t_i. We can learn the model parameters $[\theta; \omega]$ by minimizing the following loss function

$$\mathcal{L}(\mathcal{S}) = \alpha \widehat{J}(\theta) + \sum_{i=1}^{L} \mathrm{CE}(g(t_i, \mathcal{H}_{t_i}; \omega), k_i),$$

where α is a hyperparameter that controls the weight of the score matching objective and CE is the cross entropy loss.

3.2. Training: Denoising Score Matching

In practice, the direct score-based approach has limited success because of two reasons. First, the estimated score functions are inaccurate in low-density regions (Song & Ermon, 2019). Second, the derivatives of the score functions in Eq. (4) contain second-order derivatives, leading to numerical instability.

To alleviate these issues, we adopt denoising score matching (Vincent, 2011), which perturbs the data with a pre-specified noise distribution. Specifically, we add a Gaussian noise $\mathcal{N}(0, \sigma)$ to each observed event time t_i, obtaining a noise-perturbed distribution $q_\sigma(t \mid t_i)$, the intent of which is to augment samples in the low density regions of the original data distribution. Moreover, the denoising score matching objective circumvents computing second-order derivatives by replacing them with the scores of the noise distributions. Suppose for the i-th event, we have S perturbed event time denoted by $\{t_i^{\sigma,j}\}_{j=1}^{S}$, then we can substitute the objective $\widehat{J}(\theta)$ in Eq. (4) by

$$\widehat{J}^{\sigma}(\theta) = \sum_{i=1}^{L} \sum_{j=1}^{S} \frac{1}{2} \|\psi(t_i^{\sigma,j} \mid \mathcal{H}_{t_i}; \theta)^2 - \partial_t \log q_\sigma(t_i^{\sigma,j} \mid t_i)\|^2.$$

Here, $\partial_t \log q_\sigma(t \mid t_i) = \frac{t_i - t}{\sigma^2}$ is the score of the noise distribution, which directs the perturbed timestamps to move towards the original time t_i.

3.3. Parametrization

We now introduce how we construct our model. We follow Transformer Hawkes Process (Zuo et al., 2020; Zhang et al., 2020) that leverages self-attention mechanism to extract features from sequence data. Given input sequence $\mathcal{S}$, each event are firstly encoded by the summation of temporal encoding and event type embedding (Zuo et al., 2020). Then we pass the encoded sequence $\mathbf{C} \in \mathbb{R}^{L \times D}$ through the self-attention module that compute the multi-head attention output $\mathbf{A}$ as

$$\mathbf{A} = \mathrm{Concat}(\mathbf{A}_1, ..., \mathbf{A}_H)\mathbf{W}^O,$$
$$\mathbf{A}_h = \mathrm{Softmax}\left(\frac{(\mathbf{C}\mathbf{W}_h^Q)^\top \mathbf{C}\mathbf{W}_h^K}{\sqrt{d_k}}\right)\mathbf{C}\mathbf{W}_h^V.$$

Matrices $\mathbf{W}_h^Q, \mathbf{W}_h^K \in \mathbb{R}^{d \times d_k}$ and $\mathbf{W}_h^V \in \mathbb{R}^{d \times d_v}$ are query, key, value projections, and $\mathbf{W}^O \in \mathbb{R}^{(d_v * H) \times d}$ aggregates the final attention outputs. Then $\mathbf{A}$ is fed through a position-wise feed-forward neural network (FFN) to obtain the hidden representations as

$$\mathrm{FFN}(\mathbf{A}) = \max(0, \mathbf{A}\mathbf{W}_1^{\mathrm{FC}} + \mathbf{b_1})\mathbf{W}_2^{\mathrm{FC}} + \mathbf{b_2},$$

where the jth row of $\mathrm{FFN}(\mathbf{A})$ encodes the jth event and all past events up to time t_j. In practice, we stack multiple self-attention modules and position-wise feed-forward networks to construct a model with a larger capacity. We add future masks while computing the attention to avoid encoding future events.

After generating hidden representations for event sequence, we parametrize the total intensity function λ as f and the conditional distribution of next event type $p_{K \mid T}$ as g. The parametric functions are defined by

$$f(t, \mathcal{H}_{t_{j+1}}) = \mathrm{Tanh}(\mathbf{h}(j)[\mathbf{W}_1^f(t - t_j) + \mathbf{W}_2^f] + \mathbf{b}_1^f)\mathbf{w}_3^{f^\top} + \mathbf{b}_2^f,$$
$$g(t, \mathcal{H}_{t_{j+1}}) = \mathrm{Softmax}(\mathbf{h}(j)\,[\mathbf{W}_1^g(t - t_j) + \mathbf{W}_2^g] + \mathbf{b}^g),$$

where $\mathbf{W}_1^f, \mathbf{W}_2^f, \mathbf{w}_3^f, \mathbf{W}_1^g, \mathbf{W}_2^g$ are trainable weight matrix.

3.4. Uncertainty Quantification

Using the learnt score function, we can generate new events using the Langevin Dynamics (LD) and compute confidence intervals. Without loss of generality, we denote the learnt score function as $\psi(t \mid \mathcal{H}_t) = \partial_t \log p(t \mid \mathcal{H}_t)$. Suppose we aim to generate a sample for the ith event, we first generate an initial time gap $t^{(0)} \sim \pi(t)$ with π being a uniform prior distribution. Then, the LD method recursively updates the time gap given history $\mathcal{H}_{t_i}$ by

$$t^{(n)} = t^{(n-1)} + \frac{\epsilon}{2}\psi(t_{i-1} + t^{(n-1)} \mid \mathcal{H}_{t_i}) + \sqrt{\epsilon} z_n.$$

Here, ϵ is a step size and we have $z_n \sim \mathcal{N}(0, 1)$ for $n = 1, \cdots, N$. The distribution of $t^{(N)}$ is proved to approximate

the predicted distribution as $\epsilon \to 0$ and $N \to \infty$ under certain regularity conditions (Welling & Teh, 2011). Note that negative time gap can appear in both the perturbation process and the sampling process due to the added random Gaussian noise. This violates the constraint that an event's arrival time should be non-decreasing. To resolve this issue, we adopt Tweedie's formula (Efron, 2011) and add an extra denoising step after the LD procedure:

$$\widehat{t} = t^{(N)} + \sigma\psi(t_{i-1} + t^{(N)} \mid \mathcal{H}_{t_i}).$$

Here, σ is the standard deviation of the Gaussian noises. The final event time sample is then $t_{i-1} + \widehat{t}$.

Using the above sampling algorithm, we can generate event time samples under different histories for uncertainty quantification. As for event type, we sample from the learnt conditional density function $g(t_{i-1} + \widehat{t}, \mathcal{H}_{t_i}; \widehat{\omega})$ for each time sample.

4. Experiments

We demonstrate the effectiveness of our method on four real-world datasets with four baselines. Our code is publicly available at https://github.com/zichongli5/SMURF-THP.

4.1. Setup

Datasets. We experiment on four real-world datasets. Table 1 summarizes their statistics.

Table 1. Datasets statistics. From left to right columns: name of the dataset, number of event types, number of events, and average length per sequence.

Dataset	#Type	#Event	Avg. Length
StackOverflow	22	480413	64
Retweet	3	2173533	109
MIMIC-II	75	2419	4
Financial	2	414800	2074

StackOverflow (Leskovec & Krevl, 2014) StackOverflow is a website that serves as a platform for users to ask and answer questions. Users will be awarded badges based on their proposed questions and their answers to others. This dataset contains sequences of $6,633$ users' reward history in a two-year period, where each event signifies receipt of a particular type of medal.

Retweet (Zhao et al., 2015) This dataset contains $24,000$ sequences of tweets. Each sequence starts with an original tweet at time 0, and the following events signify retweeting by other users. All users are grouped into three categories based on the number of their followers. So each event is labeled with the retweet time and the retweeter's group.

MIMIC-II (Du et al., 2016) This dataset contains a subset of 650 patient's clinical visits to the Intensive Care Units in a seven-year period. Each sequence consist of visits from one particular patient, and each event is labeled with a timestamp and a set of diagnosis codes.

Financial Transactions (Du et al., 2016) This dataset contains a total of 0.7 million transaction records for a stock from the New York Stock Exchange. The long single sequence of transactions are partitioned into $2,000$ subsequences, where each event is labeled with the transaction time and the action that was taken: buy or sell.

For all the aforementioned datasets, we adhere to the same data pre-processing and train-dev-test partitioning as described in (Du et al., 2016) and (Mei & Eisner, 2017). Additionally, we apply normalization and log-normalization techniques to enhance performance. In particular, we normalize the event time in the MIMIC-II dataset, while for the Retweet and Financial datasets, we perform log-normalization of time using the formula $\frac{\log(t) - \text{Mean}(\log(t))}{\text{Var}(log(t))}$. During the testing phase, we rescale the generated events' timestamps to their original scale and assess their quality.

Baselines. We compare our method to four existing works:

- Neural Hawkes Process (NHP, Mei & Eisner (2017)), which designs a continuous-time LSTM to model the evolution of the intensity function between events;
- Noise-Contrastive Estimation for Temporal Point Process (NCE-TPP, Mei et al. (2020)), which adopts the noise-contrastive estimator to bypass the computation of the intractable normalization constant;
- Self-Attentive Hawkes Process (SAHP, Zhang et al. (2020)), which proposes a time-shifted positional encoding and adopts self-attention to model the intensity function;
- Transformer Hawkes Process (THP, Zuo et al. (2020)), which adopts the Transformer architecture to capture long-term dependencies in history and designs a continuous formulation for the intensity function.

For a fair comparison, we employ the same backbone architecture as THP and use default hyperparameters for NHP and SAHP. We implement NCE-TPP using the Transformer backbone in place of its original LSTM structure. Because these methods do not support uncertainty quantification originally, we calculate the scores from their estimated intensity functions that were learnt through maximum likelihood or noise-contrastive estimation. Then we can follow the similar LD sampling procedure of ours to generate samples using these baseline models.

Metrics. We apply five metrics for measuring the quality of the generated event samples:

• *Calibration Score* calculates the Root Mean Squared Error (RMSE) between the coverage of the confidence intervals produced by the samples and that of the desired probabilities. Specifically, with U generated samples $\{\widehat{t}_i^j\}_{j=1}^U$ for each time point t_i, we can estimate a q_s-confidence interval as $[0, t_i^{q_s}]$, where $t_i^{q_s}$ is the q_s-quantile of the U samples. Then we can compute the coverage of the estimated q_s-intervals by counting how many times the true value t_i falls in the estimated intervals, i.e., $c_s = \sum_{i=1}^L \mathbb{I}_{t_i < t_i^{q_s}}$. Calibration Score is defined as the average RMSE between the coverage and the confidence level q_s:

$$\mathrm{CS} = \sqrt{\frac{1}{S}\sum_{s=1}^{S}(c_s - q_s)^2}.$$

We consider to compute CS scores at quantiles of $\{0.5, 0.55, ..., 0.95\}$ because the samples generated below 0.5 quantiles are less useful and can be still noisy even using the denoising sampling method;

• *Continuous Ranked Probability Score (CRPS)* measures the compatibility of the estimated cumulative distribution function (cdf) $\widehat{F}$ per time t_i as

$$\mathrm{CRPS}(\widehat{F}, t_i) = \int_0^\infty (\widehat{F}(t) - \mathbb{I}_{t \geq t_i})^2 dt.$$

We compute the empirical CRPS (Jordan et al., 2017) for time t_i from the samples by

$$\mathrm{CRPS} = \frac{1}{LU}\sum_{i=1}^{L}[\sum_{j=1}^{U}|\widehat{t}_i^j - t_i| - \frac{1}{2U}\sum_{j=1}^{U}\sum_{k=1}^{U}|\widehat{t}_i^j - \widehat{t}_i^k|].$$

The total CRPS is then defined as the average among the empirical CRPS of all events.

• *Interval Length (IL)* computes the average length of the predicted intervals for a particular confidence level q_s as

$$\mathrm{IL} = \frac{1}{L}\sum_{i=1}^{L} t_i^{q_s}.$$

• *Coverage Error (CER)* calculates the mean absolute error between a coverage c_s and the corresponding confidence level q_s as

$$\mathrm{CER} = |c_s - q_s|.$$

We report IL and CER at $q_s = 0.5$ in the main results. For all of the above metrics, lower is considered "better".

• *Type Prediction Accuracy* measures the quality of the generated event's types. We take the mode of the sampled types as the prediction and then compute the accuracy.

4.2. Results

Comparison of Uncertainty Quantification. We compare different methods' performance in Table 2. We can see that SMURF-THP outperforms other baselines by large margins in terms of CS and CER. It also achieves the lowest CRPS and IL at confidence level of 0.5 for most datasets. This indicates that SMURF-THP provides the most precise and concurrently, the narrowest confidence intervals. The improvements come from two reasons: 1) SMURF-THP can generate samples in higher quality using the scores that are more accurately learned by minimizing the score matching objective. In contrast, the baselines do not initially support sampling and their scores are less accurate as they are derived from the intensity functions that are estimated by maximizing an approximated log-likelihood. 2) SMURF-THP can utilize the denoising score matching method to perturb the data points and effectively reduce the area of low-density regions in the original data space, whereas other baselines do not support this.

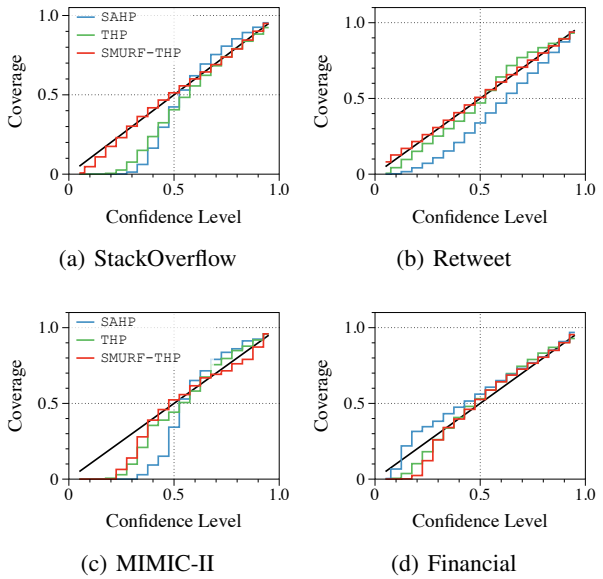
Comparison of Event Type Prediction. We also compare the event type prediction accuracy of all the methods in Table 2. NHP, NCE-TPP and SAHP predict event type by the estimated expectation of intensity, while THP and our method adopt an additional classifier for prediction. As depicted in the table, SMURF-THP attains slightly lower accuracy compared to THP, but surpasses other baselines. We attribute the accuracy loss to the fact that SMURF-THP is trained to predict event type given a ground truth timestamp but is tested on a sampled timestamp since it is unknown in the testing. As elaborated in Section 4.3, SMURF-THP can achieve comparable or even superior accuracy when tested using true timestamps.

Different Confidence Level. Figure 1 and Figure 2 compare the coverage and the interval length of the confidence intervals at different levels across four datasets. The black line in Figure 1 signifies the desired coverage. Compared with THP and SAHP, our method is much closer to the black line, which indicates that the generated confidence intervals are more accurate. Notice that the coverage error on small probabilities is larger than that on large probabilities. The reason is that sampling around 0 is more challenging as the Langevin step size is relatively larger and the score function changes more rapidly. SMURF-THP effectively mitigates such bias by learning a more precise event distribution. Figure 2 reveals that SMURF-THP also achieves a shorter interval length than THP and SAHP for most of the assessed confidence levels. Despite SMURF-THP experiencing longer intervals on small probability for the StackOverflow dataset, the corresponding coverage of the baselines is less precise. The improvement on the Retweet dataset is particularly notable since its distribution of event time exhibits a longer tail, where adding perturbation is

Table 2. Comparison of different methods' performance on four real-world datasets in terms of Calibration Score (CS), CRPS, Coverage Error (CER) and Interval Length (IL). CER and IL are both calculated at confidence level 0.5.

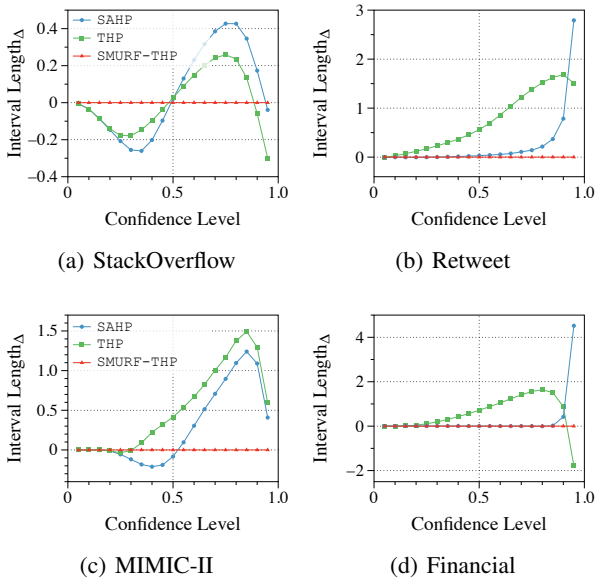
	StackOverflow					Retweet				
Methods	CS(%)(↓)	CRPS(↓)	CER(%)(↓)	IL(↓)	Acc(%)(↑)	CS(%)(↓)	CRPS(↓)	CER(%)(↓)	IL(↓)	Acc(%)(↑)
NHP	6.64±0.55	0.69±0.01	5.78±0.26	0.67±0.04	46.20±0.13	14.35±0.22	1.65±0.05	16.98±0.24	0.077±0.005	60.28±0.08
NCE-TPP	5.64±0.46	0.62±0.01	5.99±0.39	0.63±0.04	46.20±0.15	13.55±0.10	1.59±0.05	10.30±0.16	0.055±0.002	60.30±0.06
SAHP	4.48±0.24	0.48±0.02	7.73±0.47	0.55±0.01	46.22±0.06	10.06±0.35	1.12±0.02	15.65±0.03	0.061±0.001	60.32±0.08
THP	4.13±0.25	0.46±0.01	5.90±0.33	0.56±0.02	**46.48**±0.05	4.12±0.13	1.08±0.05	3.00±0.02	0.059±0.002	**60.63**±0.13
SMURF-THP	**0.65**±0.12	**0.44**±0.01	**0.48**±0.09	**0.52**±0.01	46.26±0.08	**0.71**±0.16	**0.86**±0.04	**0.76**±0.03	**0.031**±0.001	60.34±0.12
	MIMIC-II					Financial				
Methods	CS(%)(↓)	CRPS(↓)	CER(%)(↓)	IL(↓)	Acc(%)(↑)	CS(%)(↓)	CRPS(↓)	CER(%)(↓)	IL(↓)	Acc(%)(↑)
NHP	9.87±0.35	0.74±0.03	8.85±0.29	0.69±0.04	83.25±0.27	4.64±0.42	1.03±0.01	3.85±0.30	0.053±0.002	60.23±0.05
NCE-TPP	6.52±0.24	0.65±0.01	9.23±0.38	0.77±0.04	83.60±0.32	3.55±0.12	1.35±0.05	4.09±0.04	0.055±0.04	60.28±0.09
SAHP	7.49±0.32	0.61±0.01	15.70±0.52	**0.35**±0.01	83.76±0.13	3.50±0.33	1.09±0.05	3.19±0.06	0.048±0.003	60.37±0.08
THP	3.89±0.14	0.75±0.05	5.81±0.12	0.84±0.03	**84.78**±0.13	3.53±0.19	1.57±0.02	3.19±0.13	0.042±0.006	60.51±0.06
SMURF-THP	**2.87**±0.11	**0.55**±0.04	**2.33**±0.09	0.43±0.03	84.02±0.31	**2.49**±0.13	**0.84**±0.01	**2.61**±0.06	**0.042**±0.003	**61.02**±0.09

necessary for modeling scores on low-density regions.



(a) StackOverflow (b) Retweet (c) MIMIC-II (d) Financial

Figure 1. Coverage of different confidence levels on four datasets.



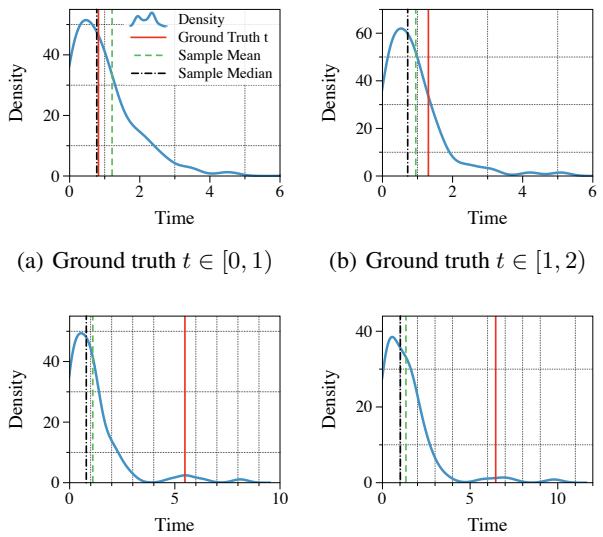
(a) StackOverflow (b) Retweet (c) MIMIC-II (d) Financial

Figure 2. Interval Length of different confidence levels on four datasets. We minus the interval length of all methods with the length of SMURF-THP (denoted as Interval Length$_\Delta$) to better show the differences.

Distribution of Samples. We visualize the distribution of samples generated by SMURF-THP for several events to study the predicted intensity and present them in Figure 3. A large proportion of the samples stay close to value 0, which is reasonable since most of the events occur within a short time after the previous one. Distributions vary as time goes further. Figure 3(c) and Figure 3(d) exhibit that SMURF-THP can still present tiny peaks around the ground truth value, indicating that our model can still capture the effect of historical events. Yet, the generated samples may be inaccurate due to the huge randomness around the large values of arrival time.

4.3. Ablation Study

Parametrization. In addition to parametrizing the intensity function, we can also parametrize the score function directly as was done in conventional score-based methods. That is, we can mask the intensity function and directly parametrize the corresponding score function with the neural network. Table 3 summarizes the results on the StackOverflow and Retweet datasets, where SMURF-THP^{S} indicates SMURF-THP with the score function parametrized. Results imply that parametrizing the intensity function fits the sequence better than parametrizing the score function.



(a) Ground truth $t \in [0, 1)$ (b) Ground truth $t \in [1, 2)$

(c) Ground truth $t \in [4, 5)$ (d) Ground truth $t \in [5, 6)$

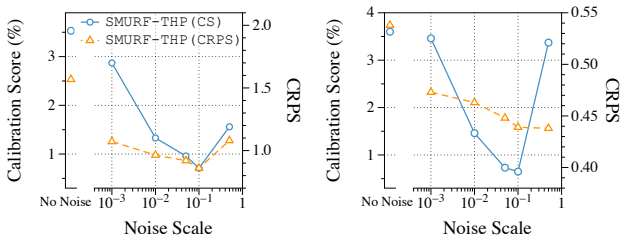
Figure 3. Distribution of samples generated by SMURF-THP and the ground truth value t in the StackOverflow dataset. The green and black lines signify the mean and median values of the samples. Ground truth of the event arrival time t is chosen randomly from time intervals $[0, 1)$, $[1, 2)$, $[4, 5)$, and $[5, 6)$.

This might occur because deriving the score function from the intensity implicitly narrows the searching space, which facilitates the model's convergence.

Table 3. Calibration Score and CRPS of SMURF-THP with two different parametrizations. The superscript s indicates a direct parametrization on the score function as opposed to no superscript indicating a parameterization on the intensity function.

	StackOverflow		Retweet	
Methods	CS(%)	CRPS	CS(%)	CRPS
SMURF-THP^{s}	6.13	0.54	3.92	1.33
SMURF-THP	**0.65**	**0.44**	**0.71**	**0.86**

Denoising. SMURF-THP adds Gaussian noise to data points and adopts denoising score matching for better performance and computational efficiency. We test the effects of different noise scales σ in Figure 4. Compared with training on clean data, adding perturbations effectively improves the performance with a suitable noise scale. Nevertheless, the selection of noise scale is tricky. Adding too small noise will lead to unstable training, which does not cover the low-density regions enough and degrades the performance. A too large noise, on the other hand, over-corrupts the data and alters the distribution significantly from the original one. We simply select σ by grid search in the above experiments and leave auto-selection for future work.



(a) Retweet (b) StackOverflow

Figure 4. Comparison of Calibration Score and CRPS of SMURF-THP with different noise scales and with no noise added.

Sampling Method. We inspect variants of our sampling algorithm, including Langevin Dynamics without the denoising step and Mirror Langevin Dynamics, on two datasets. Table 4 summarizes the results. Langevin with the denoising step yields the best performance since the denoising step corrects the perturbed distribution and ensures non-negativity. Mirror Langevin yields the worst results, likely due to the fact that SMURF-THP learns the score of a perturbed distribution which is not restricted to a constrained region.

Table 4. Calibration Score and CRPS of SMURF-THP with different sampling algorithms on StackOverflow and Retweet datasets.

	StackOverflow		Retweet	
Sampling Algorithm	CS(%)	CRPS	CS(%)	CRPS
Langevin+DS	**0.65**	**0.44**	**0.71**	**0.86**
Langevin	0.87	0.46	1.38	0.92
Mirror Langevin	8.88	0.69	6.73	1.29

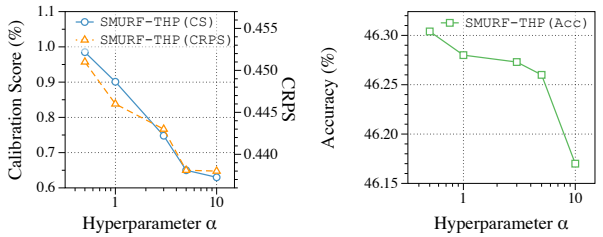
Accuracy with ground truth timestamp. We measure the type prediction accuracy of SMURF-THP given ground truth event time as the inputs of the classifier g. As presented in Table 5, the prediction accuracy of SMURF-THP exhibits an obvious increase compared with results in Table 2. This indicates the value of arrival time for type prediction and also demonstrates that SMURF-THP successfully captures the time-type dependencies.

Hyperparameter Sensitivity. We examine the sensitivity of the hyperparameter α in the training objectives to our model's prediction performance on the StackOverflow dataset. As shown in Figure 5, the Calibration Score and CRPS become lower as α grows, while the accuracy worsens. This accords with intuition since a larger α indicates more attention on fitting event time and less on predicting event type. In the above experiments, we choose the biggest α that can achieve comparable accuracy with THP.

Table 5. Event type prediction accuracy of SMURF-THP given ground truth event time as the inputs.

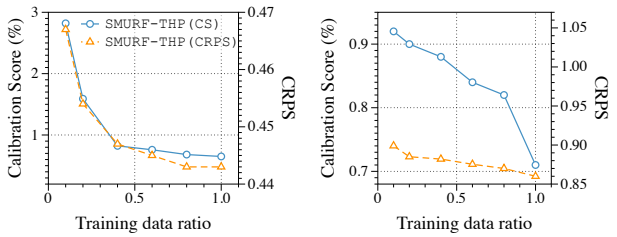
	SMURF-THP
Dataset	Acc(%)
StackOverflow	48.38
Retweet	61.65
MIMIC-II	83.84
Financial	61.62

We also train SMURF-THP with different volumes of training data to study its generalization ability. We train the model on different ratios of the dataset and present the performance in Figure 6 and Figure 7. As shown, all metrics go better as we feed more training data. Compared to the Retweet dataset, SMURF-THP is more sensitive to training ratio on the StackOverflow dataset. This is due to that the StackOverflow dataset contains less events than the Retweet, thereby SMURF-THP requires larger proportion to learn the distribution.



(a) Calibration Score and CRPS (b) Accuracy

Figure 5. Sensitivity of the hyperparameter α in the training objective to prediction performance.



(a) StackOverflow (b) Retweet

Figure 6. Calibration Score and CRPS of SMURF-THP trained with different ratios of the StackOverflow and Retweet datasets.

5. Discussion and Conclusion

We acknowledge the existence of several studies that adopt non-likelihood-based estimators to circumvent the intractable integral within the log-likelihood computation. We present our discussions on these works below. Xiao et al.

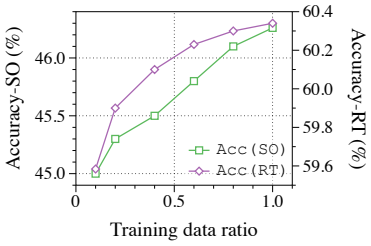


Figure 7. Prediction Accuracy of SMURF-THP trained with different ratio of the StackOverflow and Retweet datasets.

(2017) trains the model to directly predict the next event's time and type through a summation of Mean Squared Error (MSE) and cross-entropy loss. However, their model does not construct an explicit intensity function and hence doesn't support flexible sampling and uncertainty quantification. TPPRL (Li et al., 2018) employs reinforcement learning (RL) for learning an events generation policy, but they concentrate on the temporal point process (TPP) rather than the marked temporal point process (MTPP), which is the focus of our work. Furthermore, they presume the intensity function to be constant between timestamps, a limitation that hampers the accurate capture of the point process's temporal dynamics. Upadhyay et al. (2018) applies RL for MTPP which trains a policy to maximize feedback from the environment. Similar to TPPRL, it assumes a very stringent intensity function, e.g., in exponential forms, which is oversimplified to capture the complex point process in real-world applications. INITIATOR (Guo et al., 2018) and NCE-TPP (Mei et al., 2020) are both based on noise-contrastive estimations for MTPP. However, they utilize the likelihood objective for training the noise generation network, which consequently reintroduces the intractable integral. In our experiments, we include NCE-TPP's performance, as its authors have demonstrated it outperforms INITIATOR.

Several other works (Wang et al., 2020; Fox et al., 2016) explore different scopes of uncertainty quantification for the Hawkes process. That is, they provide uncertainty quantification for the parameters in conventional Hawkes process models, whereas we focus on uncertainty quantification for the predicted arrival time.

In this work, we present SMURF-THP, a score-based method for training Transformer Hawkes process models and quantifying prediction uncertainty. Our proposed model adopts score matching as the training objective to avoid intractable computations in conventional Hawkes process models. Moreover, with the learnt score function, we can sample arrival time of events using the Langevin Dynamics. This enables uncertainty quantification by calculating the associated confidence interval. Experiments on various real-world datasets demonstrate that SMURF-THP achieves state-of-the-art performance in terms of Calibration Score, CRPS and Interval Length.

References

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

Du, N., Dai, H., Trivedi, R., Upadhyay, U., Gomez-Rodriguez, M., and Song, L. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*, pp. 1555–1564. ACM, 2016.

Efron, B. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496): 1602–1614, 2011. doi: 10.1198/jasa.2011.tm11181. URL https://doi.org/10.1198/jasa.2011.tm11181. PMID: 22505788.

Fox, E. W., Schoenberg, F. P., and Gordon, J. S. Spatially inhomogeneous background rate estimators and uncertainty quantification for nonparametric hawkes point process models of earthquake occurrences. 2016.

Guo, R., Li, J., and Liu, H. INITIATOR: noise-contrastive estimation for marked temporal point process. In *IJCAI*, pp. 2191–2197. ijcai.org, 2018.

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

Hsieh, Y., Kavis, A., Rolland, P., and Cevher, V. Mirrored langevin dynamics. In *NeurIPS*, pp. 2883–2892, 2018.

Hyvärinen, A. Estimation of non-normalized statistical models by score matching. *J. Mach. Learn. Res.*, 6:695–709, 2005.

Hyvärinen, A. and Dayan, P. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Jordan, A., Krüger, F., and Lerch, S. Evaluating probabilistic forecasts with scoringrules, 2017. URL https://arxiv.org/abs/1709.04743.

Leskovec, J. and Krevl, A. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

Li, S., Xiao, S., Zhu, S., Du, N., Xie, Y., and Song, L. Learning temporal point processes via reinforcement learning. In *NeurIPS*, pp. 10804–10814, 2018.

Mei, H. and Eisner, J. The neural hawkes process: A neurally self-modulating multivariate point process. In *NIPS*, pp. 6754–6764, 2017.

Mei, H., Wan, T., and Eisner, J. Noise-contrastive estimation for multivariate point processes. In *NeurIPS*, 2020.

Omi, T., Ueda, N., and Aihara, K. Fully neural network based model for general temporal point processes. In *NeurIPS*, pp. 2120–2129, 2019.

Sahani, M., Bohner, G., and Meyer, A. Score-matching estimators for continuous-time point-process regression models. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–5. IEEE, 2016.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, pp. 11895–11907, 2019.

Song, Y., Garg, S., Shi, J., and Ermon, S. Sliced score matching: A scalable approach to density and score estimation. In *UAI*, volume 115 of *Proceedings of Machine Learning Research*, pp. 574–584. AUAI Press, 2019.

Upadhyay, U., De, A., and Rodriguez, M. G. Deep reinforcement learning of marked temporal point processes. In *NeurIPS*, pp. 3172–3182, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NIPS*, pp. 5998–6008, 2017.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural Comput.*, 23(7):1661–1674, 2011.

Wang, H., Xie, L., Cuozzo, A., Mak, S., and Xie, Y. Uncertainty quantification for inferring hawkes networks. In *NeurIPS*, 2020.

Wang, L., Zhang, W., He, X., and Zha, H. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2447–2456, 2018.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, pp. 681–688. Omnipress, 2011.

Xiao, S., Yan, J., Yang, X., Zha, H., and Chu, S. M. Modeling the intensity function of point process via recurrent neural networks. In *AAAI*, pp. 1597–1603. AAAI Press, 2017.

Yang, S.-H., Long, B., Smola, A., Sadagopan, N., Zheng, Z., and Zha, H. Like like alike: joint friendship and interest propagation in social networks. In *Proceedings of the 20th international conference on World wide web*, pp. 537–546, 2011.

Yu, S., Drton, M., and Shojaie, A. Generalized score matching for non-negative data. *J. Mach. Learn. Res.*, 20:76:1–76:70, 2019.

Yu, S., Drton, M., and Shojaie, A. Generalized score matching for general domains. *Information and Inference: A Journal of the IMA*, 11(2):739–780, 2022.

Zhang, Q., Lipani, A., Kirnap, Ö., and Yilmaz, E. Self-attentive hawkes process. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11183–11193. PMLR, 2020.

Zhao, Q., Erdogdu, M. A., He, H. Y., Rajaraman, A., and Leskovec, J. SEISMIC: A self-exciting point process model for predicting tweet popularity. In *KDD*, pp. 1513–1522. ACM, 2015.

Zhu, S., Zhang, M., Ding, R., and Xie, Y. Deep fourier kernel for self-attentive point processes. In *AISTATS*, volume 130 of *Proceedings of Machine Learning Research*, pp. 856–864. PMLR, 2021.

Zuo, S., Jiang, H., Li, Z., Zhao, T., and Zha, H. Transformer hawkes process. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11692–11702. PMLR, 2020.