Self-Attentive Hawkes Process

Qiang Zhang [1] **Aldo Lipani** [1] **Omer Kirnap** [1] **Emine Yilmaz** [1 2]

Abstract

Capturing the occurrence dynamics is crucial to predicting *which type* of events will happen next and *when*. A common method to do this is through Hawkes processes. To enhance their capacity, recurrent neural networks (RNNs) have been incorporated due to RNNs' successes in processing sequential data such as languages. Recent evidence suggests that self-attention is more competent than RNNs in dealing with languages. However, we are unaware of the effectiveness of self-attention in the context of Hawkes processes. This study aims to fill the gap by designing a *self-attentive Hawkes process* (SAHP). SAHP employs self-attention to summarise the influence of history events and compute the probability of the next event. One deficit of the conventional self-attention when applied to event sequences is that its positional encoding only considers the order of a sequence ignoring the time intervals between events. To overcome this deficit, we modify its encoding by translating time intervals into phase shifts of sinusoidal functions. Experiments on goodness-of-fit and prediction tasks show the improved capability of SAHP. Furthermore, SAHP is more interpretable than RNN-based counterparts because the learnt attention weights reveal contributions of one event type to the happening of another type. To the best of our knowledge, this is the first work that studies the effectiveness of self-attention in Hawkes processes.

1. Introduction

Humans and natural phenomena often generate a large amount of irregular and asynchronous event sequences. These sequences can be, for example, user activities on social media platforms (Farajtabar et al., 2015), high-frequency financial transactions (Bacry & Muzy, 2014), healthcare records (Wang et al., 2016), gene positions in bioinformatics (Reynaud-Bouret et al., 2010), or earthquakes and aftershocks in geophysics (Ogata, 1998). Three characteristics make these event sequences unique, their: asynchronicity, multi-modality, and cross-correlation. A sequence is asynchronous when multiple events happening in the continuous time domain are sampled with unequal intervals, in contrast to discrete sequences where events have equal sampling intervals (Zhang et al., 2018b). A sequence is multi-modal when sequences contain multiple type of events. A sequence is cross-correlated when the occurrence of one type of event at a certain time can excite or inhibit the happening of future events of the same or another type. Figure 1 shows four types of events and their mutual influence. A classic modelling problem with these sequences is to predict *which* type and *when* future events will happen.

The occurrence of asynchronous event sequences are often modelled by temporal point processes (TPPs) (Cox & Isham, 1980; Brillinger et al., 2002). They are stochastic processes with (marked) events on the continuous time domain. One special but significant type of TPPs is the Hawkes process. A considerable amount of studies have used Hawkes process as a *de facto* standard tool to model event streams, including: topic modelling and clustering of textual documents (He et al., 2015; Du et al., 2015a), construction and inference on network structure (Yang & Zha, 2013; Choi et al., 2015; Etesami et al., 2016), personalised recommendations based on users' temporal behaviour (Du et al., 2015b), discovering of patterns in social interactions (Guo et al., 2015; Lukasik et al., 2016), stance detection (Lukasik et al., 2016; Zhang et al., 2018c; 2019a) and learning causality (Xu et al., 2016). Hawkes processes usually model the occurrence probability of an event with a so called *intensity function*. For those events whose occurrence are influenced by history, the intensity function is specified as history-dependent.

The vanilla Hawkes processes specify a fixed and static intensity function, which limits the capability of capturing complicated dynamics. To improve its capability, RNNs have been incorporated as result of their success in dealing with sequential data such as speech and language. RNN-based Hawkes processes use a recurrent structure to summarise history events, either in the fashion of discrete-time (Du et al., 2016; Xiao et al., 2017b) or continuous-

[1]University College London, London, United Kingdom [2]Amazon, London, United Kingdom. Correspondence to: Qiang Zhang <qiang.zhang.16@ucl.ac.uk>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

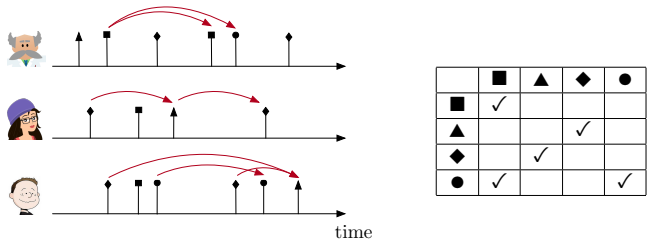


Figure 1. Three users on social media platforms exert different types of actions. The filled dark symbols in (a) define action types while the red arrows denote the influence of actions to other actions. A ✓ symbol in the cell (i, j) in (b) indicates the influence of the column event type j on the row type i on future events.

time (Mei & Eisner, 2017). This solution brings two benefits: (1) historical contributions are not necessarily additive, and (2) it allows the modelling of complex memory effects such as delays. However, recent developments in natural language processing (NLP) have led to an increasing interest in the self-attention mechanism. Although self-attention is empirically superior to RNNs in processing word sequences, it has yet to be researched whether self-attention is capable of processing event sequences that are asynchronous, multi-modal and cross-correlated.

In this work we investigate the usefulness of self-attention to Hawkes processes by proposing a *Self-Attentive Hawkes Process* (SAHP). First, we employ self-attention to measure the influence of history events to the next event by computing its probability. As self-attention relies on positional embeddings to take into account the order of events, conventional embedding methods are based on sinusoidal functions where each position is distanced by a constant shift of phase, which if used for our sequences would ignore the actual time interval between events. We remedy this deficiency by proposing a time-shifted position embedding method: time intervals act as phase shifts of sinusoidal functions. Second, we argue that the proposed SAHP model is more interpretable than the RNN-based counterparts: The learnt attention weights can reveal contributions of one event type to the happening of another.

The contributions of this paper can be summarised as follows:

- To the best of our knowledge, this work is the first to link self-attention to Hawkes processes. SAHP inherits improved capability of capturing complicated dynamics and a higher interpretability;
- To consider inter-event time intervals, we propose a novel time-shifted positional encoding that converts time intervals into phase shifts of sinusoidal functions;
- Through extensive experimentation on a synthetic dataset and three real-world datasets with different sequence lengths and different numbers of event types, we demonstrate the superiority of SAHP.

2. Notation

In this section we introduce the notation used throughout the paper.

Symbol	Description
$\mathcal{U}$	a set of event types.
$\mathcal{S}$	an event sequence.
t	the time of an event.
u, v	the type of an event.
i, j	the order number of an event in a sequence.
$N_u(t)$	the counting process for event type u.
$\mathcal{H}_t$	the set of events that happened before time t.
$\lambda^*(t)$	the conditional intensity function.
$p^*(t)$	the conditional probability density function.
$F^*(t)$	the cumulative distribution function.

3. Background

3.1. Temporal Point Processes and Hawkes Process

A temporal point process (TPP) is a stochastic process whose realisation is a list of discrete events at time $t \in \mathbb{R}^+$ (Cox & Isham, 1980; Daley & Vere-Jones, 2007). A marked TPP allocates a type (aka mark) u to each event. TPPs can be equivalently represented as a counting process $N(t)$, which records the number of events that have happened till time t. A multivariate TPP describes the temporal evolution of multiple event types $\mathcal{U}$.

We indicate with $\mathcal{S} = \{(v_i, t_i)\}_{i=1}^{L}$ an event sequence, where the tuple (v_i, t_i) is the i-th event of the sequence $\mathcal{S}$, $v_i \in \mathcal{U}$ is its event type, and t_i is its timestamp. We indicate with $\mathcal{H}_t := \{(v', t') \,|t' < t, v' \in \mathcal{U}\}$ the historical sequence of events that happened before t.

Given an infinitesimal time window $[t, t + \mathrm{d}t)$, the intensity function of a TPP is defined as the probability of the occurrence of an event (v', t') in $[t, t + \mathrm{d}t)$ conditioned on the history of events $\mathcal{H}_t$:

$$\begin{aligned}\lambda^*(t)\,\mathrm{d}t :&= p\left((v', t') : t' \in [t, t + \mathrm{d}t)|\mathcal{H}_t\right) \\ &= \mathbf{E}(\mathrm{d}N(t)|\mathcal{H}_t),\end{aligned}$$

where $\mathbf{E}(\mathrm{d}N(t)|\mathcal{H}_t)$ denotes the expected number of events in $[t, t+\mathrm{d}t)$ based on the history $\mathcal{H}_t$. Without loss of generality, we assume that two events do not happen simultaneously, i.e., $\mathrm{d}N(t) \in \{0, 1\}$.

Based on the intensity function, it is straightforward to derive the probability density function $p^*(t)$ and the cumulative distribution function $F^*(t)$ (Rasmussen, 2018):

$$p^*(t) = \lambda^*(t) \exp\left(-\int_{t_{i-1}}^{t} \lambda^*(\tau)\mathrm{d}\tau\right),$$

$$F^*(t) = 1 - \exp\left(-\int_{t_{i-1}}^{t} \lambda^*(\tau)\mathrm{d}\tau\right).$$

An Hawkes process (Hawkes, 1971) models the self-excitation of events of the same type and the mutual excitation of different event types, in an additive way. Hence, the definition of the intensity function is given as:

$$\lambda^*(t) = \mu + \sum_{(v',t')\in\mathcal{H}_t} \phi(t - t'), \tag{1}$$

where $\mu \geq 0$, named *base intensity*, is an exogenous component of the intensity function independent of the history, while $\phi(t) > 0$ is an endogenous component dependent on the history. Besides, $\phi(t)$ is a triggering kernel containing the peer influence of different event types. To highlight the peer influence represented by $\phi(t)$, we write $\phi_{u,v}(t)$, which captures the impact of a historical type-v event on a subsequent type-u event (Farajtabar et al., 2014). In this example, the occurrence of a past type-v event increases the intensity function $\phi_{u,v}(t - t')$ for $0 < t' < t$.

Most commonly $\phi_{u,v}(t)$ is parameterized as $\phi_{u,v}(t) = \alpha_{u,v} \cdot \kappa(t) \cdot \mathbb{1}_{t>0}$ (Zhou et al., 2013; Xu et al., 2016). The *excitation* parameter $\alpha_{u,v}$ quantifies the initial influence of the type-v event on the intensity of the type-u event. The *kick* function $\kappa(t)$ characterises the time-decaying influence. Typically, $\kappa(t)$ is chosen to be exponential, i.e., $\kappa(t) = \exp(-\gamma t)$, where γ is the *decaying* parameter controlling the intensity decaying speed.

To learn the parameters of Hawkes processes, it is common to use Maximum Likelihood Estimation (MLE). Other advanced and more complex adversarial learning (Xiao et al., 2017a) and reinforcement learning (Li et al., 2018) methods have been proposed, however we use MLE for its simplicity. In experiments, we use the same optimization method for our model and all baselines as done in their original papers. To apply MLE, a loss function is derived based on the negative log-likelihood. Details of derivation can be found in appendix. The log-likelihood of an event sequence $\mathcal{S}$ over a time interval $[0, T]$ is given by:

$$\mathcal{L} = \sum_{i=1}^{L} \log \lambda_{v_i}(t_i) - \int_0^T \lambda(\tau)d\tau, \tag{2}$$

where the first term is the sum of the log-intensity functions of past events, and the second term corresponds to the log-likelihood of infinitely many non-events. Intuitively, the probability that there is no event of any type in the infinitesimally time interval $[t, t + \mathrm{d}t)$ is equal to $1 - \lambda(t)\mathrm{d}t$, the log of which is $-\lambda(t)\mathrm{d}t$.

3.2. Attention and Self-Attention

Attention. The attention mechanism enables machine learning models to focus on a subset of the input sequence (Walther et al., 2004; Bahdanau et al., 2015; Zhang et al., 2018a). In Seq2Seq models with the attention mechanism the input sequence, in the encoder, is represented as a sequence of key vectors K and value vectors V, $(K, V) = [(\boldsymbol{k}_1, \boldsymbol{v}_1), (\boldsymbol{k}_2, \boldsymbol{v}_2), \ldots, (\boldsymbol{k}_N, \boldsymbol{v}_N)]$, and, in the decoder, as a sequence of query vectors, $Q = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_M]$. These query vectors are used to find which part of the input sequence is more important (Vaswani et al., 2017). Given these two sequences of vectors (K, V) and Q, the attention mechanism computes a prediction sequence $O = [\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_M]$ as follows:

$$\boldsymbol{o}_m = \left(\sum_n f(\boldsymbol{q}_m, \boldsymbol{k}_n) g(\boldsymbol{v}_n)\right) / \sum_n f(\boldsymbol{q}_m, \boldsymbol{k}_n),$$

where $m \in \{1, \ldots, M\}$, $n \in \{1, \ldots, N\}$, $\boldsymbol{q}_m \in \mathbb{R}^d$, $\boldsymbol{k}_n \in \mathbb{R}^d$, $\boldsymbol{v}_n \in \mathbb{R}^p$, $g(\boldsymbol{v}_n) \in \mathbb{R}^q$ and $\boldsymbol{o}_m \in \mathbb{R}^q$. The similarity function $f(\boldsymbol{q}_m, \boldsymbol{k}_n)$ characterises the relation between $\boldsymbol{q}_m$ and $\boldsymbol{k}_n$, whose common form is composed of: an embedded Gaussian, an inner-product, and a concatenation (Wang et al., 2018). The function $g(\boldsymbol{v}_n)$ is a linear transformation specified as $g(\boldsymbol{v}_n) := \boldsymbol{v}_n W_v$, where $W_v \in \mathbb{R}^{p\times q}$ is a weight matrix.

Self-attention. Self-attention is a special case of the attention mechanism (Vaswani et al., 2017), where the query vectors Q, like (K, V), are come from the encoder. Self-attention is a method of encoding sequences of input tokens by relating these tokens to each other based on a pairwise similarity function $f(\cdot, \cdot)$. It measures the dependency between each pair of tokens from the same input sequence. To encode positional information of tokens a positional encoder is used. Consequently, self-attention encodes both token similarity and positional information.

Self-attention is very expressive and flexible for both long-term and local dependencies, which used to be modeled by recurrent neural networks (RNNs) and convolutional neural networks (CNNs) (Vaswani et al., 2017). Moreover, the self-attention mechanism has fewer parameters and faster convergence than RNNs. Recently, a variety of Natural Language Processing (NLP) tasks have experienced large improvements thanks to self-attention (Vaswani et al., 2017; Devlin et al., 2019).

4. Self-Attentive Hawkes Process

In this section, we describe how to adapt the self-attention mechanism to Hawkes processes. A graphical representation of this adaptation is shown in Figure 2.

Event type embedding. The input sequence is made up of events. To obtain a unique dense embedding for each event type, we use a linear embedding layer,

$$\boldsymbol{tp}_v = \boldsymbol{e}_v W_E,$$

where $\boldsymbol{tp}_v$ is the type-v embedding, $\boldsymbol{e}_v$ is a one-hot vector of the type-v and W_E is the embedding matrix.

Time-shifted positional encoding. Self-attention utilises positional encoding to inject order information to a sequence. To take into account time intervals of subsequent events, we modify the conventional positional encoding. For an event (v_i, t_i), the positional encoding is defined as a K-dimensional vector such that the k-th dimension of the position embedding is calculated as:

$$pe^k_{(v_i,t_i)} = \sin\left(\omega_k \times i + w_k \times t_i\right),$$

where i is the absolute position of an event in a sequence, and ω_k is the angle frequency of the k-th dimension, which is pre-defined and will not be changed. While w_k is a scaling parameter that converts the timestamp t_i to a phase shift in the k-th dimension. Multiple sinusoidal functions with different ω_k and w_k are used to generate the multiple position values, the concatenation of which is the new positional encoding. Even and odd dimensions of $\boldsymbol{pe}$ are generated from $\sin$ and $\cos$ respectively.

Figure 3 shows how conventional and the new positional encodings work. Suppose an event (v_i, t_i) is at the $i = 14$ position of a sequence. Conventional methods calculate the values of sinusoidal functions at the $i = 14$ position as the position value of this event. Our encoding modifies this by shifting the original position i to a new position $i'_k = i + \frac{w_k t_i}{\omega_k}$, where k denotes the embedding dimension. This is equivalent to interpolating the time domain and to produce shorter equal-length time periods. Positions in a sequence are thus shifted by the time t_i. The length of time periods is decided by $\frac{w_k}{\omega_k}$. Since w_k and ω_k are dimension-specific, the shift in one dimension can be different to the one performed to the others.

History hidden vector. As an event consists of its type and timestamp, we add the positional encoding to the event type embedding in order to obtain the representation of the event (v_i, t_i):

$$\boldsymbol{x}_i = \boldsymbol{tp}_v + \boldsymbol{pe}_{(v_i,t_i)}.$$

Self-Attention. Given a series of historical events until t_i, to compute the intensity of the type-u at the timestamp t, we need to consider the influence of all types of events before it. To do this, we compute the pairwise influence of one previous event to the next event by employing self-attention. This generates a hidden vector that summarizes the influence of all previous events:

$$\boldsymbol{h}_{u,i+1} = \left(\sum_{j=1}^{i} f(\boldsymbol{x}_{i+1}, \boldsymbol{x}_j) g(\boldsymbol{x}_j)\right) / \sum_{j=1}^{i} f(\boldsymbol{x}_{i+1}, \boldsymbol{x}_j),$$

where $\boldsymbol{x}_{i+1}$ is like query ($\boldsymbol{q}$) in the attention terminology, $\boldsymbol{x}_j$ is the key ($\boldsymbol{k}$) and $g(\boldsymbol{x}_j)$ is the value ($\boldsymbol{v}$). The function $g(\cdot)$ is a linear transformation while the similarity function $f(\cdot, \cdot)$ is specified as an embedded Gaussian:

$$f(\boldsymbol{x}_{i+1}, \boldsymbol{x}_j) = \exp\left(\boldsymbol{x}_{i+1} \boldsymbol{x}_j^T\right).$$

The temporal information is provided to the model during training by preventing the model to learn about future events via masking. We implement this in the attention mechanism by masking out all values in the input sequence that correspond to future events. Hence, the intensity of one event is obtained only based on its history.

Intensity function. Since the intensity function of Hawkes processes is history-dependent, we compute three parameters of the intensity function based on the history hidden vector $\boldsymbol{h}_{u,i+1}$ via the following three non-linear transformations:

$$\begin{aligned}
\mu_{u,i+1} &= \text{gelu}\left(\boldsymbol{h}_{u,i+1} W_\mu\right), \\
\eta_{u,i+1} &= \text{gelu}\left(\boldsymbol{h}_{u,i+1} W_\eta\right), \\
\gamma_{u,i+1} &= \text{softplus}\left(\boldsymbol{h}_{u,i+1} W_\gamma\right).
\end{aligned}$$

The function gelu represents the Gaussian Error Linear Unit for nonlinear activations. We use this activation function because this has been empirically proved to be superior to other activation functions for self-attention (Hendrycks & Gimpel, 2016). softplus is used for the decaying parameter since γ needs to be constrained to strictly positive values.

Finally, we express the intensity function as follows:

$$\begin{aligned}
\lambda_u(t) = &\text{softplus}(\mu_{u,i+1} + \\
&(\eta_{u,i+1} - \mu_{u,i+1}) \exp(-\gamma_{u,i+1}(t - t_i))),
\end{aligned}$$

for $t \in (t_i, t_{i+1}]$, where the softplus is employed to constrain the intensity function to be positive. The starting intensity at $t = t_i$ is $\eta_{u,i+1}$. When t increases from t_i, the intensity decays exponentially. As $t \to \infty$, the intensity converges to $\mu_{u,i+1}$. The changing speed is decided by $(\eta_{u,i+1} - \mu_{u,i+1})$ that can be both positive and negative. This enables us to capture both excitation and inhibition effects. With inhibition we mean the effect that manifests when past events reduce the likelihood of future events to happen (Mei & Eisner, 2017).

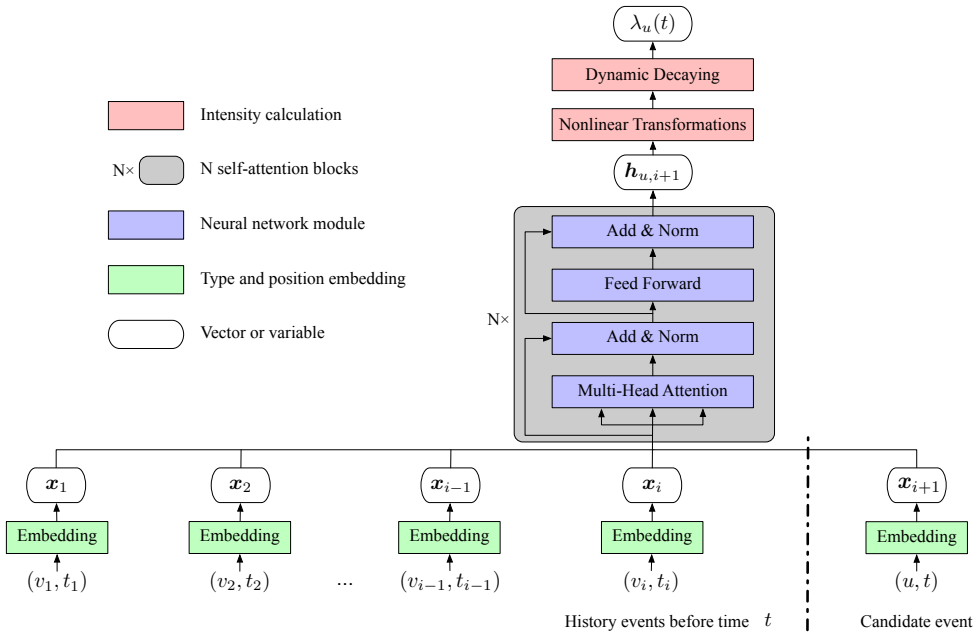


Figure 2. An event stream and the SAHP for one event type (u). The intensity function ($\lambda_u(t)$) is determined by a sequence of history events via the SAHP.

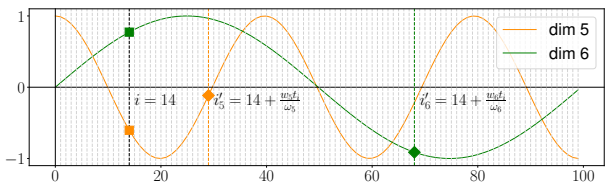


Figure 3. The time-shifted position encoding of an event with $i = 14$ in a sequence. Squares and diamonds denote conventional and new encoding values with time-shifts.

5. Experiments

To compare our method with the state-of-the-art, we conduct experiments on one synthetic dataset and three real-world asynchronous event datasets. The real-world datasets have been purposefully chosen in order to span over various properties, i.e., the number of event type ranges from 2 to 75 and the average sequence length ranges from 4 to 132. As usual, sequences from the same dataset are assumed to be drawn independently from the same process. These datasets are all available at the following weblink[1]. Each dataset is split into a training set, a validation set and a testing set. The validation set is used to tune the hyper-parameters while the testing set is used to measure the model performance. Details about the datasets can be found in Table 1 and Appendix.

[1] https://drive.google.com/drive/folders/0BwqmV0EcoUc8UklIR1BKV25YR1U

5.1. Synthetic Dataset

We generate a synthetic dataset with the open-source Python library *tick*[2]. A two-dimensional Hawkes process is generated with base intensities $\mu_1 = 0.1$ and $\mu_2 = 0.2$. The triggering kernels have a power law kernel, an exponential kernel, a sum of two exponential kernels, and a sine kernel:

$$\phi_{1,1}(t) = 0.2 \times (0.5 + t)^{-1.3}$$
$$\phi_{1,2}(t) = 0.03 \times \exp(-0.3t)$$
$$\phi_{2,1}(t) = 0.05 \times \exp(-0.2t) + 0.16 \times \exp(-0.8t)$$
$$\phi_{2,2}(t) = \max(0, \sin(t)/8) \quad \text{for } 0 \leq t \leq 4$$

In Figure 4 we show the four triggering kernels of the 2-dimensional Hawkes processes. The simulated intensities of each dimension is shown in the appendix.

5.2. Training Details

We implement the multi-head attention. This allows the model to jointly attend information from different representation subspaces (Vaswani et al., 2017). The number of heads is a hyper-parameter. We explore this hyper-parameter in the set $\{1, 2, 4, 8, 16\}$. Another hyper-parameter is the number of attention layers. We explore this hyper-parameter in the set $\{2, 3, 4, 5, 6\}$. We adapt the Adam as the basic optimiser and develop a warm-up stage for the learning rate whose initialisation is set to $1e{-}4$. To mitigate overfitting we apply dropout with rate set to 0.1. Early stopping is used when the validation loss does not decrease more than $1e{-}3$.

[2] https://github.com/X-DataInitiative/tick

Table 1. Statistics of the used datasets.

Dataset	# of Types	Sequence Length			# of Sequences		
		Min	Mean	Max	Train	Validation	Test
Synth.	2	68	132	269	3,200	400	400
RT	3	50	109	264	20,000	2,000	2,000
SOF	22	41	72	736	4,777	530	1,326
MMC	75	2	4	33	527	58	65

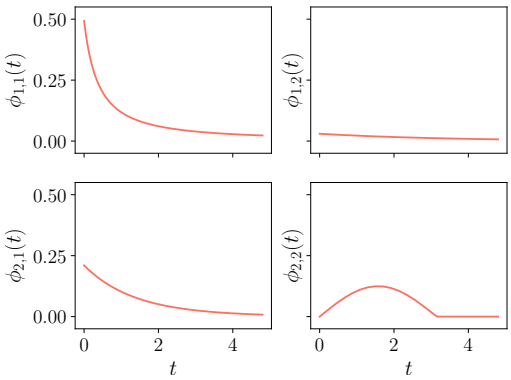


Figure 4. The four triggering kernels of the synthetic dataset with 2 event types.

5.3. Baselines

We compare our method (SAHP) against the following state-of-the-art baselines:

Hawkes Processes (HP). This is the most conventional Hawkes process statistical model which intensity is described in Eq. (1). It uses an exponential kernel;

Recurrent Marked Temporal Point Processes (RMTPP). Du et al. (2016) use RNN to learn a representation of influences from past events, and time intervals are encoded as explicit inputs;

Continuous Time LSTM (CTLSTM). Mei & Eisner (2017) use a continuous-time LSTM, which includes intensity decay and eliminates the need to encode event intervals as numerical inputs of the LSTM;

Fully Neural Network (FullyNN). Omi et al. (2019) propose to model the cumulative distribution function with a feed-forward neural network.

Log Normal Mixture (LogNormMix). Shchur et al. (2020) suggest to model the conditional probability density distribution by a log-normal mixture model.

6. Results and Discussion

For a fair comparison, we tried different hyper-parameter configurations for baselines and our model, and selected the configuration with the best validation performance. The software used to run these experiments is available at the following weblink[3].

Goodness of fit on the synthetic dataset. In order to conduct a goodness-of-fit evaluation, we used the synthetic dataset where the true intensity is known, and compared the estimated intensity against the true intensity. We chose the QQ-plot to visualise how well the proposed SAHP is able to approximate the true intensity. Figure 5 shows the QQ-plots of the estimated intensity by the five baselines and SAHP.

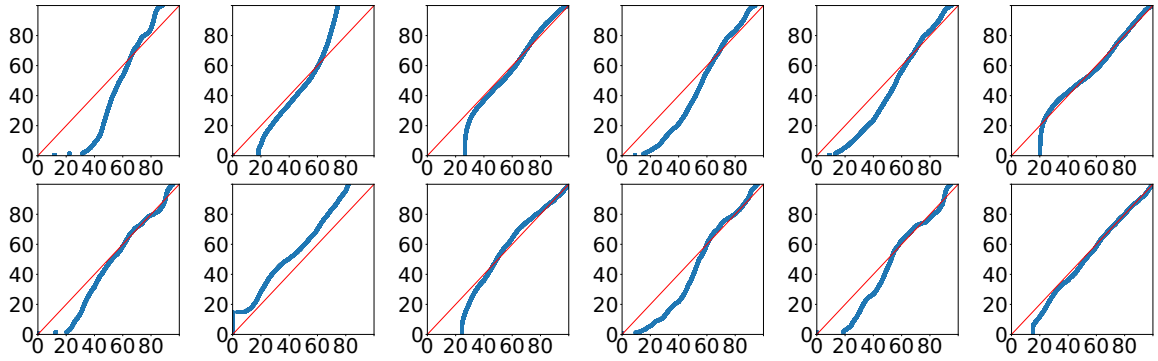
From this figure, we observe that the intensity estimated by SAHP produces the most similar distribution to the true one, which indicates that SAHP is able to best capture the underlying complicated dynamics of the synthetic dataset. Moreover, by comparing the upper and the lower sub-figures in one column, all models obtain slightly better approximations to the intensity of the second event type.

Sequence modelling. We further compare the ability of the methods to model an event sequence. As done in previous works (Mei & Eisner, 2017; Shchur et al., 2020), the negative log-likelihood (NLL) was selected as the evaluation metric. The lower the NLL is, the more capable a model is to model a specific event sequence.

In Table 2 we report the per-event NLL of these models on each test set. According to Table 2, our method significantly outperforms the baselines in all datasets. As expected, the conventional HP method is the worst in modelling an event sequence in all datasets. RMTPP and CTLSTM have very similar performance except on the Retweet dataset, where CTLSTM achieves a lower NLL than RMTPP.

Event prediction. We also evaluate the ability of the methods to predict the next event, including the event type and the timestamp, according to history. To emphasises the

[3] https://github.com/QiangAIResearcher/sahp_repo



(a) HP (b) RMTPP (c) CTLSTM (d) FullyNN (e) LogNormMix (f) SAHP

Figure 5. QQ-plot of the true *vs.* estimated intensities for each type. The x-axis and the y-axis represent the quantiles of the true and estimated intensities. For each model the top plot is for the type-1 events while the bottom plot is for the type-2 events.

Table 2. Negative log-likelihood per event on the four test sets.

Dataset	Synth.	RT	SOF	MMC
HP	2.12	9.84	3.21	1.81
RMTPP	1.85	7.43	2.44	1.33
CTLMST	1.83	6.95	2.38	1.36
FullyNN	1.55	6.23	2.21	1.03
LogNormMix	1.43	5.32	2.01	0.78
SAHP	**1.35**	**4.56**	**1.86**	**0.52**

importance of the time-shifted positional encoding, we also compare SAHP with a version (SAHP-TSE) where the new positional encoding is replaced with the standard encoding (Vaswani et al., 2017). We categorise type prediction as a multi-class classification problem. As there is class imbalance among event types, we use the macro F_1 as the evaluation metric. Also, since time interval prediction is assumed to be a real number, a common evaluation metric to evaluate these cases is to use the Root Mean Square Error (RMSE). In order to eliminate the effect of the scale of time intervals, we compute the prediction error according to the formula as follows

$$\varepsilon_i = \frac{(\hat{t}_{i+1} - t_i) - (t_{i+1} - t_i)}{t_{i+1} - t_i},$$

where $\hat{t}_{i+1}$ is the predicted timestamp while t_{i+1} is the ground truth, and $\hat{t}_{i+1} - t_i$ is the predicted time interval while $t_{i+1} - t_i$ is the true time interval. The results of type and time prediction are summarised in Tables 3 and 4.

These two tables illustrate that our model outperforms the baselines in terms of F_1 and RMSE on all the prediction tasks. We also observe that SAHP demonstrates a larger margin in type prediction for F_1. FullyNN and LogNormMix are consistently better than the other baselines in time prediction, yet LogNormMix is not good at predicting event types, which confirms the previous findings (Shchur et al., 2020). Another important finding is that the use of the time-shifted positional encoding improves the performance of our method in both tasks.

Table 3. $F_1(\%)$ of event type prediction on the four test-sets.

Dataset	Synth.	RT	SOF	MMC
HP	33.20	32.43	2.98	19.32
RMTPP	40.32	41.22	5.44	28.76
CTLMST	43.80	39.21	4.88	34.00
FullyNN	45.21	43.80	6.34	33.32
LogNormMix	42.09	45.25	3.23	32.86
SAHP-TSE	57.93	53.24	24.05	34.23
SAHP	**58.50**	**53.92**	**24.12**	**36.90**

Table 4. RMSE of event timestamp prediction on the four test sets.

Dataset	Synth.	RT	SOF	MMC
HP	42.80	1293.32	221.82	7.68
RMTPP	37.07	1276.41	207.79	6.83
CTLMST	35.08	1255.05	194.87	6.49
FullyNN	33.34	1104.41	173.92	5.43
LogNormMix	32.64	1090.45	154.13	4.12
SAHP-TSE	33.32	1102.34	143.54	4.03
SAHP	**31.16**	**1055.05**	**133.61**	**3.89**

Number of samples' influence. When we optimise the objective function Eq. (2), since it is not a closed form of the expectation, we use Monte Carlo sampling to approximate the integral. This experiment studies how the number of samples influences the SAHP's performance. The number of samples varies from 5 to 30 with step size 5. We report experimental results obtained from the StackOverflow dataset; other datasets share similar findings.

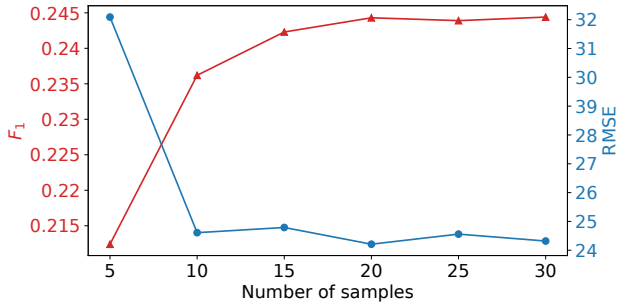


Figure 6. The influence of the number of samples of the Monte Carlo estimation on SAHP's performance for event prediction.

Figure 6 describes how the performance of event prediction changes with different number of samples. From 5 to 10 samples, there is a significant improvement on the evaluation metrics. With more than 10 samples, we observe that the performance plateaus. To reduce computational time, we use 10 as the default number of samples in Monte Carlo.

Computational efficiency. To compare the computational efficiency of our method with neural baselines, we report in Table 5 the running time on the retweet dataset with a Titan Xp GPU card. The mini-batch size is 32 and running time is averaged by 10 epochs. We make two observations: (1) CTLSTM is the least computationally efficient model, which could be due to the recurrent architecture and Monte Carlo sampling, and (2) the proposed SAHP model enjoys the same level of computational efficiency with LogNormMix and FullyNN.

Table 5. Model running time on the retweet dataset with a Titan Xp GPU card.

Model	Time (Seconds)
RMTPP	92.22
CTLSTM	134.69
LogNormMix	**85.32**
FullyNN	87.53
SAHP	86.97

Model interpretability. Apart from strong capacity in reconstructing the intensity function, the other advantage of our method is its higher interpretability. SAHP is able to reveal peer influence among event types. To demonstrate that, we extract the attention weight that the type-u events allocate to the type-v events and accumulate such attention weight over all the sequences on the StackOverflow test set. We remove the effect of the frequency of the (u, v) pairs in the dataset through dividing the accumulated attention weight via the (u, v) frequency. After normalisation, we obtain the statistical attention distribution as shown in Figure 7. The cell at the u-th row and v-th column means the statistical attention that the type-u allocates to the type-v.

Two interesting findings can be drawn from this figure: 1) for most cells in the diagonal line, when the model computes the intensity of one type, it attends to the history events of the same type; 2) for dark cells in the non-diagonal line, such as *Constituent and Caucus*, *Boosters and Enlightened*, and *Caucus and Publicist*, the model attends to the latter when computing the likelihood of the former. The first finding is attributed to the fact that attention is computed based on similarity between two embeddings while the second finding indicates the statistical co-occurrence of event types in a sequence.

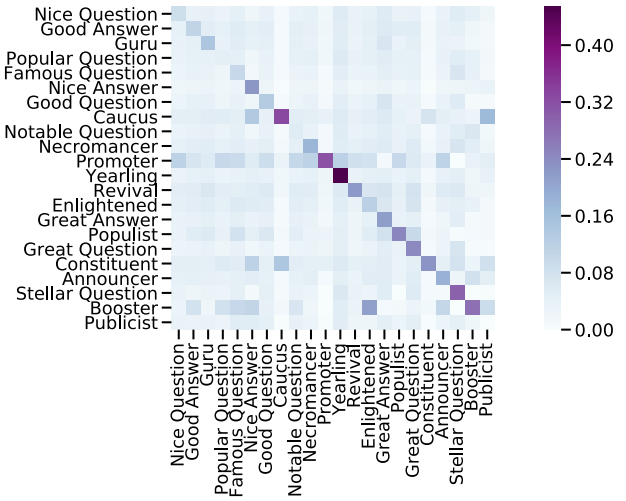


Figure 7. Expected attention weights among event types on the StackOverflow test set.

7. Related Work

Neural temporal point process. Complicated dynamics of event occurrence demands for higher capacity of Hawkes processes. To meet this demand, neural networks have been incorporated to modify the intensity function. Du et al. (2016) proposed a discrete-time RNN to encode history to fit parameters of the intensity function. Mei & Eisner (2017) designed a continuous-time Long Short-Term Memory model that avoids encoding time intervals as explicit inputs. Another two works chose not to model the intensity function. Omi et al. (2019) proposed to model the cumulative distribution function with a feed forward neural network, yet it suffers from two problems: (1) the probability density function is not normalised and (2) negative inter-event times are assigned a non-zero probability. Then, Shchur et al. (2020) suggested modelling the conditional probability density distribution by a log-normal mixture model. They only studied the one-dimensional distribution of inter-event times, neglecting mutual influence among different event types. Also, despite their claimed flexibility, it fails to achieve convincing performance on predicting event types. Above all, history events have always been encoded by a recurrent structure. Moreover, RNN and its variants have been

empirically proved to be less competent than self-attention in NLP (Vaswani et al., 2017; Devlin et al., 2019). Also, RNN-modified Hawkes processes do not provide a simple way to interpret the peer influence among events. Each historical event updates hidden states in the RNN cells but the process lacks of straightforward interpretability (Karpathy et al., 2016; Krakovna & Doshi-Velez, 2016).

Zuo et al. (2018) tries to learn network embeddings and the dynamics are described by a Hawkes process. Our model methodologically differs from this paper in two aspects: (1) We use self-attention to measure the influence of each historical event (one historical event per calculation of attention weights) on the happening of the next event (changes at different timestamps); while Zuo et al. (2018) use attention as a similarity measure between the history neighbourhood (containing multiple nodes) and the source node (remain unchanged for each sequence); (2) Zuo et al. (2018) do not consider time intervals that are important in the continuous-time domain.

Besides, recent works have advanced methods of parameter optimisation. Alternatives to the maximum likelihood estimation can be adversarial training (Xiao et al., 2017a), online learning (Yang et al., 2017), Wasserstein loss (Xiao et al., 2018), noise contrastive estimation (Guo et al., 2018) and reinforcement learning (Li et al., 2018; Upadhyay et al., 2018). This line of research is orthogonal to our work.

Positional encoding. RNN-based models are able to naturally capture position information of tokens in a sequence while convolutional neural networks (CNN) and self-attention has to rely on positional encoding to capture sequential orders. Gehring et al. (2017) equipped CNN with an absolute order numbers of input tokens; the position embeddings were learnt during model training. Vaswani et al. (2017) computed the absolute positional encoding by feeding order numbers to sinusoidal functions. In contrast, the relative positional encoding uses relative distance of the centre token to others in the sequence. Shaw et al. (2018) represented the relative position by learning an embedding matrix. Wang et al. (2019) introduced a structural position to model a grammatical structure of a sentence, which involves both the absolute and the relative strategy. However, these methods only consider the order of tokens, which ignores time intervals for temporal event sequences.

8. Conclusion

The intensity function plays an important role in Hawkes processes for predicting asynchronous events in the continuous time domain. In this paper, we propose a self-attentive Hawkes process where self-attention is adapted to enhance the expressivity of the intensity function. This method enhances the model prediction and model interpretability. For the former, the proposed method outperforms state-of-the-art methods via better capturing event dependencies; while for the latter, the model is able to reveal peer influence via attention weights. For future work, we plan to apply this work to solve practical problems such as misinformation detection (Zhang et al., 2019b) and extend it for causality analysis of asynchronous events.

Acknowledgements

This project was funded by the EPSRC Fellowship titled "Task Based Information Retrieval" and grant reference number EP/P024289/1.

References

Bacry, E. and Muzy, J.-F. Hawkes model for price and trades high-frequency dynamics. *Quantitative Finance*, 14(7): 1147–1166, 2014.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR '15)*, 2015.

Brillinger, D. R., Guttorp, P. M., Schoenberg, F. P., El-Shaarawi, A. H., and Piegorsch, W. W. Point processes, temporal. *Encyclopedia of Environmetrics*, 3:1577–1581, 2002.

Choi, E., Du, N., Chen, R., Song, L., and Sun, J. Constructing disease network and temporal progression model via context-sensitive hawkes process. In *2015 IEEE International Conference on Data Mining*, pp. 721–726. IEEE, 2015.

Cox, D. R. and Isham, V. *Point processes*, volume 12. CRC Press, 1980.

Daley, D. J. and Vere-Jones, D. *An introduction to the theory of point processes: volume II: general theory and structure*. Springer Science & Business Media, 2007.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

Du, N., Farajtabar, M., Ahmed, A., Smola, A. J., and Song, L. Dirichlet-hawkes processes with applications to clustering continuous-time document streams. In *Proceedings of the 21th ACM SIGKDD International Conference on*

Knowledge Discovery and Data Mining, pp. 219–228. ACM, 2015a.

Du, N., Wang, Y., He, N., Sun, J., and Song, L. Time-sensitive recommendation from recurrent user activities. In *Advances in Neural Information Processing Systems*, pp. 3492–3500, 2015b.

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 *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1555–1564. ACM, 2016.

Etesami, J., Kiyavash, N., Zhang, K., and Singhal, K. Learning network of multivariate hawkes processes: A time series approach. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'16. AUAI Press, 2016. ISBN 978-0-9966431-1-5.

Farajtabar, M., Du, N., Rodriguez, M. G., Valera, I., Zha, H., and Song, L. Shaping social activity by incentivizing users. In *Advances in neural information processing systems*, pp. 2474–2482, 2014.

Farajtabar, M., Wang, Y., Rodriguez, M. G., Li, S., Zha, H., and Song, L. Coevolve: A joint point process model for information diffusion and network co-evolution. In *Advances in Neural Information Processing Systems*, pp. 1954–1962, 2015.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1243–1252. JMLR. org, 2017.

Guo, F., Blundell, C., Wallach, H., and Heller, K. The bayesian echo chamber: Modeling social influence via linguistic accommodation. In *Artificial Intelligence and Statistics*, pp. 315–323, 2015.

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

Hawkes, J. On the hausdorff dimension of the intersection of the range of a stable process with a borel set. *Probability Theory and Related Fields*, 19(2):90–102, 1971.

He, X., Rekatsinas, T., Foulds, J., Getoor, L., and Liu, Y. Hawkestopic: A joint model for network inference and topic modeling from text-based cascades. In *International conference on machine learning*, pp. 871–880, 2015.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Karpathy, A., Johnson, J., and Fei-Fei, L. Visualizing and understanding recurrent networks. 2016.

Krakovna, V. and Doshi-Velez, F. Increasing the interpretability of recurrent neural networks using hidden markov models. *arXiv preprint arXiv:1611.05934*, 2016.

Li, S., Xiao, S., Zhu, S., Du, N., Xie, Y., and Song, L. Learning temporal point processes via reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 10781–10791, 2018.

Lukasik, M., Srijith, P., Vu, D., Bontcheva, K., Zubiaga, A., and Cohn, T. Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 393–398, 2016.

Mei, H. and Eisner, J. M. The neural hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems*, pp. 6754–6764, 2017.

Ogata, Y. Space-time point-process models for earthquake occurrences. *Annals of the Institute of Statistical Mathematics*, 50(2):379–402, 1998.

Omi, T., ueda, n., and Aihara, K. Fully neural network based model for general temporal point processes. In Wallach, H., Larochelle, H., Beygelzimer, A., dAlché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 2122–2132. Curran Associates, Inc., 2019.

Rasmussen, J. G. Lecture notes: Temporal point processes and the conditional intensity function. *arXiv preprint arXiv:1806.00221*, 2018.

Reynaud-Bouret, P., Schbath, S., et al. Adaptive estimation for hawkes processes; application to genome analysis. *The Annals of Statistics*, 38(5):2781–2822, 2010.

Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 464–468, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2074.

Shchur, O., Biloš, M., and Günnemann, S. Intensity-free learning of temporal point processes. In *International Conference on Learning Representations (ICLR '20)*, 2020.

Upadhyay, U., De, A., and Rodriguez, M. G. Deep reinforcement learning of marked temporal point processes.

In *Advances in Neural Information Processing Systems*, pp. 3168–3178, 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 *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 6000–6010, 2017.

Walther, D., Rutishauser, U., Koch, C., and Perona, P. On the usefulness of attention for object recognition. In *Workshop on Attention and Performance in Computational Vision at ECCV*, pp. 96–103, 2004.

Wang, X., Girshick, R., Gupta, A., and He, K. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7794–7803, 2018.

Wang, X., Tu, Z., Wang, L., and Shi, S. Self-attention with structural position representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1403–1409, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1145.

Wang, Y., Xie, B., Du, N., and Song, L. Isotonic hawkes processes. In *International conference on machine learning*, pp. 2226–2234, 2016.

Xiao, S., Farajtabar, M., Ye, X., Yan, J., Song, L., and Zha, H. Wasserstein learning of deep generative point process models. In *Advances in Neural Information Processing Systems*, pp. 3247–3257, 2017a.

Xiao, S., Yan, J., Yang, X., Zha, H., and Chu, S. M. Modeling the intensity function of point process via recurrent neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017b.

Xiao, S., Xu, H., Yan, J., Farajtabar, M., Yang, X., Song, L., and Zha, H. Learning conditional generative models for temporal point processes. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Xu, H., Farajtabar, M., and Zha, H. Learning granger causality for hawkes processes. In *International Conference on Machine Learning*, pp. 1717–1726, 2016.

Yang, S.-H. and Zha, H. Mixture of mutually exciting processes for viral diffusion. In *International Conference on Machine Learning*, pp. 1–9, 2013.

Yang, Y., Etesami, J., He, N., and Kiyavash, N. Online learning for multivariate hawkes processes. In *Advances in Neural Information Processing Systems*, pp. 4937–4946, 2017.

Zhang, Q., Liang, S., and Yilmaz, E. Variational self-attention model for sentence representation. In *Proceedings of the third workshop on Bayesian Deep Learning (NeurIPS '18)*, 2018a.

Zhang, Q., Luo, R., Yang, Y., and Liu, Y. Benchmarking deep sequential models on volatility predictions for financial time series. In *Proceedings of the Workshop on Challenges and Opportunities for AI in Financial Services (NeurIPS '18)*, 2018b.

Zhang, Q., Yilmaz, E., and Liang, S. Ranking-based method for news stance detection. In *Companion Proceedings of the The Web Conference 2018*, pp. 41–42, 2018c.

Zhang, Q., Liang, S., Lipani, A., Ren, Z., and Yilmaz, E. From stances' imbalance to their hierarchicalrepresentation and detection. In *The World Wide Web Conference*, pp. 2323–2332, 2019a.

Zhang, Q., Lipani, A., Liang, S., and Yilmaz, E. Reply-aided detection of misinformation via bayesian deep learning. In *The World Wide Web Conference*, pp. 2333–2343, 2019b.

Zhou, K., Zha, H., and Song, L. Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes. In *Artificial Intelligence and Statistics*, pp. 641–649, 2013.

Zuo, Y., Liu, G., Lin, H., Guo, J., Hu, X., and Wu, J. Embedding temporal network via neighborhood formation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2857–2866, 2018.