The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process

Hongyuan Mei HMEI@CS.JHU.EDU
Jason Eisner JASON@CS.JHU.EDU
Department of Computer Science, Johns Hopkins University
3400 N. Charles Street, Baltimore, MD 21218 U.S.A

Abstract

Many events occur in the world. Some event types are stochastically excited or inhibited—in the sense of having their probabilities elevated or decreased—by patterns in the sequence of previous events. Discovering such patterns can help us predict *which type* of event will happen next and *when*. Learning such structure should benefit various applications, including medical prognosis, consumer behavior, and social media activity prediction. We propose to model streams of discrete events in continuous time, by constructing a **neurally self-modulating multivariate point process**. This generative model allows past events to influence the future in complex ways, by conditioning future event intensities on the hidden state of a recurrent neural network that has consumed the stream of past events. We evaluate our model on multiple datasets and show that it significantly outperforms other strong baselines.

1. Introduction

Some events in the world are correlated. A single event, or a pattern of events, may help to cause or prevent future events. We are interested in learning the distribution of sequences of events (and in future work, the causal structure of these sequences). The ability to discover correlations among events is crucial to accurately predict the future of a sequence given its past, i.e., which events are likely to happen next and when they will happen.

1.1. Application Domains

We specifically focus on sequences of discrete events in continuous time ("event streams"). Modeling such sequences seems natural and useful in many applied domains:

- *Medical events.* Each patient has a stream of acute incidents, doctor's visits, tests, diagnoses, and medications. By learning from previous patients what streams tend to look like, we could predict a new patient's future from their past.
- *Consumer behavior.* Each online consumer has a stream of online interactions. By modeling the distribution over streams, we can learn purchasing patterns. E.g., buying cookies may temporarily depress the probability of buying more cookies or other desserts, while incmreasing the probability of buying milk.
- *"Quantified self" data.* Some individuals use cellphone apps to record their behaviors—such as eating specific foods, traveling among locations, performing specific work and leisure activities, sleeping and waking. By anticipating future behaviors, an app could perform helpful supportive actions, including issuing reminders or warnings and placing advance orders.

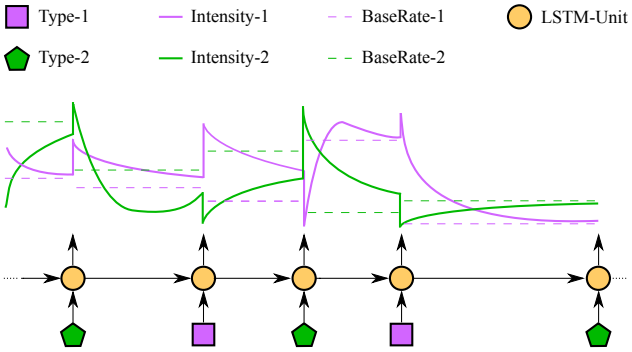


Figure 1. Drawing an event stream from a neural Hawkes process. An LSTM reads the sequence of past events (polygons) to arrive at a hidden state (orange), which determines the future "intensities" of the two types of events—their time-varying instantaneous probabilities. The intensity functions are continuous parametric curves (solid lines) determined by the most recent LSTM state, with dashed lines showing the asymptotes that they would eventually approach. In this example, events of type 1 excite type 1 but inhibit type 2. Type 2 excites itself, and excites or inhibits type 1 according to whether the count of type 2 events so far is odd or even. Those are immediate effects, shown by the sudden jumps in intensity. The events also have longer-timescale effects, shown by the shifts in the asymptotic dashed lines.

- *News events.* In principle, we could model the single stream of publicly reported events in the world, such as business news stories and significant market events.
- *Social media actions.* Previous posts, shares, comments, messages, and likes by a set of users are predictive of their future actions.
- *Scientific modeling.* For example, consider a given individual's stream of responses in a category elicitation task, "name as many vegetables as you can think of." Saying `squash` will prime `pumpkin` but will inhibit saying `squash` again. A model of such streams is a hypothesis about latent cognitive representations and priming mechanisms.
- *Dialogue events.* We could model topic shifts in political conversations, where one topic tends to lead quickly or slowly to another. Or we could model the sequence of speaker-addressee pairs as characters converse in a novel, play, or online discussion.
- *Music modeling.* We could model the stream of notes (from one instrument or many) in a musical score or performance.

1.2. Motivation for an Expressive Model Family

A basic model for event streams is the **Poisson process** (Palm, 1943), which assumes that events occur independently of one another. In a **non-homogenous Poisson process**, the (infinitesimal) probability of an event happening at time t may vary with t, but it is still independent of other events. A **Hawkes process** (Hawkes, 1971; Liniger, 2009) supposes that past events can temporarily *raise* the probability of future events, assuming that such excitation is (1) positive, (2) additive over the past events, and (3) exponentially decaying with time.

However, real-world patterns often seem to violate these assumptions. For example, (1) is violated if one event inhibits another rather than exciting it: cookie consumption inhibits cake consumption. (2) is violated when the combined effect of past events is not additive. Examples abound: The 20th advertisement does not increase purchase rate as much as the first advertisement did, and may even drive customers away. Market players may act based on their own complex analysis of market history. Musical note sequences follow some intricate language model that considers melodic trajectory, rhythm, chord progressions, repetition, etc. (3) is violated when, for example, a past event has a delayed effect, so that the effect starts at 0 and increases sharply before decaying.

We generalize the Hawkes process by determining the event intensities (instantaneous probabilities) from the hidden state of a recurrent neural network. This state is a deterministic function of the past history, updated with each successive event occurrence. It plays the same role as the state of a deterministic finite-state automaton. However, the recurrent network enjoys a continuous and infinite state space (a high-dimensional Euclidean space), as well as a learned transition function.

Our main motivation is that our model can capture effects that the Hawkes process misses. The combined effect of past events on future events can now be superadditive, subadditive, or even subtractive, and can depend on the sequential ordering of the past events. Recurrent neural networks already capture other kinds of complex sequential dependencies when applied to language modeling—that is, generative modeling of linguistic word sequences, which are governed by syntax, semantics, and habitual usage (Mikolov et al., 2010; Sundermeyer et al., 2012; Karpathy et al., 2015). We wish to extend their success (Chelba et al., 2013) to sequences of events in *continuous* time.

Another motivation for adopting a more expressive model family is to cope with missing data. Real-world datasets are often incomplete. They may systematically omit some types of events (e.g., illegal drug use, or offline purchases) which, in the true generative model, would have a strong influence on the future. They may also have stochastically missing data, where the missingness mechanism—the probability that an event is not recorded—can be complex and data-dependent (MNAR). In this setting, we can fit our model directly to the observed incomplete data streams, and use it to predict further observed incomplete streams that were generated using the same complete-data distribution and the same missingness mechanism. Note that if one knew the true complete-data distribution and the true missingness mechanism, one would optimally predict the incomplete future from the incomplete past in Bayesian fashion, by integrating over possible completions (i.e., by imputing the missing events and considering their influence on the future). Our hope is that the neural model is expressive enough that it can learn to approximate this true predictive distribution. For example, suppose that the true complete-data distribution is itself a neural Hawkes process. Then a sufficient statistic for prediction from an incomplete past would be the posterior distribution over the true hidden state that would be reached by reading the *complete* past. We hope instead to train a model whose hidden state, after it reads only the observed *incomplete* past, is nearly as predictive as this posterior distribution.

A final motivation is that in the domains discussed above, one might wish to intervene medically, economically, or socially so as to improve the future course of events. A model that can predict the result of interventions is called a causal model. Our model family can naturally be used here: any choice of parameters defines a generative story that follows the arrow of time, which can be interpreted as a causal model in which patterns of earlier events *cause*

later events to be more likely. Such a causal model predicts how the distribution over futures would change if we intervened in the sequence of events. In general, one cannot determine the parameters of a causal model based on purely observational data (Pearl, 2009). In future work, we plan to determine such parameters through randomized experiments by deploying our model family as an environment model within reinforcement learning. In this setting, an agent *tests* the effect of random interventions in order to discover their effect (exploration) and thus orchestrate more rewarding futures (exploitation).

2. Notation

We are interested in constructing distributions over **event streams** $(k_1, t_1), (k_2, t_2), \ldots, (k_I, t_I)$, where each $k_i \in \{1, 2, \ldots, K\}$ is an event type and $t_1 \leq t_2 \leq \cdots$ are times of occurrence. That is, there are K types of events, tokens of which are observed to occur in continuous time.

Given an observed event stream of this form, we can consider the *log-likelihood* ℓ of the model P:

$$\sum_{i=1}^{I} \log P((k_i, t_i) \mid \mathcal{H}_i) \tag{1}$$

where the **history** $\mathcal{H}_i$ is the prefix sequence $(k_1, t_1), (k_2, t_2), \ldots, (k_{i-1}, t_{i-1})$, and $P((k_i, t_i) \mid \mathcal{H}_i)$ is the probability that the *next* event occurs at time t_i and has type k_i. We generalize to collections of event streams in the obvious way.

Our convention is to use (lowercase) Greek letters for parameters related to the classical Hawkes process, and Roman letters for other quantities, including hidden states and affine transformation parameters. We denote vectors by boldface lowercase letters such as $\mathbf{s}$ and $\boldsymbol{\mu}$, and matrices by boldface capital Roman letters such as $\mathbf{M}$. Subscripted boldface letters denote distinct vectors or matrices: for example, $\boldsymbol{\delta}_k$, $\mathbf{h}_i$ and $\mathbf{D}_k$. Scalar quantities, including vector and matrix elements such as s_k and $\alpha_{j,k}$, are written without boldface. Function symbols are notated like their return type. All $\mathbb{R} \to \mathbb{R}$ functions are extended to apply elementwise to vectors and matrices.

3. The Model

In this section, we first review Hawkes processes, and then introduce our model one step at a time.

Formally speaking, generative models of event streams are **multivariate point processes**. A (temporal) point process is a probability distribution over functions from a given time interval to $\{0, 1\}$. A multivariate point process is formally a distribution over K-tuples of such functions. The k^{th} function indicates the times at which events of type k occurred, by taking value 1 at those times.

3.1. Self-Exciting Multivariate Point Process

A basic model of event streams is the **non-homogeneous multivariate Poisson process**. It assumes that an event of type k occurs at time t—more precisely, in the infinitesimally wide interval $[t, t + dt)$—with probability $\lambda_k(t)dt$. The value $\lambda_k(t) \geq 0$ can be regarded as a rate per unit time, just like the parameter λ of an ordinary Poisson process. λ_k is known as the **intensity function**.

A well-known generalization that captures interactions is the **self-exciting multivariate point process (SE-MPP)**, or **Hawkes process** (Hawkes, 1971; Liniger, 2009), in which past events conspire to *raise* the probability of future events. Such excitation is positive, additive over the past events, and exponentially decaying with time:

$$\lambda_k(t) = \mu_k + \sum_{i:t_i<t} \alpha_{k_i,k} \exp(-\delta_{k_i,k}(t - t_i)) \tag{2}$$

where $\mu_k \geq 0$ is the base intensity of event type k, $\alpha_{j,k} \geq 0$ is the degree to which an event of type j initially excites event type k, and $\delta_{j,k} > 0$ is the decay rate of that excitation. Upon each event occurrence, all intensities are elevated to various degrees, but then will decay toward their base rates $\boldsymbol{\mu}$.

3.2. Self-Modulating Multivariate Point Processes

The positivity constraints in the Hawkes process prevent it from learning more complicated influences. For example, the positive interaction parameters $\alpha_{j,k}$ fail to capture inhibition effects, in which past events *reduce* the intensity of future events. The positive base rates $\boldsymbol{\mu}$ fail to capture the inherent inertia of some events, which are unlikely until their cumulative excitation by past events crosses some threshold. To remove such limitations, we introduce *self-modulating* models. Here the intensities of future events are stochastically *modulated* by the past history, where the term "modulation" is meant to encompass both excitation and inhibition. Furthermore, the intensity $\lambda_k(t)$ can fluctuate non-monotonically between successive events, because the competing excitatory and inhibitory influences may decay at different rates.

3.2.1. Decomposable Self-Modulating MPP

Our first move is to enrich the Hawkes model's expressiveness while still maintaining its decomposable structure. We refer to this intermediate model as a **decomposable self-modulating multivariate point process (D-SM-MPP)**.

We relax the positivity constraints on $\alpha_{j,k}$ and μ_k, allowing them to range over $\mathbb{R}$, which allows *inhibition* ($\alpha_{j,k} < 0$) and *inertia* ($\mu_k < 0$). However, the resulting total activa-

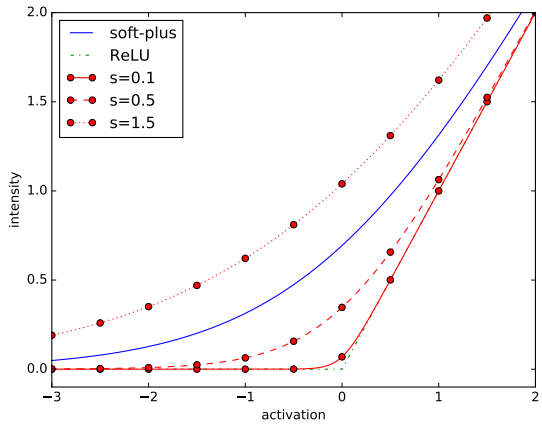


Figure 2. The soft-plus function is a soft approximation to a rectified linear unit (ReLU), approaching it as x moves away from 0. We use it to ensure a strictly positive intensity function. We incorporate a scale parameter s that controls the curvature.

tion could now be negative. We therefore pass it through a non-linear **transfer function** $f : \mathbb{R} \to \mathbb{R}_+$ to obtain a positive intensity function as required:

$$\lambda_k(t) = f(\tilde{\lambda}_k(t)) \tag{3a}$$

$$\tilde{\lambda}_k(t) = \mu_k + \sum_{i:t_i<t} \alpha_{k_i,k} \exp(-\delta_{k_i,k}(t - t_i)) \tag{3b}$$

The influence of each previous event still decays toward zero at a rate $\delta_{j,k} > 0$. Thus, as t increases between events, the intensity $\lambda_k(t)$ may rise and/or fall but eventually approaches the base rate $f(\mu_k)$.

What non-linear function f should we use? The ReLU function $f(x) = \max(x, 0)$ seems at first a natural choice. However, it returns 0 for negative x; we need to keep our intensities strictly positive at all times when an event could possibly occur, to avoid infinitely bad log-likelihood at training time or infinite log-loss at test time. A better choice would be the "soft-plus" function $f(x) = \log(1+\exp(x))$, which is strictly positive and approaches ReLU when x is far from 0. Unfortunately, "far from 0" is defined in units of x, so this choice would make our model sensitive to the units used to measure time. For example, if we switch the units of t from seconds to milliseconds, then the base intensity $f(\mu_k)$ must become 1000 times lower, forcing μ_k to be very negative and thus creating a much stronger inertial effect. To avoid this problem, introduce a scale parameter $s > 0$ and define $f(x) = s\log(1 + \exp(x/s))$. The scale parameter s controls the curvature of $f(x)$, which approaches ReLU as $s \to 0$, as shown in Figure 2. We can regard $f(x)$, x, and s as rates, with units of inverse time, so that $f(x)/s$ and x/s are unitless quantities related by soft-plus. We actually learn a separate scale parameter s_k for each event type k, which will adapt to the rate of events of that type.

In short, we instantiate equation (3a) as

$$\lambda_k(t) = f_k(\tilde{\lambda}_k(t)) = s_k \log\left(1 + \exp\left(\tilde{\lambda}_k(t)/s_k\right)\right) \tag{4}$$

3.2.2. Neurally Self-Modulating MPP

Our second move removes the restriction that the past events have independent, additive influence on $\tilde{\lambda}_k(t)$. Whereas equation (3b) decomposed $\tilde{\lambda}_k(t)$ as a simple summation, we substitute a recurrent neural network. This allows learning a complex dependence of the intensities on the number, order, and timing of past events. We refer to the result as a **neurally self-modulating multivariate point process (N-SM-MPP)**, or a **neural Hawkes process** for short.

Just as before, each event type k has an time-varying intensity $\lambda_k(t)$, which jumps discontinuously at each new event, and at other times drifts back toward a baseline intensity. In the new process, however, these dynamics depend on the hidden state $\mathbf{h}_i$ of a recurrent neural network, specifically an LSTM (Hochreiter & Schmidhuber, 1997; Graves, 2012), which has read the sequence of past pairs $(k_1, t_1), \ldots, (k_{i-1}, t_{i-1})$. This hidden state determines the intensities from time t_{i-1} onward, up until the next event k_i stochastically occurs at some time t_i. At that point, the LSTM is able to read (k_i, t_i) and so updates to state $\mathbf{h}_{i+1}$. [1] The hidden state summarizes the sequence of past events, just as in a LSTM language model (Mikolov et al., 2010), and is assumed to be a sufficient statistic for future prediction. An example trajectory of an instance of our model is shown in Figure 1.

For all times $t \in (t_{i-1}, t_i]$, we propose to derive the intensity $\lambda_k(t)$ from $\mathbf{h}_i$ as follows:

$$\lambda_k(t) = f_k(\tilde{\lambda}_k(t)) = s_k \log(1 + \exp(\tilde{\lambda}_k(t)/s_k)) \tag{5a}$$

$$\tilde{\lambda}_k(t) = \mathbf{w}_k^\top \mathbf{c}_k(t) + \mu_k \tag{5b}$$

$$\mathbf{c}_k(t) = \mathbf{h}_i \odot \exp\left(-\boldsymbol{\delta}_k(t - t_{i-1})\right) \tag{5c}$$

Equation (5c) defines $\mathbf{c}_k(t)$ to be a time-decayed copy of the hidden vector $\mathbf{h}_i$. Note that each hidden node may have an individual decay rate, which also depends on k, reflecting the time course of its individual influence on event type k. We convert this decayed vector $\mathbf{c}_k(t)$ to an intensity $\lambda_k(t)$, in typical neural network style: by passing it through an affine activation function (5b) and then the same nonlinear transfer function (5a) as before.

[1] Although we use one-layer LSTMs in our present experiments, a natural extension is to use multi-layer ("deep") LSTMs (Graves et al., 2013), in which case $\mathbf{h}_i$ is the hidden state of the top layer.

Generalizing further, we determine the base intensities $\boldsymbol{\mu}$ and decay rates $\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, \ldots, \boldsymbol{\delta}_K$ from the current LSTM state $\mathbf{h}_i$, so that they change when new events occur:[2]

$$\boldsymbol{\mu} = \mathbf{M}\mathbf{h}_i \tag{6a}$$

$$\boldsymbol{\delta}_k = \log(1 + \exp(\mathbf{D}_k \mathbf{h}_i)) \tag{6b}$$

Just as in an RNN or LSTM language model, our hidden states $\mathbf{h}_i$ (and their decayed versions $\mathbf{c}_k(t)$) are *deterministic* functions of the past events $\mathcal{H}_i$ (and the current time t). Thus, the event intensities at any time are also deterministic, like the language model's word probabilities. The stochastic part of the model is the random choice—based on these intensities—of *which* event happens next and *when* it happens. The events are in competition: an event with high intensity is likely to happen sooner than an event with low intensity, and that event is fed back into the LSTM. If no event type has high intensity, it may take a long time for the next event to occur.

3.2.3. LSTM Details

A few details are necessary to construct the LSTM. To update the LSTM state, we must provide it with a concatenated input vector representing (k_i, t_i). We encode k_i as a one-hot vector in $\mathbb{R}^K$, just as word types are encoded for LSTM language models. We encode t_i as a vector $\boldsymbol{\tau} \in \mathbb{R}^B$, by passing $\Delta_i = t_i - t_{i-1}$ (the elapsed time since the previous event) through a bank of shifted ReLUs:

$$\tau_b = \max(\Delta_i - \beta_b, 0) \text{ for } 1 \leq b \leq B \tag{7}$$

where $\boldsymbol{\beta}$ is jointly learned with the rest of the model (we fix $\beta_1 = 0$). We choose this encoding of $\Delta_i \in \mathbb{R}$ because the LSTM will apply a learned linear transformation to its input, and linear functions of $\boldsymbol{\tau}$ are piecewise-linear functions of Δ_i.

Rather than directly learning the hidden state $\mathbf{h}_1$ used at time $t = 0$, we initialize the LSTM's hidden state to $\mathbf{h}_0 = \mathbf{0}$, and have it read a special beginning-of-sequence (BOS) event (k_0, t_0), where k_0 is a special event type (i.e., expanding the LSTM input dimensionality by one) and t_0 is set to be 0. We do not generate the BOS event but only condition on it, which is why the log-likelihood formula (section 2) only sums over $i = 1, 2, \ldots$. This design is well-suited to multiple settings. In some settings, time 0 is special. For example, if we release many children into a carnival and observe the stream of their actions there, then BOS is the release event and no other events can possibly precede it. In other settings, data before time 0 are simply missing, e.g., the observation of a patient starts in midlife; nonetheless, BOS indicates the beginning of the *observed* sequence. In both kinds of settings, $\mathbf{h}_1$ characterizes the model's belief about the unknown state of the true system at time 0. Computing $\mathbf{h}_1$ by explicitly transitioning on BOS ensures that $\mathbf{h}_1$ falls in the feasible space of hidden states. More important, in future work where we can observe rich data about each event, we will be able to attach sequence metadata to the BOS event, and the LSTM can learn how this should affect $\mathbf{h}_1$.

At the other end of the sequence, we could optionally choose to identify one of the observable types in $\{1, 2, \ldots, K\}$ as a special end-of-sequence (EOS) event after which the sequence cannot possibly continue. If the model generates EOS, all intensities are permanently forced to 0, so it is not necessary for the model parameters to explain why no further events are observed.

Training the model means learning the LSTM parameters and $\boldsymbol{\beta}$, as well as all parameters mentioned in the previous section: s_k, $\mathbf{w}_k$, and $\mathbf{D}_k$ for $k \in \{1, 2, \ldots, K\}$, as well as $\mathbf{M}$.

3.2.4. Remark: Closure Under Superposition

Decomposable models are closed under superposition of event streams. Let $\mathcal{E}$ and $\mathcal{E}'$ be random event streams, on a common time interval $[0, T]$ but over disjoint sets of event types. If each stream is distributed according to a Hawkes process, then their superposition—that is, $\mathcal{E} \cup \mathcal{E}'$ sorted into temporally increasing order—is also distributed according to a Hawkes process. It is easy to exhibit parameters for such a process, using a block-diagonal matrix of $\alpha_{j,k}$ so that the two sets of event types do not influence each other. The closure property also holds for our decomposable self-modulating process, and for the same simple reason.

This is important since in various real settings, some event types tend not to interact. For example, the activities of two people Jay and Kay rarely influence each other,[3] although they are simultaneously monitored and thus form a single observed stream of events. We want our model to handle such situations naturally, rather than insisting that Kay must react to what Jay does.

Thus, we have designed our neurally self-modulating process to approximately preserve this ability to insulate event k from event j. By setting specific elements of $\mathbf{w}_k$ and $\mathbf{D}_k$ to 0, we can ensure that the intensity function $\lambda_k(t)$ depends on only a subset S of the LSTM hidden nodes. Then by setting specific LSTM parameters, we can make the nodes in S insensitive to events of type j: events of type j should close these nodes' input gates and open their forget gates, so that they simply copy their activation. Now

[2] They should therefore be subscripted by i. We omit these subscripts to simplify the notation.

[3] Their surnames might be Box and Cox, after the 19th-century farce about a day worker and a night worker unknowingly renting the same room. But any pair of strangers would do.

events of type j cannot affect the intensity $\lambda_k(t)$. Finally, to really obtain closure under superposition, we must ensure that the hidden states in S are affected in the same way when the LSTM reads $(k,1),(j,3),(j,8),(k,12)$ as when it reads $(k,1),(k,12)$, even though the intervals Δ between successive events are different. A clean solution is to have the LSTM read a richer encoding of t_i, which encodes not just a single interval Δ_i as in section 3.2.3, but all K of the intervals since the most recent event of type 1, 2, …, K respectively. With this change, we can explicitly construct a superposition process[4] with LSTM state space $\mathbb{R}^{d+d'}$—the cross product of the state spaces $\mathbb{R}^d$ and $\mathbb{R}^{d'}$ of the original processes.

If we know *a priori* that particular event types interact only weakly, we can impose an appropriate prior on the neural Hawkes parameters. And in future work with large K, we plan to investigate the use of sparsity-inducing regularizers during parameter estimation, to create an inductive bias toward models that have limited interactions, without specifying which particular interactions are present.

The superposition question is natural for event streams. It barely arises for ordinary sequence models, such as language models, since the superposition of two sentences is not well-defined unless all of the words carry distinct real-valued timestamps. However, recall that the "shuffle" of two sentences is the set of *possible* interleavings of their words—i.e., the set of superpositions that could result from assigning increasing timestamps to the words of each sentence, without duplicates. It is a standard exercise in formal language theory to show that regular languages are closed under shuffle. This is akin to our remark that neural-Hawkes-distributed random variables are closed under superposition, and indeed uses a similar cross-product construction on the finite-state automata. An important difference is that the shuffle construction does not require disjoint alphabets in the way that ours requires disjoint sets of event types. This is because finite-state automata allow nondeterministic state transitions and our processes do not.

4. Algorithms

For the proposed models, given an event stream over the time interval $[0,T]$, the log-likelihood of the parameters turns out to be given by a simple formula—the log-intensities of the events that happened, at the times they happened, minus an integral of the total intensities over the entire time range $[0,T]$:

$$\ell = \sum_{i:t_i<T} \log \lambda_{k_i}(t_i) - \underbrace{\int_{t=0}^{T} \sum_{j=1}^{K} \lambda_j(t) dt}_{\text{call this } \Lambda} \tag{8}$$

The derivation is given in Appendix A. Intuitively, the $-\Lambda$ term (which is ≤ 0) sums the log-probabilities of infinitely many *non*-events: the probability that there was *not* an event of type j in the infinitesimally wide interval $[t, t+dt)$ is $1-\lambda_j(t)dt$, whose log is $-\lambda_j(t)dt$.

We can locally maximize ℓ using any stochastic gradient method. For this, we need to be able to get an unbiased estimate of the gradient $\nabla\ell$ with respect to the model parameters. This is straightforward to obtain by back-propagation. The trick for handling the integral is that the single function evaluation $T\sum_{j=1}^{K}\lambda_j(t)$ at a random $t \sim \mathrm{Unif}(0,T)$ gives an unbiased estimate of the entire integral—that is, its expected value is Λ. Its gradient via back-propagation is therefore a unbiased estimate of $\nabla\Lambda$ (since gradient commutes with expectation). The Monte Carlo algorithm in Algorithm 1 averages over several samples to reduce the variance of this noisy estimator.

Algorithm 1 Integral Estimation (Monte Carlo)

Input: interval $[0,T]$; model parameters and events $(k_1,t_1),\ldots$ for determining $\lambda_j(t)$
$\Lambda \leftarrow 0$; $\nabla\Lambda \leftarrow \mathbf{0}$
for N samples : ▷ *e.g., take $N>0$ proportional to T*
  draw $t \sim \mathrm{Unif}(0,T)$
  for $j \leftarrow 1$ **to** K :
    Λ += $\lambda_j(t)$ ▷ *via current model parameters*
    $\nabla\Lambda$ += $\nabla\lambda_j(t)$ ▷ *via back-propagation*
$\Lambda \leftarrow T\Lambda/N$; $\nabla\Lambda \leftarrow T\nabla\Lambda/N$ ▷ *weight the samples*
return $(\Lambda, \nabla\Lambda)$

Given an event stream prefix (k_1,t_1), (k_2,t_2), …, (k_{i-1},t_{i-1}), we may wish to predict the *time* and *type* of the single next event. Let $\lambda(t) = \sum_{j=1}^{K}\lambda_k(t)$, the total intensity of all event types. The next event's time t_i has density

$$p_i(t) = P(t_i = t \mid \mathcal{H}_i) = \lambda(t)\exp\left(-\int_{t_{i-1}}^{t}\lambda(s)ds\right) \tag{9}$$

To predict a single time whose expected L2 loss is as low as possible, we should choose

$$\hat{t}_i = \mathbb{E}[t_i \mid \mathcal{H}_i] = \int_{t_{i-1}}^{\infty} t p_i(t) dt \tag{10}$$

Given the next event time t_i, the most likely type would be $\mathrm{argmax}_k\, \lambda_k(t_i)/\lambda(t_i)$. But the most likely next event type

[4] To be precise, we can approximate the superposition model arbitrarily closely, but not exactly, because a standard LSTM gate cannot be fully opened or closed. The openness is traditionally given by a sigmoid function and so falls in $(0,1)$, never achieving 1 or 0 exactly unless we are willing to set parameters to $\pm\infty$.

without knowledge of t_i is

$$\hat{k}_i = \operatorname*{argmax}_k \int_{t_{i-1}}^{\infty} \frac{\lambda_k(t)}{\lambda(t)} p_i(t) dt \tag{11}$$

The integrals in equations (10) and (11) can be estimated by Monte-Carlo sampling as before, with an inner loop to compute $p_i(t)$ by similarly sampling from (9). In equation (11), we recommend a paired comparison that uses the same t values for each k in the argmax; this also lets us share the $\lambda(t)$ and $p_i(t)$ computations across all k.

If we wish to draw longer sequences from the model, we can adopt the thinning algorithm (Lewis & Shedler, 1979; Liniger, 2009) that is commonly used for the Hawkes process, as shown in Algorithm 2. Suppose we have already sampled the first $i-1$ events. For $t \in (t_{i-1}, \infty)$, let $\lambda_k^i(t)$ denote the intensity $\lambda_k(t)$ that the model will define at time t if event t_i has not yet happened in the interval (t_{i-1}, t). In order to sample event i, we will (for each k) make a draw from the non-homogeneous Poisson process whose intensity function is λ_k^i. These K drawn sequences are in a race to have the earliest event. That event joins the multivariate event stream as event i, where it updates the LSTM state and thus modulates the intensities for event $i+1$.

A draw from this non-homogeneous Poisson process is equivalent to *independently* choosing at each time $t \in (t_{i-1}, \infty)$, with probability $\lambda_k^i(t)$, whether an event occurs. We can do this by *independently* applying rejection sampling at each time t: choose with larger probability λ^* whether a "proposed event" occurs at time t, and if it does, accept the proposed event with probability only $\frac{\lambda_k^i(t)}{\lambda^*}$. Now, drawing all of the *proposed* events in parallel is simple: it is equivalent to a single draw from a *homogenous* Poisson process with constant rate λ^*, whence the intervals between successive proposed events are IID $\mathrm{Exp}(\lambda^*)$. The inner repeat loop in Algorithm 2 lazily carries out just enough of this infinite homogenous draw to determine the time $t_{i,k}$ of the earliest event in the non-homogeneous draw.

How do we find the upper bound λ^* (see Algorithm 2)? Equations (2), (3) and (5) each define $\lambda_k^i = f_k(\tilde{\lambda}_k^i)$ where f_k is monotonically increasing (possibly identity) and $\tilde{\lambda}_k^i$ is an affine combination of positive decreasing functions. By replacing any negative coefficients in this affine combination with 0, we obtain a new function, also with domain (t_{i-1}, ∞), which is everywhere greater than λ_k^i and is decreasing. We may take λ^* to be the limit of this new function as $t \to t_{i-1}$. This is easily obtained by plugging t_{i-1} into the continuous formulas that define the function.

Algorithm 2 Data Simulation (thinning algorithm)

Input: interval $[0, T]$; model parameters
$t_0 \leftarrow 0$; $i \leftarrow 1$
while $t_{i-1} < T$: ▷ *draw event i, as it might fall in [0,T]*
  for $k = 1$ **to** K : ▷ *draw "next" event of each type*
    find upper bound $\lambda^* \geq \lambda_k^i(t)$ for all $t \in (t_{i-1}, \infty)$
    $t \leftarrow t_{i-1}$
    repeat
      draw $\Delta \sim \mathrm{Exp}(\lambda^*)$, $u \sim \mathrm{Unif}(0, 1)$
      $t \mathrel{+}= \Delta$ ▷ *time of next proposed event*
    until $u\lambda^* \leq \lambda_k^i(t)$ ▷ *accept proposal with prob* $\frac{\lambda_k^i(t)}{\lambda^*}$
    $t_{i,k} \leftarrow t$
  $t_i \leftarrow \min_k t_{i,k}$; $k_i \leftarrow \operatorname{argmin}_k t_{i,k}$ ▷ *earliest event wins*
  $i \leftarrow i + 1$
return $(k_1, t_1), \ldots (k_{i-1}, t_{i-1})$

5. Related Work

The Hawkes process has been widely used to model event streams in various applications, including topic modeling and clustering of text document streams (He et al., 2015; Du et al., 2015a), constructing and inferring network structure (Yang & Zha, 2013; Choi et al., 2015; Etesami et al., 2016), personalized recommendations based on users' temporal behavior (Du et al., 2015b), discovering patterns in social interaction (Guo et al., 2015; Lukasik et al., 2016), learning causality (Xu et al., 2016), and so on.

Recent interest has focused on expanding the expressivity of Hawkes processes. Zhou et al. (2013) describe a self-exciting process that removes the assumption of exponentially decaying influence (as we do). They replace the scaled-exponential summands in equation (2) with learned positive functions of time (the choice of function again depends on k_i, k). Lee et al. (2016) generalize the constant excitation parameters $\alpha_{j,k}$ to be stochastic, which increases expressivity. Our model also allows non-constant interactions between event types, but arranges these via deterministic functions of LSTM hidden states instead of stochastic functions. Wang et al. (2016) consider non-linear effects of past history on the future, by passing the intensity functions of the Hawkes process through a non-parametric isotonic link function g, which is in the same place as our nonlinear function f_k. In contrast, our f_k has a fixed parametric form (learning only the scale parameter), and is approximately linear when x is large. This is because we model non-linearity (and other complicated effects) with LSTM, and use f_k only to ensure positiveness of the intensity functions.

We follow this line of work with our self-modulating multivariate point processes. Our first novel contribution is to allow inhibition ($\alpha_{j,k} < 0$) and inertia ($\mu_k < 0$). Our second novel contribution is to drop the assumption of additive influence. The decomposable version of our model does preserve the usual notion that $\lambda_k(t)$ essentially sums the influences of unboundedly many past events, giving more

weight to more recent events. However, the neural version instead sums the influences of a fixed set of hidden nodes, each of which is a flexible learned function of the unbounded event history. Our hidden nodes' influences decay at different rates, just as in other models, past events of different types have their influences decay at different rates.

After doing our experiments, we discovered that Du et al. (2016) had also combined Hawkes processes with recurrent neural networks. Their neural network state $\mathbf{h}$ is very similar to ours. However, they model t_i and k_i independently. They use $\mathbf{h}$ to construct a single time-decaying intensity function $\lambda(t)$ that determines the next event's *time* t_i. Separately, they use $\mathbf{h}$ to construct a softmax distribution over the next event's *type* k_i. In contrast, we use $\mathbf{h}$ to construct a different time-varying intensity function $\lambda_k(t)$ for each event type k. Notice that time and type are *not* independent in our model, since if the next event happens at a given time t, the probability that it has type k is proportional to $\lambda_k(t)$ and hence depends on t.

Our setup also gives us a richer parameterization of the total intensity $\lambda(t)$, as a summation $\sum_k \lambda_k(t)$.[5] Even before the summation, our model (5) of each $\lambda_k(t)$ is more flexible than Du et al. (2016)'s model of $\lambda(t)$. They model $\lambda(t)$ as an exponentiated affine function of $\mathbf{h}$, scaled by a factor that decays exponentially toward zero. In our model, however, it is not the intensity that decays, but the influence on the intensity of the various components of the state $\mathbf{h}$. This can lead to more interesting nonlinear and non-monotonic variation with time. In addition, our strategy ensures that as time passes, the intensity λ_k of event type k approaches some positive learned base rate $f(\mu_k)$, rather than necessarily approaching zero as for Du et al. (2016). Another difference between the models is that we ensure positive intensities by taking f = soft-plus, whereas Du et al. (2016) take $f = \exp$; thus the combined effect of the hidden units is essentially additive for us (inspired by the classical Hawkes process), but multiplicative for Du et al..

Our decision to model the total activation $\lambda(t)$ as a sum over types is motivated by the discussion in section 3.2.4. Event types j and k may be influenced by quite different properties of the past history. Thus, it makes sense to individually model how much each event type "wants to happen," based on different projections of $\mathbf{h}$. In effect, we then determine which event is likely to happen next by modeling an explicit competition among different event types, in which their relative strengths wax and wane over time. Thus, it is easy for our model to raise the intensity of one event without changing the intensities of any other event. Du et al. (2016) really only have a single type of event whose tokens are independently labeled with numbers in $\{1, 2, \ldots, K\}$—what Liniger (2009) calls a *pseudo*-Hawkes process. Thus, for their model, the only way to raise the intensity of one single event is to raise the total intensity and then change the relative intensities to compensate. That is perhaps more difficult, though still possible, for a LSTM to learn.

[5] If Du et al. had constructed separate λ_k functions, their total intensity λ would be a sum of exps, which is more expressive than the single exp they actually use. This is analogous to how introducing latent variables into a graphical model increases its representational power.

6. Experimental Setup

We fit our various models on several simulated and real-world datasets, and evaluated them in each case by the log-likelihood that they assigned to held-out data. We also compared our approach with that of Du et al. (2016) on their prediction task. Table 1 shows statistics about each dataset that we use in this paper. The model sizes are shown in Table 2.

We used a single-layer LSTM (Graves, 2012) in section 3.2.2, selecting the number of hidden nodes from a small set $\{64, 128, 256, 512, 1024\}$ based on the performance on the dev set of each dataset. For the time feature $\boldsymbol{\tau}$, we fixed the dimensionality $B = 33$ (see section 3.2.3), giving 32 trainable parameters. We empirically found that the model performance is robust to these hyperparameters.

When estimating integrals with Monte Carlo sampling, N is the number of sampled negative observations in Algorithm 1, while I is the number of positive observations. In practice, setting $N = I$ was large enough for stable behavior, and we used this setting during training. For evaluation on dev and test data, we took $N = 10\,I$ for extra accuracy, or $N = I$ when I was very large.

For learning, we used the Adam algorithm with its default settings (Kingma & Ba, 2015). Adam is a stochastic gradient optimization algorithm that continually adjusts the learning rate in each dimension based on adaptive estimates of low-order moments. Our training objective was unregularized log-likelihood.[6] We initialized the Hawkes process parameters and s_k scale factors to 1, the β_b thresholds (section 3.2.3) to empirical quantiles of the intervals Δ_i between successive events in training data, and all other non-LSTM parameters (section 3.2.3) to small random values from $\mathcal{N}(0, 0.01)$. We performed early stopping based on log-likelihood on the held-out dev set.

[6] L_2 regularization did not appear helpful in pilot experiments, at least for our dataset size and when sharing a single regularization coefficient among all parameters.

DATASET	# OF EVENT TYPES	# OF EVENT TOKENS			SEQUENCE LENGTH		
		TRAIN	DEV	DEVTEST	MIN	MEAN	MAX
SYNTHETIC (SE-MPP)	5	482947	59683	60067	20	60.27	100
SYNTHETIC (D-SM-MPP)	5	480219	60380	59638	20	60.02	100
SYNTHETIC (N-SM-MPP)	5	478182	60407	60712	20	59.93	100
RETWEETS	3	2176116	215521	218465	50	108.77	264
MEMETRACK	5000	109415	14932	15440	1	2.92	32
MIMIC	75	9622	1250	1223	2	3.72	33

Table 1. Statistics about each dataset.

DATASET	# OF EVENT TYPES	# OF MODEL PARAMETERS		
		SE-MPP	D-SM-MPP	N-SM-MPP
SYNTHETIC	5	55	60	1127717 ($R = 256$)
RETWEETS	3	21	24	251811 ($R = 128$)
MEMETRACK	5000	50005000	50010000	21496616 ($R = 64$)

Table 2. Size of each trained model on each dataset. The number of parameters of neural Hawkes process is followed the by the number of hidden nodes R in its LSTM (chosen automatically on dev data).

6.1. Simulated Datasets

Our hope is that the neural Hawkes process is a flexible tool that can be used to fit naturally occurring data. We first checked that we could successfully fit data generated from *known* distributions.[7] When the generating distribution actually fell within our model family, could our training procedure recover the distribution in practice? When the data came from a decomposable process, could we nonetheless train our neural process to fit the distribution well?

We used Algorithm 2 to sample event streams from three different processes with randomly generated parameters: (a) a standard Hawkes process (SE-MPP, section 3.1), (b) our decomposable self-modulating process (D-SM-MPP, section 3.2.1), (c) our neural self-modulating process (N-SM-MPP, section 3.2.2). We then tried to fit each dataset with all three models.

For each dataset, we took $K = 5$ as the number of event types. To generate each event stream, we first chose the sequence length I (number of event tokens) uniformly from $\{20, 21, 22, \ldots, 100\}$ and then used the thinning algorithm to sample the first I events over the interval $[0, \infty)$. For subsequent training or testing, we treated this sequence (appropriately) as the complete set of events observed on the interval $[0, T]$ where $T = t_I$, the time of the last generated event. For each dataset, we generate 8000, 1000, 1000 and 2000 streams for the training, dev, devtest, and test sets respectively.

For SE-MPP, we sampled the parameters as $\mu_k \sim \mathrm{Unif}[0.0, 1.0]$, $\alpha_{j,k} \sim \mathrm{Unif}[0.0, 1.0]$, and $\delta_{j,k} \sim \mathrm{Unif}[10.0, 20.0]$. The large decay rates $\delta_{j,k}$ were needed to prevent the intensities from blowing up as the stream accumulated more events. For D-SM-MPP, we sampled the parameters as $\mu_k \sim \mathrm{Unif}[-1.0, 1.0]$, $\alpha_{j,k} \sim \mathrm{Unif}[-1.0, 1.0]$, and $\delta_{j,k} \sim \mathrm{Unif}[10.0, 20.0]$. For N-SM-MPP, we sampled parameters from $\mathrm{Unif}[-1.0, 1.0]$.

The results are shown in Table 3. We found that all models were able to fit the (a) and (b) datasets well with no statistically significant difference among them, but that the (c) model was substantially and significantly better at fitting the (c) dataset. In all cases, the (c) model was able to obtain a low KL divergence from the true generating model (the difference from the oracle column). This result suggests that the neural Hawkes process may be a wise choice: it introduces extra expressive power that is sometimes necessary and does not appear (at least in these experiments) to be harmful when it is not necessary.

6.2. Real-World Media Datasets

We next evaluated our model on two real-world datasets—Retweets (Zhao et al., 2015) and MemeTrack (Leskovec & Krevl, 2014).

Retweets Dataset On Twitter, novel tweets are generated from some distribution, which we do not model here. Each novel tweet then serves as a BOS event (see section 3.2.3) for a subsequent stream of retweets. We are interested in the dynamics of these streams: who retweets and when. We assume that the various streams are drawn independently from a latent process, so that they do not affect one another. The Retweets dataset includes 166076 such retweet

[7] In these pilot experiments, for both the true distributions and the learned distributions, s_k was fixed to 1 and Δ_i was encoded as a 32-bit floating point vector.

DATASETS	MODELS			
	ORACLE	SE-MPP	D-SM-MPP	N-SM-MPP
SE-MPP	-1.3711	$\mathbf{-1.3717}$ $(-1.3835, -1.3603)$	-1.3740 $(-1.3856, -1.3624)$	-1.3729 $(-1.3843, -1.3611)$
D-SM-MPP	-1.2536	-1.2600 $(-1.2691, -1.2514)$	-1.2591 $(-1.2681, -1.2506)$	$\mathbf{-1.2553}$ $(-1.2649, -1.2468)$
N-SM-MPP	-0.2708	-0.4675 $(-0.4780, -0.4570)$	-0.4663 $(-0.4769, -0.4559)$	$\mathbf{-0.3152}$ $(-0.3264, -0.3044)$

Table 3. Log-likelihood (reported in nats per event) of each model on held-out synthetic data. Each row is a synthetic dataset, labeled by its generating distribution. The "Oracle" column shows the log-probability of the held-out data according to the true generating distribution (i.e., the model and parameters that were used to generate the dataset). Each of the following columns displays the log-probability of the same data according to one of the trained models. The best of these is highlighted in bold, and a 95% bootstrap percentile confidence interval is shown for each one.

streams, and each of the retweet is marked with the retweet time (relative to the original tweet creation) and the number of followers of the retweeter.

The dataset is interesting for its temporal pattern. People like to retweet an interesting post soon after it is created and retweeted by others, but may gradually lose interest—resulting in short time intervals between retweets at the beginning but significantly longer ones in the end. In other words, the sequence is generated in a *self-exciting state*, in which previous retweets increase the intensities of future retweets, but eventually the intensities die down and events stop exciting one another. The decomposable models are essentially incapable of modeling such a phase transition, but our neural model should have the capacity to do so.

Recall that the dataset does not specify the identity of each retweeter, only his or her popularity. We therefore divide the events into three types: retweets by "small," "medium" and "large" users. Small users have fewer than 120 followers (50% of events), medium users have fewer than 1363 (45% of events), and the rest are large users (5% events). Given the past retweet history, our model must learn to predict how soon it will be retweeted again and how popular the retweeter is.

We truncated sequences to a maximum length of 264, which affected 20% of them. For computing training and test likelihoods, we treated each sequence as the complete set of events observed on the interval $[0, T]$, where 0 denotes the time of the original tweet (which is not included in the sequence) and T denotes the time of the last tweet in the (truncated) sequence.

We randomly sampled disjoint train, dev, devtest and test sets with 20000, 2000, 2000 and 5000 sequences respectively. We generated learning curves by training our models on increasingly long prefixes of the training set. In each case, we tuned hyper-parameters (e.g., number of hidden nodes) on the dev set. We show the results on the devtest set only; the final test results will be computed for the final version of the paper.

Figure 3 shows how the performance of each model on the devtest set varies with the size of the training set. As we can see, both of our self-modulating processes significantly outperform the Hawkes process.

There is no obvious *a priori* reason to expect inhibition or even inertia in this application domain, which explains why the D-SM-MPP makes only a small improvement over the best Hawkes process models. But D-SM-MPP seems to have much more stable behavior over different training sets, whereas Hawkes process training seems to sometimes get stuck in local optima.

Our full N-SM-MPP model also remains stable—and outperforms the non-neural version by a large margin. Its *consistent* superiority over the other two models is shown by the scatterplots in Figures 4 and 5, demonstrating the importance of the neural component of our model.

MemeTrack Dataset This dataset is similar in conception to Retweets, but with many more event types ($K = 5000$). It considers the reuse of fixed phrases, or "memes," in online media. It contains time-stamped instances of meme use in articles and posts from 1.5 million different blogs and news sites. The dataset spans 10 months from August 2008 till May 2009, with several hundred million documents. We are interested in how the future occurrence of a meme is affected by its past trajectory across different websites. For example, how likely is one meme to be used by a liberal news website after it has been mentioned by some conservative websites? How likely is a meme to move from an economics website into political news?

As in Retweets, we decline to model the appearance of novel memes. Each novel meme serves as the BOS event for a stream of mentions on other websites, which we do model. The K event types correspond to the different websites. Given one meme's past trajectory across websites, our model must learn to predict how soon it will be men-

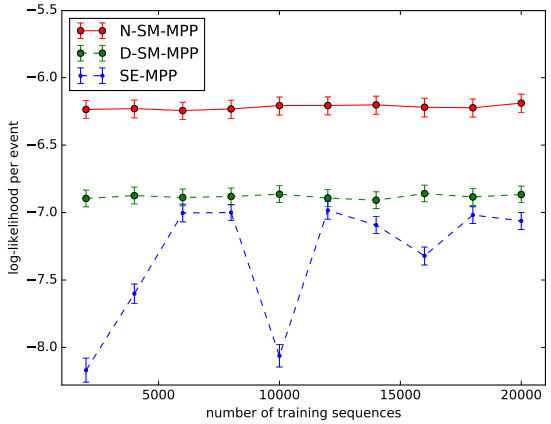


Figure 3. Learning curve (with 95% error bars) of all three models on the Retweets dataset. Our full self-modulating model significantly outperforms the decomposable one, which significantly outperforms the Hawkes process.

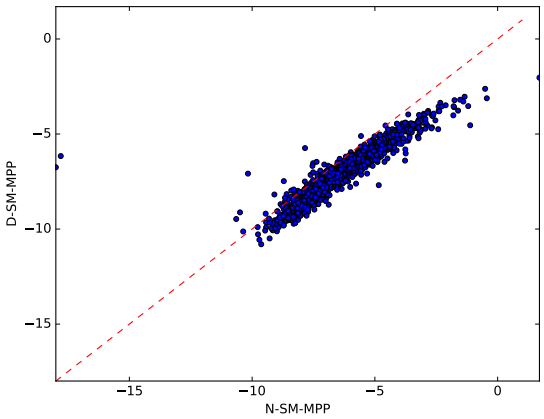


Figure 4. Scatterplot of N-SM-MPP vs. D-SM-MPP, comparing the log-likelihood of the two models (when trained on our full Retweets training set) with respect to *each* of the 2000 devtest sequences. Nearly all points fall to the right of $y = x$, since N-SM-MPP (the neural Hawkes process) is consistently more predictive than our non-neural model.

tioned again and where.

We used the version of the dataset processed by Gomez Rodriguez et al. (2013), which selected the top 5000 websites in terms of the number of memes they mentioned. We truncated sequences to a maximum length of 32, which affected only 1% of them. We randomly sampled disjoint train, dev, devtest and test sets with 37500, 5000, 5000 and 10000 sequences respectively, treating them as before.

Because our current implementation allows for only one type of BOS event, we currently ignore where the novel meme was originally posted, making the unfortunate assumption that the stream of websites is independent of the originating website. Even worse, we must assume that the stream of websites is independent of the actual text of the meme. However, as we will see, our novel models have some ability to recover from these forms of missing data.

On this dataset, the advantage of our full N-SM-MPP model was dramatic, yielding cross-entropy per event of around -8 relative to the -11 of D-SM-MPP—which in turn is far above the -60 of the Hawkes process. Figures 7 and 8 further illustrate the persistent gaps among the models. We attribute the poor performance of the Hawkes process to its failure to capture the latent properties of memes, such as their topic, political stance, or interestingness. This is a form of missing data (section 1.2), as we now discuss.

As Table 1 suggests, most memes in MemeTrack are uninteresting and give rise to only a short sequence of mentions. Thus the base mention probability is low. An ideal model would recognize that if a specific meme has been mentioned several times already, it is *a posteriori* interesting and will probably be mentioned in future as well. The Hawkes process cannot distinguish the interesting memes from the others, except insofar as they appear on more influential websites. However, this pattern can be partly captured in our D-SM-MPP by setting $\boldsymbol{\mu} < 0$ to create "inertia" (section 3.2.1). Indeed, all 5000 of its learned $\boldsymbol{\mu}$ parameters were negative, with values ranging from -10 to -30.

An ideal model would also recognize that if a specific meme has appeared much more often on conservative websites than liberal ones, it is *a posteriori* conservative and is unlikely to appear on liberal websites in future. The D-SM-MPP, unlike the Hawkes process, can similarly capture this to an extent, by having conservative websites inhibit liberal ones. 24% of its learned α parameters were negative. (We re-emphasize that this inhibition is merely a predictive effect, and probably not a direct causal mechanism.)

And our N-SM-MPP process is even more powerful. The LSTM state aims to learn sufficient statistics for predicting the future, so it can learn hidden dimensions (which fall in $(-1, 1)$) that encode useful posterior beliefs in boolean properties of the meme such as interestingness, conservativeness, timeliness, etc. The LSTM's "long short-term memory" architecture explicitly allows these beliefs to persist indefinitely through time in the absence of new evidence, without having to be refreshed by redundant new events as in the decomposable models. Also, the LSTM's hidden dimensions are computed by sigmoidal activation rather than soft-plus activation, and can be used implicitly

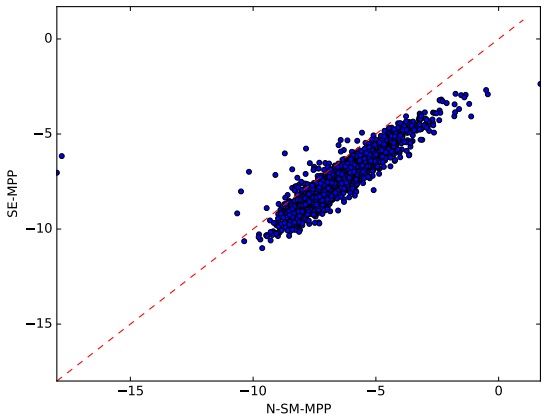
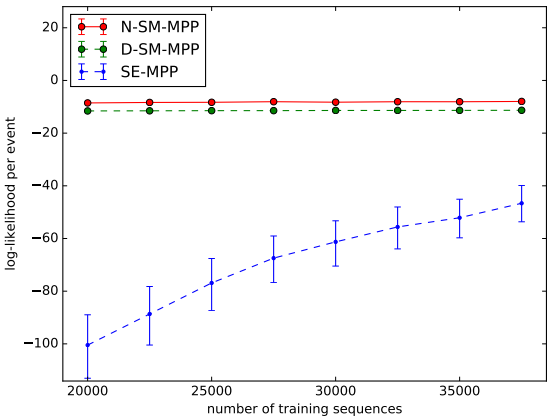


Figure 5. Scatterplot of N-SM-MPP vs. SE-MPP on the Retweets dataset. Same format and results as Figure 4.



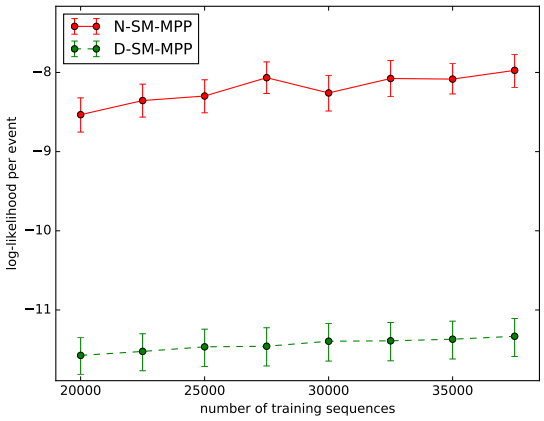


Figure 6. Learning curves (with 95% error bars) of all three models on the MemeTrack dataset. Both self-modulating models significantly outperform the Hawkes process. Zooming in on the upper curves (lower graph) shows that the full neural model significantly outperforms the non-neural one.

to perform logistic regression. The flat left side of the sigmoid resembles soft-plus and can model *inertia* as we saw above: it takes several mentions to establish interestingness. Symmetrically, the flat right side can model *saturation*: once the posterior probability of interestingness is at 80%, it cannot climb much farther no matter how many more mentions are observed.

A final potential advantage of the LSTM is that in this large-K setting, it has fewer parameters than the other models, sharing statistical strength across websites to obtain generalization. Instead of learning K^2 different $\alpha_{j,k}$ parameters, our LSTM learns $O(R^2)$ pairwise interactions among its R hidden nodes (where $R \ll K$), as well as $O(KR)$ interactions between the hidden nodes and the K event types. In this case, $K = 5000$ but $R = 64$.[8] The learning curve in Figure 6 suggests that the Hawkes process, which must explain everything in terms of excitation, may be overfitting its $O(K^2)$ interaction parameters on smaller data: it has trouble finding a setting of these parameters that obtains reasonable generalization.

6.3. Medical Prediction Task

To compare with Du et al. (2016), we evaluate our model on the prediction task and dataset that they proposed. The electrical medical records (MIMIC-II) dataset is a collection of de-identified clinical visit records of Intensive Care Unit patients for seven years. Each patient has a sequence of hospital visits, and each visit event records its time stamp and disease diagnosis. The goal is to predict which major disease will happen to each patient next, and when it will happen.

To avoid tuning on the test data, we split the original training set into a new training set and a held-out dev set. We train both our model and that of Du et al. (2016) on the new training set, and compare them on the held-out dev set. The numbers on the test set will be shown in the final version of the paper. We follow Du et al. (2016) and attempt to predict every held-out event (k_i, t_i) from its history $\mathcal{H}_i$, evaluting the prediction $\hat{k}_i$ with 0-1 loss (yielding an error rate, or ER) and evaluating the prediction $\hat{t}_i$ with L2 loss (yielding a root-mean-squared error, or RMSE). We make minimum Bayes risk predictions as explained in section 4.

[8] The parameter count of our LSTM is actually dominated by the $\mathbf{D}_k$ matrices, which form a tensor with $O(KR^2)$ entries. We could greatly reduce the number of paramteters by learning a (factorized) low-rank version. Even so, it has only half as many parameters as the other models (Table 2).

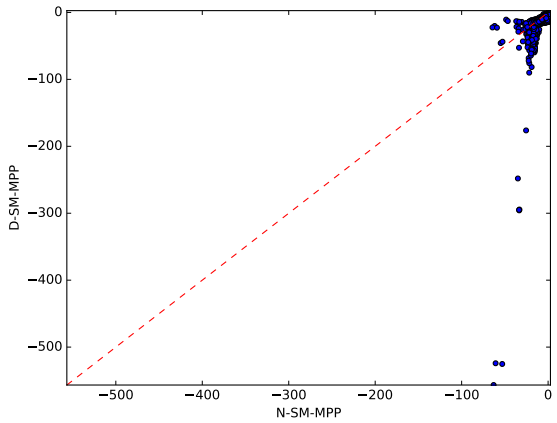


Figure 7. Scatterplot of N-SM-MPP vs. D-SM-MPP on MemeTrack. Same format as Figure 4, but this time showing 5000 devtest sequences. *Warning:* The plot is misleading because nearly the points are stacked up near the upper right corner: recall from Figure 6 that the centroid is near $(-8, -11)$. Most of the visible points are outliers where N-SM-MPP performs unusually badly—and D-SM-MPP typically does even worse.

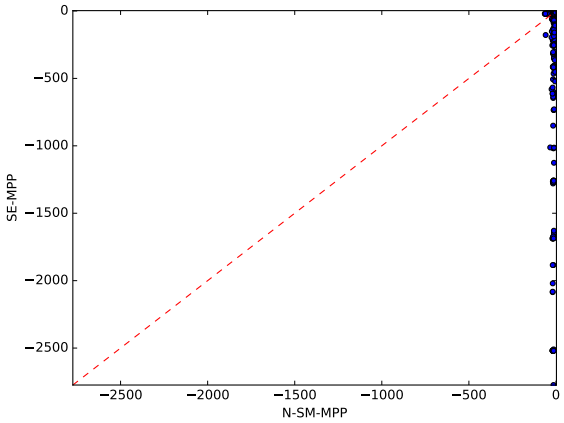


Figure 8. Scatterplot of N-SM-MPP vs. SE-MPP on MemeTrack. Same format as Figure 7.

MODELS	PREDICTION RESULTS	
	RMSE FOR TIME PREDICTION	ER FOR TYPE PREDICTION
DU MODEL	6.12 (5.89, 6.36)	12.11% (10.63%, 13.62%)
OUR MODEL	**5.37** (4.83, 5.93)	**4.97%** (3.72%, 6.22%)

Table 4. Prediction results with 95% confidence interval on MIMIC-II dataset. Our model significantly outperforms the model of Du et al. (2016) on both visit time and disease type prediction.

In Table 4, it is shown that our model significantly outperforms that of Du et al. (2016).

7. Ongoing and Future Work

We are currently exploring several extensions to deal with more complex datasets. Based on our exploration of existing datasets, we are particularly interested in handling:

- "baskets" of events (several events that are recorded as occuring simultaneously, e.g., the purchase of an entire shopping cart)
- events with structured data attached (e.g., meme text, medical records)
- external annotations of the event stream
- scalability to very large event sets
- missing events, or stochastic state transitions that are presumably due to missing events
- delayed causation, periodicity, causation by external events (e.g., clock ticks, holidays)
- hybrid of D-SM-MPP and N-SM-MPP, allowing direct influence from past events
- multiple agents each with their own state, who observe one another's actions (events)

More important, as discussed in section 1, we are interested in modeling causality. The current model might pick up that a hospital visit elevates the instantaneous probability of death, but this does not imply that a hospital visit *causes* death. (In fact, an earlier illness is usually the cause of both.) In a general model like ours that does not make specific assumptions about causality, randomized intervention experiments are required to fully determine the causal structure.

We therefore aim to embed our model within a reinforcement learner, which is able to stochastically insert or suppress certain event types and observe the effect on subsequent events. In this setting, our LSTM-based model will discover the causal effects of its actions, and the reinforcement learner will discover what actions it can take to affect future reward. Ultimately this could be a vehicle for personalized medical decision-making, as well as other decision-making. For example, a smartphone app may intervene by displaying helpful advice (not limited to advertisements), or a social work agency may intervene by visiting a home to provide timely counseling or material support.

8. Conclusion

We have proposed two extensions to the multivariate Hawkes process, a popular generative model of streams of typed, timestamped events. The Hawkes process allows interaction among the events, but it is limited to additive excitation of event types. First, we allow inhibition as well as excitation. To handle the possibility that a event type's total activation may become negative, we employ a simple transfer function to convert it back to a positive intensity. This paves the way for our second change: past events do not influence future events additively, but through their combined sequential influence on the state of a recurrent neural network (LSTM).

We have carefully motivated our extensions as important for improved data modeling. They aim to help deal with real-world phenomena, missing data, and causal modeling. Empirically, we have shown that each change yields a significantly improved ability to predict future events in naturally occurring datasets. There are several exciting avenues for further improvements.

Acknowledgments

We are grateful to Facebook for enabling this work through a gift to the second author. Nan Du kindly helped us by making his code public and answering questions. We also thank our lab group at Johns Hopkins University's Center for Language and Speech Processing for helpful comments.

References

Chelba, Ciprian, Mikolov, Tomas, Schuster, Mike, Ge, Qi, Brants, Thorsten, Koehn, Phillipp, and Robinson, Tony. One billion word benchmark for measuring progress in statistical language modeling. *Computing Research Repository*, arXiv:1312.3005, 2013. URL `http://arxiv.org/abs/1312.3005`.

Choi, Edward, Du, Nan, Chen, Robert, Song, Le, and Sun, Jimeng. Constructing disease network and temporal progression model via context-sensitive Hawkes process. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pp. 721–726. IEEE, 2015.

Du, Nan, Farajtabar, Mehrdad, Ahmed, Amr, Smola, Alexander J, and Song, Le. 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, Nan, Wang, Yichen, He, Niao, Sun, Jimeng, and Song, Le. Time-sensitive recommendation from recurrent user activities. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 3492–3500, 2015b.

Du, Nan, Dai, Hanjun, Trivedi, Rakshit, Upadhyay, Utkarsh, Gomez-Rodriguez, Manuel, and Song, Le. 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, Jalal, Kiyavash, Negar, Zhang, Kun, and Singhal, Kushagra. Learning network of multivariate Hawkes processes: A time series approach. *arXiv preprint arXiv:1603.04319*, 2016.

Gomez Rodriguez, Manuel, Leskovec, Jure, and Schölkopf, Bernhard. Structure and dynamics of information pathways in online media. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, pp. 23–32. ACM, 2013.

Graves, Alex. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012. URL `http://www.cs.toronto.edu/~graves/preprint.pdf`.

Graves, Alex, Jaitly, Navdeep, and Mohamed, Abdelrahman. Hybrid speech recognition with deep bidirectional LSTM. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pp. 273–278. IEEE, 2013.

Guo, Fangjian, Blundell, Charles, Wallach, Hanna, and Heller, Katherine. The Bayesian echo chamber: Modeling social influence via linguistic accommodation. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2015.

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

He, Xinran, Rekatsinas, Theodoros, Foulds, James, Getoor, Lise, and Liu, Yan. Hawkestopic: A joint model for network inference and topic modeling from text-based cascades. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 871–880, 2015.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Karpathy, Andrej, Johnson, Justin, and Fei-Fei, Li. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

Lee, Young, Lim, Kar Wai, and Ong, Cheng Soon. Hawkes processes with stochastic excitations. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

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

Lewis, Peter A and Shedler, Gerald S. Simulation of nonhomogeneous Poisson processes by thinning. *Naval Research Logistics Quarterly*, 26(3):403–413, 1979.

Liniger, Thomas Josef. *Multivariate Hawkes processes*. Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 18403, 2009, 2009.

Lukasik, Michal, Srijith, PK, Vu, Duy, Bontcheva, Kalina, Zubiaga, Arkaitz, and Cohn, Trevor. Hawkes processes for continuous time sequence classification: An application to rumour stance classification in Twitter. In *Proceedings of 54th Annual Meeting of the Association for Computational Linguistics*, pp. 393–398, 2016.

Mikolov, Tomas, Karafiát, Martin, Burget, Lukás, Cernocký, Jan, and Khudanpur, Sanjeev. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pp. 1045–1048, 2010.

Palm, C. *Intensitätsschwankungen im Fernsprechverkehr*. Ericsson technics, no. 44. L. M. Ericcson, 1943. URL `https://books.google.com/books?id=5cy2NQAACAAJ`.

Pearl, Judea. Causal inference in statistics: An overview. *Statistics Surveys*, 3:96–146, 2009.

Sundermeyer, Martin, Ney, Hermann, and Schluter, Ralf. LSTM neural networks for language modeling. *Proceedings of INTERSPEECH*, 2012.

Wang, Yichen, Xie, Bo, Du, Nan, Le Song, EDU, and EDU, GATECH. Isotonic Hawkes processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

Xu, Hongteng, Farajtabar, Mehrdad, and Zha, Hongyuan. Learning Granger causality for Hawkes processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

Yang, Shuang-hong and Zha, Hongyuan. Mixture of mutually exciting processes for viral diffusion. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1–9, 2013.

Zhao, Qingyuan, Erdogdu, Murat A, He, Hera Y, Rajaraman, Anand, and Leskovec, Jure. Seismic: A self-exciting point process model for predicting tweet popularity. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1513–1522. ACM, 2015.

Zhou, Ke, Zha, Hongyuan, and Song, Le. Learning triggering kernels for multi-dimensional Hawkes processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1301–1309, 2013.

A. Likelihood Function

As mentioned in section 4, given an event stream observed over the time interval $[0, T]$, the log-likelihood of the parameters of the proposed models turns out to be given by a simple formula—the log-intensities of the events that happened, at the times they happened, minus an integral of the total intensities over the entire time range $[0, T]$:

$$\sum_{i:t_i<T} \log \lambda_{k_i}(t_i) - \int_{t=0}^{T} \sum_{j=1}^{K} \lambda_j(t) dt \tag{12}$$

In this section, we will derive this formula.

First, we define $N(t) = |\{i : t_i < T\}|$ to be the count of events (of any type) preceding time t. So given the past history $\mathcal{H}_{i+1}$ (as defined in section 2), the number of occurrences for $t > t_i$ is denoted as $\Delta N(t, t_i) = N(t) - N(t_i)$. Let T_{i+1} be the random variable of the next event time and K_{i+1} be the random variable of the next event type. The cumulative distribution function and probability density function of T_{i+1} (conditioned on $\mathcal{H}_{i+1}$) are given by:

$$\begin{aligned}
F(t) &= P(T_{i+1} \leq t) = 1 - P(T_{i+1} > t) && \text{(13a)}\\
&= 1 - P(\Delta N(t, t_i) = 0) && \text{(13b)}\\
&= 1 - \exp\left(- \int_{t_i}^{t} \sum_{j=1}^{K} \lambda_j(s) ds \right) && \text{(13c)}\\
&= 1 - \exp\left(\Lambda(t_i) - \Lambda(t)\right) && \text{(13d)}\\
f(t) &= \exp\left(\Lambda(t_i) - \Lambda(t)\right) \sum_{j=1}^{K} \lambda_j(t) && \text{(13e)}
\end{aligned}$$

where $\Lambda(t) = \int_0^t \sum_{j=1}^{K} \lambda_j(s) ds$.

Moreover, given the past history $\mathcal{H}_{i+1}$ and time stamp t_{i+1} of next occurrence, the distribution of k_{i+1} is given by:

$$P(K_{i+1} = k_{i+1} \mid t_{i+1}) = \frac{\lambda_{k_{i+1}}(t_{i+1})}{\sum_{j=1}^{K} \lambda_j(t_{i+1})} \tag{14}$$

Therefore, we can derive the likelihood function as follows:

$$\mathcal{L} = \prod_{i:t_i<T} \mathcal{L}_i = \prod_{t_i<T} \{f(t_i)P(K_i = k_i \mid t_i)\} \tag{15a}$$

$$= \prod_{i:t_i<T} \{\exp\left(\Lambda(t_{i-1}) - \Lambda(t_i)\right) \lambda_{k_i}(t_i)\} \tag{15b}$$

and

$$\ell = \log \mathcal{L} \tag{16a}$$

$$= \sum_{i:t_i<T} \log \lambda_{k_i}(t_i) - \sum_{i:t_i<T} \left(\Lambda(t_i) - \Lambda(t_{i-1})\right) \tag{16b}$$

$$= \sum_{i:t_i<T} \log \lambda_{k_i}(t_i) - \Lambda(T) \tag{16c}$$

$$= \sum_{i:t_i<T} \log \lambda_{k_i}(t_i) - \int_{t=0}^{T} \sum_{j=1}^{K} \lambda_j(t) dt \tag{16d}$$