ANALYSIS OF *Neural Hawkes Process* IN *Event Based Sequences*

Abhilash Jahagirdar (abja5431)
Department of Computer Science
University of Colorado, Boulder
abja5431@colorado.edu

David Young-Jae Kim (daki7711)
Department of Computer Science
University of Colorado, Boulder
davidyjk@colorado.edu

Karthik Siddaramanna (kasi9796)
Department of Computer Science
University of Colorado, Boulder
kasi9796@colorado.edu

Madhusudhan Aithal (maai8912)
Department of Computer Science
University of Colorado, Boulder
madhuaithal@colorado.edu

October 16, 2020

ABSTRACT

Several events happen during our everyday life. One of our interest is to study the pattern of such events. With a solid understanding of how events happen we can use it in many different application such as medical treatment, purchase prediction, and behavior study. Studying event sequence has a long history starting from the Poisson Process, while being a very strong analytical tool it has a flaw in that it assumes that every event is independent with each other. The Hawkes Process tries to overcome this drawback but still has a problem that events influence each other only in a positive way and their influences are independent with other influences. We suggest that a neural network can solve this problem by giving the machine free parameters to find the inherit causality of events. For the experiment we compared the CT-LSTM[4] with the normal LSTM to see if the inclusion of the time difference directly influencing the hidden layers have any positive effect to predict future events.

***K*eywords** Event Based Sequence · Hawkes Process · Long Short Term Memory (LSTM) · Continuous-Time LSTM (CT-LSTM)

1 Introduction

Recurrent Neural Networks(RNN) are deep learning architectures that are for solving sequence based problems. Sequences are a particular order in which related events, movements, or things follow each other. We can further differentiate these sequence as *ordinal* sequence and *event* sequence. Ordinal sequence are sequences where the time interval between each event is deterministic. Examples for such events are like natural language text as in a progression of words, videos that consist of series of images, and stock market where the value is updated in a determined time. However, event sequence are sequences where time interval between each event is non-deterministic. For instance, online product purchases, criminal activity happening in a certain time period, patterns of people posting in a web forum, and individual restaurant reservations, file accesses, outgoing phone calls or text messages and e-mail, player log-ins to gaming sites, and music selections.[5]

Traditional Recurrent Neural Net(RNN) architectures for solving sequence based problems in deep learning does not consider the difference between sequences in their architecture. In this project we have explore different approaches to make the architecture adopt event based sequence, hoping to find an architecture that handles these sequences better than the traditional RNNs.

For this project we have focused on studying two papers one is "Discrete-Event Continuous-Time Recurrent Nets"[5], and the other is "The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process"[4]. The former paper concentrates on investigating whether adding a new time lapse feature improves analyzing event sequence. The authors propose continuous-time GRU(CT-GRU), a variation of GRU, that takes into account the timescale differences between events. GRU's storage is viewed as consisting of two parts one part is stored indefinitely and the other part is stored for an infinitesimal small amount of time. CT-GRU's storage consists of multiple timescales and can handle the time difference between discrete events in s special manner. GRU performs similar to CT-GRU if the time lag between inputs is also provided as an input. Two models which handle time differently end up performing similarly. The later paper focus on building a new LSTM architecture called the Continuous-Time LSTM(CT-LSTM). This architecture is built based on the Hawkes-process long known to be efficient on analyzing event sequences.

We have begun our experiment with synthetic event sequence processing data sets. This data is constructed by setting some random parameters with the hawkes process model and the neural hawkes process model then generating data based on the configured model. Then we will test how robust the models are by evaluating how much it is good at recovering the events. Another data we used is the retweet data set. This data considers retweets as an event. We will evaluate how robust the model is by giving it a task to predict at what time which tweet will be retweeted.[4]

For this project we will review some of the historical theories and reason why neural network could improve this settings in a logical way. Next we will go over the model architecture that we will use in this experiment. We will then go over the data and their characteristics. Then finally we will present our results and and discuss about the the whole process.

2 Related Theories

For this sections we will introduce some of the historical approaches that tackled event sequence and how each theory developed on top of each other. We will progressively explain why a neural network could overcome some of the shortcomings that these past theories have.

2.1 Poisson point process

The goal of this problem is to learn a probability distribution of a set of events over continuous time space where the intensity of an event occuring is a function of time. The first step in this process is to learn the distribution of a single event in discrete time space.[1]

If X is the random variable denoting number of occurences of an event in each time instance of a discretized time space, under the assumption that all such instances are indistinguishable, the $\mathbb{E}[X] = \lambda = np$ where n is number of instances and p denotes the probability of $X = 1$. The happening of an event can be represented by a tuple of {0,1} where 0 represents the event not happening and 1 represents that the event is happening. A *point process* is a distribution of {0,1}-valued functions over a time interval. From knowledge of binomial distribution and basic combinatorics, probability of observing observing k occurences of an event in a time interval is got by

$$P(\texttt{k occurences}) = \binom{n}{k}\left(\frac{\lambda}{n}\right)^k\left(1-\frac{\lambda}{n}\right)^{n-k} \tag{1}$$

Intuitively, λ represents number of events per time instance which is termed as intensity. The move from discrete time space to continuous time space can be made by taking limit as n tends to infinity. In that case,

$$P(\texttt{k occurences}) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{2}$$

To relax the assumption of having a constant intensity(λ) and to support multiple events, intensity at any time between $(t, t+dt)$ is represented by $\lambda_k(t)dt$. Here k represents event types denoted by subscripts $1, 2, ...k$. The total intensity at a time t is given by $\lambda(t) = \sum_{i=0}^{k} \lambda_i(t)$. These definitions helps us in modelling *multivariate point process* which is a distribution of k-tuples of {0,1}-valued functions over a time interval.

2.2 Hawkes Process: A Self Exciting Multivariate Point Process (SE-MPP)

The Poisson Point Process, although being a solid way estimate event sequence, has a drawback that every single event is independent with each other. While this is fine with many event sequence, such as when we are estimating the number of customers that come in a coffee shop the Poisson Process is enough because the event of a customer coming does not effect the event of another customer coming in. However, there are cases where that is not true. For instance, when a person shows up in a hospital the chances of that person showing up the next increases. Since going to the hospital indicates that one needs treatment. Thus, instead of merely calculating the mean event frequency the Hawkes process adds some more parameters to increase the value of λ for a certain period of time when an event happens and slowly decays as time goes by. Therefore, the following equation

$$\lambda_k(t) = \mu_k + \sum_{h:t_h<t} \alpha_{j,k} \exp(-\delta_{j,k}(t - t_h)) \tag{3}$$

Here the base rate $\mu_k \geq 0$. The influence of the previous event of type j on the event k is represented by $\alpha_{k_h,k} \geq 0$. The above influence decays at the rate of $\delta_{j,k} \geq 0$.

If we model event sequence processing as an SE-MPP. Each event of the past can only lead to excitation of the current event and the events from past cannot inihibit events happening in the future. This influence decays with time to a base rate which is always greater than zero.

It is easy to see the shortcoming of the above model as events in the past can inhibit certain future events. An online purchase of cookies at a certain time reduces the probability of the buyer purchasing other deserts. A positive base rate is also not desirable as the influence of certain events should cross a particular threshold for it's effect to take place. To overcome the above shortcomings the following models were proposed.

Works like [5] solving for recommendations based on user activity have used Hawkes process. Although their equation for intensity is similar to (3), the parameters $\alpha_{i,j}$ for different event pairs and base intensities are learnt through low rank matrix models(similar to PCA) as opposed to [4] which uses neural nets. [3] builds upon traditional Hawkes process by modelling $\alpha_{i,j}$ as a contagion which introduces some indeterminacy, where $\alpha_{i,j}$ is the parameter controlling the influence among events. In traditional Hawkes, this is a constant parameter and although [4] treats this parameter as a variable, it is defined as a deterministic function in hidden states of CT-LSTM. The randomness introduced by the stochastic nature of [3] is believed to increase expressivity of the process. But one thing we have to note here is that $\alpha_{i,j}$ cannot be negative as in (4).

2.3 Isotonic Hawkes Processes

[6] proposes to increase the expressivity by allowing non linear influence of past history on the intensities to support a wide variety of trends. This is achieved by using isotonic non-parametric function **g** in position of f_k defined in [4]. But f_k are parametric and are approximately linear for large values contrary to **g** which comprise of kernels like Gaussian or exponential etc. Such kernels are chosen to ease the computation of integral of the intensity. They also point out that having linear dependency between events may result in unbounded intensities which might not reflect the general phenomena. In [4], this non linearity is incorporated in the design by CT-LSTM instead of $\mathbf{f}_k$.

2.4 Hawkes Process with Inhibition: A Decomposable Self-Modulating MPP (D-SM-MPP)

A simple strategy to overcome the above limitations would be to remove the constraints of base rate $\mu_k \geq 0$ and $\alpha_{k_h,k} \geq 0$ in the below equations.

$$\tilde{\lambda}_k(t) = \mu_k + \sum_{h:t_h<t} \alpha_{k_h,k} \exp(-\delta_{k_h,k}(t - t_h)) \tag{4}$$

Removing the restrictions leads to the intensity being negative, which would not make sense because the probability of something happening has to be positive. To overcome this, a transfer function f_k which maps numbers on real line to positive real numbers is used.

$$\lambda_k(t) = f_k(\tilde{\lambda}_k(t)) \tag{5}$$

The features that are required for this function is that first it needs to be strictly positive. Even zero is undesirable because an assumption that an event will never happen is not compatible with the base requirement. Also, we would want a function that preserves the value of $\tilde{\lambda}$ when it is positive. The first function that comes in mind is the Relu function ($= max(x, 0)$). However, the Relu function does not obey the first restriction since when $\tilde{\lambda}$ is negative the value will be zero. Thus, we use a slight variation of the Relu function that satisfies both restriction

The transfer function used is

$$f(x) = s \log(1 + \exp(x/s)) \tag{6}$$

which approaches ReLU as $s \to 0$.

3 Neural Hawkes Process (CT LSTM)

3.1 Neural Hawkes Process: A Neurally Self-Modulating MPP (N-SM-MPP)

In Neural Hawkes process recurrent neural networks hidden state is used to derive the intensity value of different events. This leads to a richer representation to capture the dependence of events and predict future events. The above model is referred to as Continous-Time LSTM (CT-LSTM). The key difference compared to a normal LSTM lies in the fact that the value of hidden state decays in the continuous time interval between two events. The intensity of different events is obtained from the hidden state by training a set of parameters w_k for each event type k. The following equations show how event intensities are calculated from the hidden state and how hidden state gets calculated from the decaying continuous memory $c(t)$.

$$\lambda_k(t) = f_k(w_k h(t)) \tag{7}$$

$$h(t) = o_i \odot (2\sigma(2c(t)) - 1) \text{ for } t \in (t_{i-1}, t_i] \tag{8}$$

The update equations of a CT-LSTM look very similar to that of LSTM, instead of using the value of previous hidden state directly to perform the updates, in CT-LSTM the updates are performed based on the hidden state calculated at that time t_i. The value of the hidden state $h(t_i)$ is calculated from memory cell $c(t_i)$. The value of the memory cell decays at a specific rate. More on the memory cell decay later.

$$i_{i+1} \leftarrow \sigma\left(W_{\mathrm{i}} k_i + u_{\mathrm{i}} h(t_i) + d_{\mathrm{i}}\right) \tag{9}$$

$$f_{i+1} \leftarrow \sigma\left(W_{\mathrm{f}} k_i + u_{\mathrm{f}} h(t_i) + d_{\mathrm{f}}\right) \tag{10}$$

$$z_{i+1} \leftarrow 2\sigma\left(W_{\mathrm{z}} k_i + u_{\mathrm{z}} h(t_i) + d_{\mathrm{z}}\right) - 1 \tag{11}$$

$$o_{i+1} \leftarrow \sigma\left(W_{\mathrm{o}} k_i + u_{\mathrm{o}} h(t_i) + d_{\mathrm{o}}\right) \tag{12}$$

$$c_{i+1} \leftarrow f_{i+1} \odot c(t_i) + i_{i+1} \odot z_{i+1} \tag{13}$$

$$\bar{c}_{i+1} \leftarrow \bar{f}_{i+1} \odot \bar{c}_i + \bar{i}_{i+1} \odot z_{i+1} \tag{14}$$

$$\delta_{i+1} \leftarrow f\left(W_{\mathrm{d}} k_i + U_{\mathrm{d}} h(t_i) + d_{\mathrm{d}}\right) \tag{15}$$

The $i^t h$ input is vector $k_i \in 0, 1^K$. [4] calculates the base rate to decay after the event at time t_i. [4] calculates a decay rate parameter for the current cell state to reach the base rate. The cell decays from the current state c_{i+1} to a base rate base rate $\bar{c}_{i+1}$ based on the following equation.

$$c(t) = \bar{c}_{i+1} + \left(c_{i+1} - \bar{c}_{i+1}\right) \exp\left(-\delta_{i+1}\left(t - t_i\right)\right) \text{ for } t \in (t_i, t_{i+1}] \tag{16}$$

Thus CT-LSTM exploits the complex dynamics of recurrent network's hidden state to model discrete events in continuous time.

4 LSTM model for event sequence processing

LSTMs are not designed to handle event sequences happening at discrete times.[2] LSTM models treat the time difference between the occurrence of each event to be the same, The three ways to handle time information in LSTM are as follows.

- Ignore the timestamp values.
- Divide time into fixed slots and sample events from these slots.
- Provide time lag between different events as input along with actual event.

The first approach simply ignores the time of occurrence of an event, treating that all the events occur at equal time intervals. In the second approach there will be issues with choosing the size of the fixed slots. We experimented using the third approach.

The input for the LSTM consisted of a continuous time value representing the time lag between the current event and the past event along with the current event type. The motive is that the additional input should provide enough flexibility for the recurrent model to account for the different time lag between events.

The model has to predict both the event and time of occurrence. The model thus will have two losses associated with each prediction. A categorical loss for the event type prediction and a mean squared loss for predicting the continuous time. We optimized the parameters to reduce the sum of the above losses using Adam optimizer.

We did hyper-parameter tuning of the model by varying the size of the hidden layer of the LSTM, batch size and learning rate. More details are in the experimental section.

You can find our code implementation in the following github link : github link

5 Datasets

5.1 Hawkes Data

Hawkes data is a synthetic dataset that we are using in our experiments. Hawkes data consists of 8000 training samples, 10000 validation samples and 10000 samples as test set. Each sample in the dataset will have sequence of events. Here, event type and the time at which the event happens correspond to the event information present in each sequence. The time information of an event includes,

1. time since start event
2. time since last event
3. time since last event of same type

For our experiments, we have used *time since last event* as the input for LSTMs. There are 5 types of events in total. For each sample, the sequence length of the number of events in the sequence is randomly sampled from the set $\{20, 21, 22, ..., 100\}$. The parameters for generating the Hawkes process data are randomly sampled from Uniform distributions. μ_k and $\alpha_{j,k}$ are randomly sampled from the uniform distribution $[0.0, 1.0]$ and $\delta_{j,k}$ is randomly sampled from the uniform distribution $[10.0, 20.0]$. The event type and the corresponding event time are obtained using the thinning algorithm explained in [4]. Given these events in a sequence, the models should predict the next event type and the time at which happens.

5.2 Retweet Data

The Retweet data contains samples where each one represents the sequence of times at which the tweet was retweeted. An event type in this case is the retweet of the original tweet by a user. The original tweet in a sequence corresponds to beginning-of-the-sequence (BOS) marker. All the subsequent retweets are considered as the events of our interest. There are a total of 3 event types. These 3 events are categorized as "small", "medium", "large", based on the follower count of the retweeter. If a tweet is retweeted by a user with followers less than 120, then the event is categorized as "small". If the follower count of the retweeter is fewer than 1363, then the event corresponds to "medium" event. The rest are "large" events. The original dataset consists of 166076 retweets sequences. We have randomly sampled 20000 sequences from it for our experiments. The training set, validation set, and test set consists of 16000 sequences, 2000 sequences, and 2000 sequences respectively. The sequences are truncated to a maximum length of 264.

This dataset captures the stochastic temporal patterns, where each event can either inhibit or excite other events. We chose this dataset because it can be used to compare the performance of the RNNs using LSTM and CT-LSTM units, in an efficient manner.

6 Experiment and Results

Model	Total average NLL Loss	NLL Loss of event type	NLL Loss of event-time
LSTM	1.82	1.41	0.41
CT-LSTM (Neural Hawkes model)	1.3722	1.4735	-0.1013

Table 1: Comparison of LSTM and Neural Hawkes model on test set of Hawkes Synthetic data

Model	Total average NLL Loss	NLL Loss of event type	NLL Loss of event-time
LSTM	5,860.86	0.86	5,860.00
CT-LSTM (Neural Hawkes model)	6.1236	0.7779	5.3457

Table 2: Comparison of LSTM and Neural Hawkes model on test set of Retweet data

We did hyper parameter tuning where the learning rate was tuned from the set $[10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}]$. The hidden size and batch size were varied from 32 to 512 in the powers of 2.

6.1 Hawkes data

In case of Hawkes data the parameters in table 3 turned out to be the best, although there was not much significant difference across different runs. The performance of both the models are comparable, one interesting thing to note is that the loss with respect to predicting the time of occurrence is the main cause of higher loss in LSTM.

Learning rate	0.001
Hidden size	128
Batch size	256

Table 3: Best Hyperparameters for Hawkes synthetic data

6.2 Retweet data

In case of retweet data the parameters in table 4 turned out to be the best. The performance of the LSTM model here is significantly worse. One interesting thing to note is that the loss associated with event type prediction is comparable to that of CT-LSTM, but the time of event occurrence in an LSTM model is not being learnt indicating the superiority of CT-LSTM over LSTM in event sequence processing.

Learning rate	0.001
Hidden size	256
Batch size	32

Table 4: Best Hyperparameters for Retweet data

LSTM performs poorly compared to CT-LSTM in predicting the time of occurrence of an event. Our intuition behind this is that it is hard for LSTM to understand the representation of time difference supplied as input, and hence it cannot efficiently model the temporal patterns associated with the event sequences.

7 Discussion

To summarize our results, there wasn't much difference in the two architectures when we experimented with the hawkes synthesized data But when it comes to retweet or any real data, the two architectures had some different results. Based on our results we can conclude that the CT-LSTM did improve performance compared to the normal LSTM. That being said we will have to take these results in a careful manner because there could be some trivial or pathological errors for these results, such as implementation issues. Doing more thorough investigation would be crucial since other papers[5] suggests that there were not much difference. However, if our results turn out to be solid there are some possible explanation. One possible explanation is that hawkes data is consist of patterns that are relatively consistence throughout the whole sequence. But when it comes to retweet or any real data, the pattern can change drastically based on the time. For example, when a new tweet comes out the number of retweets would much more dense compared to when the tweets gets old and people loose interest in that specific tweet. So in this case, the CT-LSTM actually had some advantage over the regular LSTM in capturing this difference. Another explanation that could be possible is that the CT-LSTM has more free parameters than the normal LSTM, along with the fact that the two architectures have different behaviors in encoding and utilizing time.

As for future work we could do investigate event sequences that have certain traits, such as having two event that occur on the same time, $t_{i-1} = t_i$. Or we can think of scenarios where a bunch of events occur at the same time with no distinguished time difference between them. This could be the case for instance, when we are trying to analyze people's purchase history. When that is the case people could buy multiple items and their receipt will print out all of the items that is purchased in a single time slot. In mathematical terms $(E_1, t_i), (E_2, t_i)$.

Also, as of now we are just considering the relationship between the events within the sequence. However, another interesting topic is could outside events also effect the sequence or vise versa? This could be useful to analyze stock market patterns, where outside events actually influence the market price. Intuitively we can think that the CT-LSTM will be good for analyzing these cases, however, we need to be cautious of blindly using this model in causality inference. This is because, with the current model it could pick up conclusion that happen in the same time frame but does not really have a cause and effect relation, such as when ice-cream sales go up more people tend to get bitten by sharks. In this case, these two events are not causing any one of each other, but since in summer people eat more ice-cream and people go to the beach more often and shark accidents happen more often they seem to influence each other. However, our current model does not have any mechanism to filter these error out. Normally, it is known that one cannot determine the causality based on purely observational data. One proposal to overcome this fault is the incorporate reinforcement learning [4][7]. The agent will do random actions to verify if a certain intervention will cause the event pattern to change.

8 Acknowledgement

We thank for Prof. Michael C. Mozer for giving inspiration for this project.

References

[1] S.N. Chiu, D. Stoyan, W.S. Kendall, and J. Mecke. *Stochastic Geometry and Its Applications*. Wiley Series in Probability and Statistics. Wiley, 2013.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[3] Young Lee, Kar Wai Lim, and Cheng Soon Ong. Hawkes processes with stochastic excitations. *CoRR*, abs/1609.06831, 2016.

[4] Hongyuan Mei and Jason Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. In *NIPS*, 2017.

[5] Michael C. Mozer, Denis Kazakov, and Robert V. Lindsey. Discrete event, continuous time rnns. *CoRR*, abs/1710.04110, 2017.

[6] Yichen Wang, Bo Xie, Nan Du, and Le Song. Isotonic hawkes processes. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2226–2234, New York, New York, USA, 20–22 Jun 2016. PMLR.

[7] Shengyu Zhu, Ignavier Ng, and Zhitang Chen. Causal discovery with reinforcement learning, 2019.