# Order-Dependent Event Models for Agent Interactions

**Debarun Bhattacharjya** , **Tian Gao** and **Dharmashankar Subramanian**
Research AI, IBM T. J. Watson Research Center*
$\{$debarunb, tgao, dharmash$\}$@us.ibm.com

## Abstract

In multivariate event data, the instantaneous rate of an event's occurrence may be sensitive to the temporal sequence in which other influencing events have occurred in the history. For example, an agent's actions are typically driven by its own preceding actions as well as those of other relevant agents in some order. We introduce a novel statistical/causal model for capturing such an order-sensitive historical dependence, where an event's arrival rate is determined by the order in which its underlying causal events have occurred in the recent past. We propose an algorithm to discover these causal events and learn the most influential orders using time-stamped event occurrence data. We show that the proposed model fits various event datasets involving single as well as multiple agents better than baseline models. We also illustrate potentially useful insights from our proposed model for an analyst during the discovery process through analysis on a real-world political event dataset.

## 1 Introduction

There has been an explosion of datasets in recent years involving *events* of various types occurring irregularly over the timeline. Many of these involve the actions of single or multiple agents, potentially along with other pertinent observations; examples include electronic health records and wearable device data, socio-political event data, financial data around trades by automated agents, and user behavior in online retail and entertainment. Such datasets enable statistical approaches for learning about agent actions/interactions [Remondino and Correndo, 2005; Grover *et al.*, 2018].

In this paper, we treat agent actions as event occurrences and deploy machine learning techniques to capture the statistical/causal relationships between various types of events. Our model explicitly aims to capture the effect of the *order* in which preceding events have occurred. Specifically, an event's arrival rate is assumed to be determined by the recent historical order in which its underlying causal events have occurred. Our work fits within the high-level framework of *graphical event models* [Didelez, 2008; Meek, 2014; Gunawardana and Meek, 2016], which are continuous-time graphical representations of marked point processes [Cox and Lewis, 1972; Aalen *et al.*, 2008].

Although the proposed model is fairly general and widely applicable, our emphasis on order-dependence is motivated by real-world situations pertaining to agent interactions. As an illustration, consider two countries X and Y who have historically been in conflict. In politics, an escalating sequence of actions is often more likely to result in extreme actions such as declaration of war. For instance, if X first makes a negative statement about Y and then Y threatens X, it may be more likely for X to retaliate strongly and declare war on Y than if the reverse order of actions had occurred.

Explicitly recognizing the order of preceding events may also be important for modeling the behavior of individual agents. For instance, the sequence of a big loss followed by a big win may induce different behaviors in a gambler compared to the reverse sequence, or for that matter compared to the situation where they only face either a big loss or a big win. Modeling and learning about the influence of causal orders from event data could provide an analyst with an enhanced understanding of the underlying process.

**Contributions.** Our primary contributions are: (1) the formulation of a novel order-dependent event model that explicitly distinguishes the causal impact of different orders in an event dataset. As far as we are aware, this is the first model to simultaneously take a marked point process view of an event dataset and consider preceding causal event orders; (2) an efficient algorithm for learning the proposed model from an event dataset; (3) an experimental comparison with relevant baselines on event datasets involving both single and multiple agents; and (4) investigative analysis on a political event dataset extract that illustrates the benefits of explicitly identifying order-dependence during the discovery process.

## 2 Model Formulation

We first introduce some basic notation and provide relevant background before describing details of our proposed model.

### 2.1 Notation & Background

An event dataset (or event stream) is a sequence of time-stamped events of the form $D = \{(l_i, t_i)\}_{i=1}^{N}$, where $t_i$ is

[*] DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited

the occurrence time of the $i^{th}$ event, $t_i \in \mathbb{R}^+$, assumed temporally ordered between start time $t_0 = 0$ and final time $t_{N+1} = T$, and $l_i$ is an event label/type belonging to an alphabet $\mathcal{L}$ with cardinality $M = |\mathcal{L}|$. For simplicity, all equations assume a single event stream but they can be easily extended to multiple independent event streams.

**Example 1.** The event dataset in Fig. 1(a) will be a running example to illustrate the concepts. There are $N = 13$ events over event label set $\mathcal{L} = \{A, B, C\}$ with cardinality $M = 3$ over a period of around a month ($T = 30$ days). □

A mathematically principled way to model multivariate event streams is through a *marked point process*. This captures the dynamics of events occurring in continuous time using *conditional intensity functions*, which are time-varying quantities that measure the rate at which an event label occurs. In general, the conditional intensity for event label $X$ at time $t$ can be written as a function of the *history* at that time, $h_t$, i.e. it is denoted $\lambda_x(t|h_t)$ where $h_t = \{(l_i, t_i) : t_i < t\}$ represents all the preceding events at time $t$.

Graphical event models (GEMs) [Didelez, 2008; Gunawardana and Meek, 2016] provide a framework for how various event labels are generated over time, given the historical occurrences of their parents in some underlying graph. They are graphical representations of a marked point process over event labels, analogous to how Bayesian networks are graphical representations of joint distributions over random variables [Pearl, 2014]. A GEM includes a directed graph $\mathcal{G} = (\mathcal{L}, \mathcal{E})$, which has nodes for every event label $\mathcal{L}$ and directed edges $\mathcal{E}$ represented as ordered pairs from $\mathcal{L} \times \mathcal{L}$. The conditional intensity for an arbitrary label $X$ at any time $t$ depends only on historical occurrences of its parent event labels, implying that $\lambda_x(t|h_t) = \lambda_x(t|[h(\mathbf{U})]_t)$, where $\mathbf{U}$ are $X$'s parents and $[h(\mathbf{U})]_t$ is the history restricted to labels in set $\mathbf{U}$, $[h(\mathbf{U})]_t = \{(l_i, t_i) : t_i < t, l_i \in \mathbf{U}\}$.

It is important to reiterate that a GEM is merely a high-level framework – more information about the historical dependence of conditional intensity rates needs to be provided before the model can even be fully specified and subsequently learned. In this work we propose a specific model where the order in the history plays a critical role.

## 2.2 An Order-dependent Event Model

We are interested in a model where the historical order of the occurrences of a node's parent event labels in a GEM could potentially affect the rate at which it occurs at any time. Since the same event label could occur several times in the history in an event dataset, this could lead to an infinite number of distinct historical possibilities. We therefore introduce a masking function to disregard specific instances of events that repeat, only retaining distinct event occurrences.

**Definition 1.** *A* ***masking function*** $\phi(\cdot)$ *takes a sequence of event tuples as input, and returns a sub-sequence where a label is never repeated. Formally,* $\phi(\cdot)$ *takes as input some temporally ordered sequence* $s = \{(l_j, t_j)\}$ *and returns* $s' = \{(l_k, t_k) \in s : l_k \neq l_m \, for \, k \neq m\}$. *The event label order resulting from applying this masking function is obtained from ordering the labels in* $s'$ *in time, i.e.* $\{l_k : (l_k, t_k) \in s', t_k < t_m \forall k < m\}$.

Here we only consider two cases of tuple masking function $\phi(\cdot)$ due to their simplicity and potential applicability across domains: the 'first' and 'last' cases, depending on whether only the first or last occurrence of an event label in a sequence is retained to determine order. We imagine that a case's suitability depends on the application under consideration.

**Example 2** (cont.). Consider label $C$'s occurrence in Fig. 1(a) at time $t = 10$. If the 'first' masking function is applied to the history at this time, the resulting historical order is $B, C, A$, whereas 'last' results in order $B, A, C$. □

**Definition 2.** *An* ***order instantiation*** *for a set of labels* $\mathbf{Z}$ *is a permutation of a subset of* $\mathbf{Z}$. *The order instantiation at time* $t$ *in an event dataset* $\mathcal{D}$ *over a preceding time window* $w$ *can be determined by applying masking function* $\phi(\cdot)$ *to events restricted to labels* $\mathbf{Z}$ *occurring within* $[\max(t - w, 0), t)$.

**Example 3** (cont.). Suppose $C$ has parents $A$ and $B$, like in Fig. 1(b). Fig. 1(a) shows the order instantiations at each of the five occurrences of event label $C$ over its parent labels for a window of 5 days – there are two occurrences of the order $A, B$, two of order $B$ and one of $B, A$. In this particular situation, both the 'first' and 'last' masking function cases result in identical order instantiations. □

We can now formalize the proposed model:

**Definition 3.** *An* ***ordinal graphical event model*** *(OGEM) for event label set* $\mathcal{L}$ *includes:*

- *A graph* $\mathcal{G}$ *where there is a node for every event label.*
- *Windows for every node in* $\mathcal{G}$, $\mathcal{W} = \{w_X : X \in \mathcal{L}\}$.
- *A set of conditional intensity rate parameters* $\Lambda$, *one for every node and possible order instantiation with respect to the node's parents,* $\Lambda = \{\Lambda_X : X \in \mathcal{L}\} = \{\lambda_{x|\mathbf{o}} : X \in \mathcal{L}, \forall \mathbf{o}\}$. *Here* $\mathbf{o}$ *denotes an order instantiation, which is a permutation of a subset of* $X$*'s parents* $\mathbf{U}$ *– there are* $\sum_{i=0}^{|\mathbf{U}|} \frac{|\mathbf{U}|!}{i!}$ *possible orders.*

**Example 4** (cont.). Fig. 1(b) depicts an example OGEM over $\mathcal{L} = \{A, B, C\}$. Note that the graph can be cyclic and even have self-loops indicating self-dependence. The conditional intensity parameters are also shown; for instance, there are 5 parameters for $C$ – one for every order instantiation of its parents $\{A, B\}$. Parameter $\lambda_{C|A}$ is the rate at which event label $C$ occurs given that *only* $A$ (among its parents) has occurred in the recent preceding window $w_C$, whereas $\lambda_{C|A,B}$ is the rate when the recent history involves an occurrence of A followed by B. While learning from data, the order is determined by the masking function $\phi(\cdot)$. □

The closest model to an OGEM is the proximal GEM (PGEM) [Bhattacharjya *et al.*, 2018], where an event label's conditional intensity rate depends only on whether or not its parents have occurred in some recent time window. We next formalize that a PGEM is unable to distinguish conditional intensity rates from different parental orders.

**Theorem 4.** *Suppose event label* $X$ *with parents* $\mathbf{U}$ *is generated from order-dependent conditional intensity rates* $\{\lambda_{x|\mathbf{o}} : \forall \mathbf{o}\}$, *where* $\mathbf{o}$ *is an order instantiation of* $\mathbf{U}$. *For two orders* $\mathbf{o}'$ *and* $\mathbf{o}''$ *over the same subset of* $\mathbf{U}$ *s.t.* $\lambda_{x|\mathbf{o}'} \neq \lambda_{x|\mathbf{o}''}$, *a PGEM is unable to distinguish between these rates.*

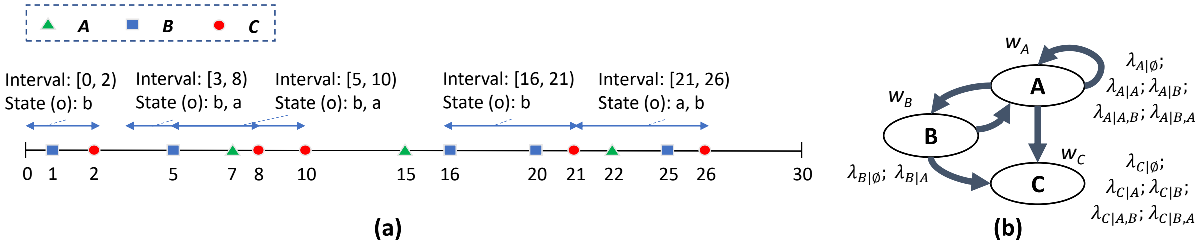

Figure 1: (a) Example event stream with $N = 13$ events of $M = 3$ types of events occurring over $T = 30$ days. The figure also indicates the order instantiations for each occurrence of event label $C$, assuming that $C$ has parents $A$ and $B$ and for a window $w_C = 5$. (b) Illustrative ordinal graphical event model with 3 nodes (event labels), each with a window and a set of conditional intensity parameters.

*Proof.* Suppose orders $\mathbf{o}'$ and $\mathbf{o}''$ are over variables $\mathbf{K} \subseteq \mathbf{U}$ variables. While learning from a dataset, both orders map to the binary parental instantiation $\mathbf{u}$ of $\mathbf{U}$ where variables in $\mathbf{K}$ and $\mathbf{U} \setminus \mathbf{K}$ are 1 and 0 respectively. Thus, both orders contribute to the estimate for $\hat{\lambda}_{x|\mathbf{u}}$ in the PGEM, which is unable to identify the true parameters $\lambda_{x|\mathbf{o}'}$ and $\lambda_{x|\mathbf{o}''}$. □

An OGEM is intended to explicitly capture the effect of the order of an event label's causes, unlike prior literature like PGEM that considers order-neutral event models. For instance, orders that are particularly influential in causing an event would have relatively high conditional intensity parameters. Understanding these influences could be beneficial for analysts during the process of discovery.

## 3 Learning

We present an approach for learning an OGEM from an event dataset $\mathcal{D}$. We treat the windows $\mathcal{W}$ as hyper-parameters, and focus on learning the graph $\mathcal{G}$ and conditional intensity parameters $\Lambda$. Like any other GEM, the OGEM graph is potentially cyclic, therefore the parents and parameters for each node/event label can be learned individually. We first show how to learn conditional intensities $\{\lambda_{x|\mathbf{o}} : \forall \mathbf{o}\}$ for a node $X$ given its parents $\mathbf{U}$ (in $\mathcal{G}$), which relies on computing ordinal summary statistics (Algorithm 1), and then briefly summarize a heuristic graph search method to learn $X$'s parents $\mathbf{U}$.

### 3.1 Learning Parameters

An OGEM is a particular kind of GEM where the conditional intensity rates are piece-wise constant over time, with rate changes occurring whenever there is a change in the order instantiation in history. The log likelihood of any particular event label $X$ for an event dataset $\mathcal{D}$ can therefore be computed using *summary statistics* of counts and durations in $\mathcal{D}$, as well as the model's conditional intensity rates:

$$\log \mathrm{L}_X(\mathcal{D}) = \sum_{\mathbf{o}} \left(-\lambda_{x|\mathbf{o}} D(\mathbf{o}) + N(x;\mathbf{o}) \ln(\lambda_{x|\mathbf{o}})\right), \quad (1)$$

where $N(x;\mathbf{o})$ refers to the number of times $X$ is observed in the dataset and that the order instantiation $\mathbf{o}$ is true in the relevant preceding window $w_X$, and $D(\mathbf{o})$ is the duration over the entire time period where the condition $\mathbf{o}$ is true. The fact that the counts and durations depend on the window $w_X$ is hidden in the notation for the sake of simplicity. From equation (1), the maximum likelihood estimates for conditional intensity parameters are $\hat{\lambda}_{x|\mathbf{o}} = \frac{N(x;\mathbf{o})}{D(\mathbf{o})}$. Thus, if the parents of a node are known, it is straightforward to compute the conditional intensity rates using the summary statistics.

**Algorithm 1** Ordinal Summary Statistics

```
 1: procedure SUMMARYSTATS(event label $X$, parents $\mathbf{U}$, window $w_X$, masking function $\phi(\cdot)$, dataset $\mathcal{D}$)
 2:     Active history $h \leftarrow \emptyset$
 3:     $N(x;\mathbf{o}) \leftarrow 0$, $D(\mathbf{o}) \leftarrow 0$, $\forall \mathbf{o}$
 4:     $D(\emptyset)$ += $t_1 - t_0$                        ▷ Increment empty set dur.
 5:     for $(l_i, t_i) \in \mathcal{D}$ do                   ▷ Scan all events in dataset
 6:         if $l_i \in \mathbf{U}$ then
 7:             Append $(l_i, t_i)$ to $h$
 8:             $\mathbf{o}$ = UpdateOrder($h$,$\phi(\cdot)$ )
 9:         if $l_i$ == $X$ then
10:             $N(x;\mathbf{o})$ += 1                         ▷ Increment count
11:         Set current time $t_c = t_i$
12:         for $(l_j, t_j) \in h$ do              ▷ Scan events in active history
13:             Set inactive time $t^* = t_j + w_X$
14:             if $t^* \geq t_{i+1}$ then
15:                 Break
16:             else                          ▷ Stay until some history is active
17:                 $D(\mathbf{o})$ += $t^* - t_c$                ▷ Increment duration
18:                 Remove this event $(l_j, t_j)$ from $h$
19:                 $\mathbf{o}$ = UpdateOrder($h$,$\phi(\cdot)$ )
20:                 Set current time $t_c = t*$
21:         $D(\mathbf{o})$ += $t_{i+1} - t_c$                    ▷ Increment duration
22:     Return counts $N(x;\mathbf{o})$ and durations $D(\mathbf{o})$, $\forall \mathbf{o}$
```

In Algo. 1, we outline how to scan the entire dataset to compute the required counts $N(x;\mathbf{o})$ and durations $D(\mathbf{o})$ for an event label $X$, given its parents $\mathbf{U}$, window $w_X$ and a masking function $\phi(\cdot)$. Computing counts is relatively easy if the order instantiation at the current time is known – whenever the label under consideration $X$ is encountered, the relevant count is incremented by one (lines 9-10).

Computing durations is more involved and requires maintaining an active history $h$. When a parent label is encountered, the corresponding event is appended to $h$ (lines 6-7). Since the order instantiation could potentially change several times between event occurrences, the entire duration between these epochs needs to be appropriately partitioned across order instantiations. These changes are identified by scanning $h$ and determining when a historical event becomes inactive before the next event occurrence (loop in lines 12-20). A subroutine 'UpdateOrder' applies the masking function to the active history and returns an order whenever the active history is modified (lines 8 and 19).

**Example 5** (cont.). Algo. 1 was run on the event dataset in Figure 1(a) to obtain counts and durations for event label $C$, with parents $\{A, B\}$ and $w_C = 5$ days. The maximum likelihood estimates of intensity rates are: $\hat{\lambda}_{C|\emptyset} = 0$, $\hat{\lambda}_{C|A} = 0$, $\hat{\lambda}_{C|B} = 0.18$, $\hat{\lambda}_{C|A,B} = 0.17$, $\hat{\lambda}_{C|B,A} = 0.33$. Similar to earlier, here the numbers are identical regardless of whether the 'first' or 'last' masking function is used. In this particular example, the rate at which $C$ happens almost doubles when $B$ happens before $A$ as compared to the reverse order. □

It may be possible for some order instantiations to never be observed in the data, resulting in counts (and therefore estimates for conditional intensities) of zero. This issue can be severe when the number of parents is large, since the number of OGEM parameters increases super-exponentially in the number of parents. As we describe in the next sub-section, our parent search approach restricts model complexity, forcing the learner to choose a small number of parents for a small dataset, making it more likely to have sufficient support in the data. For our experiments, we deal with this issue by setting the conditional intensity rate to some small *default rate*, denoted $\lambda^0$, whenever an order instantiation is not observed in the train set. This is treated as a model hyper-parameter.

### 3.2 Learning Parents

We use a score-and-search approach to find the parents of each node and therefore the underlying graph $\mathcal{G}$. A score is used to incorporate model complexity along with the log likelihood on a dataset. For instance, the Bayesian information criterion (BIC) score for an event label $X$ with parents $\mathbf{U}$ is:

$$S_X(\mathbf{U}; \mathcal{D}) = \log\mathrm{L}_X(\mathcal{D}) - \gamma \frac{|\Lambda_X|}{2} \log(T), \qquad (2)$$

where $\log\mathrm{L}_X(\mathcal{D})$ is the log likelihood for $X$ from equation (1) computed at the maximum likelihood estimates for rates, $|\Lambda_X|$ is the number of free parameters (conditional intensity rates) for $X$ in the model and $\gamma$ is a penalty weight on the complexity (second) term. Unless otherwise specified, $\gamma$ is set to 1. The overall score of a graph $\mathcal{G}$ is $S(\mathcal{G}) = \sum_X S_X(\mathbf{U}; \mathcal{D})$ since the scores are decomposable.

For our experiments, we use a forward and backward search procedure to iteratively find the best parental set $\mathbf{U}$ for each event label $X$. Specifically, we iteratively add one candidate event label $Z$ to $\mathbf{U}$ and test if it results in a better score $S_X(\mathbf{U} \cup Z)$ than the current best score. If so, we update $\mathbf{U}$ and query the next $Z$. After finishing adding as many nodes as beneficial for the score, we then iteratively test if removing an event label $Z$ from $\mathbf{U}$ would improve the score, updating $\mathbf{U}$ if it does indeed result in a better score. Such a greedy procedure is popular for learning probabilistic graphical models in general due to its efficiency and consistency, i.e. ability to recover the true graph with asymptotic data.

**Theorem 5.** *A forward backward score-based learning algorithm for OGEM graph $\mathcal{G}$ and parameters $\Lambda$ given hyperparameters $\mathcal{W}$ with summary statistics computed using Algo. 1 with either the 'first' or 'last' masking function has worst case time complexity $O(M^3 N)$, where $M$ and $N$ are the number of event labels and events respectively.*

*Proof.* For a single node, Algo. 1 runs in $O(N)$ time, assuming the 'UpdateOrder' subroutine is $O(1)$; this is possible for both the masking function cases considered. The worst case in the forward (backward) search is that all nodes will be added (removed), which is $O(M^2)$. This is repeated for all $M$ nodes to complete the entire graph and model. □

**Theorem 6.** *Let $\mathcal{G}'$ be the learned graph from a forward backward score-based structure learning algorithm for OGEM graph $\mathcal{G}$. Under the no detailed balance assumption [Gunawardana and Meek, 2016], with sufficient data, $P(\mathcal{G}' = \mathcal{G}) \rightarrow 1$ as $T \rightarrow \infty$.*

*Proof.* OGEMs fall within the piece-wise constant intensity model (PCIM) class of GEMs; we refer the reader to prior work [Gunawardana and Meek, 2016]. □

## 4 Experiments

We demonstrate the efficacy of OGEMs using the following select datasets involving single and multiple agents.

### 4.1 Datasets

**ICEWS** [O'Brien, 2010]. Socio-political events such as in the Integrated Crisis and Early Warning System (ICEWS) political event dataset are an important real-world example of numerous, asynchronous agent interaction events on a timeline. ICEWS involves dyadic events where a source actor performs an action on a target actor, for instance 'Police (Brazil) Assault Protester (Brazil).' Actors and actions are coded according to the Conflict and Mediation Event Observations (CAMEO) ontology, which was created for interactions among domestic and international actors [Gerner *et al.*, 2002]. For our first experiment, we used 4 out of 5 countries from the ICEWS extract in Bhattacharjya *et al.* [2018], which includes events involving 5 types of actors and 5 types of actions, occurring from Jan 1, 2012 to Dec 31, 2015. (One country was omitted due to the inconsistency between event labels while splitting the data into three sets for experiments.)

**Mimic-II** [Saeed *et al.*, 2011]. These are patient electronic health records from Intensive Care Unit visits over 7 years. Each patient experiences a sequence of visit events, where each event involves a time stamp and diagnosis. We filter out small sequences to obtain 650 patients with 204 disease types.

**Diabetes** [Frank and Asuncion, 2010]. Events for around 70 diabetic patients are considered: these include different types of meal ingestion, exercise and insulin dosage, along with two additional processed event labels corresponding to the increase and decrease of blood glucose measurement levels. These latter events are obtained after discretization of blood glucose measurements into three states.

**LinkedIn** [Xu *et al.*, 2017]. This includes employment and (when applicable) college enrollment related information of 2489 anonymous LinkedIn users. Each event stream includes a user's time-stamped records of professional experience, such as joining a new role in a company. We filter the data to popular companies and end up with 1000 users.

| Dataset | PGEM | OGEM | NHP |
|---|---|---|---|
| **ICEWS** | | | |
| Argentina | -1386.05 | **-1369** | **-1338.65** |
| Brazil | **-2000.47** | -2057.45 | **-1892.57** |
| Colombia | -534.46 | **-517.82** | -559.61 |
| Mexico | -796.50 | **-771.17** | -919.82 |
| **Mimic** | -495.41 | **-476.07** | - |
| **Diabetes** | -2966.23 | **-2883.62** | - |
| **LinkedIn** | -1479.26 | **-1478.38** | - |

Table 1: Log likelihood for the models on the test sets.

## 4.2 Model Fit

We conduct an experiment to evaluate how well the proposed model fits the afore-mentioned datasets.

**Experimental Setup.** Each dataset is split into three sets: train (70%), dev (15%) and test (15%), only retaining event labels that are common to all three splits. Single stream datasets like ICEWS countries are split by time, e.g. if $T = 1000$ days, then events up to time 700 days are in train. Multiple stream datasets like LinkedIn are split by stream id, e.g. for $K = 1000$ users, streams for 700 of them constitute the train set. A model's performance is measured by the log likelihood on the held-out test set. During both training and testing, we disallow positive log likelihoods to minimize over fitting, capping it at zero for any node. Hyper-parameter choices for OGEM and the baselines are as follows:

- **OGEM**: We search over a default rate hyper-parameter grid of $\lambda^0 = \{0.001, 0.005, 0.01, 0.05, 0.1\}$. Window hyper-parameter grids are dataset specific, chosen as:
  - ICEWS: $w_X = \{1, 3, 7, 10, 15, 30, 60\}$ (days) $\forall X$
  - Mimic: $w_X = \{0.1, 0.2, 0.5, 1, 1.5, 2, 5\}$ (years) $\forall X$
  - Diabetes: $w_X = \{0.01, 0.05, 0.1, 0.5, 1, 5\}$ (days) $\forall X$
  - LinkedIn: $w_X = \{2, 5, 7, 10, 15, 20\}$ (years) $\forall X$
- **PGEM**: The closest baseline is the proximal GEM, which allows different windows for different parents but does not distinguish between orders of causal events. We deploy the learning approach in Bhattacharjya *et al.* [2018], which also identifies windows using a heuristic. We use left limiting parameter $\epsilon = 0.001$ and default rate $\lambda^0$ as the only hyper-parameter with the same grid as OGEM.
- **NHP**: Primarily just for reference, we also learn a neural Hawkes process [Mei and Eisner, 2017], a state-of-the-art neural architecture for event models. Neural networks are expected to do much better than fully parametric ones on the model fitting task due to the large number of parameters. NHP does not however learn a graphical model and is less interpretable than the other models considered, making it less useful for discovery. For NHP, the only hyper-parameter is the number of epochs for training.

For all models, the optimal hyper-parameter setting is chosen by training models under various settings using the train set and finding the best performing setting on the dev set. The optimal trained model is then evaluated on the test set.

**Results.** Table 1 compares the log likelihood evaluated on test sets across models. In the OGEM column, we show the masking function case ('first'/'last') that performs better. Aside from Brazil, where PGEM performs well, OGEM exhibits superior performance. OGEM also does reasonably compared to NHP, beating it on two of the four ICEWS countries; NHP was anticipated to perform substantially better on this task. Note that NHP was not run on the multiple stream datasets because there is a peculiarity about these datasets that makes it an inappropriate baseline: they are processed to almost always have events at time $t = 0$, and the neural network exploits this by always artificially spiking the conditional intensity rate at the start time. As a result, we only compare OGEM with PGEM for these datasets.

## 4.3 Causal Orders Analysis

The power of the OGEM is that it is able to reveal orders of causal events that are influential for a particular event of interest. Here we investigate an application of OGEMs for social unrest related events on an extract of ICEWS data from January 1, 2006 till December 31, 2010. We restrict attention to the following six actors: Police, Citizen, Government, Head of Government, Protester, Military; these are among the most frequently participating actors for these countries.

**Experimental Setup.** We consider the social unrest event label 'Police; Fight; Citizen' and learn its OGEM parents and parameters, using event data corresponding to three Latin American countries – Argentina, Mexico and Venezuela. Other events were also studied but results for only one event are provided as an illustration due to space restrictions. For this analysis, we use the 'first' masking function $\phi(\cdot)$, penalty weight $\gamma = 0.1$ on the complexity term, default rate $\lambda^0 = 0.001$ and window $w_X = 30$ days.

**Results.** Selected results from our analysis are shown in Table 2. Each column in the table ranks the orders (sequences) of event labels of the identified parents based on the model's estimated conditional intensity rates. We arbitrarily choose three orders each to display. We point the reader's attention to some observations about the ways in which the preceding order affects the rate of an interaction event in ICEWS:

- In both Argentina and Mexico, missing preceding events have an impact on the rate. For instance, when the military cooperates materially with the police in Argentina, for instance to provide arms – and importantly when no other causal events occur – the rate of a police fight-related event is very high. Contrast this with the bottom order where the rate decreases, perhaps because the police's resources are consumed elsewhere.
- The impact of the order of two events is prominent for Venezuela where switching the order of two parent events alters the rate substantially. Here the recent action of a citizen making a (presumably negative) statement about the government after the government makes an international statement results in a higher rate of the event of interest than if the international government statement comes later. This is an example of the finer expressiveness of OGEMs, in accordance with Theorem 4.

| **Argentina** | | **Mexico** | | **Venezuela** | |
|---|---|---|---|---|---|
| 1. (Military; Material Coop.; Police) | 4 | 1. (Citizen; Yield; Military) | 0.5 | 1. (Govt.; Statement; Head of Govt.-Mexico)<br>2. (Citizen; Statement; Govt.) | 0.71 |
| 1. (Citizen; Reject; Head of Govt.)<br>2. (Police; Fight; Govt.) | 0.5 | 1. (Citizen; Yield; Military)<br>2. (Citizen; Demand; Govt) | 0.31 | 1. (Citizen; Statement; Govt.)<br>2. (Govt.; Statement; Head of Govt.-Mexico) | 0.04 |
| 1. (Military; Material Coop.; Police)<br>2. (Police; Fight; Govt.) | 0.12 | 1. (Citizen; Demand; Govt) | 0.08 | 1. (Citizen; Statement; Govt.) | 0.04 |

Table 2: Ranking the causal effect of sequences for the event of interest 'Police; Fight; Citizen' by intensity rate.

## 5 Related Work

Our proposed model leverages data in the form of streams of irregularly occurring time-stamped events. We briefly summarize the most relevant literature around event modeling.

### 5.1 Event Sequences

Sequences of events go by various terms, including episodes [Mannila *et al.*, 1997], narratives [Chambers and Jurafsky, 2008], storylines [Radinsky and Horvitz, 2013] and scenarios [Hashimoto *et al.*, 2014]. (Here we have used the term 'order' or 'sequence'.) Several analytical domains have long pursued event sequence models, such as in point processes in statistics [Cox and Lewis, 1972] and frequent episode mining in data mining [Mannila *et al.*, 1997]. Predicting from sequences without time-stamped information has also been widely studied in machine learning [Rudin *et al.*, 2012] and in natural language processing, particularly for narrative cloze and related tasks [Radinsky and Horvitz, 2013; Granroth-Wilding and Clark, 2016]. Our work is different from this more recent literature in that: 1) we take continuous-time event streams as input, and 2) we pursue a graphical modeling approach where the sequence of only the underlying causes matters.

### 5.2 Temporal Point Processes and Graphs

The framework of graphical event models (GEMs) [Didelez, 2008; Meek, 2014] captures the inter-event dependency in continuous-time event data by representing a marked point process, which models event dynamics using conditional intensity functions. The specific variants of GEMs typically differ in how they parametrize intensity functions, ex: through generalized linear models [Rajaram *et al.*, 2005], decision trees [Gunawardana *et al.*, 2011], forests [Weiss and Page, 2013] and proximal windows [Bhattacharjya *et al.*, 2018]. Due to the success of deep learning in many domains, deep neural network methods have also been proposed to model event streams, such as recurrent neural network models [Xiao *et al.*, 2017; Du *et al.*, 2016] and neural Hawkes processes [Mei and Eisner, 2017].

A related stream of work considers probabilistic generative models for relational event models [DuBois and Smyth, 2010; Schein *et al.*, 2015], where events have a dyadic character, i.e. they can be described in terms of a pair of actors (say, sender and receiver) that are coupled with a certain relation (say, action) like in the ICEWS dataset.

Note that there are other general graphical representations that are peripherally related to our work, including discrete time models such as dynamic Bayesian networks [Dean and Kanazawa, 1989] and graphs for time series [Eichler, 1999]. Specialized graphs for multi-agent problems have also been proposed [Grover *et al.*, 2018; Bhargava and Williams, 2019]. However, GEMs related work appears to be the most relevant graphical modeling literature; we have shown how our proposed model fits within the broader GEMs framework.

## 6 Conclusions

We have introduced a novel model for capturing order-dependent causal influences in event datasets, motivated by representing the behavior of single or multiple agents using only data about their actions/interactions. This provides a data-driven and domain-agnostic alternative to traditional approaches involving hand-crafted models requiring prior domain knowledge. We presented an efficient algorithm for learning the graphical structure and parameters of an OGEM, demonstrating comparable or better model fitting performance than baselines on various benchmark datasets.

An OGEM's ability to expose the order-sensitive nature of the rate of observing certain agent interactions provides a facet of analysis that would not be attained by alternate order-neutral graphical event models. This was highlighted theoretically but also experimentally, through an investigation conducted on agent interactions in the ICEWS political event dataset. We believe that analysts in professions such as business, intelligence and finance would find such an interpretable model and its order-related insights beneficial.

A major limitation in the current OGEM formulation is that the number of parameters is super-exponential in the number of parents. While we have partially addressed this issue here by penalizing model complexity, this opens up possibilities for future work around more compact parameter representations. Another potential challenge is around the choice of masking function to determine historical order; in domains where events recur rapidly, the 'first' and 'last' cases may be inappropriate and possibly even subvert causal analysis with poorly chosen windows. Another line of future work would therefore be to automatically learn some of the hyperparameters, notably the windows of historical dependence.

## Acknowledgments

The authors thank the anonymous reviewers for their valuable feedback. This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

## References

[Aalen *et al.*, 2008] O. O. Aalen, O. Borgan, and H. K. Gjessing. *Survival and Event History Analysis: A Process Point of View*. Springer Science & Business Media, New York, NY, USA, 2008.

[Bhargava and Williams, 2019] N. Bhargava and B. Williams. Multiagent disjunctive temporal networks. In *Proceedings of the International Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pages 458–466, 2019.

[Bhattacharjya *et al.*, 2018] D. Bhattacharjya, D. Subramanian, and T. Gao. Proximal graphical event models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 8147–8156, 2018.

[Chambers and Jurafsky, 2008] N. Chambers and D. Jurafsky. Unsupervised learning of narrative event chains. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 94305, pages 789–797, 2008.

[Cox and Lewis, 1972] D. R. Cox and P. A. W. Lewis. Multivariate point processes. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 3: Probability Theory*, pages 401–448, 1972.

[Dean and Kanazawa, 1989] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150, 1989.

[Didelez, 2008] V. Didelez. Graphical models for marked point processes based on local independence. *Journal of the Royal Statistical Society, Ser. B*, 70(1):245–264, 2008.

[Du *et al.*, 2016] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1555–1564, 2016.

[DuBois and Smyth, 2010] C. DuBois and P. Smyth. Modeling relational events via latent classes. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 803—812, 2010.

[Eichler, 1999] M. Eichler. *Graphical Models in Time Series Analysis*. PhD thesis, University of Heidelberg, Germany, 1999.

[Frank and Asuncion, 2010] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[Gerner *et al.*, 2002] D. J. Gerner, P. A. Schrodt, O. Yilmaz, and R. Abu-Jabr. Conflict and mediation event observations (CAMEO): A new event data framework for the analysis of foreign policy interactions. *International Studies Association (ISA) Annual Convention*, 2002.

[Granroth-Wilding and Clark, 2016] M. Granroth-Wilding and S. Clark. What happens next? Event prediction using a compositional neural network model. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 272–2733, 2016.

[Grover *et al.*, 2018] A. Grover, M. Al-Shedivat, J. K. Gupta, Y. Burda, and H. Edwards. Learning policy representations in multiagent systems. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1797–1806, 2018.

[Gunawardana and Meek, 2016] A. Gunawardana and C. Meek. Universal models of multivariate temporal point processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 556–563, 2016.

[Gunawardana *et al.*, 2011] A. Gunawardana, C. Meek, and P. Xu. A model for temporal dependencies in event streams. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1962–1970, 2011.

[Hashimoto *et al.*, 2014] C. Hashimoto, K. Torisawa, J. Kloetzer, M. Sano, I. Varga, J.-H. Oh, and Y. Kidawara. Toward future scenario generation: Extracting event causality exploiting semantic relation, context, and association features. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 987–997, 2014.

[Mannila *et al.*, 1997] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259–289, 1997.

[Meek, 2014] C. Meek. Toward learning graphical and causal process models. In *Proceedings of the Uncertainty in Artificial Intelligence Workshop on Causal Inference: Learning and Prediction*, pages 43–48, 2014.

[Mei and Eisner, 2017] H. Mei and J. M. Eisner. The neural Hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6754–6764, 2017.

[O'Brien, 2010] S. P. O'Brien. Crisis early warning and decision support: Contemporary approaches and thoughts on future research. *International Studies Review*, 12:87–104, 2010.

[Pearl, 2014] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2014.

[Radinsky and Horvitz, 2013] K. Radinsky and E. J Horvitz. Mining the web to predict future events. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 255–264, 2013.

[Rajaram *et al.*, 2005] S. Rajaram, T. Graepel, and R. Herbrich. Poisson-networks: A model for structured point processes. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS)*, pages 277–284, 2005.

[Remondino and Correndo, 2005] M. Remondino and G. Correndo. Data mining applied to agent based simulation. In *Proceedings of the Nineteenth European Conference on Modelling and Simulation*, pages 1–4, 2005.

[Rudin *et al.*, 2012] C. Rudin, B. Letham, A. Salleb-Aouissi, E. Kogan, and Madigan D. Sequential event prediction with association rules. In *Proceedings of the Annual Conference on Learning Theory (COLT)*, pages 615–634, 2012.

[Saeed *et al.*, 2011] M. Saeed, M. Villarroel, A. T. Reisner, G. Clifford, L.-W. Lehman, G. Moody, T. Heldt, T. H. Kyaw, B. Moody, and R. G. Mark. Multiparameter intelligent monitoring in intensive care II (MIMIC-II): A public-access intensive care unit database. *Critical Care Medicine*, 39(5):952, 2011.

[Schein *et al.*, 2015] A. Schein, J. Paisley, D. M. Blei, and H. Wallach. Bayesian Poisson tensor factorization for inferring multilateral relations from sparse dyadic event counts. In *Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1045–1054, 2015.

[Weiss and Page, 2013] J. C. Weiss and D. Page. Forest-based point process for event prediction from electronic health records. In *Machine Learning and Knowledge Discovery in Databases*, pages 547–562, 2013.

[Xiao *et al.*, 2017] S. Xiao, J. Yan, X. Yang, H. Zha, and S. M. Chu. Modeling the intensity function of point process via recurrent neural networks. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, volume 17, pages 1597–1603, 2017.

[Xu *et al.*, 2017] H. Xu, D. Luo, and H. Zha. Learning Hawkes processes from short doubly-censored event sequences. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3831–3840, 2017.