What happens next?

From Poisson counts to transformer-based event models

Andrew Bell

L2 Labs, Cornell Tech

September 2, 2026

Business processes as event streams

PO $\longrightarrow$ PO Acknowledgment $\longrightarrow$ ASN $\longrightarrow$ Invoice

Each event has at least two ingredients:

$$e_i = (t_i, k_i)$$

- t_i: when the event happened
- k_i: what kind of business event happened

Goal

Learn the behavior of when and what things should happen.

Two kinds of anomalies

Something unexpected happened.

Something expected did not happen.

Examples:

- A PO arrives at an unusual time or with unusual contents
- A customer normally sends a PO every Tuesday morning, but none arrives
- A PO is received, but the PO Ack does not appear within the normal response window

Naive approach: count events in a fixed window

Suppose we count the number of POs in a Tuesday-morning window like 6:00am to Noon.

$$N \sim \text{Poisson}(\lambda)$$

$$P(N = n) = \frac{e^{-\lambda}\lambda^n}{n!}$$

If historically $\lambda = 5$, then

$$P(N = 0) = e^{-5} \approx 0.0067.$$

Intuition

If this customer usually sends about five POs on Tuesday morning, then receiving zero is unlikely under the model.

What the Poisson model misses

The count model asks: *how many events occurred?*

But many business anomalies are about timing: *how long have we been waiting?*

For example:

PO received at 8:00 AM

PO Ack after 20 minutes: normal

PO Ack after 6 hours: suspicious

Next idea: model the waiting time

For each PO followed by a PO Ack, define

$$\Delta T_i = T^{(i)}_{\text{PO Ack}} - T^{(i)}_{\text{PO}}$$

Historical deltas might be

$$18,\ 31,\ 24,\ 27,\ 22,\ 36, \ldots \quad \text{minutes}$$

One idea is to learn a distribution over these deltas:

$$\Delta T \sim F$$

Intuition

F describes how long we normally wait between a PO and its PO Ack.

Waiting-time model: notation

$$F(t) = P(\Delta T \leq t)$$

$$S(t) = P(\Delta T > t) = 1 - F(t)$$

- $T_{\text{PO}}^{(i)}$: timestamp of the i-th PO
- $T_{\text{PO Ack}}^{(i)}$: timestamp of the corresponding PO Ack
- ΔT_i: observed waiting time for pair i
- $F(t)$: probability that the PO Ack has arrived by elapsed time t
- $S(t)$: probability that we are still waiting after elapsed time t

How do we learn F?

Empirical CDF

$$\widehat{F}(t) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[\Delta T_i \leq t]$$

Here, n is the number of historical observations. The sum counts how many observed delays satisfy $\Delta T_i \leq t$, so $\widehat{F}(t)$ is the fraction of past events that occurred within time t.

Fit a parametric model

$$\Delta T \sim \text{LogNormal}(\mu, \sigma^2), \qquad \Delta T \sim \text{Gamma}(a, b).$$

Waiting-time anomaly score

At current elapsed time $t^\star$, the simplest anomaly rule is

$$S(t^\star) < \tau$$

Example:

$$S(10 \text{ min}) = 0.82, \qquad S(30 \text{ min}) = 0.35,$$

$$S(60 \text{ min}) = 0.04, \qquad S(180 \text{ min}) = 0.001$$

- $t^\star$: elapsed time since the PO arrived
- $\tau \in (0, 1)$: alert threshold
- Small $S(t^\star)$: continued non-arrival is unlikely

Why waiting times are still not enough

A PO acknowledgment is not floating randomly through time.

- Before the PO arrives, $P(\text{PO Ack soon}) \approx 0$.
- But after the PO arrives, $P(\text{PO Ack soon})$ increases.

The event stream has structure:

$$\text{PO} \to \text{PO Ack} \to \text{ASN} \to \text{Invoice}.$$

Hawkes process: the formula

A simple Hawkes process uses

$$\lambda(t) = \mu + \sum_{t_i < t} \alpha e^{-\beta(t - t_i)}.$$

event occurs ⟶ intensity jumps ⟶ effect decays

Intuition

Past events temporarily raise the expected future event rate.

Hawkes process: notation

$$\lambda(t) = \mu + \sum_{t_i < t} \alpha e^{-\beta(t - t_i)}$$

- t: current continuous time
- $t_i < t$: timestamp of a previous event
- $\lambda(t)$: conditional intensity at time t; informally, the instantaneous event rate
- $\mu \geq 0$: baseline intensity, independent of previous events
- $\alpha \geq 0$: excitation strength from each prior event
- $\beta > 0$: decay rate of the prior event's influence
- $e^{-\beta(t-t_i)}$: exponential decay kernel

At $t = t_i$, the excitation contribution is α. As $t - t_i \to \infty$, the contribution decays to 0.

Multivariate Hawkes for business events

Now give each event type its own intensity:

$$\lambda_{\text{PO}}(t), \quad \lambda_{\text{PO Ack}}(t), \quad \lambda_{\text{ASN}}(t), \quad \lambda_{\text{Invoice}}(t).$$

A multivariate model can express relationships like

- After a PO, $\lambda_{\text{PO Ack}}(t)$ increases
- After a PO acknowledgment, $\lambda_{\text{ASN}}(t)$ increases
- After an ASN, $\lambda_{\text{Invoice}}(t)$

Intuition

Events do not merely occur at rates. They change the future rates of other event types.

The important principle: conditional intensity

The general point-process object is

$$\lambda_k(t \mid \mathcal{H}_t)$$

Intuition

Given everything that has happened so far, how strongly do we expect event type k right now? Where for business events, $k \in \{\text{PO}, \text{PO Ack}, \text{ASN}, \text{Invoice}\}$

Conditional intensity: notation

$$\lambda_k(t \mid \mathcal{H}_t)$$

- k: event type.
- t: current time.
- $\mathcal{H}_t$: complete event history strictly before time t.
- $\lambda_k(t \mid \mathcal{H}_t)$: conditional intensity for event type k at time t.

The history $\mathcal{H}_t$ may contain:

$$(t_1, k_1), (t_2, k_2), \ldots, (t_i, k_i), \qquad t_i < t.$$

Central modeling question

How should the history $\mathcal{H}_t$ determine the future intensities $\lambda_k(t \mid \mathcal{H}_t)$?

Where does $\lambda_k(t \mid \mathcal{H}_t)$ come from?

Classical models specify the functional form by hand.

But real business behavior may depend on things like the customer, supplier, weekday, order size, location, seasonality, history, ...

Modern ML re-frames the problem as: *learn a representation of history from data*.

Abstractly:

$$\lambda_k(t) = f_\phi(h_i, t), \qquad h_i = g_\theta(e_1, \ldots, e_i)$$

Learned model: notation

$$\lambda_k(t) = f_\phi(h_i, t), \qquad h_i = g_\theta(e_1, \ldots, e_i)$$

- e_i: the i-th observed business event
- h_i: learned representation of the history after observing events $1, \ldots, i$
- g_θ: history encoder with learnable parameters θ
- f_ϕ: learned intensity function with parameters ϕ
- $\lambda_k(t) = f_\phi(h_i, t)$: predicted intensity for event type k after history state h_i

Interpretation

The model learns what operational state we are in, then uses that state to predict future events.

Neural Hawkes (RNN-based)

A recurrent model updates its state one event at a time:

$$h_i = g_\theta(h_{i-1}, e_i)$$

Then the state controls the future intensities:

$$\lambda_k(t \mid h_i)$$

Paper implementation

Use a continuous-time LSTM so the hidden state can evolve both when events occur and during the gaps between events.

Transformer Hawkes

A Transformer replaces sequential memory with attention:

$$h_i = \text{Transformer}_\theta(e_1, \ldots, e_i)$$

When predicting an ASN, the model can directly attend to relevant prior events:

related PO, related PO Ack, recent history

The learned state again parameterizes

$$\lambda_k(t \mid h_i)$$

Paper implementation

Transformer Hawkes uses a causal attention mask: past visible, future hidden. This is the same attention mechanism used in decoder-only GPT-style Transformers.

Point-process loss: events vs. empty time

A standard training objective is the negative log-likelihood:

$$\mathcal{L}_{\mathrm{NLL}} = \underbrace{-\sum_{i=1}^{n} \log \lambda_{k_i}(t_i \mid \mathcal{H}_{t_i})}_{\text{event term}} + \int_0^T \sum_k \lambda_k(t \mid \mathcal{H}_t)\, dt$$

- Event term: penalizes the model when intensity is low at events that actually occurred
- Non-event term: penalizes the model for assigning high intensity across time when those predicted events do not materialize

To minimize loss, the model must put intensity in the right places, not everywhere

What does the training data look like?

For each business relationship, construct a time-ordered event stream:

$$\mathcal{S} = \{(t_1, k_1), (t_2, k_2), \ldots, (t_n, k_n)\}$$

For example:

8:05	:	PO
8:24	:	PO Ack
16:30	:	ASN
9:12	:	Invoice

During training, the model repeatedly sees:

$$\underbrace{(t_1, k_1), \ldots, (t_i, k_i)}_{\text{history}} \longrightarrow \underbrace{(t_{i+1}, k_{i+1})}_{\text{what happens next, and when}}$$

Masked-event pretraining

A later self-supervised idea is to hide events:

$$\text{PO} \rightarrow [\text{MASK}] \rightarrow \text{ASN} \rightarrow \text{Invoice}$$

The model predicts:

- what event belongs there
- when it occurred

Event-stream models can also add *void events*:

$$\text{PO} \rightarrow \text{VOID} \rightarrow \text{PO Ack} \rightarrow \text{VOID} \rightarrow \text{ASN}.$$

that explicitly represents times at which nothing happened, using a fixed time interval.

From event labels to rich business information

Classical event-stream models often use

$$e_i = (t_i, k_i)$$

For a business event, an event can be richer:

$$e_i = (t_i, \text{sender}, \text{reciever}, \text{transaction type}, \text{amount}, \text{SKUs}, \text{locations}, \ldots)$$

A natural hierarchy is

$$z_i = f_\theta(\text{message } i),$$
$$h_i = g_\phi(z_1, \ldots, z_i)$$

- z_i: representation of one business message
- h_i: representation of the operational stream state

Anomaly detection as predictive surprise

For an observed event:

$$A_{\text{event}}(i) = -\log p(e_i \mid \mathcal{H}_{t_i})$$

For an expected event that has not arrived:

$$A_{\text{absence}}(t) = -\log P(T_{\text{next}} > t \mid \mathcal{H}_t)$$

- High A_{event}: something happened that the model found unlikely
- High A_{absence}: something expected has not happened for too long

We use $-\log(\cdot)$ so that lower-probability outcomes receive larger anomaly scores; this is the standard "surprisal" transformation of probability.

Selected references

- Hawkes, A. G. (1971). "Spectra of Some Self-Exciting and Mutually Exciting Point Processes." *Biometrika*.
- Rasmussen, J. G. (2018). "Lecture Notes: Temporal Point Processes and the Conditional Intensity Function." arXiv:1806.00221.
- Du et al. (2016). "Recurrent Marked Temporal Point Processes: Embedding Event History to Vector." *KDD*.
- Mei and Eisner (2017). "The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process." *NeurIPS*.
- Zuo et al. (2020). "Transformer Hawkes Process." *ICML*.
- Liu and Hauskrecht (2021). "Event Outlier Detection in Continuous Time." *ICML*.