HawkesLLM: Semantic Uncertainty Propagation in Agentic Text Simulation

Zewei Deng [1] **Tinghan Ye** [2] **Liyan Xie** [1]

Abstract

Agentic text-simulation systems write in sequence, with each item becoming possible context for later steps. That makes uncertainty path-dependent: an early ambiguity can affect later outputs. This paper studies this problem with HawkesLLM, a framework that separates temporal influence modeling from text generation. We represent the cascade as a network whose nodes are text-generating agents. A multivariate Hawkes process models how these nodes activate over time and which earlier node outputs should influence later prompts. A language model then writes each new event from the compact memory selected by this temporal model. We evaluate the framework on a held-out Global Database of Events, Language, and Tone (GDELT) news-cascade case study. The diagnostics track semantic alignment with local held-out references and separate local drift from global drift. In this setting, HawkesLLM improves late-stage semantic alignment under a compact prompt-memory budget.

1. Introduction

Many text-simulation workflows unfold as multi-step processes. A system writes an item, keeps it in memory, and later writes with that memory in view. Here, *agentic* refers to this iterative use of generated memory: each output can become part of the next prompt. In such systems, uncertainty belongs to the trajectory, not just to one response. Ambiguity, drift, and mismatch can carry forward because later prompts depend on earlier generated content.

Recent work on uncertainty in language-agent systems and semantic uncertainty in free-form generation motivates this view (Zhao et al., 2025; Han et al., 2024; Liu et al., 2024a; Farquhar et al., 2024; Lin et al., 2023). If generated outputs become future context, uncertainty should be tracked across the sequence rather than only at the end.

We study this problem as *semantic uncertainty propagation*: how generated events stay, or fail to stay, near local held-out reference neighborhoods as the generated history grows. The references serve as semantic anchors from the same event stream. They let us ask whether the simulation remains in the same local region over time.

We propose HawkesLLM for this setting. The simulation is a cascade over text-generating nodes. A Hawkes process models when nodes activate and which earlier node outputs should matter for the next prompt. The language model then writes the next event from this selected memory. We use *memory selection* for the choice of node-specific predecessor texts and weights. This design is also motivated by evidence that long-context language models do not always use all provided context reliably (Liu et al., 2024b).

We demonstrate the framework on a GDELT news-cascade case study. The news domain is useful here because it has timestamped event streams that can be mapped to a small set of node categories. Held-out articles from the same topic window serve as imperfect local semantic references for generated events.

We make three contributions. First, we formulate semantic uncertainty propagation for iterative text simulation. Second, we develop HawkesLLM as a Hawkes-guided transition that samples event times and nodes and constructs compact node-wise prompt memory. Third, we evaluate it on the held-out news-cascade case study, where HawkesLLM improves late-stage alignment under a compact prompt-memory budget.

2. Related Work

One line of work asks how uncertainty should be handled inside multi-step agent systems (Zhao et al., 2025; Han et al., 2024; Liu et al., 2024a). In that setting, uncertainty often becomes a control signal: the agent may call a tool, ask for help, or calibrate its answer. Here the object is the generated text path. Earlier generations become part of the state that later generations read, so we track how semantic uncertainty moves along the realized cascade.

Semantic-uncertainty and black-box confidence methods

[1]Department of Industrial and Systems Engineering, University of Minnesota [2]H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology. Correspondence to: Zewei Deng <deng0379@umn.edu>.

Accepted to the ICML 2026 Workshop on Statistical Frameworks for Uncertainty in Agentic Systems, Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

usually compare possible answers to the same free-form generation problem (Farquhar et al., 2024; Lin et al., 2023). Drift work looks at what happens after generation is repeated for many steps (Spataru et al., 2024; Mohamed et al., 2025). We take the path itself as the object of evaluation: each generated event is compared with local held-out reporting, while the drift metrics separate short-range agreement with prompt memory from longer-range movement away from the seed.

Graph-cascade models give us a fixed graph, seed events, and a cascade unfolding over that graph (Kempe et al., 2015; Kleinberg, 2007). In our setting, each activation also produces text. We use a Hawkes process for the temporal layer because it turns past activations into directed, decaying influence scores between nodes (Hawkes, 1971; Ogata, 1981; Rizoiu et al., 2017). Neural temporal point processes and recent temporal-language hybrids can model richer dynamics or next-event prediction (Mei & Eisner, 2017; Zuo et al., 2020; Li et al., 2025; Liu & He, 2025; Zhou et al., 2025). The simpler parametric Hawkes model is useful here because its fitted node-to-node influence can be exposed as an inspectable memory signal for the LLM prompt. In that sense, HawkesLLM sits between LLM-based social or agent simulation (Park et al., 2023; Sun et al., 2024; Murdock et al., 2023; Zhang et al., 2025) and retrieval-augmented generation (Lewis et al., 2020): the state is a sequence of generated event texts, and the retrieved memory is driven by temporal cascade dynamics.

3. Methods

HawkesLLM has two layers. The first is a text-simulation loop: a node is scheduled, a compact memory is selected, and the LLM writes the next event. The second is a Hawkes process that supplies the schedule and memory weights. Section 3.1 defines the generic loop. Section 3.2 introduces the multivariate Hawkes model for temporal influence among nodes. Section 3.3 then plugs that model into the loop.

3.1. Agentic Text Simulation Framework

We model text propagation on a fixed directed graph (Kempe et al., 2015; Kleinberg, 2007). A node is a text-generating agent in the simulation, and an event is a timestamped activation of one node together with the text generated at that activation. Let $\mathcal{G}_0 = (\mathcal{N}, \mathcal{E})$ denote the unweighted influence graph. Here $\mathcal{N} = \{1, \dots, N\}$ is the node set, and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ is the set of allowed directed influences. We assume $\mathcal{E} = \mathcal{N} \times \mathcal{N}$, so every node can influence every other node, including itself. Any node may be activated at a generation step, and any earlier activated node can contribute memory when allowed by $\mathcal{E}$. Section 3.2 describes how we learn the effective edge weights from data.

Generation starts from a seed event $e_0 = (\tau_0, n_0, x_0)$, where τ_0 is the seed timestamp, $n_0 \in \mathcal{N}$ is the seed node, and x_0 is the seed text. The realized cascade is the event history that grows from this seed over $\mathcal{G}_0$. At each later step, the current node receives a small set of earlier node outputs as prompt memory and generates a new text. We use t for the current generation step and m for a previous event. The generated history before step t is

$$\mathcal{H}_t = \{e_m = (\tau_m, n_m, x_m) : 0 \le m < t\},$$

where τ_m is the timestamp, $n_m \in \mathcal{N}$ is the node, and x_m is the text of event m. Let $k \ge 1$ be the maximum number of node representatives allowed in the prompt. A memory policy π maps the history, next timestamp, and next node to compact weighted memory, $\mathcal{M}_t = \pi(\mathcal{H}_t, \tau_t, n_t)$. Each memory item keeps one representative from one node. If node j is retained for step t, then $r_t(j)$ denotes its latest previous generated event before τ_t. The memory has the form

$$\mathcal{M}_t \subseteq \{(j, r_t(j), w_{j,t}) : j \in \mathcal{N},\ r_t(j) < t,\ w_{j,t} > 0\}.$$

Here $w_{j,t}$ is the weight assigned to node j, and $|\mathcal{M}_t| \le k$. When $\mathcal{M}_t \neq \emptyset$, the selected weights satisfy

$$\sum_{(j, r_t(j), w_{j,t}) \in \mathcal{M}_t} w_{j,t} = 1.$$

Memory selection means choosing which node representatives, and which weights, enter the next prompt.

The text transition is then explicit. Let a_i denote the fixed instruction string for node i. Given τ_t, n_t, and $\mathcal{M}_t$, the prompt is

$$p_t = \mathrm{Prompt}\big((\tau_t, n_t),\ a_{n_t},\ \{(w_{j,t}, j, x_{r_t(j)}) : (j, r_t(j), w_{j,t}) \in \mathcal{M}_t\}\big).$$

Here $\mathrm{Prompt}(\cdot)$ denotes deterministic formatting. In our implementation, the prompt states the target node, gives a short node-style instruction, and lists the selected predecessor texts with their node labels and normalized Hawkes weights. The weights are written as text annotations, so they guide the model through the prompt rather than acting as numerical controls. Appendix A gives a representative instantiated prompt.

When $\mathcal{M}_t = \emptyset$, the prompt contains only the current event information and node instruction. Let g_{LLM} denote the text generator. The next text is generated as

$$x_t \sim g_{\mathrm{LLM}}(\cdot \mid p_t),$$

or deterministically as $x_t = g_{\mathrm{LLM}}(p_t)$ when the decoding seed is fixed. The semantic propagation update appends the completed event:

$$e_t = (\tau_t, n_t, x_t), \qquad \mathcal{H}_{t+1} = \mathcal{H}_t \cup \{e_t\}.$$

Only the text x_t is produced by the language model at step t. The timestamp, node, and memory $\mathcal{M}_t$ determine the prompt that conditions it. Figure 1 summarizes this loop.

3.2. Multivariate Hawkes Point Process

We use a multivariate Hawkes point process as the temporal influence model (Hawkes, 1971; Ogata, 1981; Rizoiu et al., 2017). It estimates when each node is likely to activate and how much past activity from one node raises the future rate of another. For fitting, let $\mathcal{D} = \{(\tau_m, n_m)\}_{m=1}^{M}$ denote an observed node-time event stream over horizon $[0, T]$. Here M is the number of observed events. Event m occurs at time $\tau_m \in [0, T]$ and belongs to node $n_m \in \mathcal{N}$. In this subsection, i indexes the node whose rate is being evaluated, and j indexes a node with a previous event. At continuous time s, let $\lambda_i(s)$ denote the conditional event rate for node i. Let μ_i denote its background rate. For each edge $(j, i) \in \mathcal{E}$, let $\phi_{j,i}(u)$ denote the excitation kernel at lag $u > 0$. It describes how much a past event from node j raises the future rate of node i. The event stream is modeled as

$$\lambda_i(s) = \mu_i + \sum_{j:(j,i)\in\mathcal{E}} \sum_{\tau_m < s,\, n_m = j} \phi_{j,i}(s - \tau_m).$$

Our main model uses an exponential kernel. Here $\alpha_{j,i} \geq 0$ is the directed excitation strength, and $\beta > 0$ is the decay rate:

$$\phi_{j,i}(u) = \alpha_{j,i} e^{-\beta u}.$$

Larger β makes influence fade faster. For the exponential kernel, the integrated excitation matrix $\mathbf{G}$ has entries $G_{j,i} = \int_0^\infty \phi_{j,i}(u)\, du = \alpha_{j,i}/\beta$ for $(j, i) \in \mathcal{E}$, and $G_{j,i} = 0$ otherwise. Stability is assessed through the spectral radius $\rho(\mathbf{G})$.

The weighted influence graph is $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \mathbf{G})$. The unweighted graph $\mathcal{G}_0$ determines which influences are allowed, while $G_{j,i}$ gives the fitted cumulative excitation from node j to node i. Later, when building prompt memory, HawkesLLM turns this fitted influence into node-wise scores by applying temporal decay to earlier generated events from each node.

For a fixed decay value β, we estimate $(\boldsymbol{\mu}, \boldsymbol{\alpha})$. Here $\boldsymbol{\mu} = (\mu_i)_{i\in\mathcal{N}}$ and $\boldsymbol{\alpha} = (\alpha_{j,i})_{(j,i)\in\mathcal{E}}$. Let $\lambda_i(s; \boldsymbol{\mu}, \boldsymbol{\alpha}, \beta)$ denote the corresponding conditional intensity. The log-likelihood term is

$$\ell_\beta(\boldsymbol{\mu}, \boldsymbol{\alpha}; \mathcal{D}) = \sum_{m=1}^{M} \log \lambda_{n_m}(\tau_m; \boldsymbol{\mu}, \boldsymbol{\alpha}, \beta) - \sum_{i\in\mathcal{N}} \int_0^T \lambda_i(s; \boldsymbol{\mu}, \boldsymbol{\alpha}, \beta)\, ds.$$

The fitted parameters solve

$$(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\alpha}}) \in \arg\max_{\boldsymbol{\mu}\geq 0,\, \boldsymbol{\alpha}\geq 0} \left\{ \ell_\beta(\boldsymbol{\mu}, \boldsymbol{\alpha}; \mathcal{D}) - \eta\Omega(\boldsymbol{\alpha}) \right\}.$$

The term $\Omega(\boldsymbol{\alpha})$ is a shrinkage penalty on excitation strengths, and $\eta \geq 0$ controls the penalty strength. We select among fixed decay values by likelihood subject to stability and use the best stable exponential fit in the held-out simulation. Appendix B reports fitting details.

3.3. Hawkes-Based Semantic Propagation

We now combine the Hawkes model with the text-generation loop. Let $e_0 = (\tau_0, n_0, x_0)$ be the seed event. The algorithm then produces L non-seed events $e_1, \ldots, e_L$.

Before step t, the history $\mathcal{H}_t = \{e_m = (\tau_m, n_m, x_m) : 0 \leq m < t\}$ contains the seed and all earlier generated events. Its node-time projection is $\mathcal{T}_t = \{(\tau_m, n_m) : e_m \in \mathcal{H}_t\}$. Given the fitted Hawkes parameters $(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\alpha}}, \hat{\beta})$, the conditional intensity for node i at candidate time r is evaluated from $\mathcal{T}_t$ and written as $\lambda_i(r \mid \mathcal{T}_t)$. At each step, the Hawkes process samples τ_t and n_t, constructs $\mathcal{M}_t$, and passes the resulting prompt to the LLM.

Algorithm 1 gives the full HawkesLLM simulation loop. Event sampling produces the next timestamp and node, the memory policy produces $\mathcal{M}_t$, and the final lines generate x_t and append e_t to the history.

3.3.1. Event Simulation by Thinning

The event-sampling block of Algorithm 1 samples the next timestamp and node from the fitted multivariate Hawkes process using Ogata-style thinning (Ogata, 1981). It first extracts $\mathcal{T}_t$, the event-time and node projection of $\mathcal{H}_t$. At $t = 1$, this projection contains only the seed time and seed node. The total intensity is $\Lambda_t(s) = \sum_{i\in\mathcal{N}} \lambda_i(s \mid \mathcal{T}_t)$. Starting from the previous event time, thinning proposes a waiting time from an exponential distribution with rate $\bar{\Lambda}$, where $\bar{\Lambda}$ is a local upper bound on Λ_t. A proposal time $\tilde{s}$ is accepted with probability $\Lambda_t(\tilde{s})/\bar{\Lambda}$. After acceptance, we set $\tau_t = \tilde{s}$ and sample the node $n_t = i$ with probability proportional to $\lambda_i(\tau_t \mid \mathcal{T}_t)$.

This sampling step uses only event times and nodes. The history texts $\{x_m : e_m \in \mathcal{H}_t\}$ enter only after τ_t and n_t are sampled, when the memory policy chooses predecessor texts for the LLM prompt.

3.3.2. Hawkes Memory Policy

The memory-policy block of Algorithm 1 constructs $\mathcal{M}_t$ for the sampled timestamp and node. We aggregate Hawkes contributions at the node level. The Hawkes state still uses all earlier event times through exponential decay, but the text prompt contains at most one representative text per retained node: the latest generated text from that node.

For each eligible node j, define

$$r_t(j) = \max\{m < t : n_m = j,\ \tau_m < \tau_t\},$$

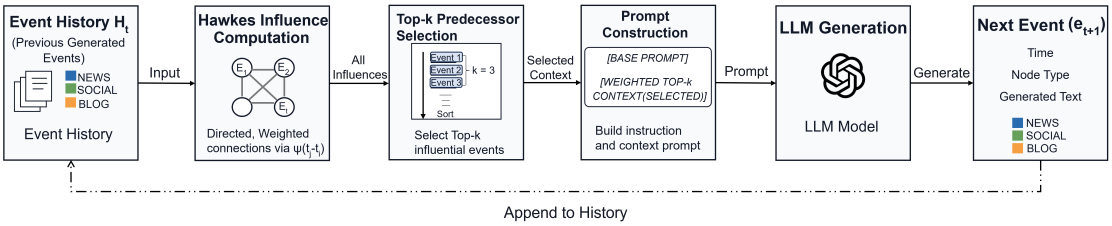

Figure 1. Sequential agentic uncertainty loop in HawkesLLM. The generated history $\mathcal{H}_t$ is the agent state; the Hawkes process selects weighted prompt memory $\mathcal{M}_t$, the LLM generates x_t, and the completed event e_t is appended to the history. Semantic alignment and local/global drift track trajectory-level uncertainty.

when such an event exists. The node-wise decayed state is

$$h_{j,t} = \sum_{m<t:\ n_m=j,\ \tau_m<\tau_t} \exp[-\hat{\beta}(\tau_t - \tau_m)].$$

The node-wise Hawkes contribution toward the current node n_t is then

$$q_{j,t} = \hat{\alpha}_{j,n_t} h_{j,t}.$$

The score is larger when node j has stronger learned excitation toward the current node and when its recent events have not decayed away.

Let

$$\mathcal{J}_t = \{j \in \mathcal{N} : (j, n_t) \in \mathcal{E},\ r_t(j) \text{ is defined}\}.$$

Let $Q_t = \sum_{\ell \in \mathcal{J}_t} q_{\ell,t}$. If $Q_t = 0$, the selected set is empty. Otherwise, define the normalized contribution $\bar{q}_{j,t} = q_{j,t}/Q_t$. The policy removes negligible nodes using a raw-score threshold ϵ_{raw} and a normalized-contribution threshold ϵ_{norm}:

$$\widetilde{\mathcal{J}}_t = \{j \in \mathcal{J}_t : q_{j,t} \geq \epsilon_{\text{raw}},\ \bar{q}_{j,t} \geq \epsilon_{\text{norm}}\}.$$

It then keeps at most k nodes with the largest remaining scores:

$$\mathcal{I}_t = \text{TopK}_k(\widetilde{\mathcal{J}}_t;\ q_{j,t}).$$

When $\mathcal{I}_t \neq \emptyset$, the prompt-memory weights are the selected scores normalized to sum to one:

$$w_{j,t} = \frac{q_{j,t}}{\sum_{\ell \in \mathcal{I}_t} q_{\ell,t}}, \qquad j \in \mathcal{I}_t.$$

The Hawkes memory policy returns

$$\mathcal{M}_t = \{(j, r_t(j), w_{j,t}) : j \in \mathcal{I}_t\}.$$

When $\mathcal{I}_t = \emptyset$, we set $\mathcal{M}_t = \emptyset$, and the prompt uses only the current event information and the style instruction for the current node. The top-k step is an engineering constraint for compact prompts; the Hawkes process itself remains the temporal influence model. The filtering thresholds are implementation parameters.

The text-update block then assembles p_t, samples x_t, and appends e_t. Thus the Hawkes process controls event timing, current node, and selected memory, while the language model verbalizes the next event.

4. Experimental Setting

4.1. Data and Event Stream

We use Global Database of Events, Language, and Tone (GDELT) article metadata (Leetaru & Schrodt, 2013). We select a recent event window for *Artemis II* coverage using the query terms `"Artemis II"` and `"Artemis 2"`. The working window spans April 1–11, 2026 UTC. GDELT timeline metadata indicate a rough scale of 10,525 matching articles over the window. The modeling dataset uses a capped sample of 250 article records. After URL/title deduplication, the final event stream contains 248 English-language events over approximately 263 hours. Tied timestamps are broken by small within-group offsets so that point-process fitting and chronological train/test splitting are well defined. The retained fields are timestamp, article domain, node, language, title, and URL.

Nodes are defined as hand-curated outlet categories, not as semantic topic labels. Table 1 reports the final five-node grouping. This grouping is a hand-curated, task-specific media taxonomy for the Artemis II window. Representative domains include local broadcast affiliates for `local_tv`, tabloid or regional commercial outlets for `mass_market`, science and space outlets for `specialist_science_tech`, finance outlets for `business_finance`, and a residual `general_news` category for the remaining domains.

Table 1. Node categories and event counts in the modeling dataset.

Node	Count
`local_tv`	69
`mass_market`	37
`specialist_science_tech`	25
`business_finance`	31
`general_news`	86

Generation setup. The text generator is Qwen2.5 run through a local Ollama backend, `ollama:qwen2.5:latest` (Bai et al., 2024), decoded with temperature 0.35, top-p 0.9, and at most 75 new

Algorithm 1 HawkesLLM semantic propagation loop

Input: seed $e_0 = (\tau_0, n_0, x_0)$, graph $\mathcal{G}_0 = (\mathcal{N}, \mathcal{E})$, budget k, length L, fitted Hawkes model $(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\alpha}}, \hat{\beta})$, thresholds $(\epsilon_{\text{raw}}, \epsilon_{\text{norm}})$, node instructions $\{a_i : i \in \mathcal{N}\}$, text generator g_{LLM}
Output: simulated history $\mathcal{H}_{L+1}$
$\mathcal{H}_1 \leftarrow \{e_0\}$
for $t = 1, \ldots, L$ **do**
  $\mathcal{T}_t \leftarrow \{(\tau_m, n_m) : e_m \in \mathcal{H}_t\}$
  $s \leftarrow \tau_{t-1}$
  Event sampling
  $\Lambda_t(r) \leftarrow \sum_{i \in \mathcal{N}} \lambda_i(r \mid \mathcal{T}_t)$
  sample τ_t by thinning from time s using Λ_t
  sample $n_t = i$ with probability $\lambda_i(\tau_t \mid \mathcal{T}_t)/\Lambda_t(\tau_t)$
  Memory policy
  $\mathcal{J}_t \leftarrow \{j \in \mathcal{N} : (j, n_t) \in \mathcal{E},\ \exists m < t \text{ with } n_m = j,\ \tau_m < \tau_t\}$
  $r_t(j) \leftarrow \max\{m < t : n_m = j,\ \tau_m < \tau_t\}$ for $j \in \mathcal{J}_t$
  $h_{j,t} \leftarrow \sum_{m<t:\ n_m=j,\ \tau_m<\tau_t} \exp[-\hat{\beta}(\tau_t - \tau_m)]$ for $j \in \mathcal{J}_t$
  $q_{j,t} \leftarrow \hat{\alpha}_{j,n_t} h_{j,t}$ for $j \in \mathcal{J}_t$
  $Q_t \leftarrow \sum_{\ell \in \mathcal{J}_t} q_{\ell,t}$
  if $Q_t = 0$ **then**
    $\mathcal{I}_t \leftarrow \emptyset$
  else
    $\bar{q}_{j,t} \leftarrow q_{j,t}/Q_t$ for $j \in \mathcal{J}_t$
    $\widetilde{\mathcal{J}}_t \leftarrow \{j \in \mathcal{J}_t : q_{j,t} \geq \epsilon_{\text{raw}},\ \bar{q}_{j,t} \geq \epsilon_{\text{norm}}\}$
    $\mathcal{I}_t \leftarrow \text{TopK}_k(\widetilde{\mathcal{J}}_t;\ q_{j,t})$
  end if
  if $\mathcal{I}_t = \emptyset$ **then**
    $\mathcal{M}_t \leftarrow \emptyset$
  else
    $w_{j,t} \leftarrow q_{j,t}/\sum_{\ell \in \mathcal{I}_t} q_{\ell,t}$ for $j \in \mathcal{I}_t$
    $\mathcal{M}_t \leftarrow \{(j, r_t(j), w_{j,t}) : j \in \mathcal{I}_t\}$
  end if
  Text update
  $p_t \leftarrow \text{Prompt}((\tau_t, n_t), a_{n_t}, \{(w_{j,t}, j, x_{r_t(j)}) : (j, r_t(j), w_{j,t}) \in \mathcal{M}_t\})$
  sample $x_t \sim g_{\text{LLM}}(\cdot \mid p_t)$
  $e_t \leftarrow (\tau_t, n_t, x_t)$, $\mathcal{H}_{t+1} \leftarrow \mathcal{H}_t \cup \{e_t\}$
end for
return $\mathcal{H}_{L+1}$

tokens. This generator is shared across node agents; the node label and node instruction determine the role being simulated at each step. Appendix C gives the remaining generation settings.

Seed event. Seed choice depends on the experimental setting. For the held-out train/test evaluation reported in Tables 2–3, each post-split simulation starts from the last training event and is evaluated over the subsequent test window, so no test-set title is used to initialize generation. For the full-data illustrative simulation and the qualitative example in Table 4, we use the earliest observed Artemis II title as the seed:

Moon rocket and weather are on NASA side for the first astronaut launch in decades.

In both settings, the seed supplies $e_0 = (\tau_0, n_0, x_0)$ and anchors the generated trajectory before the Hawkes process samples subsequent timestamp/node pairs.

4.2. Baselines

We compare HawkesLLM memory selection against two simple heuristic baselines. Both reuse the same event sequence, prompt format, text generator, and evaluation pipeline, changing only the predecessor-selection rule:

- **Chronological last-k**: the k most recent past generated events with uniform weights.
- **Random-k**: k uniformly sampled past generated events with uniform weights.

Unless otherwise stated, all methods use $k = 3$.

4.3. Train-Test Split and Matching

For the held-out evaluation, we split the 248-event dataset chronologically into train (198 events) and test (50 events). The Hawkes process is refit on train only using a stable exponential specification. Simulation is then constrained to the held-out test horizon. Appendix D gives split counts and matching details.

For each generated non-seed event, we find same-node real test events within ± 12 hours. If none exist, we relax to ± 24 hours. In total, 62 generated non-seed events are evaluated across three post-split runs. All 62 find a match, with 58 primary-window matches and 4 relaxed-window matches.

4.4. Evaluation Diagnostics

We reserve held-out texts for evaluation. Newsroom choice, outlet access, framing, and repetition across outlets all affect which article is actually written at a given time, so we evaluate local semantic agreement rather than exact continuation prediction. For a generated event $e_t = (\tau_t, n_t, x_t)$, let $\mathcal{R}_t$ be the matched set of real test-set titles from the same local region of the cascade. In this GDELT case study, those references are held-out Artemis II article titles.

When $\mathcal{R}_t \neq \emptyset$, we embed the generated text and reference texts with the evaluation embedding function $\mathbf{z}(\cdot)$. In the reported experiments, $\mathbf{z}$ is computed with the same local Ollama backend used by the saved real-vs-similarity and drift outputs, `ollama:qwen2.5:latest` (Bai et al., 2024). This makes the evaluation reproducible in the local pipeline, while keeping the metric at the level of a semantic-neighborhood diagnostic; independent embedding backends, human judgments, and factuality checks are natural extensions. Semantic alignment is the cosine similarity between the generated-text embedding and the average reference

embedding:

$$S_t = \cos\left(\mathbf{z}(x_t), \frac{1}{|\mathcal{R}_t|}\sum_{r\in\mathcal{R}_t}\mathbf{z}(r)\right),$$

where higher values indicate closer agreement with the local held-out set. If $|\mathcal{R}_t| = 1$, this is simply cosine similarity between x_t and the single matched reference text. We report mean S_t, temporal trend, and late-stage behavior, summarized by averaging S_t over the final 20% of matched simulated events within each run.

We also decompose uncertainty into global and local drift. Let x_0 be the seed text. When $\mathcal{M}_t \neq \emptyset$, let

$$\bar{\mathbf{z}}_t = \sum_{(j,r_t(j),w_{j,t})\in\mathcal{M}_t} w_{j,t}\mathbf{z}(x_{r_t(j)})$$

be the weighted predecessor centroid from the memory policy. We define

$$D_t^{\text{global}} = 1 - \cos(\mathbf{z}(x_t), \mathbf{z}(x_0)),$$
$$D_t^{\text{local}} = 1 - \cos(\mathbf{z}(x_t), \bar{\mathbf{z}}_t).$$

Global drift measures distance from the seed text. Local drift measures distance from the weighted predecessor memory that directly conditioned the current step. Local drift is undefined when no valid predecessor prompt memory exists.

5. Results

5.1. Semantic Alignment Over the Generated Trajectory

Table 2 reports held-out semantic alignment under the matched compact prompt-memory budget ($k = 3$). Late-stage S_t is the average over the final 20% of matched simulated events, as defined in Section 4.4. In this setting, HawkesLLM has the highest mean and late-stage alignment, with the largest separation near the tail of the generated trajectory.

Table 2. Held-out semantic alignment S_t under the matched compact prompt-memory budget $k = 3$. Higher values are better.

Method	Mean S_t	Trend	Late-stage S_t
HawkesLLM	**0.636**	increasing	**0.682**
Chronological last-k	0.581	decreasing	0.541
Random-k	0.621	decreasing	0.594

The trend column is computed directly from S_t over simulated time. HawkesLLM is the only method in this comparison whose semantic alignment increases over the generated trajectory. Chronological and random memory selection decrease. Figure 2 shows the corresponding trajectory-level semantic-alignment curves.

These comparisons are descriptive case-study evidence from a limited held-out sample with dependent generated events. They should be read as diagnostics for this GDELT window, not as a broad benchmark claim.

5.2. Effect of Prompt-Memory Budget

The parameter k controls prompt-memory size. For HawkesLLM, it truncates a weighted influence structure; for the baselines, it directly sets how many predecessor events are provided. Table 3 reports the k-sensitivity comparison using semantic alignment S_t.

Table 3. Prompt-memory budget sensitivity using semantic alignment S_t in a representative run. Higher is better.

Method	k	Mean S_t	S_t trend	Late-stage S_t
HawkesLLM	3	0.635	increasing	**0.703**
HawkesLLM	5	0.634	increasing	**0.703**
HawkesLLM	7	0.634	increasing	**0.703**
Chronological	3	0.578	decreasing	0.497
Chronological	5	0.556	decreasing	0.454
Chronological	7	**0.694**	flat	0.636
Random	3	0.633	decreasing	0.557
Random	5	0.597	decreasing	0.537
Random	7	0.642	decreasing	0.627

HawkesLLM changes little as k increases because most events use fewer than three meaningful weighted neighbors; extra slots rarely add useful prompt memory.

The heuristic baselines can benefit from larger prompt memory. Chronological last-k at $k = 7$ achieves higher mean semantic alignment than HawkesLLM in this representative run, but its advantage fades later in the trajectory; its late-stage S_t remains below that of HawkesLLM. The takeaway is narrower but useful: HawkesLLM is strongest when prompt memory is compact and late-stage behavior matters. A broader comparison should aggregate this k-sensitivity over repeated runs and match methods by token budget as well as by item count.

5.3. Local and Global Components of Sequential Uncertainty

Across repeated simulations, the same drift pattern appears. In the post-split setting, the average mean global drift is 0.450 ± 0.019, while the average mean local drift is 0.185 ± 0.072. Global drift exceeds local drift in all runs.

These runs show generated cascades accumulating global uncertainty while staying locally stable: the trajectory can remain close to its immediate weighted prompt memory and still move away from the seed over longer horizons. Individual runs need not be monotone; the repeated post-split runs consistently show global drift above local drift. Figure 3 illustrates this separation over a representative HawkesLLM run.

Figure 4 breaks the drift diagnostic down by node. The

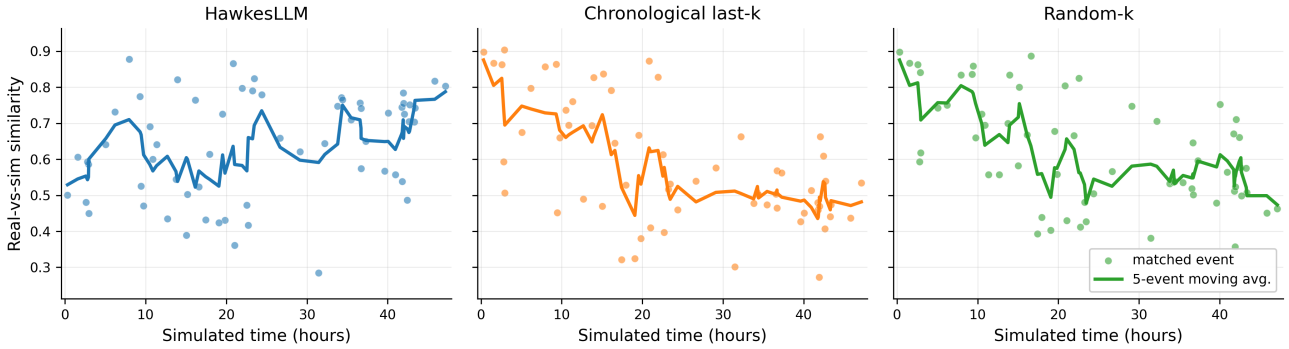

Figure 2. Sequential semantic alignment S_t over simulated time for HawkesLLM, chronological last-k, and random-k on the held-out test window. Panels show matched generated events and a 5-event moving average; higher values indicate closer alignment to the local held-out reference set. The seed event comes from the training window, so the plotted curves begin with matched generated events in the test window.

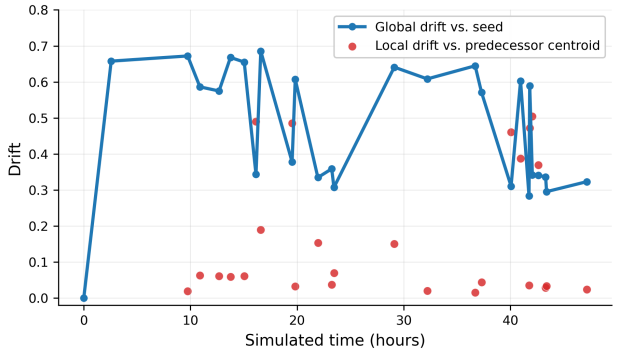

Figure 3. Global and local drift over a representative HawkesLLM run. Global drift is measured relative to the seed text, while local drift is measured relative to the weighted prompt memory.

global/local separation is visible across most node categories.

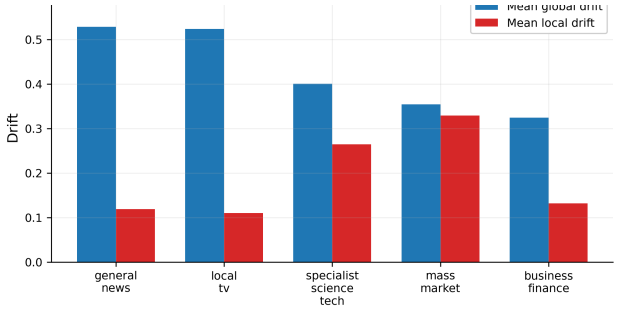

Figure 4. Node-conditioned mean global and local drift across HawkesLLM simulations. Drift varies by node.

5.4. Qualitative Propagation Example

Table 4 shows an illustrative trajectory generated by HawkesLLM. Rather than listing every event, we select representative events that show how the generated cascade changes framing across source types. The trajectory begins with a general-news seed about favorable launch conditions, then shifts toward technical mission-status language, business implications, local community framing, and eventually more public-facing launch narratives. This example is qualitative only, but it illustrates the behavior measured by the quantitative diagnostics: the sequence stays within the Artemis II semantic region while its emphasis changes across nodes and over time.

Table 4. Illustrative HawkesLLM-generated cascade. We show selected representative events from one run to highlight how source type changes the framing of the same evolving Artemis II topic.

Time	Node	Generated text
0.0	general	Moon rocket and weather are on NASA side for the first astronaut launch in decades.
0.4	general	NASA sets course for space with Artemis II, as favorable conditions align for historic crewed mission.
3.2	sci./tech	Initial data from the Artemis II mission indicates stable performance across all systems, with ongoing analysis of trajectory adjustments and spacecraft health metrics.
8.2	finance	Artemis II Mission Readies for Launch, Signaling Major Advancements in Space Exploration Ventures.
11.6	mass	NASA readies its crew for the groundbreaking Artemis II mission, signaling a major leap in space exploration under favorable conditions.
13.1	local TV	Local astronauts are getting ready for their big journey as NASA gears up for the Artemis II mission, marking an exciting step forward in space travel.
23.1	mass	NASA's Artemis II mission gets set to blast off, signaling a huge leap forward in our quest to explore the cosmos under perfect conditions.
70.3	sci./tech	Engineers at Kennedy Space Center are finalizing pre-launch procedures for Artemis II, heralding a significant step forward in NASA's lunar exploration mission.

6. Conclusion

This paper introduced HawkesLLM, a framework for monitoring semantic uncertainty propagation in iterative text-simulation systems. The Hawkes process exposes node-wise influence scores that determine which generated memories

enter future prompts, so uncertainty can be tracked over the full trajectory. In the GDELT case study, HawkesLLM yields stronger late-stage semantic alignment under a compact prompt-memory budget. Local/global drift diagnostics show how uncertainty can accumulate globally even when individual steps remain locally stable.

The gain is specific rather than sweeping. The Qwen generator already has a strong prior for plausible Artemis II text; HawkesLLM adds structure around which node memories enter the prompt. Its role is to make node-wise information flow and uncertainty monitoring explicit around the generator. Although we study news-style event propagation, the same formulation can be used for social-media narratives, multi-step agent interactions, and iterative workflows where earlier generated text becomes later context.

The study is diagnostic. It uses a sampled GDELT article list, title-level text, a hand-curated node taxonomy, and a held-out test set whose `specialist_science_tech` coverage is sparse. The generator can also introduce artifacts, including occasional mixed-language outputs. Because the reported embedding metric uses the same Qwen/Ollama model family as the generator, the next evaluation step is to pair these diagnostics with independent embedding backends, human judgments, and factuality checks. Richer event texts, additional domains, and calibrated threshold rules would help determine when semantic monitoring is reliable enough for operational use.

Impact Statement

This paper presents a research baseline for studying semantic uncertainty in multi-step text simulation. The same machinery that makes plausible news-style trajectories useful for analysis also creates misuse risks. We use the framework here as a diagnostic research instrument; deployment would require content safeguards, provenance controls, and separate reliability validation.

References

Bai, J. et al. Qwen2.5: We're all flying together. *arXiv preprint arXiv:2412.15115*, 2024.

Farquhar, S., Kossen, J., Kuhn, L., and Gal, Y. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024. doi: 10.1038/s41586-024-07421-0.

Han, J., Buntine, W., and Shareghi, E. Towards uncertainty-aware language agent. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 6662–6685. Association for Computational Linguistics, 2024.

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

Kempe, D., Kleinberg, J., and Tardos, É. Maximizing the spread of influence through a social network. *Theory of Computing*, 11(4):105–147, 2015.

Kleinberg, J. Cascading behavior in networks: Algorithmic and economic issues. In *Algorithmic Game Theory*, pp. 613–632. Cambridge University Press, 2007.

Leetaru, K. and Schrodt, P. A. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA Annual Convention*, pp. 1–49, 2013.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuttler, H., Lewis, M., Yih, W.-t., Rocktaschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020.

Li, Y., Lu, Z., Tang, F., Lai, S., Hu, M., Zhang, Y., Xue, H., Wu, Z., Razzak, I., Li, Q., and Su, J. Rhythm of opinion: A hawkes-graph framework for dynamic propagation analysis. *arXiv preprint arXiv:2502.04567*, 2025.

Lin, Z., Trivedi, S., and Sun, J. Generating with confidence: Uncertainty quantification for black-box large language models. *arXiv preprint arXiv:2305.19187*, 2023.

Liu, H., Dou, Z.-Y., Wang, Y., Peng, N., and Yue, Y. Uncertainty calibration for tool-using language agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 16781–16805. Association for Computational Linguistics, 2024a.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024b. doi: 10.1162/tacl_a_00638.

Liu, Z. and He, Y. Tpp-llm: Modeling temporal point processes by efficiently fine-tuning large language models. *arXiv preprint arXiv:2502.10867*, 2025.

Mei, H. and Eisner, J. M. The neural hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems*, volume 30, pp. 6754–6764, 2017.

Mohamed, A., Geng, M., Vazirgiannis, M., and Shang, G. LLM as a broken telephone: Iterative generation distorts information. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7493–7509. Association for Computational Linguistics, 2025.

Murdock, I., Carley, K. M., and Yagan, O. An agent-based model of reddit interactions and moderation. In *Proceedings of the 2023 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 195–202, 2023.

Ogata, Y. On lewis' simulation method for point processes. *IEEE Transactions on Information Theory*, 27(1):23–31, 1981.

Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23. ACM, 2023. doi: 10.1145/3586183.3606763.

Rizoiu, M.-A., Lee, Y., Mishra, S., and Xie, L. A tutorial on hawkes processes for events in social media. In *Frontiers of Multimedia Research*, pp. 191–218. ACM, 2017.

Spataru, A., Hambro, E., Voita, E., and Cancedda, N. Know when to stop: A study of semantic drift in text generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3656–3671. Association for Computational Linguistics, 2024.

Sun, G., Wang, Y., Niyato, D., Wang, J., Wang, X., Poor, H. V., and Letaief, K. B. Decoding echo chambers: Llm-powered simulations revealing polarization in social networks. *arXiv preprint arXiv:2408.05123*, 2024.

Zhang, L., Hu, Y., Li, W., Bai, Q., and Nand, P. Llm-aidsim: Llm-enhanced agent-based influence diffusion simulation in social networks. *Systems*, 13(1):29, 2025.

Zhao, Q., Li, D., Liu, Y., Cheng, W., Sun, Y., Oishi, M., Osaki, T., Matsuda, K., Yao, H., Zhao, C., Chen, H., and Zhao, X. Uncertainty propagation on LLM agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6064–6073. Association for Computational Linguistics, 2025.

Zhou, F., Kong, Q., Qiao, J., Wan, C., Zhang, Y., and Cai, R. Advances in temporal point processes: Bayesian, neural, and llm approaches. *arXiv preprint arXiv:2502.17528*, 2025.

Zuo, S., Jiang, H., Li, Z., Zhao, T., and Zha, H. Transformer hawkes process. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11692–11702. PMLR, 2020.

A. Instantiated Prompt Example

The prompt format is fixed across methods. HawkesLLM changes only which node-labeled predecessor texts and weights enter the memory block. The example below shows one instantiated prompt. The weights appear as text annotations, so they provide relative-importance cues rather than hard numerical controls inside the language model.

Prompt: Instantiated Example

Role. You are simulating one item in a Hawkes-driven cross-node news cascade.

Task. Write exactly one concise English sentence or headline-like update.

Constraints. Use only the weighted predecessor texts below; do not mention weights, simulations, models, or prompts. Do not copy predecessor wording verbatim. Preserve the core Artemis II subject while allowing natural semantic drift.

Target node. `local_tv`

Node style. Write like a local TV news web update: clear, public-facing, practical, and locally relatable.

Simulated time since seed. 15.04 hours

Weighted predecessor context.

1. `weight=0.81; node=local_tv;` text=*Artemis II crew making great strides in their preparations for the mission, according to latest updates from NASA. Local teams are keeping a close eye on their progress and wish them all the best for this historic journey.*
2. `weight=0.19; node=general_news;` text=*Crew aboard Artemis II continues to prepare for historic space mission, overcoming initial hurdles as NASA reports smooth progress.*

Output. Only the generated news item.

B. Hawkes Fitting Details

The full-data exponential grid included eight fixed decay values from 1/72 to 1/6 per hour. The best-overall exponential fit had $\beta = 0.1667$ and $\rho(\mathbf{G}) = 1.0535$, so it was not used for simulation. The best stable exponential fit had $\beta = 0.0833$, $\rho(\mathbf{G}) = 0.8708$, log-likelihood -556.821, AIC 1175.641, and BIC 1284.558. The Gaussian-basis comparison used truncated normal bases centered at 6, 24, and 72 hours with standard deviations 4, 10, and 20 hours, respectively; it achieved $\rho(\mathbf{G}) = 0.6836$ and log-likelihood -569.693.

For the chronological 80/20 train/test evaluation, Hawkes was refit on train only. The resulting stable exponential model has $\beta = 0.1667$ and $\rho(\mathbf{G}) = 0.8467$. That train-only refit was used for the held-out uncertainty evaluation.

C. Generation Settings

Table 5 reports the decoding and prompt-memory settings used in the simulations.

Table 5. Generation and prompt-memory settings.

Setting	Value
Generator	Qwen2.5 via Ollama
Temperature	0.35
Top-p	0.9
Max new tokens	75
Main prompt budget k	3
Normalized threshold ϵ_{norm}	0.03
Raw threshold ϵ_{raw}	10^{-5}
Event cap	40

The held-out simulations start from the last training event and then run over the test window. The full-data exploratory run used the earliest observed Artemis II title as its seed:

> Moon rocket and weather are on NASA side for the first astronaut launch in decades.

D. Held-Out Evaluation Details

The chronological 80/20 split yields 198 train events and 50 test events. Table 6 reports the per-node counts.

Table 6. Per-node train/test counts under the chronological split.

Node	Train	Test
`local_tv`	51	18
`mass_market`	22	15
`specialist_science_tech`	22	3
`business_finance`	26	5
`general_news`	77	9

Across three post-split runs, 62 generated non-seed events were evaluated. All 62 found same-node matches, with 58 within ± 12 hours and 4 using the ± 24-hour fallback. The similarity curves begin at the first matched generated test-window event, not at the training-side seed.

Additional drift summaries. In full-data repeated runs, the same global/local separation appears: average mean global drift is 0.440 ± 0.100, and average mean local drift is 0.153 ± 0.049.