EventSearch: Supportive Event Retrieval on Hypergraph for Event Reasoning

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) has greatly improved the ability of large language models to ground responses in external knowledge, yet complex event reasoning remains challenging because it requires retrieving *supportive* events. Methods based on textual similarity or entity relevance miss such events due to knowledge-base incompleteness and the nuanced nature of events. We introduce `EventSearch`, an event-oriented RAG framework for event reasoning. We first propose supportive event retrieval by building an event hypergraph index over the knowledge base. We then treat event supportiveness as random variables and maximize its expectation via a Maximum Expected Event Cover program, for which we derive a theoretical guarantee. Finally, `EventSearch` couples retrieval with reasoning in an event-oriented agentic process that fetches the information needed for complex reasoning. Experiments and in-depth analysis show that `EventSearch` outperforms competitive RAG baselines on event reasoning.

1 Introduction

Retrieval-Augmented Generation (RAG) has emerged as a transformative paradigm in NLP and web-scale information systems, addressing key limitations of large language models (LLMs) such as the static knowledge cutoff inherited from pretraining (Gupta et al., 2024). By coupling retrieval with generation, RAG can dynamically access heterogeneous, frequently updated web content, such as news streams, social media posts, and knowledge graph triples. Then they answer with improved accuracy and relevance by complex reasoning (Zhao et al., 2024). Given a query, state-of-the-art RAG retrieves pertinent web documents through: (1) improved query–source textual similarity via query enrichment (Gao et al., 2023) and retrieval-intent understanding (Oh et al., 2024); (2) entity-centric structured indexing via hierarchical trees (Sarthi et al., 2024a) and graphs (Edge et al., 2024) that capture interconnections among web entities. These advances yield strong performance across web-facing domains including finance (Li et al., 2023; Islam et al., 2023), healthcare (Dou et al., 2024), and law (Fei et al., 2023).

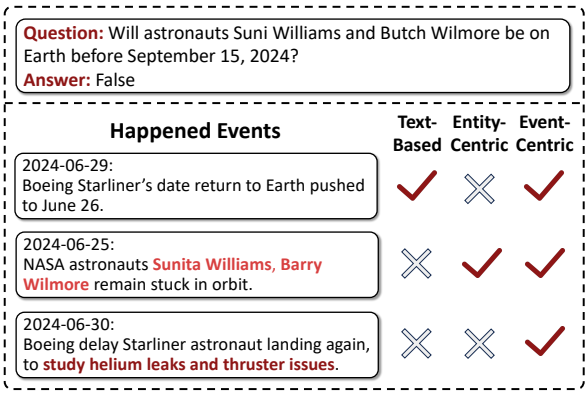


Figure 1: An event prediction example of event reasoning tasks in the Web context. Different RAG methodologies exhibit varying abilities in retrieving supportive events. `EventSearch` is able to retrieve relevant supportive events without explicit entity linkages.

While existing RAG pipelines work well on static or homogeneous knowledge, reasoning over complex, dynamic web events remains open. Event reasoning covers tasks that predict future developments or abductively infer the most plausible causes of observed occurrences, typical in web-scale event prediction (Tao et al., 2025) and multi-source fact verification (Thorne et al., 2018). Such tasks require retrieving not merely semantically relevant but *supportive* past events from noisy, partially observed web corpora. We define an event's supportiveness for a query as: **providing evidence or reasoning pathways that align with the causal chains and temporal dependencies relevant to answering the query in its information context**. Relevance and supportiveness are distinct: an event that is surface-similar to the query may be non-supportive (e.g., sharing keywords but unrelated to

the underlying causal mechanism), while a seemingly distant one can be highly supportive if it bridges critical causal or temporal gaps. The retrieval objective therefore shifts from "what looks like the query" to "what helps answer the query."

Identifying highly supportive events at web scale poses several difficulties. First, real-world web events are only partially observable: news articles or posts may omit intermediate causal links, or contain implicit references not connected in structured knowledge bases. As a result, even the supportiveness of a known event is uncertain and must be estimated rather than read off directly. Second, events are mutually influential through complex patterns of social propagation, media coverage, and entity co-occurrence that entity graphs struggle to capture, since a single event typically involves multiple participants and actions whose joint interaction cannot be decomposed into pairwise links. Text-centric RAG emphasizes keyword similarity (Gao et al., 2023), while structure-oriented RAG focuses on explicit entity relations (Edge et al., 2024). As a result, conventional retrieval may (a) return events with lexical similarity but little causal utility, or (b) miss lexically distant but supportive web events critical for reasoning. As Figure 1 illustrates, solving the web-based event prediction example requires bridging heterogeneous event streams: text-based retrieval finds the first event via surface similarity; entity-centric methods uncover the second through shared actors ("Sunita Williams" and "Barry Wilmore"); yet locating the third demands deeper causal insight to detect its latent supportiveness for the complex query.

To address this, we present `EventSearch`, an *event-centric* RAG framework for complex event reasoning over large-scale, dynamic web sources. Our contribution is twofold. **Supportive Event Retrieval:** We build an *event hypergraph index* spanning heterogeneous sources in the knowledge base. Events extracted from web documents become hyperedges over nodes denoting participants and actions, naturally capturing the multi-entity, multi-event interdependencies common on the web. To retrieve supportive events from this structure, we model event supportiveness as random variables and maximize its expectation via the *Maximum Expected Event Cover*, an optimization that jointly maximizes probabilistic supportiveness and structural connectivity. We further prove this formulation yields an optimal solution under partial observability, matching the incomplete, noisy nature of web event data. **Event-Oriented Agentic Reasoning:** We integrate supportive event retrieval into a multi-step agentic reasoning pipeline that adapts iteratively: the retrieval agent selectively searches new web sources while the reasoning agent updates its hypothesis on the evolving evidence set. This enables dynamic exploration of the web's information landscape, refining toward the answer.

We evaluate `EventSearch` on PROPHET (Tao et al., 2025) (web-scale event prediction) and Fever (Thorne et al., 2018) (multi-source fact verification), two typical web-based event reasoning benchmarks. Against strong text- and graph-oriented RAG baselines, `EventSearch` achieves significant gains, underscoring the need for event-oriented designs in web event retrieval. Our main contributions are:

- We propose `EventSearch`, an event-oriented RAG architecture that retrieves supportive events via hypergraph indexing and solves the Maximum Expected Event Cover with theoretical guarantees.
- We integrate retrieval and reasoning in a multi-step agentic RAG pipeline that iteratively refines evidence and answers through adaptive exploration of supportive events.
- Extensive experiments on PROPHET and Fever show `EventSearch`'s superior performance on event reasoning over noisy, incomplete, large-scale web event data.

2 Preliminaries

2.1 Event Representation

In NLP and knowledge representation, an event e is typically defined as a structured occurrence (Dölling et al., 2013), comprising participants (or arguments) $p \in \mathbb{P}$ that represent the involved entities, and one action a identified by a lexical trigger word (Doddington et al., 2004).

In our implementation, events may include multiple actions $a \in \mathbb{A}$ to accommodate coarser-grained definitions where a single trigger inadequately captures the scope (e.g., "protest" and "arrest" in a civil-unrest event). Since structural representations risk omitting contextual nuances, we augment them with a textual description $\mathcal{X}$, yielding $e = (\mathbb{P}, \mathbb{A}, \mathcal{X})$.

2.2 Event Reasoning

Event reasoning encompasses tasks that infer cause-and-effect relationships between events from historical web data, such as event prediction (Tao et al.,

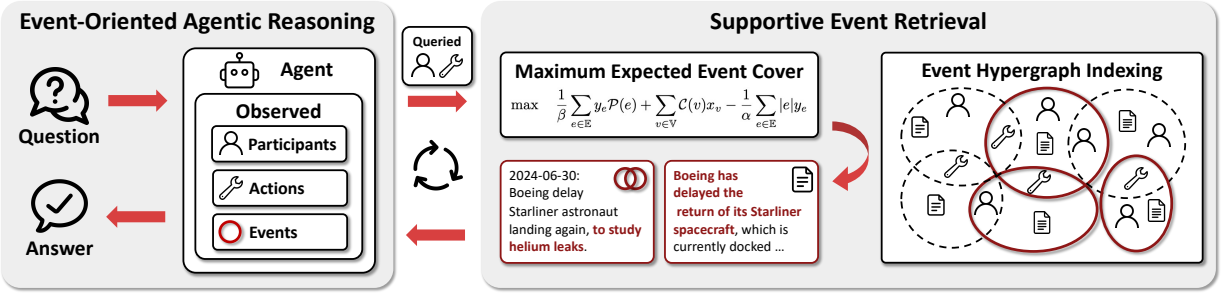


Figure 2: Overview of `EventSearch`. Given a question, our event-oriented agentic reasoning answers it in a multi-step process. In each step, it first retrieves supportive events and documents via our supportive event retrieval according to the current observed states. Then it decides to answer or update states for the next step.

2025) and fact verification (Thorne et al., 2018). They typically analyze large volumes of temporally or causally ordered information from news articles, social media posts, or other archives.

We focus on event reasoning over web news articles. For reproducibility and to avoid variability from dynamic online retrieval, we adopt an offline setting where all documents are pre-collected and processed before experimentation, while still preserving the temporal context of events, yielding an RAG setting. Given an event reasoning question $\mathcal{Q}$ and a document set $\mathbb{D}$ collecting related web information, it retrieves supportive events and documents from $\mathbb{D}$ and answers $\mathcal{Y}$:

$$\mathcal{Y} = \text{Reason}(\mathcal{Q}, \mathbb{D}). \tag{1}$$

3 Method

Our `EventSearch` framework tackles event reasoning as shown in Figure 2. It indexes the knowledge base as an event hypergraph, casts event retrieval as a maximum expected event cover program, and integrates retrieval with iterative reasoning via an agentic process toward the answer.

3.1 Supportive Event Retrieval

Event reasoning requires retrieving not just relevant but *supportive* events, those providing direct evidence or rationale for answering a question. Relevance does not imply support, and vice versa: a textually similar event may rest on unrelated causal chains, while a structurally distant event can carry the decisive evidence. Retrieving highly supportive events is hard for two reasons. First, supportiveness is only *partially observable*: whether an event truly bears on the answer is rarely written down and must be inferred. Second, event relations are *higher-order*: a single event ties together multiple participants and actions whose joint interaction cannot be faithfully decomposed into pairwise edges. Text-based RAG (query–document similarity (Ma et al., 2024)) and structure-based RAG (entity relations (Edge et al., 2024)) cope with neither difficulty and often return only superficially relevant events, missing deeper supportiveness.

To address this, `EventSearch` builds an event hypergraph index of the knowledge base. Events are extracted from documents and represented as hyperedges, with nodes for participants and actions, which preserves the multi-way semantics of each event in a single edge rather than scattering it across pairwise links. We then introduce the maximum expected event cover program, formulating retrieval as an optimization that jointly maximizes structural connectivity and probabilistic supportiveness, two signals that are individually insufficient but complementary. We also prove this program bounds optimal supportive retrieval.

To construct the index, we first extract events from the knowledge base $\mathbb{D}$. `EventSearch` begins with event extraction in each $\mathcal{D} \in \mathbb{D}$ and builds the event hypergraph. Directly extracting events causes entity coreference issues, the same entity may surface under different wording. We first extract named entities and mine events from them.

Entity Extraction Given $\mathcal{D} \in \mathbb{D}$, we prompt LLMs to extract named entities (the later event participants); see Appendix prompt (a) for the prompt and few-shot examples.

Event Extraction With participants in hand, we extract events via LLMs (Appendix prompt (b)). Each event is described in sentences (giving $\mathcal{X}$), with its date and location; few-shot demonstrations are provided for typing. We resolve event coreference by merging events with heavily overlapping participants and actions, concatenating their descriptions into $\mathcal{X}$. This yields the structured event.

Hypergraph Indexing We build the event hypergraph index from the extracted events. A hypergraph generalizes a graph so that a hyperedge, representing an event, can connect any number of vertices rather than just pairs. Formally, $\mathcal{H} = (\mathbb{V}, \mathbb{E})$. The node set $\mathbb{V} = \{v_1^{\mathcal{P}}, \ldots, v_n^{\mathcal{P}}, v_1^{\mathcal{A}}, \ldots, v_m^{\mathcal{A}}\}$ holds all participants and actions, a finite vertex set of $\mathcal{H}$[1]. $\mathbb{E} = \{e_1, \ldots, e_l\}$ is a family of non-empty subsets of $\mathbb{V}$, each a hyperedge that is also an event[2]. Each $e \in \mathbb{E}$ corresponds to an event $e = (\mathbb{P}, \mathbb{A}, \mathcal{X}, \mathcal{D})$, where $\mathbb{P}$, $\mathbb{A}$, and $\mathcal{X}$ are its participants, actions, and description, and $\mathcal{D}$ is its source document. Each e thus indexes its source $\mathcal{D}$. Unlike structure-based RAG indexing such as GraphRAG, hyperedges in $\mathcal{H}$ can span two or more vertices, enabling retrieval of higher-order event.

3.1.1 Maximum Expected Event Cover

We now elaborate on retrieval over the constructed event hypergraph. The retrieval input is the queried participants $\tilde{\mathbb{P}}$ and actions $\tilde{\mathbb{A}}$, supplied by our agent (Section 3.2). The output is the query-relevant and question-supportive events and their documents:

$$(\tilde{\mathbb{E}}, \tilde{\mathbb{D}}) = \mathrm{Retrieve}(\mathcal{H}, \tilde{\mathbb{P}}, \tilde{\mathbb{A}}). \quad (2)$$

$\tilde{\mathbb{E}}$ is the retrieved events and $\tilde{\mathbb{D}}$ the documents indexed by $\tilde{\mathbb{E}}$. We first link $\tilde{\mathbb{P}}$ and $\tilde{\mathbb{A}}$ to nodes $\tilde{\mathbb{V}} \subseteq \mathbb{V}$. Retrieval is then cast as selecting a supportive event set $\tilde{\mathbb{E}} \subseteq \mathbb{E}$. We describe its objective below.

To select supportive $\tilde{\mathbb{E}}$, we first need a measure of event supportiveness on $\mathcal{H}$. A naive option is to elicit a binary supportiveness label per event, but because real-world events are only partially observed, the exact label is unobtainable: the same event may look supportive under one causal hypothesis and irrelevant under another, and a deterministic label would collapse this ambiguity. We therefore use a Bernoulli random variable $X_e \sim \mathrm{Bernoulli}(\mathcal{P}(e))$ to indicate whether e supports answering the question, with $\mathcal{P}(e) = \Pr[X_e = 1]$ estimated by an LLM. Treating supportiveness as a random variable preserves uncertainty and lets us later reason about it in expectation. Pretrained on broad semantic and reasoning patterns, LLMs can assess an event against the query to gauge its potential usefulness; alternatives such as causal inference (Bynum and Cho, 2024) could also be used. We later show that more accurate $\mathcal{P}(e)$ yields a tighter upper bound and higher performance. A document's supportive probability is taken to be that of its event e, since each e indexes its source $\mathcal{D}$. The prompt is in the Appendix. Retrieval is thus equivalent to selecting $\tilde{\mathbb{E}}$ that maximizes the total supportive probability.

Probabilistic supportiveness alone is not enough: an LLM's estimate is noisy and may favor events that are topically plausible but disconnected from the query's actual entities and actions. We therefore also account for topological relevance of events on $\mathcal{H}$, requiring $\tilde{\mathbb{E}}$ to be structurally tied to the query nodes $\tilde{\mathbb{V}}$. We assign each node v a relevance score $\mathcal{C}(v)$ via Personal PageRank (Yang and Wang, 2024)[3], which propagates importance from $\tilde{\mathbb{V}}$ along the hyperedges so that nodes participating in events near the query receive higher scores; implementation details are in the Appendix. The selected $\tilde{\mathbb{E}}$ should cover the nodes with the highest total $\mathcal{C}(v)$, anchoring retrieval to the query's structural neighborhood. Finally, with only the two terms above, the program could trivially select *all* events to inflate the total supportive probability and the covered relevance. To prevent this shortcut, we add an event-count regularizer that penalizes the size of each chosen event, encouraging compact, decisive evidence rather than indiscriminate coverage. We define the supportiveness measure of $\tilde{\mathbb{E}}$ as:

$$f^{\tilde{\mathbb{E}}} = \frac{1}{\beta} \sum_{e \in \tilde{\mathbb{E}}} X_e + \sum_{v \in \bigcup_{e \in \tilde{\mathbb{E}}} e} \mathcal{C}(v) - \frac{1}{\alpha} \sum_{e \in \tilde{\mathbb{E}}} |e|. \quad (3)$$

α and β are hyper-parameters and $|e|$ is the number of nodes in e. $f^{\tilde{\mathbb{E}}}$ is a random variable for any $\tilde{\mathbb{E}}$, dominated by the first term; the second is topological relevance and the third is an event-count regularizer. Only the first is a sum of random variables; others are constants. Its expectation is:

$$Ef^{\tilde{\mathbb{E}}} = \frac{1}{\beta} \sum_{e \in \tilde{\mathbb{E}}} \mathcal{P}(e) + \sum_{v \in \bigcup_{e \in \tilde{\mathbb{E}}} e} \mathcal{C}(v) - \frac{1}{\alpha} \sum_{e \in \tilde{\mathbb{E}}} |e|. \quad (4)$$

We select $\tilde{\mathbb{E}}$ to maximize $Ef^{\tilde{\mathbb{E}}}$ via an integer program (IP), which encodes "select a subset of hyperedges" and "track which nodes are covered" as binary decisions. For $e \in \mathbb{E}$, the binary y_e indicates whether e is selected; for $v \in \mathbb{V}$, the binary x_v indicates whether v is covered. For any $\tilde{\mathbb{E}}$: $x_v^{\tilde{\mathbb{E}}} = 1$ iff v is covered by $\tilde{\mathbb{E}}$; $y_e^{\tilde{\mathbb{E}}} = 1$ iff $e \in \tilde{\mathbb{E}}$. Then Eq. (4) rewrites as:

$$Ef^{\tilde{\mathbb{E}}} = \frac{1}{\beta} \sum_{e \in \mathbb{E}} y_e^{\tilde{\mathbb{E}}} \mathcal{P}(e) + \sum_{v \in \mathbb{V}} \mathcal{C}(v) x_v^{\tilde{\mathbb{E}}} - \frac{1}{\alpha} \sum_{e \in \mathbb{E}} |e| y_e^{\tilde{\mathbb{E}}} \quad (5)$$

[1] We use v without superscripts when not distinguishing participants from actions.

[2] We use e for both hyperedge and event; $\mathbb{E}$ is their set.

[3] We use NetworkX's PageRank API with default parameters.

Thus, retrieving $\tilde{\mathbb{E}}$ reduces to solving the following IP, where the constraint links x_v to y_e: a node v is counted as covered only if at least one event containing it is selected.

$$\begin{aligned} \max \quad & \frac{1}{\beta}\sum_{e\in\mathbb{E}} y_e\mathcal{P}(e) + \sum_{v\in\mathbb{V}}\mathcal{C}(v)x_v - \frac{1}{\alpha}\sum_{e\in\mathbb{E}}|e|y_e \\ \text{s.t.} \quad & x_v - \sum_{e\in\mathbb{E}, v\in e} y_e \le 0, \forall v\in\mathbb{V}; \\ & x_v\in\{0,1\}, \forall v\in\mathbb{V}; y_e\in\{0,1\}, \forall e\in\mathbb{E}. \end{aligned} \tag{6}$$

3.1.2 Theoretical Guarantee

A natural worry is that, because $\mathcal{P}(e)$ is only an estimate, the IP could in principle select a substantially worse event set than the unknown ground-truth optimum. We therefore give a theoretical guarantee for IP (6): an upper bound on the expected gap between its solution and the omniscient optimum, the optimum we would obtain if the true supportiveness of every event were observable. Eq. (3) is the supportiveness measure, a random variable induced by X_e. For a definite value $s = (s_e)_{e\in\mathbb{E}}$ of the joint variable $X = (X_e)_{e\in\mathbb{E}}$, i.e., one possible realization of which events are truly supportive, Eq. (3) reduces to:

$$\frac{1}{\beta}\sum_{e\in\tilde{\mathbb{E}}} s_e + \sum_{v\in\bigcup_{e\in\tilde{\mathbb{E}}} e}\mathcal{C}(v) - \frac{1}{\alpha}\sum_{e\in\tilde{\mathbb{E}}}|e|, \tag{7}$$

for any retrieved event set $\tilde{\mathbb{E}}$. Let $\tilde{\mathbb{E}}^s \subseteq \mathbb{E}$ maximize Eq. (7) under s; this is the best retrieval an omniscient observer would make given that realization. Define $\text{OPT}(s)$ as the value of Eq. (7) at s with optimal $\tilde{\mathbb{E}}^s$. $\text{OPT}(\cdot)$ maps any s to a real number; since all s form the value space of X, $\text{OPT}(\cdot)$ is also a function on X, giving a random variable $\text{OPT}(X)$.

For each s, $\text{OPT}(s)$ is attained by the optimal $\tilde{\mathbb{E}}^s$, so $\text{OPT}(X)$ is the omniscient optimum of the supportiveness measure, the best score achievable if one could see the true realization of every X_e. The following theorem upper-bounds the expectation gap between $\text{OPT}(X)$ and $f^{\tilde{\mathbb{E}}}$ with $\tilde{\mathbb{E}}$ chosen by IP (6). In other words, even though our IP operates with only the LLM-estimated probabilities, its retrieval cannot deviate from the omniscient choice by more than this amount in expectation.

Theorem 1. *$E(OPT(X) - f^{\tilde{\mathbb{E}}}) \le \frac{2}{\beta}\sum_{e\in\mathbb{E}}(\mathcal{P}(e) - \mathcal{P}^2(e))$, where $\tilde{\mathbb{E}}$ is chosen by IP in Eq. (6).*

The proof is in the Appendix. The bound's term $\sum_{e\in\mathbb{E}}(\mathcal{P}(e) - \mathcal{P}^2(e))$ is a quadratic in the probability: the more deterministic the supportiveness judgment (e.g., all $\mathcal{P}(e) \in \{0, 1\}$), the tighter the bound, eventually reaching 0; if all $\mathcal{P}(e) = 0.5$, the bound peaks, indicating maximum uncertainty. Hence a more precise estimate of supportiveness gives a tighter bound, leading to retrievals closer to the optimal supportive events. This matches our empirical observation that stronger supportiveness estimators (better LLMs or more informative prompts) translate directly into stronger downstream reasoning, since they shrink the uncertainty.

3.2 Event-Oriented Agentic Reasoning

We now describe our integrated event-oriented agentic reasoning. The retrieval above takes queried entities and actions, but event reasoning typically cannot be solved in a single retrieval: answering one question often hinges on intermediate facts whose participants and actions are unknown until earlier evidence is read. A static one-shot retrieval, no matter how accurate, can only respond to what the question explicitly names. We therefore use a multi-step agentic RAG process (Figure 2 and the Appendix algorithm) that lets the LLM iteratively decide what to look up next based on what it has already observed.

The agent maintains the following states:

- Reasoning History $\dot{\mathbb{R}}$: previous reasoning thoughts $\mathcal{R}$, initialized empty.
- Observed Events $\dot{\mathbb{E}}$: retrieved events (initialized empty), with their source documents $\dot{\mathbb{D}}$.
- Observed Participants $\dot{\mathbb{P}}$ and Actions $\dot{\mathbb{A}}$: those observed via retrieval, initialized from the question.

At each step, we feed the current $\dot{\mathbb{R}}$, $\dot{\mathbb{E}}$, and $\dot{\mathbb{P}}$ to the agent and obtain $\mathcal{K} = \text{M}(\mathbb{R}, \dot{\mathbb{E}}, \dot{\mathbb{P}})$, where M is the agent. $\mathcal{K}$ indicates whether to answer or to keep retrieving. This decision is the key advantage of the agentic loop: the LLM, rather than a fixed budget, decides when the evidence so far is sufficient. If answering, the agent uses $\dot{\mathbb{R}}$, $\dot{\mathbb{E}}$, $\dot{\mathbb{P}}$, and $\dot{\mathbb{A}}$ (all prompts in the Appendix).

If retrieving, we parse new $\tilde{\mathbb{P}}$ and $\tilde{\mathbb{S}}$ from $\mathcal{K}$; the agent is conditioned on the already observed $\dot{\mathbb{P}}$, $\dot{\mathbb{A}}$ to avoid re-issuing previously made queries and to focus on the gaps in the current evidence set. At each step the agent also generates a thought motivating the current retrieval, then retrieves events via Eq. 2, and we update $\dot{\mathbb{E}}$ accordingly. The participants and actions of the new events update $\dot{\mathbb{P}}$ and $\dot{\mathbb{A}}$, expanding the agent's working knowledge of the relevant entities and possible follow-up queries.

Method	# $\mathcal{D}$	Brier Score ↓	
		GPT-4O	*Gemini-Pro*
Zero-Shot			
ScratchPAD (w.o. RAG) (Halawi et al., 2024)	-	25.42 ± 0.09	25.39 ± 0.41
Text-Based RAG			
Naive RAG	10.00	21.22 ± 0.30 (+4.20)	22.18 ± 0.39 (+3.21)
	20.00	20.31 ± 0.48 (+5.11)	20.98 ± 0.06 (+4.41)
APP (Halawi et al., 2024)	10.00	20.02 ± 0.26 (+5.40)	21.76 ± 0.24 (+3.60)
	20.00	19.79 ± 0.52 (+5.63)	21.06 ± 0.03 (+4.33)
HyDE (Gao et al., 2023)	20.00	20.59 ± 0.26 (+4.83)	20.87 ± 0.21 (+4.51)
RankLlama (Ma et al., 2024)	20.00	20.20 ± 1.44 (+5.22)	22.73 ± 0.08 (+2.66)
Structure-Based RAG			
HippoRAG (Gutiérrez et al., 2024)	20.00	20.94 ± 0.28 (+4.48)	22.82 ± 0.13 (+2.56)
EventSearch (Ours)	22.36	**18.48 ± 0.57 (+6.94)**	**20.10 ± 0.39 (+5.29)**

Table 1: Results on PROPHET dataset. # $\mathcal{D}$ is the average number of retrieval documents. Brier score is the lower the better. Bold stands for the best performance. Increments are compared w.o. RAG. We report mean and std values on twice runs.

To avoid infinite loops, we cap the process at l steps; if reached, the agent is forced to answer with whatever evidence has been accumulated so far.

4 Experiments

4.1 Dataset

We use PROPHET as the test dataset, a real-world event prediction benchmark (Tao et al., 2025). It contains 99 binary prediction questions mostly in 2024 (e.g., "Will Tim Walz win the VP debate against J.D. Vance?"), each paired with 100 news articles as the knowledge base; the dataset is validated by a supportiveness estimate to ensure inferability. Methods output a score $\mathcal{Y} \in [0, 1]$ in Eq. (1) as the probability that the queried event occurs. PROPHET is evaluated by the Brier Score (Brier, 1950), lower is better:

$$\texttt{Brier Score} = \frac{1}{N}\sum_{n}^{N}(\mathcal{Y}_n - \hat{\mathcal{Y}}_n)^2, \quad (8)$$

where $\hat{\mathcal{Y}} \in \{0, 1\}$ is the ground truth (whether the queried event indeed happened).

We additionally adapt the Fever dataset, a Fact Extraction and Verification task in which the system retrieves documents to verify a claim (Thorne et al., 2018). We adapt it by 1) keeping statements with "SUPPORTS" or "REFUTES" labels, and 2) merging all documents containing evidence for the remaining statements into one KB, so every statement has supporting evidence in the KB. Each statement is reformulated as: "Does the given information support the statement?" Methods predict whether the statement is supported; in Eq. (1), $\mathcal{Y} \in \{SUPPORTS, REFUTES, NOT\ ENOUGH\ INFO\}$. The metric is accuracy.

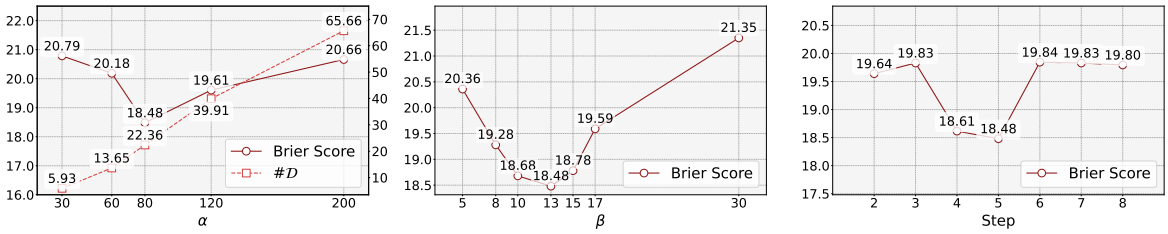
4.2 Baselines

We compare against the zero-shot non-RAG ScrathPAD; text-based RAG baselines Naive RAG, APP (Halawi et al., 2024), Rankllama (Ma et al., 2024), and HyDE (Gao et al., 2023); and the structure-based HippoRAG (Teyler et al., 2024). For comparability, all methods use the same reasoning prompt and LLMs after their respective retrieval. Prompts, baselines, and implementation details are in the Appendix.

4.3 Main Results

We compare EventSearch with competitive baselines on PROPHET in Table 1, measured by Brier Score (lower is better). EventSearch achieves the best performance across all methods, demonstrating superior accuracy in predicting future events. It outperforms the strongest text-based RAG (APP) by 1.31 points under GPT-4O and 2.68 under Gemini-Pro, and the best structure-based baseline (HippoRAG) by 2.46 and 2.72 points, respectively. validating the effectiveness of event-centric RAG. The gains are particularly notable because every baseline shares the same downstream reasoning prompt and LLM, so the improvement is fully attributable to retrieving more supportive events.

Among text-based RAG, Naive RAG and HyDE improve modestly over direct prompting but plateau due to their reliance on surface se-



(a) Regularization Study (b) Supportive Probability Study (c) Maximum Step Study

Figure 3: Study of regularization, supportive probability, and maximum step on PROPHET.

mantics. RankLlama, despite instruction-aware retrieval, still underperforms EventSearch, underscoring the need for event-aware structuring. Among structure-based RAG, HippoRAG's knowledge-graph approach beats text-based methods but trails EventSearch, as entity-centric graphs fail to encode multi-participant events or latent dependencies, two events sharing no entity but bridging the same causal chain are simply invisible to a pairwise entity graph. EventSearch's consistent gains across GPT-4O and Gemini-Pro show its model-agnostic design: the event hypergraph index and probabilistic supportive retrieval reduce dependence on the LLM's generative ability, making it versatile across models.

EventSearch also achieves the best performance on FEVER fact verification (Table 2; all methods on GPT-4o), highlighting strong generalizability across diverse event reasoning settings. By identifying and retrieving supportive events, our approach substantially improves accurate, coherent event reasoning, demonstrating its potential for event-centric systems in real-world applications.

4.4 Ablation Experiments

To assess each component's contribution, we run ablations on PROPHET by removing modules individually (Table 3). w.o. EC drops Maximum Expected Event Cover and retrieves events covering query entities directly; w.o. SS disables supportive scoring, treating events uniformly; w.o. AR removes event-oriented agentic reasoning, reducing it to single-step retrieval. Removing EC raises the Brier Score by 1.16 (18.48→19.64), confirming the importance of Maximum Expected Event Cover. Removing SS causes a larger 2.12-point increase (to 20.60), underscoring the value of probabilistic event scoring. Removing AR also degrades performance by 1.16 points, emphasizing iterative reasoning. Full EventSearch outperforms all ablations, showing that integrating event cover, supportive scoring, and agentic reasoning is essential for effective event-based reasoning.

4.5 Discussions

Regularization Study The parameter α balances event coverage and retrieval specificity in the Maximum Expected Event Cover program. As shown in Figure 3(a) on PROPHET, increasing α initially raises the number of retrieved events by prioritizing structurally coherent pathways, improving Brier Score (optimal at $\alpha = 120$). Beyond that, overly relaxed regularization degrades performance, as large α admits noisy retrieval. Moderate regularization thus balances event supportiveness and structural relevance, suppressing noise while preserving reasoning utility.

Supportive Probability Study We study β, which controls the effect of supportive probabilities in Eq. (6). PROPHET results are in Figure 3(b): performance first improves with β then declines. We explain this from two perspectives.

Theoretical Perspective: The upper bound $\frac{2}{\beta}\sum_{e\in\mathbb{E}}(\mathcal{P}(e)-\mathcal{P}^2(e))$ in Theorem 1 decreases as β grows, so performance initially improves. However, when β is excessively large the first term in Eq. (6) vanishes, leaving the structural-similarity term dominant, our method then degenerates to a common GraphRAG approach. So, at $\beta = 30$ we score 21.35, comparable to HippoRAG's 20.94.

Uncertainty Perspective: As noted, the Theorem-1 bound depends on the precision of supportiveness estimation. When supportiveness judgments are more random, a larger β is needed to shrink the bound; otherwise a smaller β suffices. The optimal β we use effectively manages this uncertainty.

Maximum Step Study The maximum reasoning step l directly affects reasoning depth. Figure 3(c) shows that increasing l from 1 to 5 progressively improves PROPHET performance, as iterative retrieval lets the system gather more supportive events. Performance plateaus beyond $l = 5$, with marginal degradation for $l > 5$ from noise accumulation or redundant retrievals. These findings

Method	ACC ↑
Text-Based RAG	
Naive RAG	91.08 ± 0.08
APP (Halawi et al., 2024)	92.29 ± 0.29
Structure-Based RAG	
HippoRAG (Gutiérrez et al., 2024)	93.16 ± 0.15
EventSearch (Ours)	$\mathbf{94.17} \pm \mathbf{0.17}$

Table 2: Results on Fever dataset. All results are based on GPT-4O. Bold stands for the best performance.

Method	Brier Score ↓	# $\mathcal{D}$
EventSearch	18.48 ± 0.41	22.36
w.o. EC	19.64 ± 1.32 (-1.16)	26.82
w.o. SS	20.60 ± 0.28 (-2.12)	16.48
w.o. AR	19.64 ± 0.21 (-1.16)	17.26

Table 3: Ablation study on PROPHET. # $\mathcal{D}$ stands for the average number of retrieval documents. EC stands for Maximum Expected Event Cover. AR is event-oriented agentic reasoning. SS is supportive score of the event.

highlight the effectiveness of multi-step agentic reasoning and the need to set l to balance thoroughness and efficiency, with $l = 5$ giving the best trade-off in our experiments.

5 Related Work

5.1 Retrieval-Augmented Generation (RAG)

Text-based methods retrieve via textual similarity, with advances in query refinement (Gao et al., 2023), self-reflection (SELF-RAG (Asai et al., 2024)), and iterative correction (Corrective RAG (Liu et al., 2024)), but largely ignore structural or interactive aspects of knowledge.

Structure-based approaches use graph structures. RAPTOR (Sarthi et al., 2024b) performs recursive abstraction for multi-scale tree retrieval; query-focused summarization (Edge et al., 2024) and HippoRAG (Teyler et al., 2024) use graph traversals for global context and long-term dependencies (Procko and Ochoa, 2024). EventRAG (Yang et al., 2025) indexes events by participating entities but does not model supportiveness, limiting it on complex event reasoning.

Agentic RAG orchestrates retrieval with autonomous agents: AMOR (Guan et al., 2024a) trains modular agents via process feedback; GEAR (Shen et al., 2024) combines graph reasoning with agentic decisions; Agent-G (Lee et al., 2025) unifies agent, retriever bank, and critic over hybrid knowledge.

5.2 Event Graph

Event graphs store abstract-level (Zhang et al., 2020) or instance-level (Gottschalk and Demidova, 2018) events and relations, supporting future forecasting (Li et al., 2018) and causality tracking (Tao et al., 2023). Unlike prior event graphs used as knowledge bases, we are the first to build an event hypergraph as a retrieval index.

5.3 Event Reasoning

A typical event reasoning task is future event prediction. Zhan et al. chain cause–effect relations via event causality graphs for structured future reasoning; Tao et al. similarly guide prediction with event schema graphs, both construct event graphs as knowledge. Other works retrieve external news and then reason (Halawi et al., 2024; Guan et al., 2024b); Ye et al. build a multi-agent forecasting framework where each agent serves a distinct utility, and Wang et al. drive LLM-aided time-series prediction with external information. In contrast, we propose an event-oriented RAG framework that can plug into other event reasoning systems.

6 Conclusion

We presented EventSearch, an event-oriented Retrieval-Augmented Generation (RAG) framework for complex event reasoning that relies on and underpins web-scale information retrieval. By constructing a web-adapted event hypergraph index and formulating the *Maximum Expected Event Cover* program, EventSearch retrieves semantically supportive events crucial for accurate reasoning. It further couples retrieval with reasoning via a multi-step, event-centric agentic process, enabling adaptive knowledge acquisition and robust inference over dynamic, heterogeneous sources. Across diverse event reasoning benchmarks, EventSearch significantly outperforms state-of-the-art RAG baselines in both precision and reasoning completeness. Our results highlight the need for event-oriented RAG systems that handle the evolving, interconnected, multi-perspective nature of web data, and suggest that tightly coupling structured event representations with generative reasoning opens new opportunities for intelligent web agents, real-time analytics, and richer event-based knowledge services.

Limitations

In this work, we use close-source LLMs for reasoning and agentic operations. However, training LLMs on this would further benefit this process. We leave it to future work.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*.

Glenn W Brier. 1950. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3.

Lucius EJ Bynum and Kyunghyun Cho. 2024. Language models as causal effect generators. *arXiv preprint arXiv:2411.08019*.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, pages 837–840. Lisbon.

Johannes Dölling, Tatjana Heyde-Zybatow, and Martin Schäfer. 2013. *Event structures in linguistic form and interpretation*, volume 5. Walter de Gruyter.

Chengfeng Dou, Ying Zhang, Yanyuan Chen, Zhi Jin, Wenpin Jiao, Haiyan Zhao, and Yu Huang. 2024. Detection, diagnosis, and explanation: A benchmark for Chinese medial hallucination evaluation. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 4784–4794, Torino, Italia. ELRA and ICCL.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Zhiwei Fei, Xiaoyu Shen, Dawei Zhu, Fengzhe Zhou, Zhuo Han, Songyang Zhang, Kai Chen, Zongwen Shen, and Jidong Ge. 2023. Lawbench: Benchmarking legal knowledge of large language models. *arXiv preprint arXiv:2309.16289*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, Toronto, Canada. Association for Computational Linguistics.

Simon Gottschalk and Elena Demidova. 2018. Eventkg: A multilingual event-centric temporal knowledge graph. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 272–287. Springer.

Jian Guan, Wei Wu, Zujie Wen, Peng Xu, Hongning Wang, and Minlie Huang. 2024a. Amor: A recipe for building adaptable modular knowledge agents through process feedback. *arXiv preprint arXiv:2402.01469*.

Yong Guan, Hao Peng, Xiaozhi Wang, Lei Hou, and Juanzi Li. 2024b. Openep: Open-ended future event prediction. *arXiv preprint arXiv:2408.06578*.

Shailja Gupta, Rajesh Ranjan, and Surya Narayan Singh. 2024. A comprehensive survey of retrieval-augmented generation (rag): Evolution, current landscape and future directions. *arXiv preprint arXiv:2410.12837*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *arXiv preprint arXiv:2405.14831*.

Danny Halawi, Fred Zhang, Chen Yueh-Han, and Jacob Steinhardt. 2024. Approaching human-level forecasting with language models. *arXiv preprint arXiv:2402.18563*.

Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. 2023. Financebench: A new benchmark for financial question answering. *arXiv preprint arXiv:2311.11944*.

Meng-Chieh Lee, Qi Zhu, Costas Mavromatis, Zhen Han, Soji Adeshina, Vassilis N Ioannidis, Huzefa Rangwala, and Christos Faloutsos. 2025. Agent-g: An agentic framework for graph retrieval augmented generation.

Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. 2023. Large language models in finance: A survey. In *Proceedings of the fourth ACM international conference on AI in finance*, pages 374–382.

Zhongyang Li, Xiao Ding, and Ting Liu. 2018. Constructing narrative event evolutionary graph for script event prediction.

Yunzhi Liu, Yizhong Wang, Zeqiu Wu, Akari Asai, and Hannaneh Hajishirzi. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2421–2425.

Hanseok Oh, Hyunji Lee, Seonghyeon Ye, Haebin Shin, Hansol Jang, Changwook Jun, and Minjoon Seo. 2024. Instructir: A benchmark for instruction following of information retrieval models. *arXiv preprint arXiv:2402.14334*.

Trevor T. Procko and Oscar Ochoa. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. 2024a. Raptor: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*.

Pritish Sarthi, Sanaullah Abdullah, Anirudh Tuli, Shreya Khanna, Amanpreet Singh Goldie, and Christopher D. Manning. 2024b. RAPTOR: Recursive abstractive processing for tree-organized retrieval. *arXiv preprint arXiv:2401.18059*.

Zhili Shen, Chenxin Diao, Pavlos Vougiouklis, Pascual Merita, Shriram Piramanayagam, Damien Graux, Dandan Tu, Zeren Jiang, Ruofei Lai, Yang Ren, and 1 others. 2024. Gear: Graph-enhanced agent for retrieval-augmented generation. *arXiv preprint arXiv:2412.18431*.

Zhengwei Tao, Zhi Jin, Xiaoying Bai, Haiyan Zhao, Chengfeng Dou, Yongqiang Zhao, Fang Wang, and Chongyang Tao. 2023. Seag: Structure-aware event causality generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4631–4644.

Zhengwei Tao, Zhi Jin, Bincheng Li, Xiaoying Bai, Haiyan Zhao, Chengfeng Dou, Xiancai Chen, Jia Li, Linyu Li, and Chongyang Tao. 2025. Prophet: An inferable future forecasting benchmark with causal intervened likelihood estimation. *arXiv preprint arXiv:2504.01509*.

Zhengwei Tao, Zhi Jin, Yifan Zhang, Xiancai Chen, Haiyan Zhao, Jia Li, Bing Liang, Chongyang Tao, Qun Liu, and Kam-Fai Wong. 2024. A comprehensive evaluation on event reasoning of large language models. *arXiv preprint arXiv:2404.17513*.

T. J. Teyler, P. Discenna, and J. W. Rudy. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *arXiv preprint arXiv:2405.14831*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819.

Xinlei Wang, Maike Feng, Jing Qiu, Jinjin Gu, and Junhua Zhao. 2024. From news to forecast: Integrating event analysis in llm-based time series forecasting with reflection. *arXiv preprint arXiv:2409.17515*.

Mingji Yang and Xiaokui Wang. 2024. Efficient algorithms for personalized pagerank computation: A survey. *arXiv preprint arXiv:2403.05198*.

Zairun Yang, Yilin Wang, Zhengyan Shi, Yuan Yao, Lei Liang, Keyan Ding, Emine Yilmaz, Huajun Chen, and Qiang Zhang. 2025. EventRAG: Enhancing LLM generation with event knowledge graphs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16967–16979, Vienna, Austria. Association for Computational Linguistics.

Chenchen Ye, Ziniu Hu, Yihe Deng, Zijie Huang, Mingyu Derek Ma, Yanqiao Zhu, and Wei Wang. 2024. Mirai: Evaluating llm agents for event forecasting. *arXiv preprint arXiv:2407.01231*.

Chuanhong Zhan, Wei Xiang, Chao Liang, and Bang Wang. 2024. What would happen next? predicting consequences from an event causality graph. *arXiv preprint arXiv:2409.17480*.

Hongming Zhang, Xin Liu, Haojie Pan, Yangqiu Song, and Cane Wing-Ki Leung. 2020. Aser: A large-scale eventuality knowledge graph. In *Proceedings of the web conference 2020*, pages 201–211.

Siyun Zhao, Yuqing Yang, Zilong Wang, Zhiyuan He, Luna K Qiu, and Lili Qiu. 2024. Retrieval augmented generation (rag) and beyond: A comprehensive survey on how to make your llms use external data more wisely. *arXiv preprint arXiv:2409.14924*.

Appendix

Proof of Theorem 1

We describe the proof of Theorem 1.

Proof. Consider a single realization $s = (s_e)_{e\in\mathbb{E}}$ of the joint variable $X = (X_e)_{e\in\mathbb{E}}$. Let $\tilde{\mathbb{E}}_s$ be the set that maximizes Eq. (7) under this realization s. Let $sol^* = \{x_v^* \mid v \in \mathbb{V}\} \cup \{y_e^* \mid e \in \mathbb{E}\}$ be an optimal solution of IP (6), and $\tilde{\mathbb{E}} = \{e \mid y_e^* = 1\}$. For a feasible solution $sol = \{x_v' \mid v \in \mathbb{V}\} \cup \{y_e' \mid e \in \mathbb{E}\}$, denote $g(sol)$ as the objective value of sol in IP (6).

For $v \in \mathbb{V}$, let $x_v^s = 1$ if there exists an event in $\tilde{\mathbb{E}}_s$ covering x_v, otherwise, let $x_v^s = 0$. Similarly, let $y_e^s = 1$ if $e \in \tilde{\mathbb{E}}_s$, otherwise, let $y_e^s = 0$. Notably, $sol_s = \{x_v^s \mid v \in \mathbb{V}\} \cup \{y_e^s \mid e \in \mathbb{E}\}$ is a feasible solution of IP (6). Since sol_s and sol^* are a feasible and an optimal solution of IP (6), respectively, it follows that $g(sol_s) - g(sol^*) \leq 0$. Furthermore, we have

$$\begin{aligned} f^{\tilde{\mathbb{E}}_s} - f^{\tilde{\mathbb{E}}} =& f^{\tilde{\mathbb{E}}_s} - g(sol_s) - (f^{\tilde{\mathbb{E}}} - g(sol^*)) \\ &+ g(sol_s) - g(sol^*) \qquad (9) \\ \leq& f^{\tilde{\mathbb{E}}_s} - g(sol_s) - (f^{\tilde{\mathbb{E}}} - g(sol^*)) \end{aligned}$$

In this realization s, by the definition of $f^{\tilde{\mathbb{E}}}$, we have

$$f^{\tilde{\mathbb{E}}_s} = \frac{1}{\beta}\sum_{e\in\mathbb{E}} y_e^s s_e + \sum_{v\in\mathbb{V}} \mathcal{C}(v) x_v^s - \frac{1}{\alpha}\sum_{e\in\mathbb{E}} |e| y_e^s$$

Thus, it deduces that

$$f^{\tilde{\mathbb{E}}_s} - g(sol_s) = \frac{1}{\beta}\sum_{e\in\mathbb{E}} y_e^s (s_e - \mathcal{P}(e)) \qquad (10)$$

Similarly, we have

$$f^{\tilde{\mathbb{E}}} - g(sol^*) = \frac{1}{\beta}\sum_{e\in\mathbb{E}} y_e^* (s_e - \mathcal{P}(e)) \qquad (11)$$

Then by Equations (9)-(11), this realization s, we have

$$\begin{aligned} f^{\tilde{\mathbb{E}}_s} - f^{\tilde{\mathbb{E}}} \leq& f^{\tilde{\mathbb{E}}_s} - g(sol_s) - (f^{\tilde{\mathbb{E}}} - g(sol^*)) \\ =& \frac{1}{\beta}\sum_{e\in\mathbb{E}} (y_e^s - y_e^*)(s_e - \mathcal{P}(e)) \end{aligned} \qquad (12)$$

Since $0 \leq \mathcal{P}(e) \leq 1$ and $y_e^s, y_e^*, s_e \in \{0, 1\}$, it follows that $(y_e^s - y_e^*)(s_e - \mathcal{P}(e)) \leq s_e - (2s_e - 1)\mathcal{P}(e)$. Thus by Eqaution (12), in this realization s, we have

$$f^{\tilde{\mathbb{E}}_s} - f^{\tilde{\mathbb{E}}} \leq \frac{1}{\beta}\sum_{e\in\mathbb{E}} (s_e - (2s_e - 1)\mathcal{P}(e)) \qquad (13)$$

By Equation (13), and by enumerating all realizations on random variables X_e where $e \in \mathbb{E}$, it follows that

$$E(\mathrm{OPT}(X) - f^{\tilde{\mathbb{E}}}) \leq \frac{2}{\beta}\sum_{e\in\mathbb{E}} \mathcal{P}(e) - \mathcal{P}^2(e).$$

□

Implementation Details

Event Hypergraph we use `gpt-4o-mini-2024-07-18` for Entity Extraction, Event Extraction, and Event Typing in Event Hypergraph Indexing. We use networkx[4] toolkit for hypergraph implementation. Since networkx doesn't have hypergraph operation, we use DiGraph to implement hypergraph. In our implementation, the event hyperedge is a node in the DiGraph, and its participant nodes and action nodes connect to it. We solve the integer program in Eq. (6) by Gurobi Python API[5]. We use default hyper-parameters for Personal PageRank in networkx package and find it works well. More details about the Personal PageRank is in the Appendix 6. We show the prompts for both PROPHET and Fever in the Appendix 6.

Event-Oriented Agentic Reasoning We use LangGraph[6] framework to implement the event-oriented agentic reasoning. We wrap the supportive events retrieval as a tool in LangGraph, and use the function call module of LLMs to call this tool. We show the tool implementation in Figure 5 in the Appendix. When do entity linking on the hypergraph, we find two most similar nodes of each query participant or action. We use `all-MiniLM-L6-v2`[7] model in similarity computing. In maximum expected event cover retrieval, we set regularization parameter $\alpha = 80$, and supportive encouraging parameter $\beta = 200$ in Eq. 6. We later conduct more analysis on these two hyper-parameters. In event-oriented agentic reasoning, we set the maximum step $l = 5$. All experiments in this work are under twice runs. We report the mean and std values. The version for GPT-4O is `gpt-4o-2024-08-06` while Gemini-Pro is `gemini-1.5-pro-latest`.

Event-Oriented Agentic Reasoning

We show the event-oriented agentic reasoning in the algorithm 1.

[4] https://networkx.org/
[5] https://www.gurobi.com/
[6] https://www.langchain.com/langgraph
[7] https://sbert.net/

Algorithm 1 Event-Oriented Agentic Reasoning

Require: Question Q, max steps l
Ensure: Final answer

Initialize $\dot{\mathbb{R}} \leftarrow \emptyset$, $\tilde{\mathbb{E}} \leftarrow \emptyset$, $t \leftarrow 0$,
$\tilde{\mathbb{P}} \leftarrow \text{ParticipantsOf}(Q)$, $\tilde{\mathbb{A}} \leftarrow \text{ActionsOf}(Q)$
while $t < l$ **do**
  $\mathcal{K} \leftarrow \text{M}(\dot{\mathbb{R}}, \dot{\mathbb{E}}, \dot{\mathbb{P}}, \dot{\mathbb{A}})$
  if $\mathcal{K}$ indicates **Retrieval then**
    $\dot{\mathbb{P}}, \dot{\mathbb{A}}, \mathcal{R}_{\text{new}} \leftarrow Parse(\mathcal{K})$
    $\mathbb{R} \leftarrow \mathbb{R} \cup \{\mathcal{R}_{\text{new}}\}$
    $(\tilde{\mathbb{E}}, \tilde{\mathbb{D}}) = \text{Retrieve}(\mathcal{H}, \dot{\mathbb{P}}, \dot{\mathbb{A}})$ ▷ Eq.(2)
    $\dot{\mathbb{E}} \leftarrow \dot{\mathbb{E}} \cup \tilde{\mathbb{E}}$, $\dot{\mathbb{D}} \leftarrow \dot{\mathbb{D}} \cup \tilde{\mathbb{D}}$
    $\dot{\mathbb{P}} \leftarrow \dot{\mathbb{P}} \cup \bigcup_{e \in \tilde{\mathbb{E}}} \text{ParticipantsOf}(e)$
    $\dot{\mathbb{A}} \leftarrow \dot{\mathbb{A}} \cup \bigcup_{e \in \tilde{\mathbb{E}}} \text{ActionsOf}(e)$
    $t \leftarrow t + 1$
  else
    $\text{Answer}(\dot{\mathbb{R}}, \dot{\mathbb{E}}, \dot{\mathbb{D}}, \dot{\mathbb{P}}, \tilde{\mathbb{A}})$
    exit loop
if $t \geq l$ **then** ▷ Force final answer
  $\text{Answer}(\dot{\mathbb{R}}, \dot{\mathbb{E}}, \dot{\mathbb{D}}, \dot{\mathbb{P}}, \dot{\mathbb{A}})$

Event e			Participant p		Action a	
p / e	a / e	e / $\mathcal{Q}$	e / p	p / $\mathcal{Q}$	e / a	a / $\mathcal{Q}$
1.98	1.33	461.30	2.82	324.42	1.29	476.52

Table 4: Statistics of event hypergraph on PROPHET. e, p, a, $\mathcal{Q}$ stand for event, event participant, event action, and the question. We report the average numbers.

Event Hypergraph Statistics

The statistical profile of PROPHET event hypergraphs, in Table 4, reveals three key structural properties. First, the average of 1.98 participants a per event e indicates frequent multi-nodes interactions. Second, the moderate action $\mathcal{A}$ multiplicity (1.33 a/e).

Notably, the scale statistics demonstrate significant complexity: each question $\mathcal{Q}$ contains 461.30 events ($e/\mathcal{Q}$), implying *high-frequency* of event patterns. The participant-event ratio (324.42 $p/\mathcal{Q}$ vs. 2.82 e/p) further reveals *asymmetric role distributions*: while most participants engage in 2-3 events, a minority serve as hubs across dozens of interactions. These characteristics collectively justify our `EventSearch`'s capacity for indexing events.

Cost Analysis

We analyze in detail the cost consumption of our method to illustrate its practical efficiency. In the process of hypergraph construction, the token consumption is mainly concentrated in four steps: participant extraction, event extraction, relevance score computation, and summarization. As shown in Figure 4, each of these steps contributes only a fraction of the total, and together they remain within a reasonable range. The total consumption for processing a single data instance is 337,760 tokens. By utilizing the `gpt-4o-mini-2024-07-18` model, the overall monetary cost of a single data point is merely \$0.2266, which demonstrates that our approach is highly cost-effective. Considering the scale of our dataset and the complexity of the hypergraph extraction pipeline, this level of expenditure is negligible compared to the benefits gained in analytical depth and accuracy. Therefore, our method not only achieves strong performance in terms of information extraction and representation, but also imposes minimal computational overhead, making it suitable for large-scale and real-time applications.

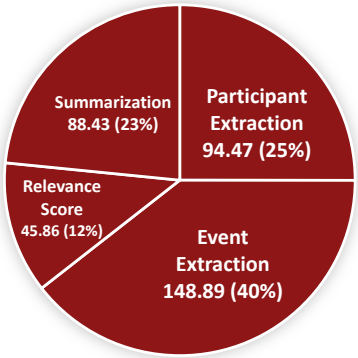


Figure 4: Cost analysis on PROPHET. The amount of token consumed per unit of data, measured in thousands.

Event Diversity

The retrieved events are desired to be diversified, leading to more information. We further inspect the retrieved event diversity by IP (6). We add additional the following constraints to adapt it into a new IP:

$$\sum_{e \in \mathbb{E}, v \in e} y_e \leq 3, \forall v \in \mathbb{V}. \tag{14}$$

These constraints restrict the number of nodes from the same event should not exceed 3. The results on PROPHET are in Table 5. We find the performance drops. The results indicate that our retrieval on the event hypergraph already has sufficient event diversity for event reasoning.

Method	Brier Score ↓
IP (6)	18.48 ± 0.41
IP (14)	19.63 ± 0.99

Table 5: Diversity inspectation of program Eq.(6).

Personal Page Rank

The Personal PageRank (PPR) algorithm is implemented as follows to acquire the relevance scores $\mathcal{C}(v)$ of each node v on the hypergraph $\mathcal{H}$.

Let $\mathbf{M}$ be the transition probability matrix of the hypergraph $\mathcal{H}$. For a hypergraph with n nodes, $\mathbf{M}$ is an $n \times n$ matrix, where the element M_{ij} represents the probability of transitioning from node i to node j.

We start by initializing the relevance scores of all nodes. Let $\mathcal{C}^{(0)}(v)$ be the initial relevance score of node v. We set $\mathcal{C}^{(0)}(v)$ based on the initial assignment of relevance score $\mathcal{C}_{\mathcal{V}}$ for all nodes v on $\mathcal{H}$. That is, $\mathcal{C}^{(0)}(v) = \mathcal{C}_{\mathcal{V}}(v)$.

The PPR algorithm then iteratively updates the relevance scores. In the tth iteration, the relevance score of node v, $\mathcal{C}^{(t)}(v)$, is calculated as follows:

$$\begin{aligned}\mathcal{C}^{(t)}(v) =&(1-\alpha) \sum_{u \in \text{In}(v)} M_{uv} \mathcal{C}^{(t-1)}(u) \\ &+ \alpha \mathcal{C}^{(0)}(v)\end{aligned} \tag{15}$$

where $\text{In}(v)$ is the set of in-neighbors of node v, α is a teleportation probability (a scalar value between 0 and 1, we set $\alpha = 0.15$). The first term $(1-\alpha) \sum_{u \in \text{In}(v)} M_{uv} \mathcal{C}^{(t-1)}(u)$ represents the contribution from the in-neighbors of node v, and the second term $\alpha \mathcal{C}^{(0)}(v)$ is the teleportation term, which helps to prevent the algorithm from getting trapped in cycles.

We repeat this iterative process until the change in the relevance scores between two consecutive iterations is below a certain threshold $\epsilon = 1^{-6}$. That is, we stop when

$$\sum_{v \in V} |\mathcal{C}^{(t)}(v) - \mathcal{C}^{(t-1)}(v)| < \epsilon \tag{16}$$

where V is the set of all nodes in the hypergraph $\mathcal{H}$. At the end of the iteration process, the final relevance score of each node v is $\mathcal{C}(v) = \mathcal{C}^{(t)}(v)$, which will be used in the maximum expected event cover program for event retrieval.

Baselines

`ScrathPAD`: This is the zero-shot ScrathPAD prompting method without RAG. We adopt the scratchpad prompt introduced by (Halawi et al., 2024). For other RAG methods, to ensure comparability, **we uniformly use this prompt for reasoning**. We show this prompt in the Appendix 6 (f).

`Naive RAG`: Given the length of news articles, we perform summarization of these articles in advance. The RAG method then retrieves relevant news articles by calculating the embedding similarity between the question and the news summaries. For this purpose, we employ all-MiniLM-L6-v2 models in SentenceTransformer[8]. After retrieving the news, we utilize the scratchpad prompt for reasoning.

`APP`: This method, introduced by (Halawi et al., 2024), is specifically text-based RAG designed for future forecasting. It also begins with summarizing the news articles. Subsequently, it uses an LLM to compute the relevance score. Following this, it too makes use of the scratchpad prompt for the reasoning process.

`Rankllama`: Rankllama is a retrieval method capable of understanding complex retrieval instructions (Ma et al., 2024). It encodes both the question and the news articles (using summaries of the news) with the model. After the retrieval step, it provides answers in the format of the scratchpad prompt.

`HyDE`: For a given query, this method leverages an instruction - following language model (such as InstructGPT) to generate a "hypothetical document" that captures relevance patterns (Gao et al., 2023). This is a query refinement method. In the event prediction scenario, we generate potential future events that could influence the answer. Then, relevant news articles are retrieved.

`HippoRAG`: This is a representative graphrag framework inspired by the human brain's long-term memory system. It enhances large language models (LLMs) by enabling efficient multi-hop reasoning and knowledge integration from external documents, using a knowledge graph and personalized PageRank algorithm (Gutiérrez et al., 2024).

Ethics Statement

Research-Only Purpose Our algorithm is strictly designed for academic and research purposes. Any commercial exploitation, malicious intent (including but not limited to defamation, harassment, or discriminatory practices), or misuse beyond its intended scope is explicitly prohibited.

[8] https://sbert.net

```
class FindEventsByEntitiesAndActionsInput(BaseModel):
      entities: list[str] = Field(description="Names of the
            input named entities rather than date or time. It
            must be directly a list of string of entities.
            They can not be dates or time. Eg. ['A', 'B',
            'C']."),
            required = True)
      actions: list[str] = Field(description="Names of the
            input actions. It must be directly a list of
            string of actions. Eg. ['A', 'B', 'C']."),
            required = True)


tool = StructuredTool.from_function(
      Func = _find_events_by_paricipants_and_actions,
      Name = "FindEventsByEntitiesAndActions",
      Description = "Find events of a certain event type on
            the graph which are simultaneously related to the
            input named entities and actions.",
      args_schema = FindEventsByEntitiesAndActionsInput)
```

Figure 5: Core code for retrieval tool and its inputs.

Data Accountability Disclaimer The creators of this algorithm bear no responsibility for any financial, unauthorized, unethical, or harmful application of the algorithm.

Prohibition of Harmful Outcomes Any deployment of this algorithm that leads to financial, physical, psychological, or socio-economic harm is categorically condemned. Users are urged to implement safeguards against such risks.

Prompts

We show all prompts (a-f) in the Appendix. The prompts of (a) Named Entity Extraction, (b) Event Extraction, (c) Supportive Probability, (d) Agentic Plan for PROPHET, (e) Agentic Answer for PROPHET, (f) ScratchPAD for PROPHET, (g) Agentic Plan for Fever, (l) Agentic Answer for Fever.

(a) Named Entity Extraction

Your task is to extract named entities from the given paragraph. Don't extract date, time. Respond with a list of entities.
Document: $\mathcal{D}$
Output:

(b) Event Extraction

Your task is to extract all events about the given named entities from the given passages. Each event contains at least one participant and one action. Also, summarize the event.
Pay attention to the following requirements:
- Each event is in the JSON format as ["participants": ["", "", "", ...], "action": "", "time": "", "location": "", "text": "", ...].
Document: $\mathcal{D}$
Named Entities: $\mathbb{P}$
Output:

(c) Supportive Probability

Please consider the following forecasting question and its background information. After that, I will give you a news article and ask you to rate its relevance with respect to the forecasting question.
Question: $\mathcal{Q}$
Question Background: *background*
Resolution Criteria: *resolution criteria*
Article: $\mathcal{D}$
Please rate the relevance of the article to the question, at the scale of 1-6
1 – irrelevant
2 – slightly relevant
3 – somewhat relevant
4 – relevant
5 – highly relevant
6 – most relevant
Guidelines:
- If the article has events of similar types which may happened on different subjects, it also consider relevant to the question.
- You don't need to access any external sources. Just consider the information provided.
- If the text content is an error message about JavaScript, paywall, cookies or other technical issues, output a score of 1.
Rating:

(d) Agentic Plan for PROPHET

Your task is to answer the question about reasoning and predicting a future event in several turns via operation on an event graph. You need to answer the question step by step. In each step, you can answer the question or retrieve more events from a pre-built event graph via calling tools. Previous Memory records summarization of all previous retrieved information and thoughts. Observed Entities and Observed actions store named entities and actions you already retrieved. You must take next step by considering Previous Memory, Observed Entities, and Observed Actions. Answer the question know when you think the information is enough for reasoning the answer. You must generate the thought of calling the tools. You can only use observed entities and actions as arguments of the tool call. Previous Tool Calls records previous tool calling you have made. Don't make duplicated tool call with the same arguments that have been made before. Don't mention in the output words.
Question: $\mathcal{Q}$
Question Background: *background*
Previous Memory (could be empty): $\mathbb{R}$
ALL Observed Entities (could be empty): $\mathbb{P}$
ALL Observed Actions (could be empty): $\mathbb{A}$
Previous Tool Calls: *tool calls*
Your Thought:

(e) Agentic Answer for PROPHET

Question: $\mathcal{Q}$
Question Background: *background*
Resolution Criteria: *resolution criteria*
I have observed events related to the question.
Observed Events: $\mathbb{E}$
Instructions:
Your goal is to aggregate the information and make a final prediction.
1. Provide at least 3 reasons why the answer might be no: {Insert your thoughts}
2. Provide at least 3 reasons why the answer might be yes: {Insert your thoughts}
3. Rate the strength of each of the reasons given in the last two responses: { Insert your rating of the strength of each reason }
4. Aggregate your considerations: {Insert your aggregated considerations}
5. Output your answer (a number between 0 and 1) with an asterisk at the beginning and end of the decimal: {Insert your answer}

(f) ScratchPAD for PROPHET

Question: $\mathcal{Q}$
Question Background: *background*
Resolution Criteria: *resolution criteria*
We have retrieved the following information for this question: *retrieved articles*
Instructions:
1. Provide at least 3 reasons why the answer might be no: { Insert your thoughts }
2. Provide at least 3 reasons why the answer might be yes: { Insert your thoughts }
3. Rate the strength of each of the reasons given in the last two responses: { Insert your rating of the strength of each reason }
4. Aggregate your considerations: { Insert your aggregated considerations }
5. Output your answer (a number between 0 and 1) with an asterisk at the beginning and end of the decimal: { Insert your answer }

(g) Agentic Plan for Fever

Your task is to answer the question about decide whether a claim totally is supported, refuted by given information. If only part of the statement is supported while other part is refutes by the given information, it is still considered as refuted. You must complete this task in several turns via operation on an event graph. You need to answer the question step by step. In each step, you can answer the question or retrieve more events from a pre-built event graph via calling tools. If you think the information is enough, answer the question and don't make new tool call. If you choose to retrieve, make new tool call and don't answer the question because you need to retrieve more information. Previous Memory records summarization of all previous retrieved information and thoughts. Observed Entities and Observed actions store named entities and actions you already retrieved. You must take next step by considering Previous Memory, Observed Entities, and Observed Actions. You must generate the thought of calling the tools. You can only use observed entities and actions as arguments of the tool call. Previous Tool Calls records previous tool calling you have made. Don't make duplicated tool call with the same arguments that have been made before. Don't mention in the output words.
Question: $\mathcal{Q}$
Previous Memory (could be empty): $\mathbb{R}$
ALL Observed Entities (could be empty): $\mathbb{P}$
ALL Observed Actions (could be empty): $\mathbb{A}$
Previous Tool Calls: *tool calls*
Your Thought:

(l) Agentic Answer for Fever

Question: $\mathcal{Q}$
Given events: $\mathbb{E}$
Given articles: {articles}
Your Answer: