TUCH: TEMPORAL UNFOLDING VIA CAUSAL HYPOTHESES FOR LLM-BASED FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Forecasting requires estimating the probability of a future outcome from evidence that becomes available over time. Existing LLM-based forecasting methods remain largely *temporally flat*, organizing evidence without explicitly reasoning across successive stages of real-world time. We introduce *temporal unfolding* and instantiate it in TUCH, where causal hypotheses represent the model's current estimate of the event's causal state, are updated with new evidence, and guide subsequent retrieval. To provide step-level supervision, we construct reference trajectories from resolved forecasting questions and train the model with supervised fine-tuning and process-outcome reinforcement learning. Experiments on PROPHET and ForecastBench show that TUCH consistently outperforms same-backbone temporally flat baselines, while strong LLMs under the same temporal-unfolding scaffold show mixed performance relative to the trained TUCH models.

1 INTRODUCTION

Large language models (LLMs) have shown strong performance on forecasting tasks, where the goal is to estimate the probability of a future event (Halawi et al., 2024; Ye et al., 2024). Existing approaches range from single-pass retrieval-augmented prediction (Chandak et al., 2026; Turtel et al., 2025a;b; Lee et al., 2025; Guan et al., 2024) to multi-round agentic search (Jin et al., 2025; Ye et al., 2024). Despite their differences, these methods largely organize forecasting around associative evidence acquisition. Evidence is retrieved according to its relevance to the forecasting question, current queries, or reasoning context, and then combined to support the prediction (Dettki et al., 2026; Zou et al., 2022). Multi-round methods may revise queries, retrieved context, or intermediate reasoning, but these revisions occur across computational rounds. These methods combine evidence from different stages of an event without organizing reasoning around successive stages of real-world time. We refer to such formulations as *temporally flat*, as illustrated in Figure 1.

Forecasting, however, differs from retrieval-based question answering because the target outcome has not yet occurred and cannot simply be retrieved (Bazaga et al., 2025). Reasoning about how current conditions and mechanisms may develop into future outcomes can support such forecasts (Zhang et al., 2025). We refer to this as *temporally causal reasoning*. As new evidence arrives, it changes which mechanisms remain plausible and which evidence is most relevant to retrieve next.

To support this reasoning, we track the state of the unfolding event. We use *causal state* to denote the underlying configuration of mechanisms and developments driving the outcome. The model estimates this state from available evidence and updates its estimate as new evidence arrives. We call this formulation *temporal unfolding*. The forecasting process advances along real-world time, with the current causal-state estimate guiding subsequent evidence acquisition, as illustrated in Figure 1.

A temporal scaffold specifies when evidence is processed, but does not ensure reliable updates to the model's causal-state estimate. Prior work reports inconsistent or overly conservative forecast revisions under new information (Yuan et al., 2026) and errors in updating natural-language belief states that can be reduced through training (Lidayan et al., 2026). These findings motivate training models for temporal unfolding rather than relying on the inference scaffold alone.

We propose TUCH (**T**emporal **U**nfolding via **C**ausal **H**ypotheses). TUCH uses causal hypotheses to represent the model's current estimate of the event's causal state, update this estimate with newly available evidence, and guide retrieval at subsequent temporal steps. Since existing forecasting

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

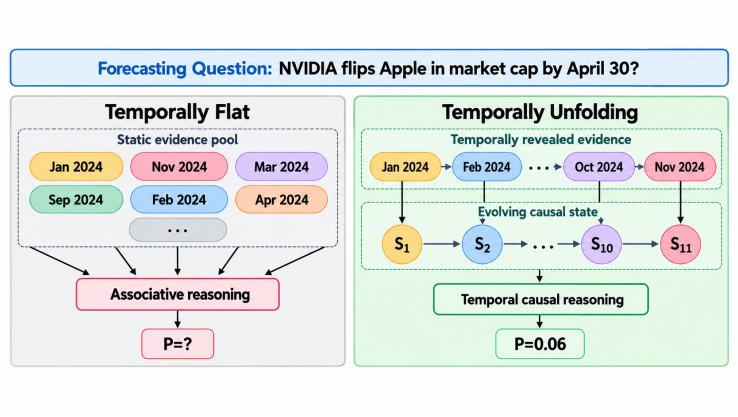

Figure 1: **Comparison between temporally flat forecasting and temporal unfolding.** Temporally flat methods do not track successive updates to the event's causal state over real-world time, whereas temporal unfolding updates the current causal-state estimate and uses it to guide subsequent evidence acquisition.

datasets provide final outcomes but no step-level supervision for this process, we construct reference trajectories and train the model with supervised fine-tuning (SFT) and process-outcome reinforcement learning (RL) (Lightman et al., 2023; Yao et al., 2026).

Our evaluation addresses three complementary questions. First, we evaluate whether temporal unfolding improves forecasting performance over temporally flat methods. Compared with the strongest temporally flat baseline, the multi-round Search-R1, TUCH reduces the Brier score on PROPHET (Tao et al., 2026) and ForecastBench (Karger et al., 2025) by 18.4% and 23.2% with SFT, and by 28.6% and 31.8% after process-outcome RL. Second, we examine whether model scale and the scaffold alone are sufficient. Strong pretrained LLMs given the same temporal-unfolding scaffold but no process training do not consistently match the trained TUCH models. On ForecastBench, GPT-4o trails TUCH-RL by 7.5 Brier points. Third, we examine the mechanisms underlying these gains. Ablations show consistent contributions from hypothesis memory, hypothesis-guided retrieval, and the process reward.

Our contributions are:

- We introduce temporal unfolding, a forecasting formulation in which the forecasting process advances along real-world time, tracking the event's evolving causal state and conditioning evidence acquisition on the current estimate of that state.
- We instantiate this formulation in TUCH, where causal hypotheses represent the model's estimate of the event's causal state, persist across temporal steps, and guide subsequent retrieval.
- We construct a hindsight-guided data pipeline that turns resolved forecasting questions into step-level supervision for learning causal-hypothesis updates across temporal steps.
- We develop a training procedure that combines step-level SFT with process-outcome RL to teach the model how to update causal hypotheses using intermediate supervision and final-outcome feedback.
- We validate both the formulation and its realization through same-scale comparisons with temporally flat methods, strong LLMs under the same temporal-unfolding scaffold, and controlled ablations of hypothesis memory, hypothesis-guided retrieval, and the process reward.

2 Related Work

Existing temporally flat methods for LLM-based forecasting broadly fall into two groups, single-pass forecasting and multi-step agentic methods.

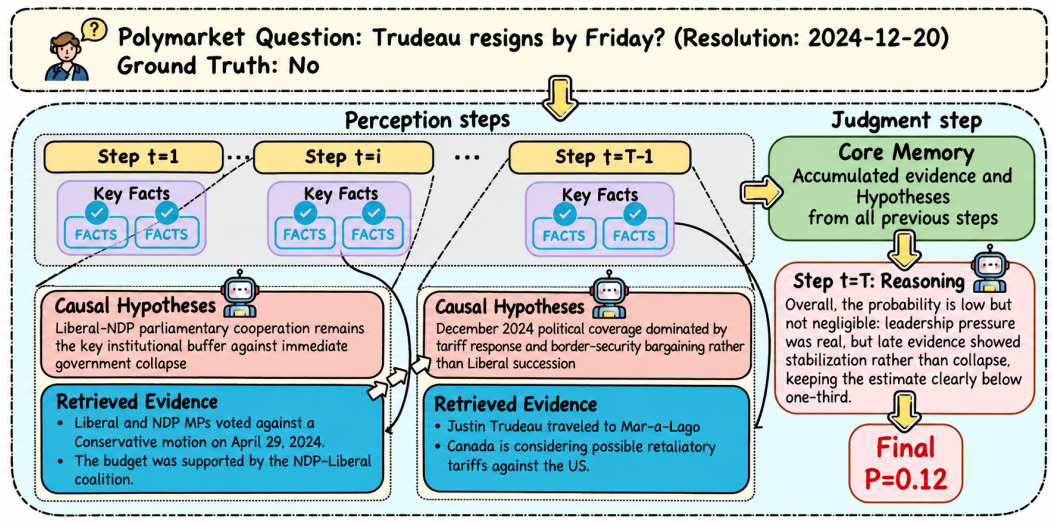

Figure 2: **Overview of the TUCH framework.** The left side shows the perception steps. At each step, the model updates its causal hypotheses using the core memory and newly retrieved evidence. The updated hypotheses guide retrieval for the next temporal step and are stored in the core memory together with the current evidence. The right side shows the judgment step, where the model consolidates the accumulated evidence and hypotheses into a calibrated probability.

2.1 Single-Pass Forecasting

Most LLM-based forecasting methods retrieve relevant evidence and produce a probability forecast in a single pass. Prior work has improved this paradigm through retrieval augmentation, self-play, preference optimization, supervised fine-tuning, and outcome-based reinforcement learning (Turtel et al., 2025a;b). OpenForecaster (Chandak et al., 2026) is a representative trained system in this family.

These methods differ in training and retrieval design, but share the same temporal limitation under our definition. Evidence from different stages of an event may be included in the forecasting context, but these stages are not represented as successive updates to an evolving causal-state estimate. We therefore categorize these methods as temporally flat.

2.2 Multi-Step Agentic Methods

Agentic methods extend forecasting with iterative search, tool use, and reasoning. Search-R1 (Jin et al., 2025) learns to issue multiple search queries during step-by-step reasoning, while MIRAI (Ye et al., 2024) evaluates forecasting agents that iteratively access structured historical events and news through external tools. PROPHET (Tao et al., 2026) also evaluates agentic retrieval for event forecasting. ProphetBLF (Murphy, 2026) maintains a linguistic belief state that is updated after each step of an iterative tool-use loop.

Although these methods iterate, their updates are organized around search or tool-use rounds rather than successive stages of real-world time. Under our definition, they therefore remain temporally flat.

TUCH instead advances through real-world temporal steps, using persistent causal hypotheses to represent its estimate of the event's causal state and guide retrieval at the next step.

3 TUCH

3.1 Overview

We study calibrated binary forecasting. Each instance is represented as $x = (q, c, d)$, where q is a binary forecasting question, c specifies the resolution criteria, and d is the resolution date. We set the

evidence cutoff to $\tau = d - 30$ days and exclude all evidence published after τ. Given the observed outcome $y \in \{0, 1\}$, the model predicts $p_\theta(x) = P_\theta(y = 1 \mid x)$. We evaluate predictions using the Brier score

$$\mathrm{BS}(p_\theta(x), y) = (p_\theta(x) - y)^2 . \tag{1}$$

TUCH formulates forecasting as temporal unfolding. As the forecasting process advances along real-world time, the model tracks the event's evolving causal state, representing its current estimate as causal hypotheses. At each step, newly retrieved evidence is combined with the accumulated history to update these hypotheses, which then guide evidence retrieval for the next step. The evidence and hypotheses from previous steps are retained in a core memory. At the final step, the accumulated history is used to produce the probability p. Section 3.2 formalizes this process.

Learning this process requires supervision beyond the final outcome. We therefore construct step-level reference hypotheses from resolved Polymarket forecasting questions and use them to train the model with supervised fine-tuning and process-outcome reinforcement learning. Sections 3.3 and 3.4 describe the data construction and training procedures.

3.2 Temporal Unfolding via Causal Hypotheses

As illustrated in Figure 2, TUCH partitions the available evidence history up to the cutoff τ into $T-1$ consecutive real-world intervals. These intervals correspond to $T-1$ perception steps, followed by a final judgment step at $t = T$. Let M_θ denote the forecasting model and $\phi(\cdot)$ a frozen, ℓ_2-normalized sentence encoder. The key-fact corpus F is partitioned into $F_1, \ldots, F_{T-1}$ by publication time, where F_t contains the facts published during interval t. At the first step, before any causal hypotheses have been formed, we retrieve

$$e_1 = \mathrm{Top}_K(q, F_1, \phi),$$

where K is the evidence budget for each step.

At perception step t, the model receives the forecasting question q, resolution criteria c, resolution date d, the core memory H_t, and the newly retrieved evidence e_t. The core memory contains the evidence and causal hypotheses from previous steps, with $H_1 = \varnothing$. The model generates

$$P_t = M_\theta(q, c, d, H_t, e_t), \tag{2}$$

where $P_t = \{\rho_{t,i}\}_{i=1}^{M_t}$ contains M_t causal hypotheses. Each $\rho_{t,i}$ contains a prediction statement, $\mathrm{pred}(\rho_{t,i})$, and an evidence-grounded explanation of the forecast-relevant mechanism or development underlying it. The set P_t explicitly represents the model's current estimate of the event's causal state at step t.

For $t = 1, \ldots, T-2$, the prediction statements in P_t guide retrieval from the next temporal interval. Each hypothesis retrieves

$$r_t = \lfloor K/M_t \rfloor$$

facts from F_{t+1} according to cosine similarity under ϕ. The retrieved facts are combined across hypotheses

$$B_{t+1} = \bigcup_{i=1}^{M_t} \mathrm{Top}_{r_t}\left(\mathrm{pred}(\rho_{t,i}), F_{t+1}, \phi\right). \tag{3}$$

If overlapping results leave fewer than K unique facts, the remaining slots are filled with unused facts that have the highest similarity to any current prediction. The resulting set forms e_{t+1}. In this way, evidence acquisition at each step is conditioned on the current causal hypotheses rather than directly on the original question alone.

After each perception step, the current evidence and causal hypotheses are appended to the core memory

$$H_{t+1} = \big((P_1, e_1), \ldots, (P_t, e_t)\big). \tag{4}$$

The ordered memory preserves the sequence of observed evidence and the model's evolving causal-state estimates.

At the final judgment step, the model conditions on the complete history

$$H_T = \big((P_1, e_1), \ldots, (P_{T-1}, e_{T-1})\big)$$

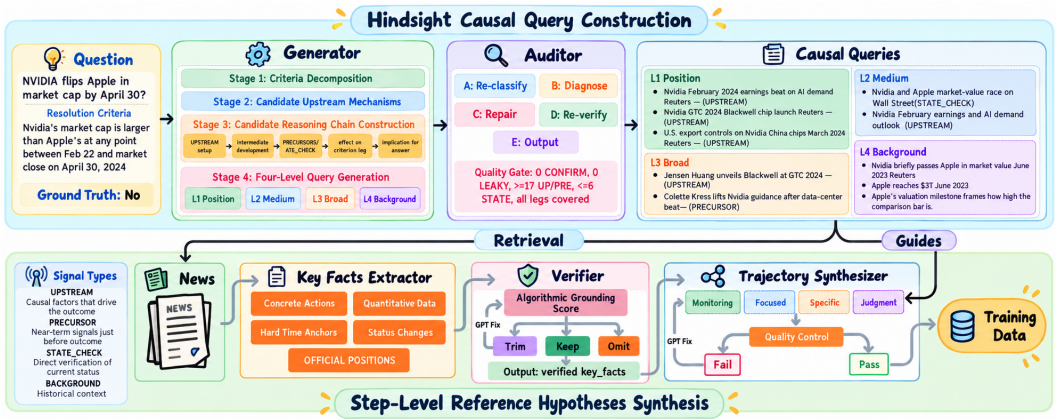

Figure 3: **Training-data construction pipeline. Top:** hindsight causal queries guide retrieval of temporally ordered news evidence and provide a shared scaffold for synthesis. **Bottom:** retrieved articles are distilled into grounded key facts and used to synthesize step-level reference hypotheses, followed by quality control and human validation.

and produces the final probability

$$p = M_\theta(q, c, d, H_T), \qquad p \in [0, 1]. \tag{5}$$

The same T-step structure and output format are used during SFT, RL, and inference.

3.3 Data Construction

We construct step-level supervision from resolved Polymarket binary forecasting questions. We instantiate the temporal unfolding process with $T = 12$ at monthly granularity, covering more than 99% of the forecasting horizons in our training and evaluation data. Hindsight is used only for training-data construction. Evaluation evidence is constructed independently without access to question outcomes, as described in Section 4.1.

As shown in Figure 3, the pipeline first uses hindsight causal queries to retrieve temporally ordered news evidence and then synthesizes step-level reference hypotheses. The queries provide a shared scaffold for retrieval and synthesis, aligning both with mechanisms relevant to the observed resolution.

3.3.1 Hindsight Causal Query Construction

Using the known resolution as hindsight, we construct queries targeting upstream conditions, mechanisms, and intermediate developments related to the observed outcome rather than directly confirming the resolution. The generator, auditor, and repair components are implemented with GPT-5.4 (OpenAI, 2026).

The generator decomposes each resolution criterion into testable components, identifies candidate upstream mechanisms and causal chains, and converts them into four levels of queries (L1 to L4). Each query is assigned a specificity level and a causal signal type. More specific queries target individual upstream drivers, while broader queries track intermediate states and developments.

A separate auditor checks each query before acceptance. It verifies the assigned specificity and causal signal type, relevance to the forecasting question, direct outcome leakage, and post-resolution information. Queries that fail these checks are repaired and audited again.

We manually audit queries from 200 randomly sampled training questions across different specificity levels. In 96% of the sampled questions, no inspected query is flagged for relevance, specificity, outcome leakage, or temporal validity. Flagged queries in the remaining questions are repaired or removed before use.

These hindsight causal queries are used only during training-data construction and are never provided to the forecasting model during SFT, RL, or inference. The prompts used for query generation and auditing are provided in Appendix E.

3.3.2 STEP-LEVEL REFERENCE HYPOTHESES SYNTHESIS

To keep the core memory compact, we distill each retrieved article into at most 25 atomic, source-grounded key facts. GPT-5.4 is used for key-fact extraction, grounding verification against the source article, and conservative repair with operations such as KEEP, TRIM, and OMIT.

A GPT-5.4 synthesizer then constructs reference causal hypotheses from the temporally organized key facts. The hindsight causal queries provide a shared scaffold for retrieval and synthesis, producing a coherent trajectory that is refined as new evidence becomes available.

A GPT-5.4 quality-control pipeline checks format, resolution consistency, future-information leakage, and later-step relevance, regenerating failures for up to three attempts. During pipeline development, 10% of 40 manually inspected steps contained future-information leakage, motivating the trajectory-level GPT-Fix module in Figure 3 and an explicit leakage check.

After finalizing the pipeline, we validated 600 synthesized steps from 200 randomly sampled training questions across early, middle, and late stages. Annotators saw only the forecasting question and evidence available up to each step, not the resolved outcome. No evidence-grounding errors or future-information leakage were found in the 600 inspected steps. Of these, 574 (95.7%) passed the causal-plausibility check. The remaining 26 were flagged only for causal plausibility, so no additional human filtering was applied.

After filtering and regeneration, the final training set contains 1,378 reference trajectories from questions resolved before November 2024, retaining only samples with sufficient full-text news coverage that pass the quality and leakage checks. Further details are provided in Appendix A.

3.4 TRAINING METHOD

During SFT, the model follows the reference trajectories constructed in Section 3.3.2, using reference hypotheses to form teacher-conditioned histories and retrieval trajectories. During RL and inference, the process is fully self-conditioned, with the model generating its own causal hypotheses to guide subsequent retrieval. Reference trajectories are used as supervision but are not provided as model input during RL or inference.

Supervised Fine-Tuning. At each perception step, the model is trained to generate the reference causal hypotheses from the reference history and retrieved evidence. At the final step, it predicts the reference judgment from the accumulated history. Training uses next-token cross-entropy under teacher forcing. The hindsight causal queries in Section 3.3.1 are used only for data construction and are never provided to the forecasting model.

Process-Outcome Reinforcement Learning. Starting from the SFT model, we optimize the full T-step process with DAPO (Yu et al., 2025). Each rollout uses self-conditioned retrieval. The outcome reward is the negative Brier score of the final prediction. The process reward compares generated and reference causal hypotheses at each perception step. We embed both sets with the retrieval encoder and compute an optimal one-to-one matching with the Hungarian algorithm. Each step combines F1-based coverage with the mean semantic similarity of the matched hypotheses, and the process reward averages these scores across perception steps.

Following GDPO (Liu et al., 2026), we normalize the outcome and process rewards separately within each prompt group. The rollout advantage is

$$A_i = z_g\left(R_{\text{out}}^{(i)}\right) + \lambda z_g\left(R_{\text{pro}}^{(i)}\right), \tag{6}$$

where $z_g(\cdot)$ denotes a within-group z-score with a standard-deviation floor to prevent amplification of near-zero variance, and λ controls the relative weight of the process signal.

Table 1: Performance comparison on PROPHET and ForecastBench. Best results are in bold, and second-best results are underlined. Gemini 2.0 Flash was unavailable at the time of our ForecastBench evaluation.

Method	**PROPHET** Brier ($\times 100$) $\downarrow$	**ForecastBench** Brier ($\times 100$) $\downarrow$
Temporally flat baselines		
Vanilla RAG	36.40	31.03
OpenForecaster	37.50	31.80
Search-R1	30.40	25.22
Large LLMs under the temporal-unfolding scaffold		
GPT-4o	28.01	24.67
Gemini 2.0 Flash	21.13	–
DeepSeek-V3	27.07	25.83
Qwen2.5-72B-Instruct	28.53	28.65
TUCH *variants*		
TUCH-SFT	24.80	<u>19.37</u>
TUCH-RL	<u>21.70</u>	**17.19**

4 Experimental Setup

4.1 Evaluation Datasets and Metrics

We evaluate on PROPHET (Tao et al., 2026) and ForecastBench (Karger et al., 2025). PROPHET provides forecasting questions with an accompanying news corpus, which we use without modification. For ForecastBench, we use 274 resolved binary questions from Polymarket, Manifold, Metaculus, and Infer, deduplicated against the training set and spanning diverse event categories. All selected ForecastBench questions resolve after March 2025, while the training questions resolve before November 2024.

ForecastBench does not provide an accompanying news corpus, so we construct the evaluation evidence independently of the training-data pipeline and without access to question outcomes. The resulting evidence corpus is built once and shared across all evaluated methods. Full construction details are provided in Appendix C.2.

We report Brier scores multiplied by 100, with lower values indicating better probabilistic forecasts.

4.2 Baselines

Our comparisons address two distinct questions. First, we compare TUCH with temporally flat forecasting methods at the same model scale to evaluate the forecasting formulation while controlling for backbone size. Second, we evaluate stronger pretrained LLMs under the same temporal-unfolding scaffold as TUCH but without task-specific process training to test whether model scale and the inference scaffold alone are sufficient.

Temporally flat baselines. Using Qwen2.5-7B-Instruct as the shared backbone, we reproduce Vanilla RAG, OpenForecaster (Chandak et al., 2026), and Search-R1 (Jin et al., 2025) following their original training and inference procedures. For the trainable baselines, OpenForecaster and Search-R1 use the same underlying training-question set as TUCH, while retaining their method-specific data construction and training objectives. This setup controls for backbone size and, where applicable, the training-question set.

Large LLMs under the temporal-unfolding scaffold. We evaluate GPT-4o (OpenAI, 2024), DeepSeek-V3 (DeepSeek-AI, 2025), Qwen2.5-72B-Instruct (Qwen et al., 2025), and Gemini 2.0 Flash (Google DeepMind, 2025) under the same temporal-unfolding scaffold as TUCH. At inference time, these models use the same temporal partitions, per-step evidence budget, causal-hypothesis format, hypothesis-guided retrieval procedure, and final judgment format. They receive no task-

specific process training. This comparison tests whether strong pretrained reasoning ability and the inference scaffold alone are sufficient without learning the temporal-unfolding process.

TUCH variants. TUCH-SFT is trained on the step-level supervision described in Section 3.3. TUCH-RL further optimizes the model with process-outcome reinforcement learning.

4.3 Training Details

We use E5-large-v2 (Wang et al., 2024) as the sentence encoder for hypothesis-guided retrieval and process-reward computation. Using Qwen2.5-7B-Instruct as the backbone, we perform SFT on 1,378 reference trajectories with $T = 12$ at monthly granularity. We then apply DAPO to 1,000 training prompts with $G = 8$ rollouts per prompt. Each rollout executes the full T-step process with self-conditioned retrieval. Training is conducted on 8 NVIDIA H200 GPUs. Hyperparameters and implementation details are provided in Appendix B.

5 Results

5.1 Main Results

Table 1 summarizes the main results on PROPHET and ForecastBench. TUCH-SFT achieves Brier scores of 24.80 and 19.37 on the two benchmarks, respectively, and process-outcome RL further improves them to 21.70 and 17.19. TUCH achieves the best performance among same-scale methods on both benchmarks and the best overall result on ForecastBench.

Temporal unfolding improves over temporally flat baselines. Search-R1 is the strongest temporally flat baseline on both benchmarks, with Brier scores of 30.40 on PROPHET and 25.22 on ForecastBench. Using the same Qwen2.5-7B-Instruct backbone, TUCH-SFT improves over Search-R1 by 18.4% and 23.2%, while TUCH-RL improves by 28.6% and 31.8%, respectively. Since Search-R1 already uses multi-round retrieval and reinforcement learning, these gains indicate that simply adding iterative retrieval and RL is not sufficient. Instead, they support the effectiveness of organizing forecasting around real-world temporal progression and persistent causal hypotheses that guide evidence acquisition. Notably, TUCH-SFT already outperforms Search-R1 before process-outcome RL is introduced, indicating that the gain is not attributable to the RL stage alone.

The temporal-unfolding scaffold alone is not sufficient. Strong pretrained LLMs under the same temporal-unfolding scaffold do not consistently match the trained TUCH models. On ForecastBench, GPT-4o is the strongest available control at 24.67, compared with 19.37 for TUCH-SFT and 17.19 for TUCH-RL. On PROPHET, TUCH-SFT outperforms GPT-4o, DeepSeek-V3, and Qwen2.5-72B-Instruct, while TUCH-RL reaches 21.70.

These results separate the effect of the inference scaffold from that of process learning. Providing temporal steps, causal hypotheses, and hypothesis-guided retrieval to a strong pretrained model is not sufficient to reproduce the gains of TUCH. The remaining gap suggests that performance depends not only on the structure of inference, but also on learning how to update causal hypotheses as evidence accumulates and use them consistently across temporal steps.

Reinforcement learning further improves TUCH. TUCH-RL improves over TUCH-SFT by 12.5% on PROPHET and 11.3% on ForecastBench. This consistent improvement shows that process-outcome RL provides additional gains beyond supervised process learning. We isolate the contribution of the process reward in Section 5.2.

5.2 Ablation Study

We ablate three mechanisms used to realize temporal unfolding in TUCH (Table 2). Hypothesis memory preserves causal hypotheses across temporal steps. Hypothesis-guided retrieval uses the current hypotheses to select evidence for the next step. The process reward directly supervises the intermediate unfolding process during RL. The two SFT ablations are retrained from the same Qwen2.5-7B-Instruct backbone with the same training split, supervision construction, and hyper-

Table 2: Component-wise ablation of TUCH on PROPHET and ForecastBench. Bold rows denote full TUCH variants, while indented rows denote ablations.

Method	PROPHET Brier ($\times 100$) $\downarrow$	ForecastBench Brier ($\times 100$) $\downarrow$
TUCH-SFT	**24.80**	**19.37**
w/o hypothesis memory	27.90	20.13
question-only retrieval	29.39	21.56
TUCH-RL	**21.70**	**17.19**
w/o process reward	23.10	18.27

parameters as TUCH-SFT. The RL ablation keeps the full RL configuration and removes only the process reward.

Effect of hypothesis memory. We remove prior causal hypotheses from the core memory while retaining hypothesis-guided retrieval. Hypotheses are still generated at each step and used for next-step retrieval, but are not preserved as context for later steps. This increases the Brier score by 3.10 points on PROPHET. The benefit of causal hypotheses is therefore not limited to the step in which they are generated, as preserving them across steps allows later reasoning to build on earlier causal estimates. Because temporal partitioning and hypothesis-guided retrieval remain unchanged in this ablation, the degradation specifically reflects the value of cross-step hypothesis memory within temporal unfolding.

Hypothesis-guided vs. question-only retrieval. We replace hypothesis-guided retrieval with question-only retrieval at every temporal step while retaining the temporal structure and hypothesis memory. During SFT, the ablation uses the same reference hypothesis targets and teacher-conditioned hypothesis history as TUCH-SFT. Only the retrieval query is changed from the current hypotheses to the original forecasting question, so the retrieved evidence may differ between the two variants. This increases the Brier score by 4.59 points on PROPHET and also degrades performance on ForecastBench. The result shows that causal hypotheses contribute not only through hypothesis memory, but also as signals for subsequent retrieval. Because the hypotheses are updated as the event unfolds, they can continually redirect evidence acquisition toward the mechanisms and developments that are most relevant at the current stage. This allows retrieval to follow the evolving reasoning process rather than repeatedly returning to the original forecasting question.

Effect of process reward. Finally, we remove the process reward R_{pro} while keeping the remaining RL setup unchanged. Outcome-only RL already improves over TUCH-SFT, but adding the process reward further reduces the Brier score by 1.40 points on PROPHET and also improves performance on ForecastBench. This shows that the process reward provides additional gains beyond the final outcome reward.

6 Conclusion

Forecasting requires reasoning over evidence that becomes available over time, yet existing LLM-based methods remain temporally flat when they do not track the event's evolving causal state along real-world time. We propose TUCH, which instantiates temporal unfolding using causal hypotheses that represent the model's current estimate of this state, persist across temporal steps, and guide subsequent retrieval. To learn how to update these hypotheses, TUCH is trained with step-level supervision constructed from resolved forecasting questions. Across PROPHET and ForecastBench, TUCH outperforms same-scale temporally flat baselines, while strong pretrained LLMs under the same temporal-unfolding scaffold do not consistently match the trained TUCH models. Process-outcome reinforcement learning provides further gains over supervised fine-tuning. The results show the benefit of learning to update causal hypotheses as evidence accumulates and use them to guide retrieval across temporal steps.

AI USE STATEMENT

Generative AI tools were used to polish the manuscript and to construct synthetic training data for our experiments. Specifically, they were used to generate hindsight causal queries from resolved forecasting questions, distill retrieved articles into grounded key facts, and synthesize step-level reference causal hypotheses from temporally organized evidence. Details of the synthetic-data construction process are provided in Section 3.3 and the appendices. All AI-assisted content was reviewed by the authors. The authors take full responsibility for the final content of the manuscript and the use of the generated data.

ETHICS STATEMENT

This work studies LLM-based forecasting through temporal unfolding and causal hypotheses. The framework is intended as a research method for making forecasting more structured and inspectable, rather than as a system that provides guaranteed predictions about real-world events. We use public forecasting questions and publicly available news evidence, without private user data or human-subject experiments. As with other news-based forecasting systems, outputs may reflect biases in news coverage, forecasting platforms, or teacher-generated reference hypotheses. Evidence grounding and quality-control checks are applied during data construction, but they do not guarantee factual completeness, causal correctness, or reliability for consequential decision-making.

REPRODUCIBILITY STATEMENT

We provide the data-construction procedure, training setup, evaluation protocol, and implementation details in Sections 3.1 and 4 and the appendices. Appendix A describes the training data construction, Appendix B reports the SFT and RL settings, Appendix C details the evaluation datasets, Appendix D lists the evaluated model versions and knowledge cutoffs, and Appendix E provides the prompts used for causal-query construction and verification. Some experiments rely on closed-source LLM APIs whose availability and model versions may change over time, which may limit exact reproduction of those results.

REFERENCES

Adrián Bazaga, Rexhina Blloshmi, Bill Byrne, and Adrià de Gispert. Learning to reason over time: Timeline Self-Reflection for improved temporal reasoning in language models, 2025. URL https://arxiv.org/abs/2504.05258.

Nikhil Chandak, Shashwat Goel, Ameya Prabhu, Moritz Hardt, and Jonas Geiping. Scaling Open-Ended reasoning to predict the future, 2026. URL https://arxiv.org/abs/2512.25070.

DeepSeek-AI. DeepSeek-V3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Hanna M. Dettki, Charley M. Wu, and Bob Rehder. Do LLMs share Human-Like biases? Causal reasoning under prior knowledge, irrelevant context, and varying compute budgets, 2026. URL https://arxiv.org/abs/2602.02983.

Google DeepMind. Gemini 2.0 Flash model card. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-2-0-Flash-Model-Card.pdf, April 2025. Model card.

Yong Guan, Hao Peng, Xiaozhi Wang, Lei Hou, and Juanzi Li. OpenEP: Open-Ended future event prediction, 2024. URL https://arxiv.org/abs/2408.06578.

Danny Halawi, Fred Zhang, Chen Yueh-Han, and Jacob Steinhardt. Approaching Human-Level forecasting with language models, 2024. URL https://arxiv.org/abs/2402.18563.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09516.

Ezra Karger, Houtan Bastani, Chen Yueh-Han, Zachary Jacobs, Danny Halawi, Fred Zhang, and Philip E. Tetlock. ForecastBench: A dynamic benchmark of AI forecasting capabilities, 2025. URL https://arxiv.org/abs/2409.19839.

Sang-Woo Lee, Sohee Yang, Donghyun Kwak, and Noah Y. Siegel. Advancing event forecasting through massive training of large language models: Challenges, solutions, and broader impacts, 2025. URL https://arxiv.org/abs/2507.19477.

Kalev Leetaru and Philip A. Schrodt. GDELT: Global data on events, location, and tone. In *ISA Annual Convention*, 2013.

Aly Lidayan, Jakob Bjorner, Satvik Golechha, Kartik Goyal, and Alane Suhr. ABBEL: Learning Natural-Language belief states for Memory-Efficient interaction, 2026. URL https://arxiv.org/abs/2512.20111.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. GDPO: Group reward-Decoupled normalization policy optimization for multi-reward RL optimization, 2026. URL https://arxiv.org/abs/2601.05242.

Kevin Murphy. Agentic forecasting using sequential Bayesian updating of linguistic beliefs, 2026. URL https://arxiv.org/abs/2604.18576.

OpenAI. GPT-4o System Card, August 2024. URL https://openai.com/index/gpt-4o-system-card/.

OpenAI. Introducing GPT-5.4, March 2026. URL https://openai.com/index/introducing-gpt-5-4/.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Zhengwei Tao, Pu Wu, Zhi Jin, Xiaoying Bai, Haiyan Zhao, Chengfeng Dou, Xiancai Chen, Jia Li, Linyu Li, Chongyang Tao, and Wentao Zhang. PROPHET: An inferable future forecasting benchmark with causal intervened likelihood estimation, 2026. URL https://arxiv.org/abs/2504.01509.

Benjamin Turtel, Danny Franklin, and Philipp Schoenegger. LLMs can teach themselves to better predict the future, 2025a. URL https://arxiv.org/abs/2502.05253.

Benjamin Turtel, Danny Franklin, Kris Skotheim, Luke Hewitt, and Philipp Schoenegger. Outcome-based reinforcement learning to predict the future, 2025b. URL https://arxiv.org/abs/2505.17989.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training, 2024. URL https://arxiv.org/abs/2212.03533.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020. URL https://arxiv.org/abs/2002.10957.

Jiarui Yao, Ruida Wang, Hao Bai, and Tong Zhang. Future-KL regularized GRPO: Process-Level credit assignment from f-divergence regularization, 2026. URL https://arxiv.org/abs/2601.10201.

Chenchen Ye, Ziniu Hu, Yihe Deng, Zijie Huang, Mingyu Derek Ma, Yanqiao Zhu, and Wei Wang. MIRAI: Evaluating LLM agents for event forecasting, 2024. URL https://arxiv.org/abs/2407.01231.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An Open-Source LLM reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Zhangdie Yuan, Zifeng Ding, and Andreas Vlachos. Do language models update their forecasts with new information?, 2026. URL https://arxiv.org/abs/2509.23936.

Meiru Zhang, Auss Abbood, Zaiqiao Meng, and Nigel Collier. The power of simplicity in LLM-Based event forecasting. In Ehsan Kamalloo, Nicolas Gontier, Xing Han Lu, Nouha Dziri, Shikhar Murty, and Alexandre Lacoste (eds.), *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*, pp. 454–470, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-264-0. doi: 10.18653/v1/2025.realm-1.32. URL https://aclanthology.org/2025.realm-1.32/.

Andy Zou, Tristan Xiao, Ryan Jia, Joe Kwon, Mantas Mazeika, Richard Li, Dawn Song, Jacob Steinhardt, Owain Evans, and Dan Hendrycks. Forecasting future world events with neural networks, 2022. URL https://arxiv.org/abs/2206.15474.

A Data Construction Details

A.1 Training Data Statistics

Our final training set contains 1,378 resolved binary forecasting questions from Polymarket. All training questions resolve before November 2024, preceding the evaluation periods of both PROPHET and ForecastBench. Table 3 reports the topic distribution of the training set.

Table 3: Topic distribution of the Polymarket training questions. Percentages are rounded to one decimal place and may not sum exactly to 100%.

Category	Count	Percentage
Politics	390	28.3%
Geopolitics	257	18.7%
Crypto	241	17.5%
Culture	185	13.4%
Sports	114	8.3%
Business/Corporate	75	5.4%
Tech/AI	59	4.3%
Science/Health	30	2.2%
Legal/Regulatory	19	1.4%
Macro/Economy	8	0.6%
Total	**1,378**	**100.0%**

A.2 Training Evidence Construction

For each training question, we construct temporally organized news evidence using the hindsight causal queries described in Section 3.3. These queries are used only to construct training supervision and are never used during evaluation. News articles are retrieved from the GDELT (Leetaru & Schrodt, 2013) corpus and organized according to the temporal unfolding procedure described in Section 3.2.

B Implementation Details

B.1 Supervised Fine-Tuning

We fine-tune Qwen2.5-7B-Instruct on the 1,378 synthesized reference trajectories. Table 4 summarizes the SFT configuration.

Table 4: Hyperparameters for supervised fine-tuning.

Hyperparameter	**Value**
Base model	Qwen2.5-7B-Instruct
Training samples	1,378
Learning rate	1.0×10^{-5}
LR scheduler	Cosine
Warmup fraction	0.1
Epochs	2
Max tokens per GPU	32,768
Max context length	32,768
Training framework	slime

B.2 Reinforcement Learning

Starting from the SFT checkpoint, we further optimize the model with DAPO on 1,000 training prompts. Each prompt produces eight rollouts, and each rollout executes the full 12-step temporal unfolding process with self-conditioned retrieval. Table 5 summarizes the RL configuration.

Table 5: Hyperparameters for reinforcement learning.

Hyperparameter	**Value**
Training prompts	1,000
Rollouts per prompt	8
Evidence budget K	60
Process reward weight λ	0.3
KL coefficient	0.02
Epochs	1
Batch size	32
Max response length per call	2,048
Max context length	32,768
Training framework	slime

C Evaluation Dataset Details

C.1 PROPHET

We evaluate on the L1 subset of PROPHET, which contains 612 binary forecasting questions. PROPHET provides its own accompanying evidence corpus, which we use without modification.

PROPHET does not provide category labels for its forecasting questions. We recover the official Polymarket market tags by matching market IDs for 605 of the 612 questions. For the remaining seven questions, we assign categories based on the question text. Table 6 reports the resulting distribution.

C.2 ForecastBench

Our ForecastBench evaluation set contains 274 resolved binary forecasting questions collected from Polymarket, Manifold, Metaculus, and Infer. All questions resolve after March 2025 and are deduplicated against the training set.

Table 6: Category distribution of the PROPHET evaluation set, derived primarily from official Polymarket market tags.

Category	Count	Percentage
Sports	229	37.4%
Politics	202	33.0%
Entertainment	117	19.1%
Crypto	30	4.9%
Economy/Finance	17	2.8%
Science/Health	15	2.5%
Weather	2	0.3%
Total	**612**	**100.0%**

Unlike the training-data construction pipeline, ForecastBench evidence is constructed without access to question outcomes. For each question, we recall candidate news from the GDELT GKG index between $d - 365$ days and the evidence cutoff $\tau = d - 30$ days, using entities and topic keywords extracted from the question. Candidates are then reranked by the cosine similarity between the article title and the question text, its resolution criteria, or its background description, computed with a MiniLM (Wang et al., 2020) encoder. Within each month, a fixed share of the quota is reserved for lower-similarity articles to avoid a homogeneous evidence pool. Up to 900 top-ranked articles are retained per question, organized into monthly evidence pools, and distilled into grounded key facts. The resulting evidence corpus is constructed once and shared across all evaluated methods.

Table 7 reports the category distribution of the resulting evaluation set.

Table 7: Category distribution of the ForecastBench evaluation set constructed in this work. The dataset contains 274 resolved binary forecasting questions.

Category	Count	Percentage
Tech/AI	50	18.2%
Sports	49	17.9%
Economy/Finance	46	16.8%
Geopolitics	38	13.9%
Other	34	12.4%
Politics	24	8.8%
Crypto	14	5.1%
Entertainment	11	4.0%
Science/Health	8	2.9%
Total	**274**	**100.0%**

D Model Versions and Knowledge Cutoffs

Table 8 summarizes the model versions and knowledge cutoffs used in our experiments. For models whose providers do not explicitly document a knowledge cutoff, we report estimates based on publicly available information.

E Prompts

E.1 Hindsight Causal Query Generation

The following prompt is used to construct hindsight causal queries for training-data generation. It decomposes the resolution criteria, identifies candidate upstream mechanisms and reasoning chains, and generates four levels of news-oriented retrieval queries. The known outcome is used only as

Table 8: Model versions and knowledge cutoffs used in our experiments.

Model	Knowledge Cutoff	Source
GPT-4o	October 2023	OpenAI documentation
Gemini 2.0 Flash	August 2024	Google model card
DeepSeek-V3	July 2024	Estimated from public information
Qwen2.5-72B-Instruct	June 2024	Estimated from public information
Qwen2.5-7B-Instruct	June 2024	Estimated from public information

supervision during training-data construction, while the generated queries are constrained to avoid direct outcome confirmation.

Hindsight Causal Query Generation

```
System prompt:
You are a query generator for pre-event forecasting.
Your job is to generate search queries that retrieve upstream NEWS ARTICLES, precursor signals, state checks
      , and background news coverage useful for forecasting a binary event before the outcome was publicly
      confirmed. The downstream retrieval pool contains news articles only; every query must be phrased to
      retrieve a specific news article.
You must NOT rely on retrieved articles.
You must NOT generate queries that merely retrieve the final answer.
You must generate candidate upstream queries from the resolution criteria, known ground truth, and plausible
       forecasting mechanisms, using only your pretraining knowledge.

All output must be in English. Output STRICT JSON ONLY.

User prompt:
You are given a binary prediction question, its Yes-resolution criteria, the known ground-truth answer, and
      the resolution date.

Your task is to perform Stages 1 through 4 sequentially and produce a single JSON object that contains the
      output of every stage. Use the known ground-truth answer as supervision to identify what kinds of
      upstream news, precursor signals, state checks, and background news coverage would have helped an
      analyst infer that correct answer before the outcome was publicly confirmed. However, the final
      generated queries must not be final-answer lookup queries. They must be upstream forecasting queries:
      queries that help recover the pre-outcome news needed to reason toward the known correct answer,
      rather than queries that directly retrieve the answer itself.

Input:
Question: {question}
Resolution date: {end_date}
Ground truth: {ground_truth}
Resolution criteria for Yes:
{criteria_yes}
Bucket: {bucket}

Stage 1: Criteria decomposition.
Decompose the resolution criteria into independently testable criterion legs. Identify the overall logical
      structure, such as SINGLE, AND, OR, THRESHOLD, ABSENCE, MIXED, or OTHER. For each leg, specify its
      leg_id, leg_name, leg_type, Yes condition, No condition, and forecast-relevant news types.

Stage 2: Candidate upstream mechanism generation.
For each criterion leg, generate candidate mechanisms that an analyst should search for before the
      resolution date. A candidate mechanism is a search-worthy hypothesis about what kind of upstream news
      could help forecast that leg, not a verified cause.

Stage 3: Candidate reasoning chain construction.
Construct candidate reasoning chains that connect upstream mechanisms to forecast-relevant logic. Each chain
       should connect an upstream setup, intermediate development, precursor or state check, effect on the
      criterion leg, and implication for the overall answer.

Stage 4: Four-level query generation.
Generate four levels of search queries. Each level must contain exactly 25 queries.

Level 1 is the most precise and should target specific upstream events, named actors, official statements,
      votes, lawsuits, agency announcements, economic data, market levels, regulatory actions, or pre-
      resolution state checks.

Level 2 uses medium-precision topical phrasings around the same upstream mechanisms as Level 1.
```

```
Level 3 contains broad issue-area queries that retrieve pre-outcome topic coverage while remaining aligned
      with the resolution criteria.

Level 4 contains background queries that provide contextual information for interpreting the event.

Each query must be phrased to retrieve a specific news article, such as the kind of headline a journalist
      would write. Avoid meta-analytic, retrospective, or umbrella-concept queries that no single news
      article would carry.

Each query must be formatted as:
{
  "query": "...",
  "explanation": "(QUERY_TYPE, leg_id) ..."
}

QUERY_TYPE must be one of UPSTREAM, PRECURSOR, STATE_CHECK, or BACKGROUND.
The explanation must be no longer than 25 words and must begin with the query type and criterion leg
      identifier.

Hard rules:
- Do not search for the final answer.
- Do not generate final confirmation queries.
- Do not use prediction-market or betting terminology.
- Do not include the exact resolution date verbatim.
- Do not fabricate unsupported exact numbers, dates, percentages, prices, vote totals, or official titles.
- For conjunctive criteria, every required leg must be covered in both Level 1 and Level 2.
- STATE_CHECK queries are allowed but must not dominate Level 1.

Output strict JSON only. The JSON must include:
- criteria_decomposition
- candidate_mechanisms
- candidate_reasoning_chains
- keywords_by_level
```

E.2 Causal Query Verification and Repair

The following prompt audits the generated causal queries before retrieval. It reclassifies each query, checks for outcome leakage, confirmation-oriented queries, temporal validity, news retrievability, criterion coverage, and level placement, and repairs problematic queries when necessary.

Causal Query Verification and Repair

```
System prompt:
You are an auditor and repairer for pre-event forecasting search queries.

Your job is to audit a generated query set and repair it so the final queries retrieve upstream news
      articles, precursor signals, state checks, and background news coverage.

You distinguish between:
- queries that help retrieve news articles and forecast the outcome before it was publicly confirmed
- queries that merely retrieve the final answer after the outcome was known

Preserve good queries. Repair only weak or contaminated queries. Ensure the final output satisfies the
      schema and quality rules.

All output must be in English. Output STRICT JSON ONLY. No prose outside the JSON. No markdown fences.

User prompt:
You are auditing generated search queries for pre-event forecasting quality.

Input:
Question: {question}
Resolution criteria: {criteria_yes}
Ground truth: {ground_truth}
Resolution date: {end_date}

Criteria decomposition:
{criteria_decomposition_json}

Generated queries:
{keywords_by_level_json}

Step A: Re-classify.
```

```
For every query in every level, independently determine its type from the query text itself. Do not trust
      the existing prefix in the explanation.

Possible labels:
UPSTREAM, PRECURSOR, STATE_CHECK, BACKGROUND, CONFIRMATION, LEAKY

Step B: Diagnose.
For each query, check whether it is leaky, confirmation-oriented, non-news-retrievable, placed in the wrong
      level, or based on unsupported exact figures or names. Also check whether every level contains exactly
       25 queries and whether every criterion leg is covered in Level 1 and Level 2.

Step C: Repair.
Repair problematic queries with minimal changes while preserving good queries unchanged.

Repair priority:
1. Remove leakage and final-answer wording.
2. Rewrite Level-1 confirmation queries into earlier upstream, precursor, or state-check queries.
3. Rewrite non-news-retrievable queries into specific news-article queries.
4. Replace weak or duplicate queries when a criterion leg is missing.
5. Fix wrong-level placements.
6. Remove unsupported exact numbers, dates, vote totals, or official titles.
7. Reduce excess STATE_CHECK queries in Level 1 if needed.

Each repaired query must preserve the format:
{
  "query": "...",
  "explanation": "(QUERY_TYPE, leg_id) ..."
}

Step D: Re-verify.
Re-classify the final repaired set of queries. The final audit summary must reflect the repaired set, not
      the original generated set.

Step E: Write the final JSON.
The output must contain:
- audit_summary
- repaired keywords_by_level

The final set passes only if:
- Level 1 contains zero CONFIRMATION queries and zero LEAKY queries.
- Level 1 contains at least 17 UPSTREAM or PRECURSOR queries.
- Level 1 contains at most 6 STATE_CHECK queries.
- Every criterion leg is covered in Level 1.
- Every criterion leg is covered in Level 2.
- Every level contains exactly 25 queries.

Output strict JSON only. Do not include prose outside the JSON.
```