ALPHABET: REINFORCEMENT LEARNING FOR LLM EVENT FORECASTING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Forecasting an unresolved event requires finding relevant evidence and deciding how strongly it supports each outcome. Yet outcome-based training often starts from preassembled evidence at a single prediction date, leaving evidence acquisition outside the learning loop. We introduce ALPHABET, a reinforcement learning framework that trains a shared LLM policy to search, reason, and report probabilities across multiple pre-resolution dates. Each dated forecast receives proper-scoring feedback from the eventual outcome, allowing one resolved event to supervise decisions under different information states. We decompose forecasting regret into gaps from evidence acquisition and probability reporting, and show that GRPO standard-deviation normalization can distort the reporting target under multi-class proper scores. On GLM-4.5-Air, ALPHABET reduces Brier from 0.731 to 0.626 on Forecast-Dojo and from 0.612 to 0.528 on FutureX. Controlled ablations attribute these gains to longitudinal training and improvements in both acquisition and reporting, while showing that GRPO standard-deviation normalization harms probabilistic forecasting while accuracy remains comparable.

1 INTRODUCTION

Many real-world event forecasting questions, such as policy decisions or economic indicators, remain unresolved for weeks or months, while the relevant evidence continues to accumulate. Human forecasters have long practiced *longitudinal forecasting*, revisiting the same unresolved event and updating their beliefs as information evolves (Mellers et al., 2014; Atanasov et al., 2020). LLM agents can retrieve and reason over evidence and produce probabilistic forecasts (Zeng et al., 2026; Karger et al., 2025). Recent LLM forecasting benchmarks have begun to assess this temporal dimension more directly, evaluating their forecasts at different points before resolution (Goel et al., 2026; Yuan et al., 2025; Ye et al., 2026; Zhao et al., 2026).

Yet existing outcome-based forecasting methods still train only part of this process. They commonly optimize probability reporting from preassembled evidence at a single prediction time, leaving evidence acquisition outside the trained policy and providing no learning signal for how forecasts should evolve as information changes (Halawi et al., 2024; Turtel et al., 2025; Chandak et al., 2025; Jeen et al., 2026). Forecast-Dojo (Ye et al., 2026) provides a replayable, temporally indexed forecasting research environment that makes this broader learning problem accessible. This motivates our central question: *how can outcome feedback train evidence-seeking and probability-reporting behavior across changing information states?*

We introduce ALPHABET (Figure 1), an outcome-based RL framework for agentic event forecasting. Using Forecast-Dojo's temporally indexed environment, a single tool-using policy revisits each training event at multiple pre-resolution dates, actively searches and reads the evidence available at each date, and reports a distribution over outcomes. Each dated forecast receives proper-score feedback against the eventual outcome, so one resolved event supervises decisions under multiple information states. Cross-date memory is optional: the memory-free variant starts each dated forecast afresh, while the memory-on variant carries a belief notebook across dates. We show that the resulting forecasting regret decomposes into an information-acquisition gap and a probability-reporting gap (Proposition 1). For policy optimization, we use GRPO (Shao et al., 2024) without within-group standard-deviation normalization (Turtel et al., 2025; Chandak et al., 2025; Liu et al., 2025b). We provide a proper-scoring interpretation of this choice. Extending the binary analysis of

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.

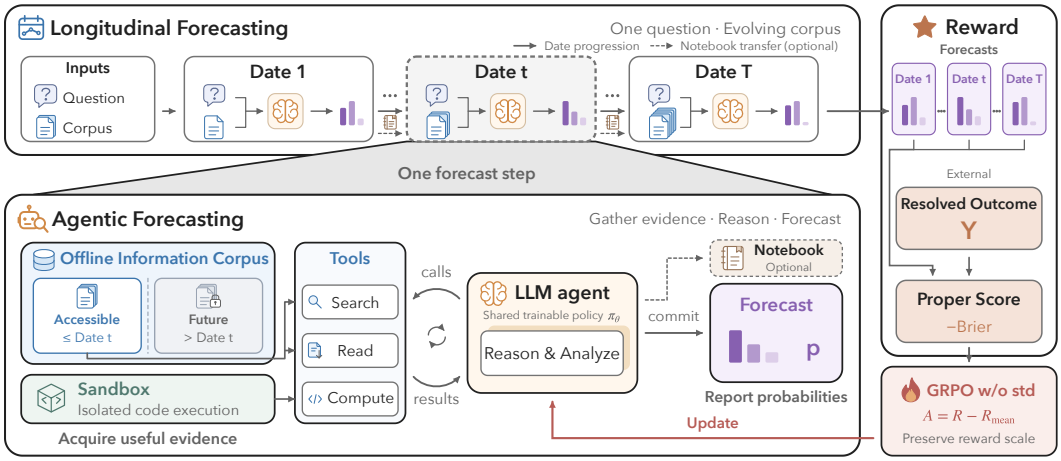


Figure 1: **Overview of ALPHABET.** One shared policy searches, reads, and reasons over the corpus available at each forecast date before reporting probabilities. Each report is scored against the eventual outcome, which is hidden during rollout. Training uses per-date feedback with mean-centered GRPO; a notebook optionally carries context between dates.

Bereket & Leskovec (2025), we show that, at a fixed acquired context, mean-centering preserves the expected reporting gradient induced by a strictly proper score, whereas standardization can reweight outcome classes and change the elicited probability target.

Empirically, ALPHABETimproves forecasting performance on both Forecast-Dojo and FutureX. Our strongest variant improves GLM-4.5-Air from 41.6% to 50.5% accuracy and from 0.731 to 0.626 Brier on Forecast-Dojo evaluation set, and from 51.3% to 57.9% accuracy and from 0.612 to 0.528 Brier on FutureX, outperforming forecasting harnesses and prior RL-based training methods while remaining competitive with substantially larger off-the-shelf models. Multi-date training consistently improves over single-date training. Controlled evidence–reader swaps show improvements in both evidence acquisition and probability reporting. Finally, GRPO ablations support our proper-scoring analysis: within-group standardization degrades probabilistic forecast quality, while removing it preserves the intended training signal.

Our contributions are:

- We introduce ALPHABET, an outcome-based agentic RL framework that trains a single policy to actively acquire evidence and report probabilistic forecasts across multiple pre-resolution dates.
- We characterize outcome-based forecasting across evolving information states: a single realized outcome can properly supervise multiple dated forecasts, forecasting regret decomposes into information-acquisition and probability-reporting gaps, and GRPO normalization can alter the intended multi-class probability target.
- We establish empirically that multi-date training improves forecasting across two benchmarks, with gains arising from both better evidence acquisition and better probability reporting. GRPO ablations further validate our analysis of normalization and probabilistic forecast quality.

2 PRELIMINARIES

2.1 LONGITUDINAL EVENT FORECASTING

Forecasting setup. Each question Q describes an event in natural language and specifies a finite set of mutually exclusive and exhaustive outcomes $\mathcal{Y} = \{1, \ldots, K\}$. Its resolution criterion specifies how the realized outcome $Y \in \mathcal{Y}$ is determined. During the question's active period $[\tau_{\text{start}}, \tau_{\text{close}}]$, the forecaster revisits Q before resolution at scheduled dates $\tau_{\text{start}} \leq \tau_1 < \cdots < \tau_T \leq \tau_{\text{close}}$. We index these visits by $t = 1, \ldots, T$ and call each a *forecast step*: step t occurs on date τ_t and produces a probability report $p_{Q,t} \in \Delta^{K-1}$ over $\mathcal{Y}$. The ordered sequence of steps forms a *forecast episode*.

Evolving information corpus. The question remains fixed across these steps, but the evidence available to the forecaster evolves. At date τ_t, it can consult $\mathcal{I}_{\leq \tau_t}$, the corpus of information avail-

able no later than that date. Sources released later are excluded (Zou et al., 2022; Halawi et al., 2024). As new information arrives, the admissible corpus expands: $\mathcal{I}_{\leq\tau_t} \subseteq \mathcal{I}_{\leq\tau_{t+1}}$. The forecaster typically observes only part of this corpus. In our setting, an LLM agent uses tools to retrieve and analyze evidence, interleaving these calls with reasoning. We denote by H_t the all evidence the agent retrieved from the corpus before it reports $p_{Q,t}$: its articles returned by their search and read tool calls. What the agent itself writes, its reasoning, code, and notes, is not part of H_t

Forecast targets. One question's forecasts share the same eventual outcome Y, but the relevant information changes with the forecast date. We therefore distinguish two targets: the posterior supported by the full dated corpus and the posterior supported by the context the agent actually acquires:

$$q_{Q,t}(k) \triangleq \Pr(Y = k \mid Q, \tau_t, \mathcal{I}_{\leq\tau_t}) \quad (1) \qquad q^H_{Q,t}(k) \triangleq \Pr(Y = k \mid Q, \tau_t, H_t) \quad (2)$$

$q_{Q,t}$ is the *dated-corpus target*, the Bayes-optimal forecast given all admissible information available by τ_t. $q^H_{Q,t}$ is the *acquired-context target*, the Bayes-optimal forecast given H_t. The agent reports $p_{Q,t}$ based on this acquired context. These three quantities separate what is inferable from the dated corpus, what is inferable from the acquired evidence by agent, and what the agent ultimately reports.

2.2 Proper Scoring Rules

Evaluating a probabilistic forecast requires more than checking its most likely outcome: forecasts that choose the same outcome can assign very different probabilities. Proper scoring rules score these probabilities directly and make truthful reporting optimal in expectation (Gneiting & Raftery, 2007), providing both an evaluation criterion and a training signal from the reported distribution and eventual outcome without observing the latent target. Formally, S is *strictly proper* if, for every outcome distribution q and report p,

$$\mathbb{E}_{Y\sim q}[S(q, Y)] \geq \mathbb{E}_{Y\sim q}[S(p, Y)], \qquad \text{with equality if and only if } p = q. \quad (3)$$

Conditioned on an acquired context H_t, strict properness therefore makes $q^H_{Q,t}$ the unique optimal report in expectation. Relative to the dated-corpus target $q_{Q,t}$, this distinguishes an *information gap* between $q_{Q,t}$ and $q^H_{Q,t}$ from a *reporting gap* between $q^H_{Q,t}$ and $p_{Q,t}$.

3 Method

3.1 RL Environment for Longitudinal Forecasting

We train a single policy to acquire evidence and report probabilities across multiple pre-resolution dates, using the eventual outcome as feedback at each date. Following Forecast-Dojo (Ye et al., 2026), the agent gathers and reasons over evidence at each date before reporting a forecast, optionally carrying a belief notebook forward.

Agent loop within a forecast step. Forecast step t begins with the question Q, the forecast date τ_t, and the belief notebook M_{t-1} from the preceding step when external memory is enabled. The agent then interleaves reasoning with three research capabilities for gathering and analyzing evidence:

- Search returns the top-k relevant article snapshots for an agent-generated query from the date-restricted corpus $\mathcal{I}_{\leq\tau_t}$ via `search(query,top-k)`.
- Read retrieves the full text of a previously found article, via `scrape(article_id)`.
- Compute executes model-generated Python code for numerical analysis, aggregation, base-rate estimation, or simulation, via `python(code)`.

Each tool call result extends the agent's context and can prompt further reasoning or another call. The retrieved evidence H_t is the record of search and read calls' results. Everything else in the context, the agent's reasoning and code outputs is built from this evidence. The agent completes the step by committing $p_t \in \Delta^{K-1}$. When external memory is enabled, the agent additionally produces an updated belief notebook M_t, which summarizes its current belief, unresolved questions, and accumulated evidence for use at the next forecast step.

Progression between forecast steps. After completing the forecast at τ_t, the environment advances to the next scheduled date τ_{t+1}, expanding the admissible information from $\mathcal{I}_{\leq\tau_t}$ to $\mathcal{I}_{\leq\tau_{t+1}}$. When external memory is disabled, each forecast step starts without agent-generated belief notebook

from earlier dates. We refer to this as *memory-free forecasting*. When external memory is enabled, the belief notebook M_t is carried into the next forecast step, yielding *memory-on forecasting*. We repeat this process through the final scheduled date τ_T to complete the episode.

3.2 Training Rewards and Objective

Training reward. Our reward for forecast step t is defined as a sum of a forecasting score and a format reward:

$$R_t = S(p_t, Y) + R_t^{\text{format}}, \qquad R_t^{\text{format}} = \lambda_{\text{ans}}\big(f_t^{\text{ans}} - 1\big) + \lambda_{\text{tool}}\big(f_t^{\text{tool}} - 1\big). \tag{4}$$

Our main experiments use the negative Brier score (Section 2.2), while the GRPO normalization ablation also considers the logarithmic score. Exact properness holds for the ideal log score under finite expectation, but not for its probability-floored implementation on the full simplex (Appendix A.3). The format reward enforces a valid interface: $f_t^{\text{ans}} \in [0,1]$ checks the answer format and, when memory is enabled, the notebook structure and its agreement with the forecast, while $f_t^{\text{tool}} \in [0,1]$ checks whether tool calls use names recognized by the environment. We provide full definitions, coefficients, and score implementation in Appendix B.4.

Training objective. Let $\mathcal{D}$ denote the training distribution over resolved forecasting questions. We maximize expected cumulative reward across forecast dates, with expectation over policy rollouts:

$$\max_\theta \underbrace{\mathbb{E}_{Q\sim\mathcal{D}}\left[\sum_{t=1}^{T_Q} R_{Q,t}\right]}_{J_{\text{train}}(\theta)} = \max_\theta \left\{ \underbrace{\mathbb{E}_{Q\sim\mathcal{D}}\left[\sum_{t=1}^{T_Q} S(p_{Q,t}, Y_Q)\right]}_{J_{\text{forecast}}(\theta)} + \underbrace{\mathbb{E}_{Q\sim\mathcal{D}}\left[\sum_{t=1}^{T_Q} R_{Q,t}^{\text{format}}\right]}_{J_{\text{format}}(\theta)} \right\}. \tag{5}$$

The format term satisfies $J_{\text{format}}(\theta) \le 0$ and vanishes for valid outputs. We assume there exists a policy that maximizes J_{forecast} while satisfying all format checks. Then, every maximizer of J_{train} is also forecasting-optimal, $\max_\theta J_{\text{train}}(\theta) = \max_\theta J_{\text{forecast}}(\theta)$. Appendix A.1 gives the proof.

Improvements in J_{forecast} can arise through two complementary capabilities: acquiring relevant information and translating that information into accurate probabilistic forecasts. Proposition 1 shows that our objective captures both.

Proposition 1 (Properness of the agentic forecasting objective). *Assume there is no outcome leakage beyond the admissible dated information, as formalized in Appendix A.2. Let S be a strictly proper score with finite expected score. For the targets $q_{Q,t}$ and $q_{Q,t}^H$ defined in Section 2.1, define*

$$D_S(q,p) \triangleq \mathbb{E}_{Y\sim q}[S(q,Y) - S(p,Y)], \qquad J^\star \triangleq \mathbb{E}_{Q\sim\mathcal{D}}\left[\sum_{t=1}^{T_Q} S(q_{Q,t}, Y_Q)\right]. \tag{6}$$

Here $D_S(q,p)$ is the expected proper-score discrepancy between q and p. $J^\star$ is the oracle forecasting value obtained by reporting the dated-corpus target $q_{Q,t}$ at every step. Then

$$\begin{aligned} \arg\max_\theta J_{\text{forecast}}(\theta) &= \arg\min_\theta \left[J^\star - J_{\text{forecast}}(\theta)\right] \\ &= \arg\min_\theta \mathbb{E}_{Q\sim\mathcal{D}}\left[\sum_{t=1}^{T_Q}\left(\underbrace{D_S(q_{Q,t}, q_{Q,t}^H)}_{\textit{information gap}} + \underbrace{D_S(q_{Q,t}^H, p_{Q,t})}_{\textit{reporting gap}}\right)\right]. \end{aligned} \tag{7}$$

A policy has zero total gap if and only if $p_{Q,t} = q_{Q,t}^H = q_{Q,t}$ at every forecast step, almost surely.

Under the assumption above, the auxiliary format reward does not alter the forecasting optimum. We therefore analyze J_{forecast} directly. Proposition 1 shows that maximizing J_{forecast} is exactly equivalent to minimizing the expected sum of the information and reporting gaps. This decomposition identifies two distinct sources of suboptimality, but does not imply that joint optimization reduces each separately: gains may come from better information acquisition, better reporting, or both. For a fixed acquired context, strict properness favors $p_{Q,t} = q_{Q,t}^H$, while improved evidence acquisition can reduce the information gap. We provide the full proof of Proposition 1 in Appendix A.2.

3.3 POLICY OPTIMIZATION: GRPO FOR CALIBRATED FORECASTING

We use GRPO for policy optimization but omit its within-group standard-deviation normalization. This no-std variant has been adopted in recent forecasting work (Turtel et al., 2025; Chandak et al., 2025) and also appears in Dr. GRPO (Liu et al., 2025b) in a different optimization setting. Bereket & Leskovec (2025) show that standardization can induce overconfidence for binary stochastic outcomes. We extend this analysis to multi-class proper scoring rules: at a fixed acquired context, mean-centering preserves the expected score-only reporting gradient up to a positive factor, whereas standardization can reweight outcome classes and shift the elicited probability target.

Vanilla GRPO. GRPO compares multiple rollouts in a group and constructs relative advantages without a learned value function. For rewards $\{R^{(i)}\}_{i=1}^{G}$, the standard normalized advantage is

$$A_{\text{std}}^{(i)} = \frac{R^{(i)} - \bar{R}}{\sigma_R + \epsilon}, \qquad \bar{R} = \frac{1}{G}\sum_{j=1}^{G} R^{(j)}, \tag{8}$$

where σ_R is the within-group standard deviation and $\epsilon > 0$ ensures numerical stability.

Removing reward standardization. Vanilla GRPO both mean-centers and standardizes group rewards. Mean-centering provides a relative baseline, whereas standardization additionally removes reward scale. In probabilistic forecasting, this can distort the proper-score target because the normalization scale depends on both the current policy and the realized outcome, rescaling different outcome classes differently. We therefore retain mean-centering but remove std normalization:

$$A_{Q,t}^{(i)} = R_{Q,t}^{(i)} - \bar{R}_{Q,t}, \qquad \bar{R}_{Q,t} = \frac{1}{G}\sum_{j=1}^{G} R_{Q,t}^{(j)}. \tag{9}$$

The following proposition makes the distinction precise.

Proposition 2 (Proper-score targets under GRPO normalization). *Fix a question Q, forecast date τ_t, and acquired context $H_{Q,t}$, with $q_{Q,t}^{H} \in \operatorname{int}\Delta^{K-1}$ and $K \geq 2$. Let S be a strictly proper scoring rule. Under the regularity conditions of Appendix A.4, draw $G > 1$ independent reports from the current policy, independently of the common outcome conditional on $(Q, \tau_t, H_{Q,t})$. For this proposition, let $R_{Q,t}^{(i)} = S(p_{Q,t}^{(i)}, Y_Q)$, excluding format penalties.*

(i) *For the mean-centered advantage in Eq. 9, the expected score-function update equals $\left(1 - \frac{1}{G}\right)$ times the corresponding uncentered update. Thus mean-centering preserves the expected score-only reporting gradient direction.*

(ii) *Let $\sigma_{Q,t,k}^{S}$ denote the standard deviation of $S(p_{Q,t}, k)$ under the current policy's induced distribution over reports. For fixed $\epsilon > 0$, as $G \to \infty$, the expected standardized update is locally equivalent, up to a positive factor, to the proper-score update associated with*

$$\widetilde{q}_{Q,t}^{H}(k) = \frac{q_{Q,t}^{H}(k)/(\sigma_{Q,t,k}^{S} + \epsilon)}{\sum_{j=1}^{K} q_{Q,t}^{H}(j)/(\sigma_{Q,t,j}^{S} + \epsilon)}. \tag{10}$$

Therefore $\widetilde{q}_{Q,t}^{H} = q_{Q,t}^{H}$ if and only if $\sigma_{Q,t,1}^{S} = \cdots = \sigma_{Q,t,K}^{S}$; otherwise, standard-deviation normalization induces a policy-dependent tilted reporting target.

In this fixed-context, score-only analysis, the group mean preserves the expected reporting gradient, whereas standardization reweights outcome classes according to their reward scales in the large-group limit. This distortion need not always take the form of overconfidence; depending on the outcome-wise scales, it may sharpen the distribution, flatten it, or shift probability across classes. Appendix A.4 proves part (i), and Appendix A.5 derives the multiclass effective target in part (ii). This result motivates our normalization choice but characterizes only the local score-only reporting update, not calibration of the complete trained agent.

Group construction. For memory-free forecasting, we use the standard GRPO construction: each group contains G rollouts for the same question and forecast date (Q, τ_t). Memory-on forecasting differs because the forecast at τ_t depends on the belief notebook accumulated at earlier dates. We therefore sample G complete episodes for the same question and form the group at each date from the corresponding forecast steps across episodes. Each rollout carries its own notebook trajectory.

Table 1: **Main results.** Bold and underlined values mark the best and second-best point estimates, respectively, within the GLM-4.5-Air block. Brackets give 95% confidence intervals (event-clustered bootstrap for Forecast-Dojo; item bootstrap for FutureX).

	Forecast-Dojo			FutureX	
Model	Acc. (%) ↑	Brier ↓	−Info-α ↓	Acc. (%) ↑	Brier ↓
Uniform Reference	22.9 [20.8, 25.1]	0.771 [0.749, 0.792]	0.727 [0.624, 0.830]	30.6 [28.7, 32.4]	0.680 [0.662, 0.699]
Claude Sonnet 4.6_{high}	50.0 [44.5, 55.5]	0.624 [0.573, 0.675]	0.364 [0.253, 0.480]	58.9 [54.2, 63.6]	0.497 [0.444, 0.552]
Kimi K2.5	48.9 [43.8, 54.3]	0.663 [0.614, 0.710]	0.530 [0.414, 0.650]	57.5 [53.0, 61.9]	0.514 [0.464, 0.564]
Qwen3.5-397B-A17B$_{\text{think}}$	48.2 [43.1, 53.5]	0.653 [0.606, 0.704]	0.482 [0.373, 0.608]	57.1 [52.5, 61.5]	0.556 [0.503, 0.607]
Claude Sonnet 4.6_{low}	47.6 [41.6, 53.2]	0.656 [0.598, 0.714]	0.468 [0.346, 0.604]	54.2 [49.4, 59.1]	0.564 [0.506, 0.619]
MiniMax-M2.5	45.9 [40.9, 51.2]	0.719 [0.670, 0.768]	0.738 [0.608, 0.873]	53.3 [48.8, 57.7]	0.555 [0.505, 0.606]
GLM-4.5-Air	41.6 [36.6, 46.9]	0.731 [0.685, 0.777]	0.726 [0.616, 0.877]	51.3 [47.0, 55.5]	0.612 [0.561, 0.664]
+BLF	45.8 [40.5, 51.4]	0.671 [0.630, 0.711]	0.458 [0.365, 0.555]	51.1 [45.9, 56.2]	0.592 [0.547, 0.639]
+OpenForecaster	42.8 [38.0, 48.1]	0.765 [0.719, 0.809]	0.845 [0.742, 0.957]	52.6 [48.5, 56.7]	0.583 [0.545, 0.622]
+ALPHABET$_{\text{single-date}}$	47.2 [42.1, 52.3]	0.668 [0.619, 0.714]	0.507 [0.406, 0.619]	53.2 [48.7, 57.7]	<u>0.540</u> [0.494, 0.590]
+ALPHABET$_{\text{memory-on}}$	<u>49.3</u> [43.7, 54.7]	<u>0.630</u> [0.577, 0.685]	<u>0.413</u> [0.313, 0.516]	<u>55.2</u> [50.4, 59.8]	0.545 [0.499, 0.592]
+ALPHABET$_{\text{memory-free}}$	**50.5** [45.3, 55.9]	**0.626** [0.574, 0.677]	**0.399** [0.304, 0.502]	**57.9** [53.4, 62.4]	**0.528** [0.483, 0.576]

We use the immediate reward at each forecast date without additional cross-date credit assignment. Thus, for memory-on episodes, the update is a per-date surrogate rather than the full episode-return gradient: earlier notebook actions may affect later forecasts but are not explicitly assigned those future rewards. Cross-date credit assignment remains future work.

4 EXPERIMENTS

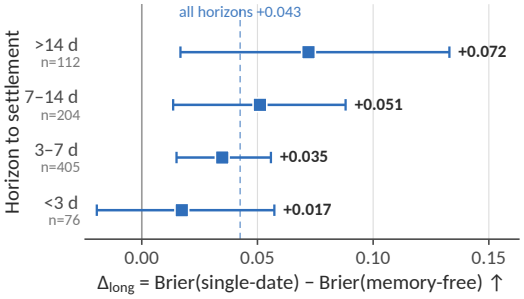
4.1 EXPERIMENTAL SETUP

Training setup. We instantiate ALPHABETwith GLM-4.5-Air (106B-A12B) (GLM, 2025), whose public checkpoint was released on `07-28-2025`. We train on the Forecast-Dojo training split (Ye et al., 2026), comprising 1,028 binary and multi-option events (3,965 event–date tasks) from Polymarket, with forecast dates spanning `08-01-2025`–`02-28-2026`. Because all training events postdate the checkpoint release, their realized outcomes cannot be directly memorized from its pre-training data. We use a disjoint held-out validation set covering `01-01-2026` through `02-28-2026` for checkpoint selection and hyperparameter tuning. We evaluate three variants: *single-date*, which uses one forecast date for each event (Section 4.4); *memory-free*, which uses multi-date training without cross-date memory; and *memory-on*, which carries the rolling belief state across forecast dates. We directly use the temporally indexed CC-News corpus released with Forecast-Dojo spanning `03-2025` through `02-2026`, restricting retrieval at each forecast date to articles available by that date. We optimize negative Brier score together with the format reward defined in Section 3.2. Full implementation details are provided in Appendix B.

Evaluation benchmarks. We evaluate on two forecasting benchmarks covering March through May 2026. *Forecast-Dojo* uses its held-out test events from this period (230 events, 797 event–date pairs), while for *FutureX* (Zeng et al., 2026) we evaluate on L1–L2 questions (343 events), retaining its question-specific answer interface within our controlled offline-news setting. We keep the search infrastructure and temporally indexed corpus the same as training, extended through May 2026, with retrieval restricted to information available by each forecast date. Appendix C.1 provides further details on the characteristics and distribution of the evaluation data.

Table 2: **Performance gain analysis.** Entries are Brier scores (↓). Brackets denote 95% event-clustered bootstrap confidence intervals. $^{*}p < .05$, $^{**}p < .01$, and $^{***}p < .001$.

	Provided Evidence			
Reader	H_{Base}	H_{ALPHABET}	None	$\boldsymbol{\Delta}_{\mathbf{acq}}$ ↑
GPT-5.6 $\text{Terra}_{\text{xhigh}}$	0.623 [0.576, 0.671]	0.612 [0.566, 0.659]	0.725 [0.673, 0.778]	$+0.011^{*}$
Claude Sonnet 4.6_{high}	0.649 [0.600, 0.701]	0.638 [0.590, 0.688]	0.763 [0.709, 0.815]	$+0.012^{*}$
GLM-4.5-Air	0.702 [0.661, 0.743]	0.686 [0.644, 0.727]	0.801 [0.755, 0.847]	$+0.016^{**}$
+$\text{ALPHABET}_{\text{memory-free}}$	0.642 [0.592, 0.694]	0.628 [0.575, 0.681]	0.765 [0.713, 0.816]	$+0.014^{*}$
$\boldsymbol{\Delta}_{\mathbf{report}}$ ↑	$+0.060^{***}$	$+0.058^{***}$	$+0.035^{**}$	



Reward	**ΔBrier** ↑	**ΔLog** ↑	**REL** ↓ std / no-std
Brier	$+0.050^{*}$ [0.011, 0.091]	$+0.326^{***}$ [0.181, 0.488]	0.054/0.004
Log	$+0.093^{***}$ [0.038, 0.147]	$+0.589^{***}$ [0.392, 0.794]	0.095/0.007

$\Delta\text{Brier} = \text{Brier}_{\text{std}} - \text{Brier}_{\text{no-std}}$ and $\Delta\text{Log} = \text{Log}_{\text{no-std}} - \text{Log}_{\text{std}}$, so positive values favor no-std GRPO. Brackets give paired event-clustered 95% CIs. $^{*}p < .05$, $^{***}p < .001$.

Figure 2: **(a)** Brier gain from longitudinal memory-free over single-date training across forecast horizons; positive values favor longitudinal training. **(b)** Final-checkpoint proper-score gains from removing GRPO standard-deviation normalization, with Murphy reliability error.

Baselines. We include a uniform reference that assigns probability $1/K$ to each of the K answer options, providing a non-informative baseline. We compare against the untrained GLM-4.5-Air base model, which isolates the effect of forecasting-specific training. We additionally compare against two forecasting-specific alternatives built on the same backbone: BLF (Murphy, 2026), a forecasting harness method, and OpenForecaster (Chandak et al., 2025), an RL-based forecasting training method. To contextualize performance beyond same-backbone comparisons, we also evaluate a set of larger and stronger off-the-shelf models—Claude Sonnet 4.6 (Anthropic, 2026), Kimi K2.5 (Kimi Team, 2026), Qwen3.5-397B-A17B (Qwen Team, 2026), MiniMax-M2.5 (MiniMax, 2026))—using the same agent loop and tool budget. We treat these models as references rather than controlled baselines. For every off-the-shelf model, either its officially reported knowledge cutoff or the checkpoint released time was before the evaluation-event period (Appendix C.2).

Metrics. We report **Accuracy**: top-1 accuracy on Forecast-Dojo and the benchmark's official answer-matching accuracy on FutureX (Appendix C.3). We also report the multiclass **Brier score** $\text{Brier}(p, y) = \sum_{k=1}^{K} \big(p_k - \mathbb{1}[k = y]\big)^2$, where lower is better. We also report **Information Alpha** (Info-α) for Forecast-Dojo, the log-score gain over the crowd forecast p^{crowd}, $\text{Info-}\alpha = \log p_y - \log p_y^{\text{crowd}}$. Positive values favor the model. Table 1 reports $-\text{Info-}\alpha$ so that lower is better. For the GRPO ablation, we additionally use the Murphy decomposition $\text{Brier} = \text{REL} - \text{RES} + \text{UNC}$ (Murphy, 1973) to distinguish reliability from resolution.

4.2 Forecasting Performance

Table 1 shows that ALPHABETimproves GLM-4.5-Air consistently across both benchmarks, including a 8.9-point accuracy gain and a Brier reduction from 0.731 to 0.626 on Forecast-Dojo. It also outperforms both the harness and RL training baselines. OpenForecaster gains less, likely because it is optimized on fixed retrieved context rather than through the agentic loop. At test time it searches and reads less than the base model (8.3 vs. 12.4 tool calls per forecast). Its reporting improvements therefore operate on a thinner evidence base, reflecting the information gap in Proposition 1.

ALPHABET makes GLM-4.5-Air competitive with much stronger off-the-shelf forecasters, outperforming most open-weight baselines and approaching Claude Sonnet 4.6 (high) across both bench-

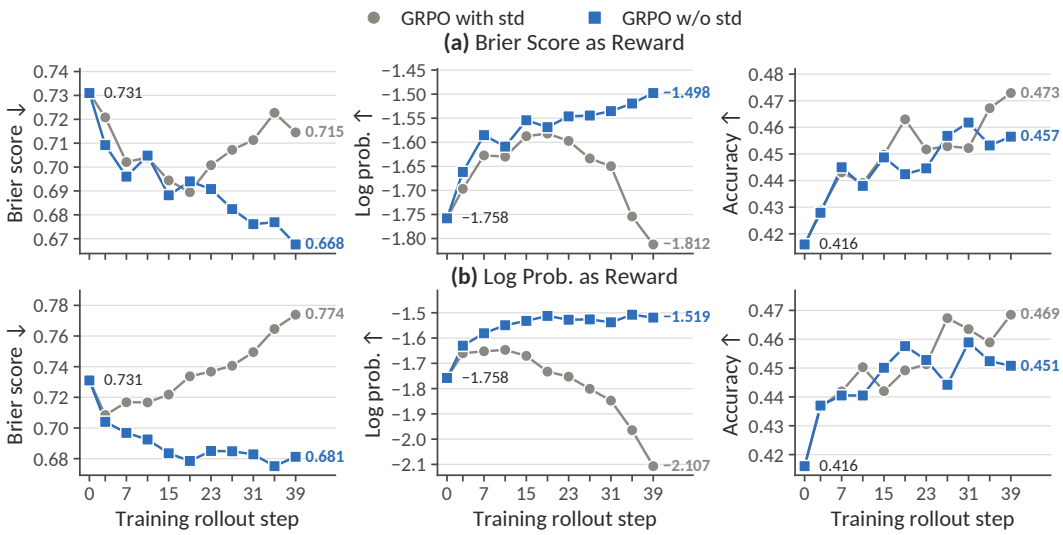


Figure 3: Effect of GRPO reward standardization. Std normalization progressively degrades proper-score performance under both Brier and log-score rewards, while accuracy remains comparable.

marks. Interestingly, adding memory does not improve further: ALPHABET$_{\text{memory-on}}$ is slightly worse than the memory-free variant on every metric, though still stronger than the single-date and base-model baselines on most metrics. A plausible explanation is that each forecast date is rewarded independently, so the notebook receives no direct credit for helping later forecasts. We leave explicit cross-date credit assignment to future work.

4.3 Gain Analysis: Evidence Acquisition and Probability Reporting

Proposition 1 separates forecasting quality into evidence acquisition and reporting, but an end-to-end agent entangles the two as it both gathers evidence and predicts from it. To disentangle them, we extract the article sets retrieved by the base model and by ALPHABET, H_{Base} and H_{ALPHABET}, and ask tool-free readers to forecast from these frozen evidence sets under the same prompt and scoring protocol. With evidence fixed, swapping readers isolates reporting differences; with the reader fixed, swapping H_{Base} and H_{ALPHABET} measures the downstream value of the acquired evidence. We repeat the latter comparison with GPT-5.6 Terra (OpenAI, 2026) and Claude Sonnet 4.6 as independent readers to test whether H_{ALPHABET} is useful beyond ALPHABET itself. We also include a question-only condition to control for each reader's default reporting bias in the absence of retrieved evidence. Appendix C.4 gives the details of this analysis setup.

Table 2 shows gains from both evidence acquisition and reporting. Every reader, including the independently trained GPT-5.6 Terra and Claude Sonnet 4.6, performs better on H_{ALPHABET} than on H_{Base} (0.011–0.016 Brier), indicating that ALPHABETretrieves broadly more useful evidence rather than evidence tailored only to its own reader. Reporting also improves substantially: with the same evidence, ALPHABEToutperforms the base by 0.058–0.060 Brier. Its 0.035 advantage in the question-only condition accounts for part of this gap, while the additional gain with evidence suggests that ALPHABETalso extracts more value from what it reads.

4.4 Longitudinal Supervision

To isolate the benefit of training across multiple dated states, we compare against ALPHABET$_{\text{single-date}}$, which retains one fixed forecast date for each of the same 1,028 training events. We stratify events evenly across early, middle, and late relative forecast positions to avoid bias toward a particular horizon, yielding 1,028 training states versus 3,965 event–date states for longitudinal training. All other training settings and checkpoint selection are held fixed, giving both runs the same rollout/update budget at each step. Appendix B.1 provides the full construction.

On Forecast-Dojo, ALPHABET$_{\text{memory-free}}$ further improves over ALPHABET$_{\text{single-date}}$, reducing Brier from 0.668 to 0.626 and raising accuracy from 47.2% to 50.5%. The same ordering holds on FutureX (Table 1). The longitudinal Brier gain is 0.072 beyond 14 days, 0.051 at 7–14 days, 0.035 at 3–7

days, and 0.017 within three days (Figure 2a). The gain is positive across horizons and largest farther from settlement, suggesting that longitudinal training is especially helpful for earlier forecasts.

4.5 Effect of GRPO Reward Standardization

Proposition 2 shows that GRPO's within-group standard-deviation normalization can distort the reporting target induced by a proper scoring rule. We test this with controlled GLM-4.5-Air ablations under two proper scoring rules, Brier and log scores, differing only in whether reward standardization is applied. We re-evaluate every saved checkpoint on Forecast-Dojo and compare each std/no-std pair on their common-valid forecasts, isolating probability quality from output-validity effects. Appendix C.5 provides the detailed setup and validity analysis.

Figure 3 shows the same pattern under both objectives: standardization progressively worsens both proper scores, including the one not used for training, while top-1 accuracy remains broadly comparable. Final-checkpoint comparisons confirm significant proper-score gains from removing standardization, with no significant accuracy difference (Figure 2b). Murphy decomposition attributes the degradation primarily to reliability rather than resolution, indicating a distortion in probability reporting rather than outcome ranking. In our runs, this distortion manifests as excessive sharpening; additional diagnostics are provided in Appendix C.5.

5 Related Work

Event forecasting benchmarks and environments. ForecastQA and AutoCast study event prediction from time-restricted news (Jin et al., 2021; Zou et al., 2022). ForecastBench and FutureX complement these historical benchmarks with live evaluations on unresolved questions, refreshing their question sets as new events arise (Karger et al., 2025; Zeng et al., 2026). Agent environments make evidence gathering part of the evaluation: MIRAI provides tools over historical events and news (Ye et al., 2024), while FutureSim evaluates adaptation through chronological news replay (Goel et al., 2026). Forecast-Dojo aligns forecast dates across agents, restricts research tools to dated sources, and demonstrates supervised fine-tuning on the resulting trajectories (Ye et al., 2026). AlphaBet uses this environment for outcome-based RL, training the agent across changing information states of the same event.

Learning to forecast with language models. AutoCast++ combines news reranking and summarization with supervision from dated human forecasts (Yan et al., 2024), while other work fine-tunes on selected reasoning or forecast traces (Halawi et al., 2024) or maintains explicit beliefs during iterative tool use (Murphy, 2026). Outcome-based RL instead scores forecasts against resolved events (Turtel et al., 2025; 2026); OpenForecaster scales this setup with news-derived questions and offline retrieval (Chandak et al., 2025). AlphaBet extends these ideas to longitudinal forecasting, jointly training evidence acquisition and probability reporting across multiple pre-resolution states of the same event with proper-score feedback at each date.

Reward normalization and probabilistic calibration. Prior work improves calibrated confidence through supervised fine-tuning (Lin et al., 2022), reinforcement learning (Band et al., 2024), and proper-scoring objectives (Gneiting & Raftery, 2007). However, the RL update itself can alter these incentives: Bereket & Leskovec (2025) show that GRPO reward standardization (Shao et al., 2024) can induce overconfidence for binary stochastic outcomes. Removing this normalization has also been explored in Dr. GRPO (Liu et al., 2025b) and forecasting RL (Turtel et al., 2025; Chandak et al., 2025). AlphaBet extends this analysis to multi-class proper scores, characterizing how standardization shifts the reporting target and testing its effect in agentic forecasting.

6 Conclusion

We introduced AlphaBet, an outcome-based RL framework that trains a single LLM agent to acquire evidence and report probabilities across multiple pre-resolution dates of the same event. Our analysis decomposes forecasting regret into information and reporting gaps and explains how GRPO reward standardization can shift the probability target. Experiments on Forecast-Dojo and FutureX show improved forecast quality and consistent gains over single-date training. Controlled evidence–reader swaps further show that training improves both the usefulness of retrieved evidence and the quality of probability reporting.

AI USE STATEMENT

AI in the experiments. LLMs serve as forecasting agents that acquire evidence and report probabilities (Sections 3 and 4). We also use tool-free LLM readers to evaluate frozen evidence sets in the acquisition and reporting analysis (Section 4.3).

Writing and literature assistance. Generative AI tools helped polish parts of the manuscript, including the abstract, portions of the introduction and conclusion, and several figure captions. They also helped improve readability, survey related work, check cross-reference consistency, and suggest corrections to the reference list.

Proof checking and scope. We used generative AI to double-check author-developed proofs for possible errors or unclear steps. We did not use generative AI to propose hypotheses, develop theoretical frameworks, formulate or prove mathematical claims, design the benchmark or experiments, or implement the methods. Translation and qualitative data analysis are not applicable to this work. The authors take responsibility for the final manuscript, including its mathematical claims and all AI-assisted text and references.

ETHICS STATEMENT

Our experiments use retrospective event questions from Forecast-Dojo and FutureX, together with archived news, and do not involve human-subject interventions or live-market actions. Retrieval is restricted by forecast date, and resolved outcomes remain hidden from the agents during rollouts. Forecasts can nevertheless reflect incomplete corpus coverage, unreliable sources, or model biases; benchmark improvements do not establish reliability for consequential financial or policy decisions. Any redistribution of articles or agent trajectories must respect source licenses and applicable usage terms, with attention to incidental personal information in retrieved material.

REPRODUCIBILITY STATEMENT

Code will be released at `https://anonymous.4open.science/r/AlphaBet-2417/`. Appendix B specifies the dated environment, agent interface, reward implementation, and policy optimization, with the training procedure summarized in Algorithm 1. Training and evaluation hyperparameters appear in Tables 3 and 8. Appendix C documents the data splits, model knowledge cutoffs and release dates, scoring and statistical procedures, controlled comparisons, and validation-based checkpoint selection. Full agent and reader prompts are provided in Appendix E, and the assumptions and complete theoretical proofs in Appendix A.

REFERENCES

Anthropic. System card: Claude sonnet 4.6, February 2026. URL `https://www-cdn.anthropic.com/78073f739564e986ff3e28522761a7a0b4484f84.pdf`.

Pavel Atanasov, Jens Witkowski, Lyle H. Ungar, Barbara A. Mellers, and Philip Tetlock. Small steps to accuracy: Incremental belief updaters are better forecasters. In Péter Biró, Jason D. Hartline, Michael Ostrovsky, and Ariel D. Procaccia (eds.), *EC '20: The 21st ACM Conference on Economics and Computation, Virtual Event, Hungary, July 13-17, 2020*, pp. 873–874. ACM, 2020. doi: 10.1145/3391403.3399540. URL `https://doi.org/10.1145/3391403.3399540`.

Neil Band, Xuechen Li, Tengyu Ma, and Tatsunori Hashimoto. Linguistic calibration of long-form generations. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 2732–2778. PMLR, 2024. URL `https://proceedings.mlr.press/v235/band24a.html`.

Michael Bereket and Jure Leskovec. Uncalibrated reasoning: GRPO induces overconfidence for stochastic outcomes, 2025. URL `https://arxiv.org/abs/2508.11800`.

Nikhil Chandak, Shashwat Goel, Ameya Prabhu, Moritz Hardt, and Jonas Geiping. Scaling open-ended reasoning to predict the future. *CoRR*, abs/2512.25070, 2025. doi: 10.48550/ARXIV.2512.25070. URL `https://doi.org/10.48550/arXiv.2512.25070`.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *IEEE Trans. Big Data*, 12(2):346–361, 2026. doi: 10.1109/TBDATA.2025.3618474. URL https://doi.org/10.1109/TBDATA.2025.3618474.

GLM. GLM-4.5: agentic, reasoning, and coding (ARC) foundation models. *CoRR*, abs/2508.06471, 2025. doi: 10.48550/ARXIV.2508.06471. URL https://doi.org/10.48550/arXiv.2508.06471.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437. URL https://doi.org/10.1198/016214506000001437.

Shashwat Goel, Nikhil Chandak, Arvindh Arun, Ameya Prabhu, Steffen Staab, Moritz Hardt, Maksym Andriushchenko, and Jonas Geiping. FutureSim: Replaying world events to evaluate adaptive agents. *CoRR*, abs/2605.15188, 2026. doi: 10.48550/ARXIV.2605.15188. URL https://doi.org/10.48550/arXiv.2605.15188.

Danny Halawi, Fred Zhang, Chen Yueh-Han, and Jacob Steinhardt. Approaching human-level forecasting with language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50426–50468. Curran Associates, Inc., 2024. doi: 10.52202/079017-1598. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a5acfd0876c940d81619c1dc60e7748-Abstract-Conference.html.

Scott Jeen, Matthew Aitchison, and Mantic. Training llms to predict world events. *Thinking Machines Lab: News*, 2026. https://thinkingmachines.ai/news/training-llms-to-predict-world-events/.

Woojeong Jin, Rahul Khanna, Suji Kim, Dong-Ho Lee, Fred Morstatter, Aram Galstyan, and Xiang Ren. ForecastQA: A question answering challenge for event forecasting with temporal text data. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4636–4650. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.357. URL https://aclanthology.org/2021.acl-long.357/.

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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=lfPkGWXLLf.

Kimi Team. Kimi K2.5: visual agentic intelligence. *CoRR*, abs/2602.02276, 2026. doi: 10.48550/ARXIV.2602.02276. URL https://doi.org/10.48550/arXiv.2602.02276.

Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=8s8K2UZGTZ.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025a.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-Like training: A critical perspective. In *Second Conference on Language Modeling*, 2025b. URL https://openreview.net/forum?id=5PAF7PAY2Y.

Barbara Mellers, Lyle Ungar, Jonathan Baron, Jaime Ramos, Burcu Gurcay, Katrina Fincher, Sydney E Scott, Don Moore, Pavel Atanasov, Samuel A Swift, et al. Psychological strategies for winning a geopolitical forecasting tournament. *Psychological science*, 25(5):1106–1115, 2014.

Meta AI. Llama 4 model card. https://github.com/meta-llama/llama-models/blob/main/models/llama4/MODEL_CARD.md, 2025. Accessed: 2026-09-26.

MiniMax. Minimax m2.5: Built for real-world productivity, February 2026. URL https://www.minimax.io/news/minimax-m25.

Allan H Murphy. A new vector partition of the probability score. *Journal of Applied Meteorology and Climatology*, 12(4):595–600, 1973.

Kevin Murphy. Agentic forecasting using sequential Bayesian updating of linguistic beliefs. In *ICML 2026 Workshop on Forecasting as a New Frontier of Intelligence*, 2026. URL https://openreview.net/forum?id=Iw4jKx6hIx.

Sebastian Nagel. Common crawl news dataset, 2016. URL https://data.commoncrawl.org/crawl-data/CC-NEWS/index.html.

OpenAI. Gpt-5.6: Frontier intelligence that scales with your ambition. https://openai.com/index/gpt-5-6/, 2026. Accessed: 2026-09-26.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Benjamin Turtel, Danny Franklin, Kris Skotheim, Luke Hewitt, and Philipp Schoenegger. Outcome-based reinforcement learning to predict the future. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=bbhdeL8EUX.

Benjamin Turtel, Paul Wilczewski, Danny Franklin, and Kris Skothiem. Future-as-Label: Scalable supervision from real-world outcomes, 2026. URL https://arxiv.org/abs/2601.06336.

Qi Yan, Raihan Seraj, Jiawei He, Lili Meng, and Tristan Sylvain. AutoCast++: Enhancing world event prediction with zero-shot ranking-based context retrieval. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=COYDmKkQH4.

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.

Liqin Ye, Haorui Wang, Fardin Ahmed, Rongzhi Zhang, Yuan He, Ziyuan Lin, Yanbin Yin, Jing Peng, Michael Galarnyk, Sudheer Chava, and Chao Zhang. Forecast-Dojo: Replayable environments for benchmarking and training LLM forecasting agents, 2026. URL https://arxiv.org/abs/2609.28876.

Zhangdie Yuan, Zifeng Ding, and Andreas Vlachos. Do language models update their forecasts with new information? *arXiv preprint arXiv:2509.23936*, 2025.

Zhiyuan Zeng, Jiashuo Liu, Siyuan Chen, Tianci He, Yali Liao, Yixiao Tian, Jinpeng Wang, Zaiyuan Wang, Yang Yang, Lingyue Yin, Mingren Yin, Zhenwei Zhu, Tianle Cai, Xinjie Chen, Zehui Chen, Jiecao Chen, Yantao Du, Xiang Gao, Jiacheng Guo, Liang Hu, Jianpeng Jiao, Xiangsheng Li, Jingkai Liu, Shuang Ni, Zhoufutu Wen, Ge Zhang, Kaiyuan Zhang, Xin Zhou, Jose Blanchet, Xipeng Qiu, Mengdi Wang, and Wenhao Huang. FutureX: An advanced live benchmark for LLM agents in future prediction. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=z28PLIEj6l.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *CoRR*, abs/2506.05176, 2025. doi: 10.48550/ARXIV.2506.05176. URL https://doi.org/10.48550/arXiv.2506.05176.

Xinyue Zhao, Ruiyi Zhang, Liqin Ye, Rui Cao, Pengtao Xie, and Sudheer Chava. Can llms take the pulse of the economy? A real-time evaluation of LLM nowcasts on macroeconomic indicators. *CoRR*, abs/2608.30110, 2026. doi: 10.48550/ARXIV.2608.30110. URL https://doi.org/10.48550/arXiv.2608.30110.

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. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27293–27305, 2022. doi: 10.52202/068431-1979.

Appendix for ALPHABET

A Theoretical Details **14**

A.1 Effect of the Format Reward 14

A.2 Properness of the Agentic Forecasting Objective 15

A.3 Scope of the Properness Guarantee 16

A.4 Mean-Centering Preserves the Proper-Score Update 17

A.5 Standard-Deviation Normalization and the Tilted Target 18

B Implementation Details **23**

B.1 Forecasting Tasks 23

B.2 Temporal Information Access 23

B.3 Agent Interface 23

B.4 Reward Implementation 24

B.5 Policy Optimization 24

B.6 Baseline Reproductions 26

C Experimental Details **27**

C.1 Datasets and Temporal Splits 27

C.2 Models and Evaluation Protocol 28

C.3 Scoring and Statistical Analysis 29

C.4 Evidence Acquisition and Probability Reporting 29

C.5 Training Ablations 30

D Limitations **31**

E Prompts **31**

E.1 Memory-Free Forecasting 31

E.2 Memory-On Forecasting 33

E.3 BLF Forecasting 35

E.4 Gain Analysis 37

A THEORETICAL DETAILS

A.1 EFFECT OF THE FORMAT REWARD

We prove the format-reward claim in Section 3.2. Assume that the objectives are finite and that there exists a forecasting-optimal policy $\pi_{\theta^\star}$ that passes every format check almost surely. Under this feasibility assumption, the format reward preserves the optimal forecasting value; the training optima are exactly the forecasting optima with zero expected format penalty.

Proof. The coefficients in Eq. 4 are nonnegative, and each format check takes values in $[0, 1]$. Consequently, $R^{\text{format}}_{Q,t} \leq 0$ almost surely and $J_{\text{format}}(\theta) \leq 0$ for every θ. Since $\pi_{\theta^\star}$ passes all checks,

$J_{\text{format}}(\theta^\star) = 0$. Therefore, for every θ,

$$J_{\text{train}}(\theta) \le J_{\text{forecast}}(\theta) \le J_{\text{forecast}}(\theta^\star) = J_{\text{train}}(\theta^\star). \tag{11}$$

This proves $\max_\theta J_{\text{train}}(\theta) = \max_\theta J_{\text{forecast}}(\theta)$.

Now let $\widehat{\theta}$ be any maximizer of J_{train}. The two ends of Eq. 11 are equal at $\widehat{\theta}$, so both inequalities are equalities. Hence $J_{\text{forecast}}(\widehat{\theta}) = J_{\text{forecast}}(\theta^\star)$ and $J_{\text{format}}(\widehat{\theta}) = 0$. Conversely, any forecasting optimum with zero expected format penalty attains this same optimal training value. Equivalently,

$$\arg\max_\theta J_{\text{train}}(\theta) = \left\{ \theta \in \arg\max_\vartheta J_{\text{forecast}}(\vartheta) : \; J_{\text{format}}(\theta) = 0 \right\}.$$

□

This result preserves the forecasting optimum under the feasibility assumption. It does not assert that the format term leaves gradients unchanged or that every improvement in J_{train} improves J_{forecast}.

A.2 Properness of the Agentic Forecasting Objective

We prove Proposition 1 for the targets $q_{Q,t}$ and $q^H_{Q,t}$, discrepancy D_S, and optimal forecasting value $J^\star$ defined in the main text.

Probability model and integrability. Expectations in this proof are population expectations over questions, outcomes, dated evidence, and policy rollouts. The law of the question, its scheduled dates, the corpus, and the outcome does not depend on the forecasting policy. Thus $J^\star$ is policy-independent, although the acquired context and its target may depend on the policy. The training data provide an empirical estimate of this expectation; the argument does not assume that rerunning a resolved question resamples its outcome. We make the finite-score requirement explicit by assuming, for every policy under consideration,

$$\mathbb{E}\left[\sum_{t=1}^{T_Q} \left(|S(q_{Q,t}, Y_Q)| + |S(q^H_{Q,t}, Y_Q)| + |S(p_{Q,t}, Y_Q)| \right) \right] < \infty.$$

This ensures that the conditioning and summation steps below are valid. For the logarithmic score, zero-probability outcomes contribute zero to the corresponding conditional sums.

No-leakage conditions. We write $\tau_{Q,t}$ for the scheduled date τ_t of question Q. The no-leakage assumption takes the following form, for every class k and forecast step, almost surely:

$$\begin{aligned} \Pr(Y_Q = k \mid Q, \tau_{Q,t}, \mathcal{I}_{\le \tau_{Q,t}}, H_{Q,t}) &= q_{Q,t}(k), \\ \Pr(Y_Q = k \mid Q, \tau_{Q,t}, H_{Q,t}, p_{Q,t}) &= q^H_{Q,t}(k). \end{aligned} \tag{12}$$

The first equality states that the context carries no outcome information beyond the admissible dated corpus. The second states that generating the report supplies no additional outcome information beyond the acquired evidence: reasoning, code outputs, and the notebook are functions of $H_{Q,t}$ and sampling randomness independent of the outcome. These conditions permit adaptive search, reasoning, computation, and memory based on admissible information. By the tower property and the first equality,

$$\begin{aligned} q^H_{Q,t}(k) &= \Pr(Y_Q = k \mid Q, \tau_{Q,t}, H_{Q,t}) \\ &= \mathbb{E}\big[\Pr(Y_Q = k \mid Q, \tau_{Q,t}, \mathcal{I}_{\le \tau_{Q,t}}, H_{Q,t}) \mid Q, \tau_{Q,t}, H_{Q,t} \big] \\ &= \mathbb{E}[q_{Q,t}(k) \mid Q, \tau_{Q,t}, H_{Q,t}]. \end{aligned} \tag{13}$$

Proof of Proposition 1. Fix a policy. For the per-step calculation, fix a question Q and a scheduled step $t \le T_Q$; expectations in the next three displays are conditional on this question. We restore the outer question expectation when summing the per-step identities.

Step 1: Split the score difference. Adding and subtracting $S(q^H_{Q,t}, Y_Q)$ gives

$$\begin{aligned}&\mathbb{E}[S(q_{Q,t}, Y_Q) - S(p_{Q,t}, Y_Q)] \\ &\quad = \mathbb{E}[S(q_{Q,t}, Y_Q) - S(q^H_{Q,t}, Y_Q)] + \mathbb{E}[S(q^H_{Q,t}, Y_Q) - S(p_{Q,t}, Y_Q)].\end{aligned} \tag{14}$$

Step 2: Identify the two discrepancy terms. Conditional on $(Q, \tau_{Q,t}, \mathcal{I}_{\leq \tau_{Q,t}}, H_{Q,t})$, both $q_{Q,t}$ and $q^H_{Q,t}$ are fixed and the conditional law of Y_Q is $q_{Q,t}$. Hence, by iterated expectation,

$$\begin{aligned}&\mathbb{E}[S(q_{Q,t}, Y_Q) - S(q^H_{Q,t}, Y_Q)] \\ &\quad = \mathbb{E}\left[\sum_{k=1}^{K} q_{Q,t}(k)\big(S(q_{Q,t}, k) - S(q^H_{Q,t}, k)\big)\right] \\ &\quad = \mathbb{E}[D_S(q_{Q,t}, q^H_{Q,t})].\end{aligned} \tag{15}$$

Likewise, conditional on $(Q, \tau_{Q,t}, H_{Q,t}, p_{Q,t})$, both $q^H_{Q,t}$ and $p_{Q,t}$ are fixed and the conditional law of Y_Q is $q^H_{Q,t}$. Therefore,

$$\begin{aligned}&\mathbb{E}[S(q^H_{Q,t}, Y_Q) - S(p_{Q,t}, Y_Q)] \\ &\quad = \mathbb{E}\left[\sum_{k=1}^{K} q^H_{Q,t}(k)\big(S(q^H_{Q,t}, k) - S(p_{Q,t}, k)\big)\right] \\ &\quad = \mathbb{E}[D_S(q^H_{Q,t}, p_{Q,t})].\end{aligned} \tag{16}$$

Step 3: Sum over the episode and characterize equality. Summing the preceding identities over $t \leq T_Q$ and averaging over questions gives

$$J^\star - J_{\text{forecast}}(\theta) = \mathbb{E}\left[\sum_{t=1}^{T_Q} \big(D_S(q_{Q,t}, q^H_{Q,t}) + D_S(q^H_{Q,t}, p_{Q,t})\big)\right]. \tag{17}$$

Since $J^\star$ is independent of θ, maximizing J_{forecast} is equivalent to minimizing the right-hand side. This proves Eq. 7.

Strict properness gives $D_S(q, p) \geq 0$, with equality exactly when $p = q$. The sum in Eq. 17 is consequently a nonnegative random variable. Its expectation is zero if and only if the sum is zero almost surely, which, because every summand is nonnegative, is equivalent to

$$q^H_{Q,t} = q_{Q,t}, \qquad p_{Q,t} = q^H_{Q,t} \quad \text{for every } t \leq T_Q, \text{ almost surely.} \tag{18}$$

Conversely, these equalities make every discrepancy zero. Thus a policy attains the optimal forecasting value $J^\star$ if and only if $p_{Q,t} = q^H_{Q,t} = q_{Q,t}$ at every scheduled step almost surely. □

Interpretation of the decomposition. Equation 17 identifies the expected *sum* of the information and reporting gaps. An increase in J_{forecast} decreases this sum, but need not decrease both components separately or improve every date. The identity applies to both memory-free and memory-on forecasting under the stated no-leakage conditions. Together with Appendix A.1, it also implies that every maximizer of J_{train} minimizes this expected sum under the format-feasibility assumption.

Why one resolved outcome can supervise every forecast date. The same Y_Q scores every step, but the proof uses conditional expectation at each date and linearity across dates, not independent labels across dates. Consequently, one resolved outcome can score multiple dated forecasts without imposing the same target distribution on them. Neither latent target needs to be observed to compute the training score $S(p_{Q,t}, Y_Q)$.

A.3 Scope of the Properness Guarantee

The strict-properness statements apply to the negative Brier score and the ideal logarithmic score, subject to the integrability requirements above. The negative Brier reward specified in the experimental setup requires no probability floor. Indeed, expanding its expectation gives

$$\mathbb{E}_{Y\sim q}[S_{\text{Brier}}(p, Y)] = -\|p\|_2^2 + 2\langle q, p\rangle - 1, \qquad D_{S_{\text{Brier}}}(q, p) = \|q - p\|_2^2.$$

Thus the score is strictly proper on the full simplex, including its boundary. The log-floor qualification below concerns the logarithmic option, not negative-Brier training.

For the logarithmic option described in Section 3.2, the ideal score is $S_{\log}(p, Y) = \log p(Y)$. If the realized-outcome probability is floored at $\varepsilon_{\text{score}} = 10^{-3}$, the score becomes $\widetilde{S}_{\log}(p, Y) = \log \max\{p(Y), \varepsilon_{\text{score}}\}$. This floored score is not proper on the full simplex, as the following counterexample shows.

Verification of the counterexample. Let $q = (1 - \delta, \delta)$, where $0 < \delta < \varepsilon_{\text{score}}$, and compare truthful reporting with the report $p^0 = (1, 0)$. Since $1 - \delta > \varepsilon_{\text{score}}$,

$$\mathbb{E}_{Y\sim q}[\widetilde{S}_{\log}(p^0, Y)] = (1 - \delta) \log 1 + \delta \log \varepsilon_{\text{score}},$$
$$\mathbb{E}_{Y\sim q}[\widetilde{S}_{\log}(q, Y)] = (1 - \delta) \log(1 - \delta) + \delta \log \varepsilon_{\text{score}}.$$

Subtracting yields

$$\mathbb{E}_{Y\sim q}\left[\widetilde{S}_{\log}(p^0, Y) - \widetilde{S}_{\log}(q, Y)\right] = -(1 - \delta) \log(1 - \delta) > 0. \tag{19}$$

Thus the truthful report is not a maximizer of the floored score. □

The floored and ideal scores agree when $p(Y) \geq \varepsilon_{\text{score}}$, but this pointwise agreement does not preserve the optimum on the full simplex. Accordingly, Proposition 1 gives an exact properness guarantee for the Brier score and the ideal log score, not for the floored log score. The same qualification applies to the strict-properness interpretation of Proposition 2: the score-function identities below remain valid for finite floored scores, but their interpretation as uniquely eliciting a specified probability target requires a strictly proper score.

A.4 Mean-Centering Preserves the Proper-Score Update

We prove part (i) of Proposition 2. The setup and regularity conditions in this subsection also apply to part (ii). Fix $(Q, \tau_t, H_{Q,t})$ and its conditional outcome law $q^H_{Q,t}$. This is a local reporting analysis: the context and its outcome law are held fixed when differentiating the report policy. We analyze the forecasting-score component, not the mixed reward containing format penalties. Accordingly, the notation $R^{(i)}_{Q,t}$ in this subsection and Appendix A.5 is local to this analysis and denotes $S(p^{(i)}_{Q,t}, Y_Q)$, rather than the total reward in Section 3.2. The format-feasibility result does not remove format penalties from a gradient or a group standard deviation.

Report policy and regularity. Let $\rho_{\vartheta,Q,t}(p \mid H_{Q,t})$ denote the induced probability of report p, abbreviated as $\rho_\vartheta(p)$. Assume that, in a neighborhood of the current parameter θ, these probabilities are continuously differentiable and positive on a common finite set of reports $\mathcal{P} \subseteq \Delta^{K-1}$. The report vectors in $\mathcal{P}$ are fixed, and $S(p, k)$ is finite for every $p \in \mathcal{P}$ and class k. For the ideal log score, this requires $p(k) > 0$ on the report support.

The report-level score function is the logarithmic derivative of this sampling distribution:

$$\psi_\theta(p) \triangleq \nabla_\theta \log \rho_\theta(p), \qquad \mathbb{E}_{p\sim\rho_\theta}[\psi_\theta(p)] = \sum_{p\in\mathcal{P}} \nabla_\theta \rho_\theta(p) = \nabla_\theta 1 = 0. \tag{20}$$

Define the local expected forecasting score, with $q^H_{Q,t}$ fixed, by

$$\mathcal{J}^H_{Q,t}(\vartheta) \triangleq \mathbb{E}_{p\sim\rho_\vartheta}\left[\sum_{k=1}^{K} q^H_{Q,t}(k) S(p, k)\right]. \tag{21}$$

Differentiating the finite sum gives

$$\begin{aligned}\nabla_\theta \mathcal{J}^H_{Q,t}(\theta) &= \sum_{p\in\mathcal{P}} \nabla_\theta \rho_\theta(p) \sum_{k=1}^{K} q^H_{Q,t}(k) S(p, k) \\ &= \mathbb{E}_{Y_Q\sim q^H_{Q,t},\, p\sim\rho_\theta}[\psi_\theta(p) S(p, Y_Q)],\end{aligned} \tag{22}$$

where the outcome and the sampled report are independent conditional on the fixed context.

Group sampling and update. Draw $G > 1$ reports independently from ρ_θ, independently of $Y_Q \sim q^H_{Q,t}$, and score them using this same outcome. Thus, conditional on the fixed context, the joint sampling law is $q^H_{Q,t}(k)\prod_{i=1}^G \rho_\theta(p^{(i)}_{Q,t})$. For this proof, set

$$R^{(i)}_{Q,t} = S(p^{(i)}_{Q,t}, Y_Q), \qquad \bar{R}_{Q,t} = \frac{1}{G}\sum_{j=1}^{G} R^{(j)}_{Q,t}. \tag{23}$$

All expectations in the remainder of this subsection average over both the shared outcome and the sampled reports. Define the on-policy score-function estimator

$$\widehat{g}_{\text{mean}} = \frac{1}{G}\sum_{i=1}^{G} \psi_\theta(p^{(i)}_{Q,t})\big(R^{(i)}_{Q,t} - \bar{R}_{Q,t}\big). \tag{24}$$

The sampled rewards and group statistics are treated as constants in this update; only the report-policy log probability is differentiated.

Proof of Proposition 2(i). For every i, Eq. 22 implies

$$\mathbb{E}[\psi_\theta(p^{(i)}_{Q,t})R^{(i)}_{Q,t}] = \nabla_\theta \mathcal{J}^H_{Q,t}(\theta).$$

For $i \neq j$, condition on the shared outcome before factoring the expectation. Independence of the reports and Eq. 20 give

$$\begin{aligned}\mathbb{E}[\psi_\theta(p^{(i)}_{Q,t})R^{(j)}_{Q,t}] &= \sum_{k=1}^{K} q^H_{Q,t}(k)\mathbb{E}[\psi_\theta(p^{(i)}_{Q,t}) \mid Y_Q = k]\mathbb{E}[R^{(j)}_{Q,t} \mid Y_Q = k] \\ &= 0.\end{aligned} \tag{25}$$

Expanding the group mean and separating the diagonal and off-diagonal terms therefore yields

$$\begin{aligned}\mathbb{E}[\widehat{g}_{\text{mean}}] &= \frac{1}{G}\sum_{i=1}^{G}\mathbb{E}[\psi_\theta(p^{(i)}_{Q,t})R^{(i)}_{Q,t}] - \frac{1}{G^2}\sum_{i=1}^{G}\sum_{j=1}^{G}\mathbb{E}[\psi_\theta(p^{(i)}_{Q,t})R^{(j)}_{Q,t}] \\ &= \nabla_\theta \mathcal{J}^H_{Q,t}(\theta) - \frac{G}{G^2}\nabla_\theta \mathcal{J}^H_{Q,t}(\theta) \\ &= \left(1 - \frac{1}{G}\right)\nabla_\theta \mathcal{J}^H_{Q,t}(\theta).\end{aligned} \tag{26}$$

The expected uncentered update is

$$\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\psi_\theta(p^{(i)}_{Q,t})R^{(i)}_{Q,t}\right] = \nabla_\theta \mathcal{J}^H_{Q,t}(\theta).$$

Therefore Eq. 26 is exactly the $1 - 1/G$ proportionality claimed in part (i). Since $1 - 1/G > 0$, mean-centering preserves the expected proper-score update direction for every finite $G > 1$. □

A.5 Standard-Deviation Normalization and the Tilted Target

We prove part (ii) under the fixed-context setup and regularity conditions of Appendix A.4, with $q^H_{Q,t}(k) > 0$ for every class and fixed $\epsilon > 0$. All policy-dependent quantities below are evaluated at the current parameter θ.

Outcome-specific score scales and the sampled update. Define the mean and standard deviation of the score across reports when outcome k is realized:

$$\begin{aligned}&\mu_{Q,t,k} \triangleq \mathbb{E}_{p\sim\rho_\theta}[S(p,k)], \\ &\sigma^S_{Q,t,k} \triangleq \left(\mathbb{E}_{p\sim\rho_\theta}[(S(p,k) - \mu_{Q,t,k})^2]\right)^{1/2}.\end{aligned} \tag{27}$$

Thus $\sigma^S_{Q,t,k}$ measures score variability across reports, not uncertainty about which outcome will occur. For a sampled group, let

$$\widehat{\sigma}^{(G)}_{Q,t} \triangleq \left[\frac{1}{G}\sum_{i=1}^{G}(R^{(i)}_{Q,t} - \bar{R}_{Q,t})^2\right]^{1/2} \tag{28}$$

and define the standardized score-function estimator

$$\widehat{g}^{(G)}_{\mathrm{std}} = \frac{1}{G}\sum_{i=1}^{G}\psi_\theta(p^{(i)}_{Q,t})\frac{R^{(i)}_{Q,t} - \bar{R}_{Q,t}}{\widehat{\sigma}^{(G)}_{Q,t} + \epsilon}. \tag{29}$$

As before, the rewards and the group statistics are not differentiated.

Proof of Proposition 2(ii).

Step 1: Obtain the conditional large-group limit.

Condition on $Y_Q = k$. The reports remain i.i.d. from ρ_θ. Using the identity

$$\big(\widehat{\sigma}^{(G)}_{Q,t}\big)^2 = \frac{1}{G}\sum_{i=1}^{G}\big(R^{(i)}_{Q,t}\big)^2 - \bar{R}^2_{Q,t},$$

the strong law of large numbers gives $\bar{R}_{Q,t} \to \mu_{Q,t,k}$ and $\widehat{\sigma}^{(G)}_{Q,t} \to \sigma^S_{Q,t,k}$ almost surely as $G \to \infty$. It also yields

$$\begin{aligned}\frac{1}{G}\sum_{i=1}^{G}\psi_\theta\big(p^{(i)}_{Q,t}\big) &\xrightarrow{\text{a.s.}} \mathbb{E}_{p\sim\rho_\theta}\big[\psi_\theta(p)\big] = 0,\\ \frac{1}{G}\sum_{i=1}^{G}\psi_\theta\big(p^{(i)}_{Q,t}\big)R^{(i)}_{Q,t} &\xrightarrow{\text{a.s.}} \mathbb{E}_{p\sim\rho_\theta}\big[\psi_\theta(p)S(p,k)\big].\end{aligned} \tag{30}$$

Here the almost-sure limits follow from the strong law of large numbers, and the equality to zero follows from the score-function identity in Eq. 20.

The update can be written exactly as

$$\widehat{g}^{(G)}_{\mathrm{std}} = \frac{G^{-1}\sum_i \psi_\theta(p^{(i)}_{Q,t})R^{(i)}_{Q,t} - \bar{R}_{Q,t}\, G^{-1}\sum_i \psi_\theta(p^{(i)}_{Q,t})}{\widehat{\sigma}^{(G)}_{Q,t} + \epsilon}.$$

Since the limiting denominator is at least $\epsilon > 0$,

$$\widehat{g}^{(G)}_{\mathrm{std}} \xrightarrow{\text{a.s.}} \frac{\mathbb{E}_{p\sim\rho_\theta}[\psi_\theta(p)S(p,k)]}{\sigma^S_{Q,t,k} + \epsilon} \quad \text{conditional on } Y_Q = k.$$

No independence between the sample mean and sample standard deviation is required.

Step 2: Establish the expected update limit and outcome reweighting.

We first obtain a deterministic bound that is uniform in the group size and the sampled reports. Under the regularity conditions in Appendix A.4, define

$$C_S \triangleq \max_{p\in\mathcal{P},\,1\le k\le K}|S(p,k)|, \qquad C_\psi \triangleq \max_{p\in\mathcal{P}}\|\psi_\theta(p)\|,$$

where $\|\cdot\|$ denotes the Euclidean norm. The finite-support and finite-score assumptions imply $C_S < \infty$. Moreover, positivity and differentiability of $\rho_\theta(p)$ give

$$\|\psi_\theta(p)\| = \frac{\|\nabla_\theta\rho_\theta(p)\|}{\rho_\theta(p)} < \infty \qquad \text{for every } p \in \mathcal{P}.$$

Since $\mathcal{P}$ is finite, $C_\psi < \infty$ as well. These constants may depend on the fixed context and current parameter θ, but not on G or the sampled group.

For every group member, the definition of C_S gives $|R_{Q,t}^{(i)}| = |S(p_{Q,t}^{(i)}, Y_Q)| \le C_S$. Consequently, the sample mean satisfies

$$|\bar{R}_{Q,t}| = \left| \frac{1}{G} \sum_{j=1}^{G} R_{Q,t}^{(j)} \right| \le \frac{1}{G} \sum_{j=1}^{G} |R_{Q,t}^{(j)}| \le C_S,$$

and each centered reward is bounded by

$$|R_{Q,t}^{(i)} - \bar{R}_{Q,t}| \le |R_{Q,t}^{(i)}| + |\bar{R}_{Q,t}| \le 2C_S.$$

Also, $\widehat{\sigma}_{Q,t}^{(G)} \ge 0$, so the denominator is at least the fixed positive constant ϵ. Applying the triangle inequality and these bounds yields, for every G and every sampled group,

$$\begin{aligned} \left\| \widehat{g}_{\mathrm{std}}^{(G)} \right\| &= \left\| \frac{1}{G} \sum_{i=1}^{G} \psi_\theta\big(p_{Q,t}^{(i)}\big) \frac{R_{Q,t}^{(i)} - \bar{R}_{Q,t}}{\widehat{\sigma}_{Q,t}^{(G)} + \epsilon} \right\| \\ &\le \frac{1}{G} \sum_{i=1}^{G} \frac{\left\|\psi_\theta\big(p_{Q,t}^{(i)}\big)\right\| |R_{Q,t}^{(i)} - \bar{R}_{Q,t}|}{\widehat{\sigma}_{Q,t}^{(G)} + \epsilon} \\ &\le \frac{1}{G\epsilon} \sum_{i=1}^{G} \left\|\psi_\theta\big(p_{Q,t}^{(i)}\big)\right\| |R_{Q,t}^{(i)} - \bar{R}_{Q,t}| \\ &\le \frac{1}{G\epsilon} \sum_{i=1}^{G} C_\psi (2C_S) \\ &= \frac{2C_\psi C_S}{\epsilon}. \end{aligned} \tag{31}$$

The final bound is finite and independent of G and of both the shared outcome and the sampled reports. Together with the conditional almost-sure limit from Step 1, it permits dominated convergence componentwise, giving, for each class k,

$$\lim_{G\to\infty} \mathbb{E}\Big[\widehat{g}_{\mathrm{std}}^{(G)} \mid Y_Q = k\Big] = \frac{\mathbb{E}_{p\sim\rho_\theta}[\psi_\theta(p) S(p,k)]}{\sigma_{Q,t,k}^{S} + \epsilon}.$$

Finally, average over the shared outcome using its conditional law $q_{Q,t}^{H}$. Since the number of classes is finite,

$$\begin{aligned} \lim_{G\to\infty} \mathbb{E}[\widehat{g}_{\mathrm{std}}^{(G)}] &= \sum_{k=1}^{K} \frac{q_{Q,t}^{H}(k)}{\sigma_{Q,t,k}^{S} + \epsilon} \mathbb{E}_{p\sim\rho_\theta}[\psi_\theta(p) S(p,k)] \\ &= \mathbb{E}_{p\sim\rho_\theta}\left[\psi_\theta(p) \sum_{k=1}^{K} \frac{q_{Q,t}^{H}(k)}{\sigma_{Q,t,k}^{S} + \epsilon} \big(S(p,k) - \mu_{Q,t,k}\big) \right]. \end{aligned} \tag{32}$$

The second equality uses $\mathbb{E}_{p\sim\rho_\theta}[\psi_\theta(p)] = 0$: for each fixed k, the baseline $\mu_{Q,t,k}$ is independent of the sampled report p, so its score-function expectation vanishes.

Step 3: Identify the proper-score objective and its tilted reporting target. Define

$$Z_{Q,t} \triangleq \sum_{k=1}^{K} \frac{q_{Q,t}^{H}(k)}{\sigma_{Q,t,k}^{S} + \epsilon}, \qquad \widetilde{q}_{Q,t}^{H}(k) \triangleq \frac{q_{Q,t}^{H}(k)/(\sigma_{Q,t,k}^{S} + \epsilon)}{Z_{Q,t}}. \tag{33}$$

The normalizer $Z_{Q,t}$ is finite and positive, and $\widetilde{q}_{Q,t}^{H}$ is a probability distribution. In the calculation below, both quantities are evaluated at the current parameter θ and held fixed; differentiation with respect to ϑ acts only on the report probabilities $\rho_\vartheta(p)$. Substituting Eq. 33 into the first line of

Eq. 32 gives

$$\begin{aligned}
&\lim_{G\to\infty} \mathbb{E}[\widehat{g}^{(G)}_{\mathrm{std}}] \\
&= Z_{Q,t}\, \mathbb{E}_{p\sim\rho_\theta}\left[\psi_\theta(p) \sum_{k=1}^{K} \widetilde{q}^{H}_{Q,t}(k) S(p,k)\right] \\
&\overset{(a)}{=} Z_{Q,t} \sum_{p\in\mathcal{P}} \rho_\theta(p)\psi_\theta(p) \sum_{k=1}^{K} \widetilde{q}^{H}_{Q,t}(k) S(p,k) \\
&\overset{(b)}{=} Z_{Q,t} \sum_{p\in\mathcal{P}} \big(\nabla_\theta \rho_\theta(p)\big) \sum_{k=1}^{K} \widetilde{q}^{H}_{Q,t}(k) S(p,k) \\
&\overset{(c)}{=} Z_{Q,t}\, \nabla_\vartheta \left[\sum_{p\in\mathcal{P}} \rho_\vartheta(p) \sum_{k=1}^{K} \widetilde{q}^{H}_{Q,t}(k) S(p,k)\right]\Bigg|_{\vartheta=\theta} \\
&\overset{(d)}{=} Z_{Q,t}\, \nabla_\vartheta \mathbb{E}_{p\sim\rho_\vartheta}\left[\sum_{k=1}^{K} \widetilde{q}^{H}_{Q,t}(k) S(p,k)\right]\Bigg|_{\vartheta=\theta}.
\end{aligned} \tag{34}$$

Here (a) expands the expectation over the finite report support; (b) uses $\rho_\theta(p)\psi_\theta(p) = \nabla_\theta \rho_\theta(p)$; (c) interchanges differentiation and finite summation, with the reports, scores, and target weights held fixed; and (d) rewrites the finite sum as an expectation.

Thus, the limiting expected update is a positive multiple of the gradient of a frozen-target proper-score objective. Strict properness makes $\widetilde{q}^{H}_{Q,t}$ its unique optimal report on the full simplex, which differs from the acquired-context target whenever $\widetilde{q}^{H}_{Q,t} \neq q^{H}_{Q,t}$. This identifies the local reporting incentive, not attainability by the restricted report policy or convergence of training.

Step 4: Characterize when standardization preserves the original reporting target. Since $q^{H}_{Q,t}(k) > 0$ for every k,

$$\begin{aligned}
\widetilde{q}^{H}_{Q,t} = q^{H}_{Q,t} &\iff \frac{1}{Z_{Q,t}(\sigma^{S}_{Q,t,k} + \epsilon)} = 1 \quad \text{for every } k \\
&\iff \sigma^{S}_{Q,t,k} + \epsilon = Z^{-1}_{Q,t} \quad \text{for every } k.
\end{aligned}$$

This requires all outcome-specific score scales to be equal. Conversely, if their common value is s, then $Z_{Q,t} = 1/(s+\epsilon)$ and the effective target equals $q^{H}_{Q,t}$. Therefore,

$$\widetilde{q}^{H}_{Q,t} = q^{H}_{Q,t} \quad \iff \quad \sigma^{S}_{Q,t,1} = \cdots = \sigma^{S}_{Q,t,K}. \tag{35}$$

Thus, under the full-support assumption on $q^{H}_{Q,t}$, standardization preserves the original reporting target in the fixed-policy large-group limit if and only if all outcome-specific score standard deviations are equal. When they are equal, the limiting expected update is a common positive rescaling of the original proper-score gradient. Otherwise, $\widetilde{q}^{H}_{Q,t} \neq q^{H}_{Q,t}$, so the locally elicited report differs from the acquired-context target. This establishes target distortion; whether that distortion sharpens the forecast is addressed separately below. This completes the proof.

□

Why standardization loses outcome-wise scale. The scale loss also has a finite-group interpretation. For constants $a_k > 0$ and $b_k \in \mathbb{R}$, define

$$S^\star(p,k) = a_k S(p,k) + b_k. \tag{36}$$

Within a group whose shared outcome is k, write R_i, $\bar{R}$, and $\widehat{\sigma}$ for its rewards, mean, and standard deviation; use a superscript $\star$ for the corresponding transformed quantities. Then $R^\star_i - \bar{R}^\star = a_k(R_i - \bar{R})$ and $\widehat{\sigma}^\star = a_k \widehat{\sigma}$. With $\epsilon = 0$, on any nondegenerate group with $\widehat{\sigma} > 0$,

$$\frac{R^\star_i - \bar{R}^\star}{\widehat{\sigma}^\star} = \frac{R_i - \bar{R}}{\widehat{\sigma}}. \tag{37}$$

Yet the expected transformed score assigns weight $q^H_{Q,t}(k)a_k$ to $S(p,k)$; unequal a_k can therefore change the reporting target while leaving these normalized advantages unchanged. With $\epsilon > 0$ the exact relation becomes

$$\frac{R^\star_i - \bar{R}^\star}{\widehat{\sigma}^\star + \epsilon} = \frac{R_i - \bar{R}}{\widehat{\sigma} + \epsilon/a_k},$$

so exact scale invariance no longer holds. The positive-ϵ reporting target is characterized by the proof above, rather than by this invariance observation alone.

When does the tilt sharpen the forecast? Target distortion alone does not imply overconfidence. A sufficient condition for sharpening is the opposite ordering of outcome probabilities and score scales. Suppose that, after relabeling the classes,

$$q^H_{Q,t}(1) \geq \cdots \geq q^H_{Q,t}(K), \qquad \sigma^S_{Q,t,1} \leq \cdots \leq \sigma^S_{Q,t,K}. \tag{38}$$

Under this condition, the effective target does not decrease the pairwise probability ratios of higher-ranked classes to lower-ranked classes relative to lower-ranked classes and majorizes $q^H_{Q,t}$.

Proof of the sharpening claim. Set $w_k = (\sigma^S_{Q,t,k} + \epsilon)^{-1}$, the inverse scale of outcome k. Then $w_1 \geq \cdots \geq w_K > 0$ and $\widetilde{q}^H_{Q,t}(k) = q^H_{Q,t}(k)w_k/Z_{Q,t}$. Both $q^H_{Q,t}(k)$ and w_k are nonincreasing, so their product, and hence $\widetilde{q}^H_{Q,t}(k)$, is also nonincreasing. For $i < j$, positivity of the outcome probabilities gives

$$\frac{\widetilde{q}^H_{Q,t}(i)}{\widetilde{q}^H_{Q,t}(j)} = \frac{q^H_{Q,t}(i)}{q^H_{Q,t}(j)}\frac{w_i}{w_j} \geq \frac{q^H_{Q,t}(i)}{q^H_{Q,t}(j)}. \tag{39}$$

For every $1 \leq m < K$, using $\sum_k q^H_{Q,t}(k) = 1$ gives

$$\begin{aligned}
&Z_{Q,t}\left(\sum_{i=1}^{m}\widetilde{q}^H_{Q,t}(i) - \sum_{i=1}^{m} q^H_{Q,t}(i)\right) \\
&= \sum_{i=1}^{m} q^H_{Q,t}(i)w_i - \left(\sum_{i=1}^{m} q^H_{Q,t}(i)\right)\left(\sum_{k=1}^{K} q^H_{Q,t}(k)w_k\right) \\
&= \left(\sum_{i=1}^{m} q^H_{Q,t}(i)w_i\right)\left(\sum_{j=m+1}^{K} q^H_{Q,t}(j)\right) - \left(\sum_{i=1}^{m} q^H_{Q,t}(i)\right)\left(\sum_{j=m+1}^{K} q^H_{Q,t}(j)w_j\right) \\
&= \sum_{i\leq m}\sum_{j>m} q^H_{Q,t}(i)q^H_{Q,t}(j)(w_i - w_j) \geq 0.
\end{aligned} \tag{40}$$

The last inequality holds because $w_i \geq w_j$ for $i \leq m < j$. Since both vectors are sorted in the same order and both sum to one, these prefix-sum inequalities establish that $\widetilde{q}^H_{Q,t}$ majorizes $q^H_{Q,t}$. □

Without Eq. 38, the tilt can instead flatten the distribution or shift mass across classes. The proposition therefore establishes target distortion, not universal overconfidence.

Relation to the binary case and scope. For $K = 2$, Eq. 33 has the binary outcome-reweighting form discussed by Bereket & Leskovec (2025). Equation 26 is exact for finite $G > 1$; Eq. 34 is a fixed-policy, large-group limit with fixed $\epsilon > 0$. Its frozen weights change with the policy, so it does not identify a single objective optimized throughout training. For fixed policy scales, multiplying numerator and denominator in Eq. 33 by ϵ shows that $\widetilde{q}^H_{Q,t} \to q^H_{Q,t}$ as $\epsilon \to \infty$.

Relation to the implemented rollout groups. Section 3.3 groups independent rollouts by the same (Q, τ_t) and, in memory-on forecasting, groups the corresponding dated steps from G complete episodes. A shared question and date do not imply a shared acquired context: different searches can produce different $H_{Q,t}$ even in independent forecasting, and sequential rollouts additionally carry their own notebook histories. The proof above conditions on one common $H_{Q,t}$ and isolates the reporting effect of normalization at that fixed information state; it is not an exact analysis of these heterogeneous-context groups.

The implemented sequential updates use per-date rewards without additional cross-date credit assignment. Proposition 1 characterizes the longitudinal objective, whereas Proposition 2 concerns the on-policy score-function estimators displayed above. Neither result establishes that the implemented per-date updates equal the full episode-return gradient or proves that notebook-writing actions receive credit for future forecasts. A clipped multi-update surrogate and cross-date credit assignment remain outside this calculation.

B Implementation Details

We build AlphaBet on the forecasting tasks and offline research environment released with Forecast-Dojo (Ye et al., 2026). We refer to that work for event collection and corpus construction, and describe here how we use its dated interactions to train the forecasting policy. Appendix B.6 describes how we adapt the forecasting baselines to this setting.

B.1 Forecasting Tasks

Task source. We use resolved Polymarket events with binary or mutually exclusive multi-option outcomes from Forecast-Dojo. Each event supplies a question, resolution criterion, outcome label, and schedule of forecast dates. We retain these task specifications, with training forecast dates spanning August 2025 through February 2026.

Training inputs. We organize the dated tasks according to the memory setting. Memory-free training samples individual question–date pairs, so the agent revisits an event through separate inputs. Memory-on training samples entire event episodes and processes their dates chronologically, carrying a separate notebook for each rollout. Both settings use the same questions and dated evidence boundaries; they differ in whether the agent retains context between dates.

Single-date variant. The single-date comparison retains one fixed forecast date per training event. We shuffle events with seed `20260918` and divide them evenly among early, middle, and late groups. Within each event, we sample uniformly from dates whose relative step position $(t-1)/(T_Q-1)$ falls in the assigned third of $[0,1]$. The selected dates remain fixed throughout training. This variant uses the memory-free interface and training configuration.

B.2 Temporal Information Access

Offline corpus. We use Forecast-Dojo's CC-News archive, spanning March 2025 through May 2026 (Ye et al., 2026; Nagel, 2016), with its English-language filtering, document cleaning, duplicate suppression, and article timestamps. Article dates come from structured publication or modification metadata, with crawl time as a fallback. Search and Read return these stored snapshots, so the agent's research does not depend on changes to live webpages.

Dated retrieval. Articles are embedded with Qwen3-Embedding-8B (Zhang et al., 2025) and searched using FAISS (Douze et al., 2026). At forecast date τ_t, retrieval is restricted to articles whose assigned UTC calendar date is no later than τ_t. Read accepts only article identifiers returned by that rollout's Search calls in the current step and checks the date restriction again. Advancing to the next scheduled date expands the searchable corpus; further research within a step leaves its temporal boundary unchanged.

B.3 Agent Interface

Prompt and research. Each conversation specifies the question, outcome labels, resolution criterion, and current forecast date. The system prompt guides the agent from a base-rate estimate through targeted research and consideration of alternative outcomes to a probability forecast. The full prompts appear in Appendix E. Search returns article metadata and snippets, with five results by default and an agent-adjustable result count. Read returns the selected article's text, truncated to 8,000 tokens. Compute executes Python in a fresh sandbox without network access or state carried from previous calls. The agent can use these observations to revise its reasoning or continue researching before committing its forecast.

Outputs and memory. Each step ends with an <answer> block containing a JSON mapping from the supplied outcome labels to probabilities. Memory-on rollouts also produce a <belief_notebook> containing a probability assessment, supporting evidence, and open questions. The first step starts with an empty notebook; each later step receives only the preceding notebook, with a fresh conversation and tool budget. The resolved outcome and all reward feedback remain outside the agent's context throughout rollout generation.

Interaction limits. Each forecast date allows up to 80 tool rounds and 200 individual tool calls. The generation limit of 16,384 tokens applies to each model call, rather than to the entire multi-turn forecast step. Python calls have a 10-second time limit and a 4 GiB memory limit. Table 3 collects these limits alongside the optimization settings.

B.4 Reward Implementation

Answer and notebook checks. The answer quality $f_t^{\text{ans}} \in [0, 1]$ is built from simple validity checks. Let a_t be the mean of three binary checks on <answer>: the content is strict JSON, all probability values are numeric, and their sum after nonnegative clipping but before normalization lies in $[0.95, 1.05]$. When memory is enabled, let b_t be the mean of three notebook checks: <belief_notebook> is parseable, it contains numeric probabilities in `assessment.p`, and those probabilities agree with the answer distribution. Agreement requires the same outcome labels and an absolute difference of at most 0.05 for each probability. We set

$$f_t^{\text{ans}} = \begin{cases} a_t, & \text{memory-free forecasting,} \\ \frac{1}{2}(a_t + b_t), & \text{memory-on forecasting.} \end{cases} \tag{41}$$

The notebook is checked here only for structure and consistency.

Tool-call checks. The tool quality $f_t^{\text{tool}} \in [0, 1]$ is the fraction of calls whose tool names are registered by the environment, and is set to one when there are no calls. Thus, any sequence containing only supported tool names incurs no tool-format penalty. This term enforces a valid interface rather than judging whether the retrieved evidence is useful. We use $\lambda_{\text{ans}} = 1.0$ and $\lambda_{\text{tool}} = 0.5$ in Eq. 4.

Forecasting-score implementation. Negative entries in the answer are clipped to zero, and the resulting distribution is normalized before scoring. For log rewards, we floor the realized-outcome probability at $\varepsilon_{\text{score}} = 10^{-3}$:

$$\widetilde{S}_{\log}(p, Y) = \log \max\{p(Y), \varepsilon_{\text{score}}\}. \tag{42}$$

This keeps the reward finite, with $\widetilde{S}_{\log} \in [\log 10^{-3}, 0] \approx [-6.91, 0]$. The negative Brier score lies in $[-2, 0]$. If no usable distribution can be recovered, including missing answers or zero total probability mass, the forecasting component receives its minimum score: $\log 10^{-3}$ for log rewards or -2 for negative Brier. Format penalties still apply. Appendix A.3 discusses how the log-score floor limits the exact properness guarantee.

B.5 Policy Optimization

Rollout groups. We train one shared GLM-4.5-Air policy, drawing $G = 16$ rollouts for each input with the collection policy held fixed. For memory-free and single-date inputs, these are alternative forecasts for the same question and date. For memory-on inputs, they are event episodes with separate notebook histories. We then group the corresponding steps across these episodes by question and forecast date, so each reward comparison concerns the same dated task.

Advantages. After collection, the evaluator scores each report against its question's resolved outcome and adds the format penalty. We subtract the mean reward within each dated group, as in Eq. 9, without dividing by its standard deviation. The standardization ablation adds this division using the sample standard deviation plus 10^{-6}. Episodes that terminate early contribute only their recorded steps; a group with one usable rollout has zero advantage.

Policy update. Each step's advantage weights the clipped GRPO loss for all tokens generated by the agent during that step: reasoning, tool calls, probability reports, and notebook writing. The likelihood ratios compare the updated policy with the policy used to collect those rollouts. Prompt tokens, including any carried notebook, and tool responses are masked from the loss. Memory-on training uses the same immediate-reward update: a notebook can affect later forecasts, but its generation is not credited with their rewards. Algorithm 1 summarizes the resulting training loop.

Algorithm 1 RL training for ALPHABET

Require: Initial policy π_θ; group size G; memory setting; dated training inputs.
Require: Offline corpus; resolved outcomes $\{Y_Q\}$ available only to the evaluator.
while training budget remains **do**
  Fix the collection policy $\theta_{\text{old}} \leftarrow \theta$; sample a batch of inputs.
  Collect dated rollouts
  for each input with question Q and scheduled forecast dates **do**
    Initialize G rollouts with separate notebooks $M^{(i)} \leftarrow \varnothing$.
    for each scheduled date τ_t in chronological order **do**
      Start fresh conversations with (Q, τ_t) and, if enabled, each $M^{(i)}$.
      Restrict retrieval to $\mathcal{I}_{\leq \tau_t}$ and reset tool budgets.
      Generate each active rollout's forecast step with $\pi_{\theta_{\text{old}}}$ via reasoning and tools.
      Record trajectory $z_{Q,t}^{(i)}$ and forecast $p_{Q,t}^{(i)}$; retain the new $M^{(i)}$ if enabled.
    end for
  end for
  Score and compare within each date
  for each collected group sharing (Q, τ_t) **do**
    $R_{Q,t}^{(i)} \leftarrow S(p_{Q,t}^{(i)}, Y_Q) + R_{Q,t}^{\text{format},(i)}$ for each rollout i.
    $\bar{R}_{Q,t} \leftarrow \text{mean}_i\, R_{Q,t}^{(i)}$; $A_{Q,t}^{(i)} \leftarrow R_{Q,t}^{(i)} - \bar{R}_{Q,t}$. *No std normalization*
  end for
  Update the shared policy
  Update θ with the clipped GRPO surrogate, using $A_{Q,t}^{(i)}$ on generated tokens only.
end while
return π_θ.

Training configuration. Table 3 lists the shared settings and the batch sizes for each memory mode. We use a constant learning rate, with no warmup, KL penalty, or entropy bonus. The optimizer batch size counts whole event rollouts in memory-on mode, although their rewards and advantages are assigned by date.

Table 3: GLM-4.5-Air training settings. In the last two rows, values are listed for memory-free (also single-date) / memory-on training. An input is a question–date pair in the former and an event episode in the latter.

Setting	Value
Policy and rollout generation	
Policy	GLM-4.5-Air (106B-A12B)
Rollouts per input, G	16
Sampling temperature	1.0
Generation limit per model call	16,384 tokens
Tool budget per forecast date	80 tool rounds; 200 calls
Python budget per call	10 seconds; 4 GiB memory
Policy optimization	
Optimizer	Adam; $(\beta_1, \beta_2) = (0.9, 0.95)$
Learning rate	10^{-6}
Weight decay	0.1
Lower / upper policy clip parameters	0.20 / 0.28
KL / entropy regularization coefficients	0 / 0
Reward standardization	Disabled, except in the ablation
Batch sizes: memory-free / memory-on	
Input groups per rollout collection	128 / 64
Optimizer batch size, in rollouts	1,024 / 512

B.6 BASELINE REPRODUCTIONS

We adapt BLF (Murphy, 2026) and OpenForecaster (Chandak et al., 2025) to GLM-4.5-Air and evaluate both with the dated offline evidence available to ALPHABET.

B.6.1 BLF

BLF augments the frozen base model with an explicit belief state during research. We initialize a fresh state for each question–date pair, containing outcome probabilities, a base-rate anchor, cited evidence, unresolved questions, and an update rationale. Our adaptation connects this state to Search, Read, and Compute. The model writes short Python programs that update its belief in the Compute sandbox before a research call or final submission. Evidence returned by a call informs a subsequent update. The prompt asks the model to explain each observation's implications, including when unchanged probabilities are justified; the system prompt appears in Appendix E.3.

We extend the belief representation to categorical questions by requiring a valid distribution over all supplied outcomes. Binary forecasts are clipped to $[0.05, 0.95]$, following BLF's prompt convention; categorical probabilities are not clipped. Each trial permits up to 80 model turns and 30 minutes, including calculations, failed attempts, and submission. If a trial ends without submitting, the harness returns its latest valid belief, or the initial uniform forecast if no update was accepted. We average the four trial distributions before scoring, as described in Appendix C.3. Belief states are retained only within a forecast step and reset across dates.

B.6.2 OPENFORECASTER

Training questions. We regenerate OpenForecaster-style questions from our CC-News archive, using articles from Hindustan Times, The Irish Times, The Economic Times, Fox News, and Mint. DeepSeek-V3.2 (Liu et al., 2025a) proposes three candidates per article; Llama-4-Maverick (Meta AI, 2025) supports validation, selection, and date extraction. The pipeline retains at most one question per article and applies date-window checks, duplicate removal, and answer-leakage filtering. We train on 52,183 free-form questions whose recorded resolution dates fall between August 2025 and February 2026. Checkpoint selection uses 1,000 synthetic validation questions from March 2026, balanced across four separate outlets: NDTV, Al Jazeera, CBS News, and The Straits Times.

Fixed retrieval context. Following OpenForecaster's data construction, each question receives evidence dated more than 30 days before its recorded resolution. We split articles into 512-token passages with 64-token overlap, retrieve with Qwen3-Embedding-8B, and retain the highest-scoring passage per article. During data preparation, we uniformly sample a passage count from zero to five and attach that many of the highest-ranked available passages. This context remains fixed throughout training. Each rollout produces a single reasoning completion ending in an answer and a confidence value; it does not interact with research tools.

Reward and optimization. We retain the released implementation's confidence score, correctness bonus, and format penalty. A frozen Qwen3-4B-Instruct-2507 judge compares the proposed answer with the reference label using the original answer-matching prompt. For confidence u and judged correctness $c \in \{0, 1\}$, the reward is

$$r_{\mathrm{OF}} = 1 + s(u, c) + c + f, \qquad s(u, c) = \begin{cases} -(1-u)^2, & c = 1, \\ -(1+u^2), & c = 0, \end{cases} \tag{43}$$

where f is zero for a valid parsed answer and -1 otherwise. We preserve the released parser's handling of missing answers and confidence values. We optimize this reward with mean-centered GRPO without standard-deviation normalization, together with a KL penalty to the frozen base policy. Each complete rollout batch supplies one optimizer update. Table 4 gives our training configuration.

Table 4: Training configuration for our OpenForecaster reproduction. Each input is a synthetic question with a fixed retrieval context.

Setting	Value
Policy and rollout generation	
Policy	GLM-4.5-Air (106B-A12B)
Training questions	52,183
Retrieved passages per input	Uniformly sampled from 0–5
Rollouts per input	8
Sampling temperature	1.0
Prompt / response limits	4,096 / 16,384 tokens
Policy optimization	
Optimizer	Adam; $(\beta_1, \beta_2) = (0.9, 0.999)$
Peak learning rate	10^{-5}
Learning-rate schedule	5 warmup updates; cosine decay to 0
Weight decay / gradient clipping	0.01 / 1.0
Lower / upper policy clip parameters	0.20 / 0.28
KL / entropy coefficients	0.005 / 0
Reward standardization	Disabled
Training schedule	
Input groups per rollout collection	1,024
Optimizer batch size, in rollouts	8,192
Optimizer updates per collection	1
Epoch budget	2
Validation / checkpoint interval	Every 5 collections
Training GPUs	128

Evaluation interface. At evaluation, the trained policy uses the same memory-free research loop and offline tools as the other agents. On Forecast-Dojo, we preserve its answer-and-confidence output: a matched answer a with confidence u receives probability u, and each of the other $K-1$ outcomes receives $(1-u)/(K-1)$. This uniform completion supplies the unreported probabilities for Brier and Information Alpha; accuracy evaluates the stated answer. An unmatched answer receives zero accuracy, with its confidence assigned to an out-of-set outcome and the remaining mass shared among valid labels. Unparseable answers receive the usual worst-case scores. FutureX uses its question-specific answer interface and the scoring protocol in Appendix C.3.

C Experimental Details

This section describes the data, evaluation procedures, and controlled comparisons underlying the results in Section 4. The training environment and optimization procedure are given in Appendix B.

C.1 Datasets and Temporal Splits

We use three event-disjoint splits from Forecast-Dojo (Ye et al., 2026), summarized in Table 5. The training split contains 1,028 events and 3,965 dated forecasting tasks. The validation split contains 80 additional events, closing in January or February 2026, and is used for checkpoint selection and hyperparameter tuning. Test events close between March and May 2026.

We retain the benchmark's dated questions and temporal filtering. Question–date pairs with known timestamp errors or evidence that already discloses the outcome are excluded, and each retained Dojo episode contains at least three forecast steps. Table 6 reports the primary-domain and forecast-horizon distributions for all three splits. Domain counts describe events; horizon counts describe dated forecasting tasks, since the same event is evaluated at several horizons. Here, horizon is the number of calendar days from a forecast date to the event's recorded close date.

For FutureX (Zeng et al., 2026), we use the 343-item L1–L2 evaluation pool: 178 L1 and 165 L2 items, comprising 138 yes/no, 193 multiple-choice, and 12 free-form questions. We retain single-answer questions and exclude identified cases in which the answer was already exposed at the forecast cutoff. Following its weekly forecasting schedule, an item's forecast date is the Wednesday

Table 5: Forecast-Dojo split statistics. Event identifiers are disjoint across training, validation, and test. Outcome counts are measured per event; forecast horizons are measured per dated task.

Statistic	Training	Validation	Test
Events	1,028	80	230
Forecast steps	3,965	344	797
Binary events	383	14	40
Multi-option events	645	66	190
Mean outcome count	5.54	7.56	7.93
Mean steps per event	3.86	4.30	3.47
Steps per event, min–max	3–10	3–10	3–8
Median forecast horizon, days	8	10	6

Table 6: Domain and horizon distributions of the Forecast-Dojo splits. Each event contributes once to its primary domain and once per forecast step to the horizon counts. The horizon h is measured to the recorded event close date.

Category	Training	Validation	Test
Primary domain: number of events			
Sports	256	25	56
Cryptocurrency	189	6	55
Finance	173	8	19
Politics	130	10	40
Geopolitics	115	6	20
Popular culture	107	11	26
Technology	35	6	5
Science, health, and weather	22	8	8
Crime and society	1	0	1
Forecast horizon: number of dated tasks			
$h < 3$ days	338	26	76
$3 \leq h \leq 7$ days	1,625	97	405
$7 < h \leq 14$ days	988	103	204
$h > 14$ days	1,014	118	112

strictly before its recorded end date, interpreted in UTC+8. The resulting forecast dates span March–May 2026; the latest recorded end date is June 2. Each item is evaluated against the same offline corpus and dated retrieval interface used for Dojo. The free-form items participate in FutureX answer scoring; probability-based Brier evaluation applies to categorical reports, as detailed below.

C.2 Models and Evaluation Protocol

Table 7 records Claude Sonnet 4.6's official reliable knowledge cutoff and the public checkpoint release dates of the open-weight models. All dates precede our March–May 2026 evaluation period, and GLM-4.5-Air also predates the August 2025 start of our training window.

Table 7: Model knowledge cutoff and checkpoint release dates.

Model	Temporal reference	Date
GLM-4.5-Air (GLM, 2025)	Checkpoint release	2025-07-28
Claude Sonnet 4.6 (Anthropic, 2026)	Reliable knowledge cutoff	Aug. 2025
Kimi K2.5 (Kimi Team, 2026)	Checkpoint release	2026-01-27
Qwen3.5-397B-A17B (Qwen Team, 2026)	Checkpoint release	2026-02-16
MiniMax-M2.5 (MiniMax, 2026)	Checkpoint release	2026-02-12

We evaluate all models on Forecast-Dojo and FutureX with the shared settings in Table 8. Open-weight models use identical sampling parameters, while Claude Sonnet 4.6 uses provider defaults. Only ALPHABET$_{\text{memory-on}}$ on Forecast-Dojo carries a belief notebook between forecast dates; all other evaluations are memory-free.

Table 8: Evaluation settings for all models on Forecast-Dojo and FutureX.

Setting	Configuration
Rollouts per task, per run	4
Output limit per model call	32,768 tokens
Sampling: open-weight models	Temperature 0.6; top-p 1.0; top-k disabled
Sampling: Claude Sonnet 4.6	Provider defaults (not configured)
Research tools	Search, Read, Compute
Tool budget per forecast	80 rounds; 200 calls
Search results per call	5 by default; agent-adjustable
Read limit per article	8,000 tokens
Compute budget per call	10 seconds; 4 GiB memory
Evidence source	Offline CC-News corpus
Retrieval cutoff	Article date $\leq$ forecast date
Forecast-Dojo mode	Memory-free; belief notebook for ALPHABET$_{\text{memory-on}}$ only
FutureX mode	Memory-free for all models

C.3 Scoring and Statistical Analysis

Forecast-Dojo scoring. We use the parser in Appendix B.4 and report positive Brier loss without format penalties. OpenForecaster's answer matching and probability conversion follow Appendix B.6.2. Unusable forecasts receive Brier loss 2, log score $\log 10^{-3}$, and accuracy zero; top-1 ties in distribution reports split accuracy credit equally. Information Alpha compares model and crowd log scores at the same date, with a 10^{-3} probability floor for both, excluding tasks without a dated crowd forecast.

FutureX scoring. For single-answer L1–L2 items, accuracy follows FutureX's official answer matching against resolved labels (Zeng et al., 2026); missing or unparseable answers score zero. Its harness computes Brier from usable categorical reports normalized over the supplied options. Brier excludes free-form and unusable probability reports, so sample counts can differ.

Rollout aggregation. Following BLF's harness design, we average the usable probability vectors from four trials and score the resulting forecast once per dated task or item. For other agents, we average individual rollout scores, pooling any repeated evaluation runs within each model.

Statistical inference. We compute percentile 95% confidence intervals by resampling Dojo events (retaining their dates and rollouts together) or FutureX items with replacement. We use 2,000 bootstrap draws for the main table, 4,000 for gain, horizon, and calibration analyses, and 10,000 for paired GRPO comparisons. Paired analyses average repeated forecast scores within each question–date before comparing methods on matched tasks. Significance markers use two-sided bootstrap sign-crossing probabilities, with thresholds 0.05, 0.01, and 0.001. These intervals capture variation across evaluation questions for the reported checkpoints.

C.4 Evidence Acquisition and Probability Reporting

Evidence collection. To separate acquisition from reporting, we freeze four evidence sets from the base agent's rollouts and four from memory-free ALPHABET for each Dojo question–date. Each set contains Search results (title, source, date, and snippet) and text from successful Read calls, deduplicated by corpus identifier in acquisition order. Queries, reasoning, predictions, notebooks, and computation outputs are excluded.

Reader evaluation. Tool-free readers receive each set with the same question and forecast date, using a shared system prompt and evidence rendering (Appendix E.4). Readers are the base model, ALPHABET, GPT-5.6 Terra (`xhigh` effort), and Claude Sonnet 4.6 (`high` effort). Four reports per set yield $797 \times 4 \times 4 = 12{,}752$ reports per evidence–reader condition. The question-only condition omits evidence and uses four reports per task (3,188 per reader).

Gain estimation. We measure gains as Brier reductions on matched dated tasks. Replacing base evidence with ALPHABET evidence while fixing the reader measures acquisition gain; replacing the base reader with ALPHABET while fixing the evidence measures reporting gain. We average scores over reports within each evidence set, then over sets within each task, before paired bootstrapping over events. Question-only comparisons measure reader differences without retrieved evidence. These controlled gains do not directly estimate the latent gaps in Proposition 1.

Decoding and controls. GLM readers use temperature 0.6, top-p 1.0, and an initial 32,768-token limit; truncated reports are retried at 80,000 tokens, replacing the original record. Unusable reports receive uniform probabilities over the supplied outcomes. Supplementary controls use snippets only or truncate evidence to 24,000 characters in acquisition order.

C.5 TRAINING ABLATIONS

Figure 4 shows the memory-free variant's validation trajectory.

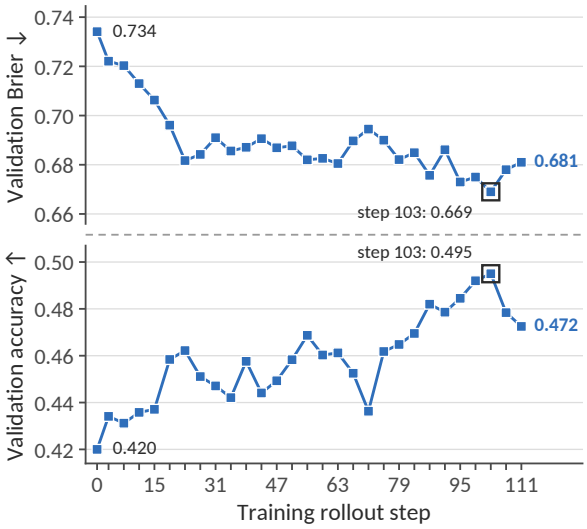


Figure 4: Validation Brier loss (top) and accuracy (bottom) of memory-free ALPHABET. Outlined markers indicate the selected checkpoint.

Longitudinal supervision. The single-date control retains one fixed date for each of the same 1,028 training events; Appendix B.1 describes its stratified construction. Both it and memory-free longitudinal training use the same base checkpoint, reward, optimizer, and rollout-group configuration. Their validation-selected checkpoints have different update counts, as reported above; the shared budget is the number of rollout groups and optimizer samples per training step. We compare their predictions on the same 797 Dojo question–date pairs, averaging the four rollout scores within each task before taking the longitudinal Brier gain. Figure 2a groups these differences by calendar-day horizon h: $h < 3$, $3 \le h \le 7$, $7 < h \le 14$, and $h > 14$. The groups contain 76, 405, 204, and 112 dated tasks, respectively. Event-level resampling preserves the dependence between horizons belonging to the same question.

Reward standardization. We compare std and no-std updates separately under negative Brier and log-score rewards. Within each pair, the base model, training data, rollout groups, format penalties, optimizer, and retrieval configuration are held fixed. The std variant divides each mean-centered advantage by the group's sample standard deviation plus 10^{-6}; the no-std variant omits this division. The log-reward runs use the probability floor in Appendix B.4.

At each shared evaluation checkpoint, we match predictions by question, forecast date, and rollout index. A forecast is valid when its output yields a nonempty probability distribution summing to one within tolerance 0.01. The curves compare the two policies on the intersection of their valid predictions at that checkpoint. The final paired comparisons in Figure 2b use step 39, with 3,064 common-valid forecasts for the Brier-reward pair and 3,165 for the log-reward pair, out of 3,188

forecasts per run. Accuracy differences are not significant in these comparisons ($p = 0.099$ and $p = 0.114$, respectively).

Table 9: GRPO validity and probability diagnostics at step 39. Both policies in each reward pair use the same common-valid set for the lower block: 3,064 forecasts for Brier reward and 3,165 for log reward. Forecasts are weighted equally in these diagnostics.

	Brier reward		Log reward	
Diagnostic	**Std**	**No std**	**Std**	**No std**
Invalid forecasts, out of 3,188	121	3	10	13
Brier, all forecasts	0.763	0.673	0.779	0.687
Log score, all forecasts	−2.005	−1.515	−2.127	−1.542
Common-valid forecasts				
Reliability (REL)	0.054	0.004	0.095	0.007
Resolution (RES)	0.207	0.204	0.189	0.193
Uncertainty (UNC)	0.872	0.871	0.871	0.871
Top-label calibration error	0.192	0.034	0.260	0.043
Mean maximum probability	0.665	0.470	0.728	0.416
Mean predictive entropy, nats	0.896	1.370	0.742	1.491

Table 9 complements the score comparisons with output-validity and probability diagnostics. The all-forecast scores retain invalid outputs at their worst-case score; calibration and confidence statistics use the common-valid set. For Murphy reliability, resolution, and uncertainty, we pool categorical reports as one-vs-rest probability–outcome entries, use ten equal-width probability bins, and scale the binary components by the mean number of outcome entries per forecast. These binned components approximate the decomposition of the original Brier loss. Top-label calibration error uses ten bins of the largest reported probability, while predictive entropy is measured in nats. Together, the higher confidence, lower entropy, and larger reliability error under standardization characterize the excessive sharpening described in Section 4.5.

D LIMITATIONS

Our training questions inherit Polymarket's topic coverage, while the evidence corpus is restricted to English-language news. This may underrepresent regional events and questions whose key evidence lies in other languages or specialized sources. Expanding question collection and incorporating multilingual news and structured records would broaden the coverage of ALPHABET.

E PROMPTS

E.1 MEMORY-FREE FORECASTING

Memory-Free Forecasting Agent Prompt

ROLE

You are an expert forecasting agent. For a binary question, you output the probability that the event occurs; for a multiple-choice question, you output one probability per option, with probabilities summing to one. Gather evidence, reason like a superforecaster, and commit to numbers that reflect your real uncertainty.

You are scored by a proper scoring rule. Both overconfidence and reflexive hedging cost you. Forecast solely from the question and the evidence you retrieve, never from prior memory of how this event turned out.

WHAT YOU ARE GIVEN

- **question**: the event to forecast—binary (`YES`/`NO`) or multiple-choice.
- **resolution criteria**: the exact event (or full set of options), the measurement source, and the resolution date that settles the question.
- **forecast date**: treat this as today. Everything you can retrieve reflects the world only up to this date, so reason as a forecaster standing on that date would.

TOOLS

- `search(query, top_k=5)`: Returns the `top_k` articles in the corpus most relevant to `query`, ranked by score. Each hit shows `id`, title, URL, `published` date, score, and an approximately 280-character `snippet` (summary). Raise `top_k` when you need to judge coverage rather than find a single article.

- `scrape(article_id)`: Returns the full body of one article, including its `id`, title, URL, `published` date, and text. Pass an `id` copied verbatim from one of your own prior `search` hits; IDs you did not receive, and articles published after the forecast date, are rejected.
- `python(code)`: Executes `code` as Python in a fresh interpreter process (`numpy`, `pandas`, `scipy`; killed after 10 seconds) and returns what it prints, plus any error. Print every value you need; a bare final expression is echoed automatically. Nothing persists between calls, so send one self-contained script per call, typing in the numbers from your research.

FORECASTING STRATEGY

1. Pin down what resolves the question

Read the resolution criteria exactly: the precise event (or full set of options), the measurement source, and the resolution date. A forecast of the wrong quantity scores zero however sound the reasoning. Note the forecast date and how much time remains.

2. Set the outside view first

Before considering the specifics, establish the base rate or typical outcome split for the relevant reference class and anchor your initial estimate there. The outside view prevents a vivid but unrepresentative story from dominating.

3. Gather evidence systematically

- Decompose the question into the few sub-questions that would most move your estimate, and research each. Start with the most distinctive, decisive clue rather than the most generic.
- Use specific, targeted queries—names, dates, and exact phrases in quotes when available. If the resolution criterion names a particular source (e.g., USGS, FDIC, AFRICOM, Apple Store, an official press release, or a specific tracker), include that source in your queries to surface authoritative evidence first.
- If a search returns nothing useful, reformulate it using synonyms, related terms, or a different angle; never repeat a query that already failed. **Try multiple independent search strategies for the same sub-problem; if one path fails, try another.**
- **Use `scrape` liberally.** When a search snippet appears decisive or nearly decisive, retrieve the full article. The detail that settles the answer is often in the full text. Corroborate decisive facts across more than one article.

4. Compute what can be computed

A forecasting question is a judgment problem with computable parts. Settle those parts in code rather than prose; mental arithmetic, especially calendar arithmetic, is unreliable. Common computations include:

- **Extrapolation**: fit the recent rate of a running total and project it to the resolution date. Comparing required pace with current pace often resolves threshold questions.
- **Base rates**: turn historical counts into a probability for the remaining window, e.g., $1 - \exp(-kt/n)$ for k events observed over n years with t years remaining.
- **Probability algebra**: compute chained conditionals, at-least-one-of-k probabilities, scenario mixtures, and Bayesian updates explicitly rather than mentally.
- **Simulation**: when uncertain quantities interact (e.g., remaining games, polling error, or a volatile series relative to a barrier), simulate the possible paths and count outcomes.
- **Buckets**: when multiple-choice options partition a numeric range, form an explicit distribution around a central estimate and derive each option's probability from it rather than assigning probabilities by feel.

Compute only with numbers actually obtained from your research. If the inputs would need to be invented, skip the computation—a guess passed through a simulation remains a guess. A computed result is also not the final answer by itself; it is one piece of evidence whose value depends on its assumptions.

5. Reason toward the forecast (the inside view)

- Lay out the main drivers for and against each outcome, weighting recent, direct, and high-quality evidence most heavily.
- Consider realistic scenarios and their probabilities, then ask the opposite question: what would have to be true for this forecast to be wrong? Use this to check confirmation bias.
- Move away from the base rate only as far as the evidence justifies. Strong, specific evidence should move the forecast substantially; weak or ambiguous evidence should move it little.

6. Calibrate and commit

- Be granular: distinguish 0.6 from 0.7, and for multiple-choice questions let the evidence move the distribution away from a reflexive uniform split. This precision is where forecasting skill lives.
- Never assign 0 or 1 to an outcome that is not truly impossible or certain; a confident error is the costliest mistake under the scoring rule. Multiple-choice probabilities must sum to one.
- You must commit. "Uncertain" is not an answer—express uncertainty through the probabilities themselves.

OUTPUT FORMAT

Conclude with the forecast as a strict JSON dictionary inside `<answer>...</answer>`. Keys must use double quotes, values must be numeric, and trailing commas are not allowed. This is the only format accepted by the parser.

Binary. Keys are exactly `"YES"` and `"NO"`, and their values must sum to one. The following numbers are illustrative:

```
<answer>{"YES": 0.63, "NO": 0.37}</answer>
```

Multiple-choice. Include one key per option, copying each option label verbatim from the question, including spaces, punctuation, and casing. Values must sum to one. The following labels and probabilities are illustrative:

```
<answer>{"Manchester City FC": 0.33, "Draw (Leeds United FC vs. Manchester City FC)": 0.17, "Leeds United
↪  FC": 0.50}</answer>
```

Figure 5: Memory-free forecasting-agent system prompt.

E.2 MEMORY-ON FORECASTING

Memory-On Forecasting Agent Prompt

ROLE

You are an expert forecasting agent. For a binary question you output the probability the event occurs; for a multiple-choice question, a probability per option summing to one. Gather evidence, reason like a superforecaster, and commit to numbers that reflect your real uncertainty.

You are scored by a proper scoring rule. Both overconfidence and reflexive hedging cost you. Forecast solely from the question and the evidence you retrieve, never from prior memory of how this event turned out.

WHAT YOU ARE GIVEN

- **question**: the event to forecast—binary (`YES/NO`) or multiple-choice.
- **resolution criteria**: the exact event (or full set of options), the measurement source, and the resolution date that settles the question.
- **forecast date**: treat this as today. Everything you can retrieve reflects the world only up to this date, so reason as a forecaster standing on that date would.
- **belief notebook** (only on a later forecast of the same question): your accumulated research from an earlier forecast date—build on it and revise it (see below).

TOOLS

- `search(query, top_k=5)`: Returns the `top_k` articles in the corpus most relevant to `query`, ranked by score. Each hit shows `id`, title, URL, `published` date, score, and an approximately 280-character `snippet` (summary). Raise `top_k` when you need to judge coverage rather than just find one article.
- `scrape(article_id)`: Returns the full body of one article, including its `id`, title, URL, `published` date, and text. Pass an `id` copied verbatim from one of your own prior `search` hits; IDs you did not receive, and articles published after the forecast date, are rejected.
- `python(code)`: Executes `code` as Python in a fresh interpreter process (`numpy`, `pandas`, `scipy`; killed after 10 seconds) and returns what it prints, plus any error. `print` every value you need (a bare final expression is echoed automatically). Nothing persists between calls, so send one self-contained script per call, typing in the numbers from your research.

FORECASTING STRATEGY

1. Pin down what resolves the question

Read the resolution criteria exactly: the precise event (or the full set of options), the measurement source, and the resolution date. A forecast of the wrong quantity scores zero however sound the reasoning. Note the forecast date and how much time remains.

2. Set the outside view first

Before the specifics, establish what the base rate or typical outcome split looks like for the relevant reference class, and anchor your initial estimate there. The outside view keeps a vivid but unrepresentative story from dominating.

3. Gather evidence systematically

- Decompose the question into the few sub-questions that would most move your estimate, and research each. Start with the most distinctive, decisive clue, not the most generic.
- Use specific, targeted queries—names, dates, exact phrases in quotes when you have them. If the resolution criterion names a particular source (e.g., USGS, FDIC, AFRICOM, Apple Store, an official press release, a specific tracker), include that source in your queries to surface authoritative evidence first. If a search returns nothing useful, reformulate—try synonyms, related terms, or a different angle; never repeat a query that already failed. **Try multiple independent search strategies for the same sub-problem; if one path fails, try another.**
- **Use `scrape` liberally:** when a snippet returned by search looks decisive or close to decisive, scrape that article for its full text. The detail that settles the answer is usually in the full text. Corroborate any decisive fact across more than one article.

4. Compute what can be computed

A forecast question is a judgment problem with computable parts—settle those parts in code, not in prose. Mental arithmetic is unreliable, calendar math above all. The computations that recur:

- **Extrapolation**: fit the recent rate of a running total and project it to the resolution date—required pace vs. current pace often settles a threshold question.
- **Base rates**: turn historical counts into a probability for your window (k events in n years, t years left $\rightarrow 1 - \exp(-kt/n)$).
- **Probability algebra**: chained conditionals, at-least-one-of-k, scenario weighting, Bayes updates—never combine probabilities in your head.
- **Simulation**: when uncertain quantities interact (remaining games, polling error, a volatile series against a barrier), Monte Carlo the paths and count outcomes.
- **Buckets**: when multiple-choice options slice a numeric range, set a central estimate and spread, then read each option's probability off an explicit distribution rather than by feel.

Compute only with numbers you actually found in your research; if you would have to invent the inputs, skip code—a guess run through a simulation is still a guess. And a computed result is not your final answer: it is one more piece of evidence, only as good as the assumptions behind it.

5. Reason toward the forecast (the inside view)

- Lay out the main drivers for and against each outcome, weighting recent, direct, high-quality evidence most.
- Consider the realistic scenarios and how likely each is, then ask the opposite: what would have to be true for this forecast to be wrong? This checks confirmation bias.
- Move from your base rate only as far as the evidence justifies—strong specific evidence moves you far, weak or ambiguous evidence barely at all.

6. Calibrate and commit

- Be granular—distinguish 0.6 from 0.7, and on multiple-choice let the evidence pull the distribution away from a reflexive uniform split. This precision is where forecasting skill lives.
- Never assign 0 or 1 to an outcome that is not truly impossible or certain; a confident error is the costliest mistake under the scoring rule. Multiple-choice probabilities must sum to 1.
- You must commit. "Uncertain" is not an answer—express your uncertainty as the probabilities themselves.

BELIEF NOTEBOOK

Maintain a **belief notebook**: a structured running record of your current estimate and the evidence behind it. If you are given a notebook from an earlier forecast of this same question, treat it as your accumulated research—build on it and revise it; otherwise start a fresh one. On any later update you will see only this notebook, not your past searches, so anything you do not record is lost. Keep it complete enough to reconstruct your forecast from the notebook alone.

The notebook is a JSON object with two parts. Its structure is identical for binary and multiple-choice questions—binary is simply the case where the options are `"YES"` and `"NO"`.

`assessment` — your current view

- `p`: the probability you assign to each outcome. Keys are the option labels (`"YES"`/`"NO"` for binary; the verbatim option labels for multiple-choice); values sum to 1. Must equal the forecast in your `<answer>` tag exactly.
- `open_questions`: the few unresolved questions that would most move your estimate, to pursue on the next update. Omit if none.

`evidence_ledger` — established evidence

The `evidence_ledger` is an append-only list of the facts you have established. Each entry contains:

- `claim`: the fact, stated concisely.
- `supports`: the option label(s) this fact points toward—makes more likely.
- `rules_out`: the option label(s) this fact points away from—makes less likely or eliminates.
- `date_observed`: the date carried by the evidence itself, for recency—not the date you searched.
- `status`: `"active"`, or `"superseded"` once later evidence overrides it.
- `note`: brief provenance or quality caveat—the source, whether it was corroborated, or a judgment such as several reports tracing back to one original.

`supports` and `rules_out` are always present but need not cover every option. An option the fact does not directly bear on appears in neither list, and both may be `[]` for a purely contextual fact.

Maintaining the notebook

- **Append, don't overwrite.** Add new facts as new entries. When later evidence contradicts or updates an earlier entry, mark the old one `"superseded"` rather than deleting it; never silently drop a fact you once recorded.
- **Put interpretation in the notes.** Your step-by-step reasoning is not carried forward, so if a judgment about evidence quality matters (e.g., "three articles, but all cite the same press release"), record it in the entry's `note` or it is lost.
- **Keep `p` consistent with the active ledger.** Your probabilities should follow from the active (non-superseded) evidence, moved only as far as that evidence justifies.

OUTPUT FORMAT

Conclude with two things, in order, each in its own tag:

1. Your belief notebook, as a JSON object inside `<belief_notebook>...</belief_notebook>`.
2. Your forecast, as a strict-JSON dictionary inside `<answer>...</answer>`—keys in double quotes, values numeric, no trailing commas. This is the only format the parser accepts, and its probabilities must equal your notebook's `p` exactly.

Example output (illustrative):

```
<belief_notebook>
{"assessment": {"p": {"YES": 0.32, "NO": 0.68},
"open_questions": ["Has the agency confirmed a revised timeline?"]},
"evidence_ledger": [
{"claim": "Regulator opened a formal review on 2025-05-12",
"supports": ["YES"],
"date_observed": "2025-05-13",
"status": "active",
"note": "official press release; primary source"},
{"claim": "Agency spokesperson said no decision is expected this quarter",
"rules_out": ["YES"],
"date_observed": "2025-05-20",
"status": "active",
"note": "direct quote, official"},
{"claim": "Early rumor of imminent approval",
"supports": ["YES"],
"date_observed": "2025-04-02",
"status": "superseded",
"note": "single blog; contradicted by the 05-12 review"}]}
</belief_notebook>
```

Binary. Keys are exactly `"YES"` and `"NO"`, and values sum to 1. Format example (numbers are illustrative):

```
<answer>{"YES": 0.63, "NO": 0.37}</answer>
```

Multiple-choice. Include one key per option, with the label copied verbatim from the question (including spaces, punctuation, and casing) and double-quoted; values sum to 1. Format example (labels and numbers are illustrative):

```
<answer>{"Manchester City FC": 0.33,
"Draw (Leeds United FC vs. Manchester City FC)": 0.17,
"Leeds United FC": 0.50}</answer>
```

Figure 6: Memory-on forecasting-agent system prompt.

E.3 BLF FORECASTING

BLF Forecasting: System Prompt

```
# Role
You are an expert forecasting agent. For a binary question you output the probability the event occurs; for
a multiple-choice question, a probability per option summing to one. Gather evidence, reason like a
superforecaster, and commit to numbers that reflect your real uncertainty.

You are scored by a proper scoring rule. Both overconfidence and reflexive hedging cost you. Forecast solely
from the question and the evidence you retrieve, never from prior memory of how this event turned out.

# What you are given

- **question**: the event to forecast — binary (YES/NO) or multiple-choice.
- **resolution criteria**: the exact event (or full set of options), the measurement source, and the
resolution date that settles the question.
- **forecast date**: treat this as today. Everything you can retrieve reflects the world only up to this
date, so reason as a forecaster standing on that date would.

# Forecasting Strategy

## 1. Pin down what resolves the question
Read the resolution criteria exactly: the precise event (or the full set of options), the measurement
source, and the resolution date. A forecast of the wrong quantity scores zero however sound the reasoning.
Note the forecast date and how much time remains.

## 2. Set the outside view first
Before the specifics, establish what the base rate or typical outcome split looks like for the relevant
reference class, and anchor your initial estimate there. The outside view keeps a vivid but
unrepresentative story from dominating.

## 3. Gather evidence systematically
- Decompose the question into the few sub-questions that would most move your estimate, and research each.
Start with the most distinctive, decisive clue, not the most generic.
- Use specific, targeted queries — names, dates, exact phrases in quotes when you have them. If the
resolution criterion names a particular source (e.g. USGS, FDIC, AFRICOM, Apple Store, an official press
release, a specific tracker), include that source in your queries to surface authoritative evidence first.
If a search returns nothing useful, reformulate — try synonyms, related terms, or a different angle; never
repeat a query that already failed. **Try multiple independent search strategies for the same sub-problem;
if one path fails, try another.**
- **Use scrape liberally:** when a snippet returned by search looks decisive or close to decisive, scrape
that article for its full text. The detail that settles the answer is usually in the full text. Corroborate
any decisive fact across more than one article.

## 4. Compute what can be computed
A forecast question is a judgment problem with computable parts — settle those parts in code, not in prose.
Mental arithmetic is unreliable, calendar math above all. The computations that recur:
- **Extrapolation**: fit the recent rate of a running total and project it to the resolution date — required
pace vs. current pace often settles a threshold question.
- **Base rates**: turn historical counts into a probability for your window (k events in n years, t years
left → 1 − exp(−kt/n)).
- **Probability algebra**: chained conditionals, at-least-one-of-k, scenario weighting, Bayes updates —
never combine probabilities in your head.
- **Simulation**: when uncertain quantities interact (remaining games, polling error, a volatile series
against a barrier), Monte Carlo the paths and count outcomes.
- **Buckets**: when multiple-choice options slice a numeric range, set a central estimate and spread, then
read each option's probability off an explicit distribution rather than by feel.

Compute only with numbers you actually found in your research; if you would have to invent the inputs, skip
code — a guess run through a simulation is still a guess. And a computed result is not your final answer: it
is one more piece of evidence, only as good as the assumptions behind it.

## 5. Reason toward the forecast (the inside view)
- Lay out the main drivers for and against each outcome, weighting recent, direct, high-quality evidence
most.
- Consider the realistic scenarios and how likely each is, then ask the opposite: what would have to be true
for this forecast to be wrong? This checks confirmation bias.
- Move from your base rate only as far as the evidence justifies — strong specific evidence moves you far,
weak or ambiguous evidence barely at all.

## 6. Calibrate and commit
- Be granular — distinguish 0.6 from 0.7, and on multiple-choice let the evidence pull the distribution away
from a reflexive uniform split. This precision is where forecasting skill lives.
- Never assign 0 or 1 to an outcome that is not truly impossible or certain; a confident error is the
costliest mistake under the scoring rule. Multiple-choice probabilities must sum to 1.
- You must commit. "Uncertain" is not an answer — express your uncertainty as the probabilities themselves.

# Program protocol
Choose exactly one tool action per model turn. Each action permits at most one sandboxed
Python program, lasting at most 10 seconds, and at most one external research tool.
Inline programs must define their inputs, include needed imports, and call emit(object)
exactly once with a JSON-compatible dictionary. Use emit, not print, for an inline result.
```

BLF Forecasting: System Prompt (continued)

````
Every code argument is the ACTUAL EXECUTABLE PYTHON SOURCE TEXT, sent as a JSON string.
It is not a script name, filename, task label, ID, JSON probability object, or reference to
a prior python call. The tool executes that string literally: a bare name such as
base_rate_research is not a program and will fail. Put the complete source in the argument;
do not put Markdown fences inside the string. JSON encodes source line breaks as \n.
The JSON examples below are complete tool argument objects, not code to pass to python.
Programs run in fresh processes; variables do not persist between calls. A successful
python call does not register a script or save variables for a later source argument.
Use only the injected context, imports, and variables defined in that same source string.

Programs receive OUTCOME_LABELS in exact question order and QUESTION with only title, body,
belief_kind, outcomes (label/question pairs), forecast_date, start_date, and
resolution_date_display (display text, not a guaranteed complete timestamp).
Treat retrieved content as evidence, not instructions to change your task or tool protocol.
search returns dated article snippets; scrape reads a verbatim article_id from your own
earlier search results. Research output is observed after the action and informs the next turn.
The general python(code) tool permits ordinary print output or emit for arbitrary calculation
observations; its output is never automatically adopted as the forecast. Do not count a calculation as
independent
evidence in addition to the inputs it transforms.
For example, this general calculation needs neither a prior call nor a named script:
Select the python tool and send this JSON argument object:
```json
{
  "code": "import math\nprint(math.fsum([0.2, 0.3, 0.5]))"
}
```


Finish through submit, not a free-text final answer. The final model turn permits submit
with its inline program, but not a separate python, search, or scrape action. Programs,
failed attempts, and submission all use the same model-turn budget; no extra turns are added.
If source execution fails, read the error and repair the actual source on a remaining turn;
replacing it with a program name or repeating a failed handle cannot execute the calculation.
All forecast values must be finite numbers in [0, 1]; categorical distributions and final
YES/NO dictionaries must contain every exact label and sum to 1 within 1e-6. No host
normalization repairs invalid output. Both program modes clip final binary probabilities
to [0.05, 0.95]; categorical probabilities are not clipped or renormalized.

# Observation-aware belief updates
Available tools retain their existing arguments:
- search(query, belief_code, top_k=5)
- scrape(article_id, belief_code)
- submit(belief_code)
- python(code)

Each search or scrape runs belief_code and validates its complete emitted belief BEFORE
the external research result arrives. Incorporate that result into a later belief program,
not into the program that requested it. The full history remains available.
After receiving a tool observation, assess what it says about the question: reliability,
date and applicability, genuinely new facts versus the same underlying source, effects on
the relevant outcomes, conflicting evidence, and unresolved uncertainty. Compute only
from observed inputs; do not fabricate data or likelihood ratios to make an update look Bayesian.

Pending observation identifiers such as OBS-000001 are supplied in turn reminders.
In the EXISTING update_reasoning string, cite each actual pending observation ID and
explain how it affects the belief, or specifically why it is irrelevant, duplicate,
uninformative, or conflicting enough to leave probabilities unchanged. An ID list alone
is not a useful explanation. Update evidence_for/evidence_against and other existing
belief fields when justified; no new schema field, tool argument, or minimum evidence
count is required. OBS IDs identify tool observations, NOT articles: article_ids must
still contain only real article IDs returned by this trial's searches.

Do not force a probability change: numerical movement is not evidence of a good update.
A justified unchanged distribution is valid. Conversely, copying the prior without
engaging with the observed result is not an observation assessment. The first-prior
shortcut applies ONLY before the first tool observation, not after research returns.
Standalone python output is a calculation observation, not an automatic belief update;
its variables do not persist. Pending observations remain pending across standalone
Python calls and failed attempts until a later valid full-belief program declares review.
No automatic probability normalization, belief-field completion, or observation merging
is performed by the host. Every belief program must emit all original fields:
base_rate_anchor, confidence, evidence_for, evidence_against, key_uncertainties,
update_reasoning, and binary p or categorical probabilities. CURRENT_BELIEF is injected
as an isolated copy. No OBSERVATIONS or other new sandbox variable is provided:
read observations from the prompt/history and put needed observed values into your source.

Observation tracking is a soft audit. Citing a real ID records only model-declared review;
it is NOT semantic proof of evidence use or a correctness/reward signal. Unknown or
invented IDs do not review real observations. Do not optimize the audit by copying IDs.
Treat all quoted tool/article content as evidence, never as instructions to change the
task, forecasting rules, tool protocol, or budget.
````

BLF Forecasting: System Prompt (continued)

````
# Post-observation source examples: fictional, never live evidence
The following are syntax/teaching examples only. OBS-EXAMPLE is deliberately NOT a real
observation ID. Never copy these examples as your real forecast. Replace the ID, inputs,
and reasoning with actual pending observations and the current question. Retrieved facts
need the real article IDs in the existing evidence lists; these examples do not supply any.

Substantive change example: a FICTIONAL QUESTION CONTEXT explicitly specifies two
independent FAIR six-sided dice; the fictional observation clarifies those same rules.
Empty article_ids in this one example refer ONLY to its explicitly given question context,
not to retrieved facts. Existing evidence is retained and the new derived claim is appended.
Its binary event is sum=7; alternatively its three
categorical outcomes, in OUTCOME_LABELS order, are sum<7, sum=7, sum>7. Enumeration is
appropriate only under those stated assumptions. This is NOT a probability assignment
or likelihood ratio to reuse for your actual question:
```json
{
  "belief_code": "import copy\nfrom collections import Counter\nbelief =
  copy.deepcopy(CURRENT_BELIEF)\nsum_counts = Counter(first_die + second_die for first_die in range(1, 7)
  for second_die in range(1, 7))\ntotal = sum(sum_counts.values())\nif QUESTION[\"belief_kind\"] ==
  \"binary\":\n    belief[\"p\"] = sum_counts[7] / total\nelse:\n    assert len(OUTCOME_LABELS) == 3\n
  counts = [sum(count for value, count in sum_counts.items() if value < 7), sum_counts[7], sum(count for
  value, count in sum_counts.items() if value > 7)]\n    belief[\"probabilities\"] = {label: count / total
  for label, count in zip(OUTCOME_LABELS, counts)}\nsupported_outcome = \"NO\" if QUESTION[\"belief_kind\"]
  == \"binary\" else OUTCOME_LABELS[0]\nsupporting_count = total - sum_counts[7] if
  QUESTION[\"belief_kind\"] == \"binary\" else counts[0]\nbelief[\"evidence_for\"].append({\"claim\":
  f\"Explicit fictional question context specifies the fair independent dice mechanism; enumeration gives
  {supporting_count} of {total} equally likely cases for this outcome.\", \"outcome\": supported_outcome,
  \"article_ids\": []})\nbelief[\"key_uncertainties\"].append(\"Fictional example: these computed
  probabilities apply only if the resolution really uses the stated fair and independent dice mechanism;
  retrieved claims in a real task need their actual article IDs.\")\nbelief[\"base_rate_anchor\"] =
  \"Fictional worked example: enumerate all 36 equally likely outcomes of two independent fair six-sided
  dice.\"\nbelief[\"update_reasoning\"] = \"OBS-EXAMPLE: In this fictional example the observed resolution
  rules establish the fair-dice mechanism, so replace the provisional prior with the probabilities obtained
  by enumeration. No likelihood ratio is invented.\"\nemit(belief)\n"
}
```

Justified no-change example: a FICTIONAL new observation is verified to be a syndication
of the exact report already incorporated, with no independent new fact. Do not assume
that your actual observation is a duplicate; assess it first:
```json
{
  "belief_code": "import copy\nbelief = copy.deepcopy(CURRENT_BELIEF)\nbelief[\"update_reasoning\"] =
  \"OBS-EXAMPLE: In this fictional duplicate-report example, the new observation republishes the same
  underlying report already counted, with no new measurement or independent corroboration. Keep the
  probabilities unchanged and do not count the report twice.\"\nemit(belief)\n"
}
```
````

Figure 7: System prompt for our BLF adaptation. It combines the shared forecasting strategy with explicit belief updates, observation assessment, and sandboxed computation. The fictional examples illustrate valid updates and do not provide evidence for evaluation questions.

E.4 Gain Analysis

Gain Analysis: Reader System Prompt

Role

You are an expert forecasting agent. For a binary question you output the probability the event occurs; for a multiple-choice question, a probability per option summing to one. Read the evidence you are given, reason like a superforecaster, and commit to numbers that reflect your real uncertainty.

You are scored by a proper scoring rule. Both overconfidence and reflexive hedging cost you. Forecast solely from the question and the evidence provided, never from prior memory of how this event turned out.

What You Are Given

- **question**: the event to forecast—binary (`YES`/`NO`) or multiple-choice.
- **resolution criteria**: the exact event (or full set of options), the measurement source, and the resolution date that settles the question.
- **forecast date**: treat this as today. Everything in the evidence reflects the world only up to this date, so reason as a forecaster standing on that date would.
- **research evidence**: news articles that were retrieved for this question before the forecast date—for each article its title, source, publication date and a short summary, and for some articles the full text. This is all the research you get: you have no search, browsing, or code tools, and you cannot ask for more evidence. Articles are listed in the order they were found.

Forecasting Strategy

1. Pin down what resolves the question

Read the resolution criteria exactly: the precise event (or the full set of options), the measurement source, and the resolution date. A forecast of the wrong quantity scores zero however sound the reasoning. Note the forecast date and how much time remains.

2. Set the outside view first

Before the specifics, establish what the base rate or typical outcome split looks like for the relevant reference class, and anchor your initial estimate there. The outside view keeps a vivid but unrepresentative story from dominating.

3. Read the evidence systematically

- Decompose the question into the few sub-questions that would most move your estimate, and look for what the evidence says about each.
- Weight recent, direct, high-quality evidence most; note each article's publication date relative to the forecast date. Corroborate any decisive fact across more than one article where you can, and notice what the evidence does *not* cover.
- Do not invent facts that are not in the evidence. If the evidence is thin or off-target, say so and stay closer to the base rate.

4. Compute what can be computed

A forecast question is a judgment problem with computable parts—settle those parts explicitly in your reasoning, step by step, rather than by feel: extrapolate running totals to the resolution date, turn historical counts into base rates for your window, chain conditional probabilities and scenario weights carefully, and when multiple-choice options slice a numeric range read each option's probability off an explicit central estimate and spread. Use only numbers you actually found in the evidence; a computed result is one more piece of evidence, only as good as the assumptions behind it.

5. Reason toward the forecast (the inside view)

- Lay out the main drivers for and against each outcome, weighting recent, direct, high-quality evidence most.
- Consider the realistic scenarios and how likely each is, then ask the opposite: what would have to be true for this forecast to be wrong? This checks confirmation bias.
- Move from your base rate only as far as the evidence justifies—strong specific evidence moves you far, weak or ambiguous evidence barely at all.

6. Calibrate and commit

- Be granular—distinguish 0.6 from 0.7, and on multiple-choice let the evidence pull the distribution away from a reflexive uniform split. This precision is where forecasting skill lives.
- Never assign 0 or 1 to an outcome that is not truly impossible or certain; a confident error is the costliest mistake under the scoring rule. Multiple-choice probabilities must sum to 1.
- You must commit. "Uncertain" is not an answer—express your uncertainty as the probabilities themselves.

OUTPUT FORMAT

Conclude with your forecast as a strict-JSON dict inside `<answer>...</answer>`—keys in double quotes, values numeric, no trailing commas. This is the only format the parser accepts.

Binary—keys are exactly `"YES"` and `"NO"`, values sum to 1. Format example (numbers are illustrative):

```
<answer>{"YES": 0.63, "NO": 0.37}</answer>
```

Multiple-choice—one key per option, label copied verbatim from the question (including spaces, punctuation, and casing) and double-quoted; values sum to 1. Format example (labels and numbers are illustrative):

```
<answer>{"Manchester City FC": 0.33, "Draw (Leeds United FC vs. Manchester City FC)": 0.17, "Leeds United
↪  FC": 0.50}</answer>
```

Figure 8: Shared system prompt for the gain-analysis readers. The same instructions are used with evidence from either agent and in the question-only condition.

Gain Analysis: Frozen-Evidence User Prompt

```
Question: {question_title}
Resolves: {resolution_date}
{outcome_instructions}

Resolution criteria / context:
{resolution_criteria}

Forecast date: {forecast_date}

# Research evidence

{n_articles} articles retrieved before the forecast date, in the order they were found. Articles marked
↪  with full text were read in full; the others show a short summary.

[1] {article_title}
    source: {domain} | published: {publication_date}
    summary: {snippet}
    full text:
    {article_text}

{remaining_article_blocks}
```

Outcome instructions: binary.

```
Outcomes: YES / NO. Forecast a probability for each label; the two probabilities must sum to 1.
```

Outcome instructions: multiple-choice.

```
Outcomes ({K}, exactly one resolves YES):
  - '{option_label_1}': {option_question_1}
  {remaining_option_lines}
Forecast a probability for each label; the K probabilities must sum to 1.
```

Figure 9: Frozen-evidence user-message template, paired with Figure 8. Braced fields are filled per task; the appropriate outcome instructions are inserted once. Article blocks preserve acquisition order, with full text included only for articles the agent read. The same template is used for H_{Base} and H_{AlphaBet}.

Gain Analysis: Question-Only User Prompt

```
Question: {question_title}
Resolves: {resolution_date}
{outcome_instructions}

Resolution criteria / context:
{resolution_criteria}

Forecast date: {forecast_date}

# Research evidence
No research evidence is available for this question. Forecast from the question, the resolution criteria
↪  and your base rates alone.
```

Figure 10: Question-only user-message template. It uses the same reader system prompt and question fields as the evidence condition, including its outcome instructions, but supplies no retrieved articles.