# FORECAST-DOJO: REPLAYABLE ENVIRONMENTS FOR BENCHMARKING AND TRAINING LLM FORECASTING AGENTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

We introduce `Forecast-Dojo`, a replayable environment for benchmarking and training LLM forecasting agents. It combines resolved prediction-market questions with dated news, allowing agents to research an event and revisit their predictions at successive historical dates. The same tasks and tools support repeated evaluation, collection of training interactions, and feedback from recorded outcomes without waiting for new events to resolve. `Forecast-Dojo` contains 1,568 Polymarket events, split by time into training and evaluation periods, and 18.8M dated news articles. In an evaluation of 12 models, research tools lower Brier score for all 12. Forecasts also improve as events unfold, with the largest gains at steps where more newly dated evidence is recorded. Every model still trails historical market forecasts in both Brier score and accuracy. A belief notebook carried between dates lowers research cost but does not consistently improve forecast quality. A supervised fine-tuning demonstration uses the collected trajectories to improve a student's accuracy from 34.8% to 42.8% on later, held-out events.

## 1 INTRODUCTION

Large language model (LLM) agents are increasingly used to forecast real-world events by actively searching for evidence, reasoning under uncertainty, and producing probabilistic predictions (Halawi et al., 2024; Zeng et al., 2025; Seed, 2026; Karger et al., 2025). Forecasting is inherently time-dependent: the evidence available to a forecaster changes as new information arrives. The same question can therefore pose a different prediction problem at different times, requiring the forecaster to update its belief as new evidence emerges.

Existing forecasting benchmarks capture only part of this process (Table 1). Live benchmarks (Karger et al., 2025; Zeng et al., 2025; Zhao et al., 2026; Zhang et al., 2026) pose unresolved questions, so the forecasting problem evolves naturally with the world, but they run on wall-clock time. A past forecasting condition cannot be recreated for a model released later, and outcomes arrive only at resolution, which slows evaluation and makes training impractical. Historical benchmarks (Jin et al., 2021; Halawi et al., 2024; Chandak et al., 2025) reconstruct past information cutoffs, making resolved events immediately scorable and reusable. However, they typically evaluate each forecast at a single historical point rather than revisiting the same question across multiple points in time. The ideal setting combines the two: the forecasting problem evolves as new evidence becomes available, yet each past step can be replayed with the same task and information cutoff. Models can then be compared under identical conditions and scored immediately, and their forecasting trajectories become usable for training.

We introduce `Forecast-Dojo`, a replayable environment that reconstructs resolved real-world events as sequences of historical forecast steps (Figure 1). At each step, an LLM agent can search a temporally restricted information corpus, inspect full documents, and use computational tools to produce a probabilistic forecast using documents whose assigned dates satisfy that cutoff. Each step can be reset and rerun across models or repeated trials, while steps from the same event can be traversed sequentially with persistent agent memory. The recorded outcome is retained for immediate scoring and is not supplied in the forecasting prompt. This common interaction interface supports



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.

Table 1: **Summary of existing forecasting benchmarks.** The criteria target the capabilities used to construct and study Forecast-Dojo. *Aligned evaluation* indicates that the same forecasting task is evaluated at a pre-specified sequence of forecast dates or evidence states shared across models. *Market belief* indicates that historical prediction-market prices are available as a reference. ✓: provided; ✗: not documented in the cited work; ✓✗: system- or agent-dependent.

| Benchmark | Dated replay | Agent research | Aligned evaluation | Agent memory | Market belief | Train/eval split |
|---|---|---|---|---|---|---|
| ForecastQA (Jin et al., 2021) | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ |
| Autocast (Zou et al., 2022) | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ |
| MIRAI (Ye et al., 2024) | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ |
| ForecastBench (Karger et al., 2025) | ✗ | ✓✗ | ✗ | ✗ | ✗ | ✗ |
| FutureX (Zeng et al., 2025) | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ |
| Prophet Arena (Yang et al., 2025b) | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ |
| EvolveCast (Yuan et al., 2025) | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ |
| BTF-2 (Liptay et al., 2026) | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ |
| FutureSim (Goel et al., 2026) | ✓ | ✓ | ✓✗ | ✓ | ✗ | ✗ |
| **Forecast-Dojo** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

both benchmarking and learning: held-out events evaluate agents under controlled conditions, while training events generate forecasting trajectories and outcome feedback for learning.

Empirically, we evaluate 12 models on 230 held-out events without tools, with research tools, and with research tools plus a belief notebook carried between dates. Research tools lower Brier for all 12 models, and memory-free Brier falls from 0.670 in the first third of an event to 0.606 in the last third, while forecasts without tools stay flat. The improvements concentrate at steps with more new evidence. Every model still trails the historical market forecasts in Brier and accuracy, with the best Brier at 0.546 against 0.498 for the market. Across the five proprietary models, the belief notebook reduces estimated API cost by a median of 24%, but it lowers Brier for only 6 of the 12 models, and none of these gains is significant (Section 5.2). Finally, a supervised fine-tuning study shows that an agent trained on `Forecast-Dojo` trajectories outperforms its base model on later, held-out events (Section 5.5).

Our contributions are:

- **A replayable forecasting environment.** `Forecast-Dojo` replays each resolved event as a fixed sequence of dated forecast steps with an optional belief notebook, and scores every step against the realized outcome for evaluation or training.
- **A dataset of resolved events and dated news.** It covers 1,568 Polymarket events with 6,122 forecast steps, split by time into 1,338 training and 230 evaluation events, with evidence from 18.8M dated CC-News articles.
- **A benchmark study beyond overall rankings.** We compare 12 models with historical market forecasts, examine how forecasts change as events unfold and how memory affects quality and cost, and show how the collected interactions can be used to train an agent.

## 2 Related Work

**Forecasting with dated evidence.** ForecastQA restricts news by time, while Autocast pairs forecasting questions with dated articles and historical human forecasts (Jin et al., 2021; Zou et al., 2022). ForecastBench and FutureX collect predictions on unresolved events (Karger et al., 2025; Zeng et al., 2025). Related settings evaluate repeated forecasts, macroeconomic nowcasts, and simulated market decisions (Yang et al., 2025b; Zhao et al., 2026; Cheng et al., 2026). `Forecast-Dojo` uses resolved events to support repeatable research and immediate outcome-based feedback, while keeping market probabilities outside the forecasting prompt.

**Interactive research and forecast revision.** Bench to the Future uses frozen research corpora, and BTF-2 records traces to distinguish information gathering from judgment (Wildman et al., 2025; Liptay et al., 2026). MIRAI provides code-based access to events and news, while WorldReasoner

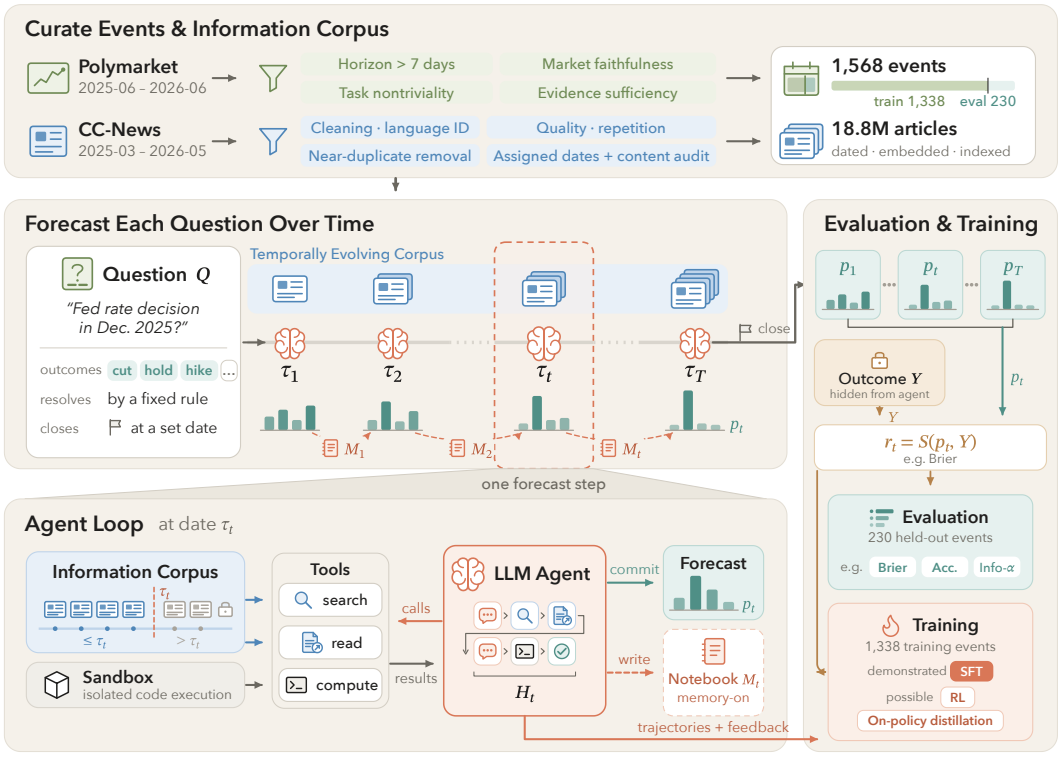


Figure 1: **Overview of Forecast-Dojo.** Top: resolved Polymarket events are filtered and split by time into training and evaluation events, and CC-News articles are cleaned, dated, and indexed. Middle left: each question is forecast at a fixed sequence of dates as the visible corpus grows, with an optional belief notebook $M_t$ passed between steps. Bottom left: within one step, the agent searches and reads articles dated on or before $\tau_t$, runs code in a sandbox, and commits a forecast $p_t$. Right: the realized outcome $Y$ stays hidden from the agent. The evaluator scores each forecast against it, using held-out events for evaluation and training events for learning.

evaluates outcome, evidence, and reasoning quality (Ye et al., 2024; Chi et al., 2026). FutureSim studies long-horizon adaptation in a replayed world, where agents jointly decide how to research, maintain memory, revise forecasts, and progress through time (Goel et al., 2026). `Forecast-Dojo` instead fixes time progression in the experimental design: each event is replayed at the same historical checkpoints for all agents, which yields matched longitudinal trajectories for controlled comparison and reusable interactions with outcome feedback for learning. This complements work on evidence-driven revision, probability coherence, and iterative research workflows (Yuan et al., 2025; Paleka et al., 2025; Murphy, 2026).

**Environments for training agents.** MLE-Dojo provides executable machine learning engineering tasks and feedback for evaluation and training (Qiang et al., 2025). `Forecast-Dojo` follows this environment-centered approach, specifying the forecasting interaction and its feedback while leaving learning algorithms separate. Prior forecasting work trains on question collections or selected model-generated forecasts (Zou et al., 2022; Halawi et al., 2024), and recent methods use reinforcement learning, outcome-based fine-tuning, and news-derived questions (Levy, 2026; Jeen et al., 2026; Chandak et al., 2025). Our contribution is the shared task and tool interface for collecting research interactions and evaluating agents on separate events; the SFT experiment demonstrates one use of that interface.

## 3 Forecasting as an Interactive Task

We formulate forecasting as an interactive task over a sequence of forecast steps for the same event. At each step, the agent researches evidence available up to the current date and produces a probabilistic forecast. Across steps, the evidence boundary advances and explicit memory may persist.

**Events, forecast steps, and episodes.** A question $Q$ specifies an event, its resolution criteria, and a finite set of mutually exclusive outcomes $\mathcal{Y} = \{1, \ldots, K\}$. Let $Y \in \mathcal{Y}$ denote the realized outcome. At ordered dates $\tau_1 < \cdots < \tau_T$ before the event closes, where $T$ varies across events, the agent performs a *forecast step*: it researches the event and reports a probability distribution $p_{Q,t} \in \Delta^{K-1}$. The ordered forecast steps for the same question form a *forecast episode*. All steps concern the same eventual outcome, but they differ in the historical evidence available at the forecast date.

**Within-step interaction.** Let $\mathcal{I}$ denote a fixed information corpus. At date $\tau_t$, the agent is exposed only to

$$\mathcal{I}_{\leq \tau_t} = \{d \in \mathcal{I} : \mathrm{date}(d) \leq \tau_t\}, \qquad \mathcal{I}_{\leq \tau_t} \subseteq \mathcal{I}_{\leq \tau_{t+1}}, \tag{1}$$

where $\mathrm{date}(d)$ denotes the timestamp assigned to document $d$. Within a forecast step, the agent may issue search queries, inspect retrieved articles, and use computation before submitting its forecast (see Section 4.4). Let $H_t$ denote the within-step interaction history, including the agent's research actions and the resulting tool observations. The agent produces $p_{Q,t}$ conditioned on $H_t$. Because agents choose their own queries and which documents to inspect, different rollouts at the same date from an agent can follow different research paths even under the same dated evidence boundary.

**Progression across forecast steps.** After each report, the environment advances to the next scheduled date and starts a fresh interaction. We define two modes for carrying information across forecast steps: *memory-free* and *memory-on*. In memory-free forecasting, each step starts without information produced at previous steps. In memory-on forecasting, the agent produces a belief notebook $M_t$ that summarizes its current assessment, supporting evidence, and open questions. At the next step, $M_t$ is provided alongside the question and new forecast date, allowing the agent to update its forecast from its prior assessment as new evidence becomes available.

**Outcome feedback.** Each forecast step ends with a probability report $p_{Q,t}$. Once the realized outcome $Y$ is available, the environment assigns feedback $r_{Q,t} = S(p_{Q,t}, Y)$, where $S$ is an outcome-based scoring rule. The outcome and feedback are not part of the agent's forecasting context. This separates the interaction that produces a forecast from the feedback assigned to it, allowing the same task interface to support different downstream evaluation or learning procedures.

# 4 FORECAST-DOJO

Section 3 defines the forecasting task abstractly. We now describe how Forecast-Dojo instantiates its events, information corpus, forecast episodes, runtime, and evaluation records.

## 4.1 FORECASTING EVENTS

We construct our candidate pool from resolved Polymarket[1] binary and mutually exclusive multi-option events whose full lifetimes fall between June 2025 and June 2026. We construct a daily market-price reference from UTC-day mean YES prices, which we call the *market belief*. For binary events, the YES price determines the probability of YES and its complement; for multi-option events, we normalize the option-level YES prices to obtain a distribution over the mutually exclusive outcomes.

Within this pool, we select events along four dimensions. (1) *Forecasting horizon*: the tradeable lifetime must exceed 7 days, so that each question supports multiple forecast steps. (2) *Market activity*: most days must have sufficient trading activity, which reduces sparse or stale prices and favors questions with sustained attention. (3) *Task nontriviality*: we remove events whose market already assigns near-certain probability to the realized outcome, as well as events whose market history is nearly flat and directionless. (4) *Evidence sufficiency*: following Joren et al. (2025), an LLM judge must rate the evidence retrieved two days before the recorded close as SUFFICIENT, followed by a content audit. This criterion does not establish sufficient evidence at earlier forecast dates. Several filters use the realized outcome, so the benchmark is not a random sample of Polymarket events (Appendices A.2 and C.4).

[1] https://polymarket.us/

### 4.2 INFORMATION CORPUS

**Corpus construction.** `Forecast-Dojo` requires broad historical evidence whose availability can be reconstructed at each forecast date. We build the corpus from CC-News (Nagel, 2016), which provides large-scale news coverage together with crawl timestamps that support historical reconstruction. We process CC-News archives from March 2025 through May 2026 using a quality pipeline adapted from FineWeb (Penedo et al., 2024), including text cleaning, language identification, repetition and document-quality filtering, and near-duplicate removal. We suppress duplicate URLs, identical titles within a seven-day window, and near-verbatim body matches to reduce repeated coverage. After filtering, the corpus contains approximately 18.8M articles, which we embed with Qwen3-Embedding-8B (Zhang et al., 2025) and index with FAISS (Douze et al., 2026).

**Temporal integrity.** We assign each article a UTC calendar day $t(d)$ using the closest-to-crawl rule. Among valid structured publication and modification timestamps, we choose the one closest to the crawl time (breaking ties toward the later timestamp), cap it at the crawl time, and use its UTC day. If neither structured timestamp is valid, we use the crawl day. These timestamps come from standard structured metadata fields (Guha et al., 2016; Schema.org; Google Search Central). Search and article reads admit $t(d) \leq \tau_t$, so articles assigned to the forecast day itself are included. This is a day-level metadata boundary; it does not certify intraday availability or reconstruct earlier versions of an edited page. Event construction additionally screens for content-level outcome leakage. Appendix A.1 gives the full dating rule and explains how repeated crawls of the same page are handled.

### 4.3 TEMPORAL TASK CONSTRUCTION

**Forecast-date selection.** A forecast step for every day of an event's lifetime would let long-lived events dominate the samples. We therefore use a sublinear schedule: after a two-day buffer before the recorded close, an event with $n$ candidate days receives $T = \mathrm{clamp}\,(\mathrm{round}(\sqrt{n}), 3, 10)$ forecast steps. We partition the event history into $T$ temporal bins and select one date from each, favoring (1) *market-belief movement*, larger changes in the market belief as a proxy for changes in expectations, and (2) *news volume*, more articles in the corpus overall, a global signal rather than an event-specific measure of relevance. After leakage filtering, events in both splits must retain at least three forecast dates (Appendix A.3).

**Temporal train–evaluation split.** We split complete events, rather than individual forecast steps, into non-overlapping temporal windows. Training events must start and close within `[2025-06, 2026-03)` and evaluation events within `[2026-03, 2026-06)`, and events crossing either boundary are excluded, so no event appears in both splits. The training split has 1,338 events (602 binary) with 5,325 forecast steps, and the evaluation split has 230 events (40 binary) with 797 steps (Table 7).

### 4.4 AGENT INTERACTION

**Tool interface.** At each forecast step, the agent can call three tools. SEARCH, via `search(query, top_k)`, returns identifiers, titles, dates, retrieval scores, and snippets for the top-$k$ articles in $\mathcal{I}_{\leq \tau_t}$. READ, via `scrape(article_id)`, returns the full text of a retrieved article. COMPUTE, via `python(code)`, runs model-generated code for numerical analysis, base-rate estimation, or simulation. Python runs in an isolated sandbox with networking disabled and no access to benchmark data such as the task files, corpus, and index (Appendix A.5). The agent may interleave reasoning with repeated tool calls, and each tool output joins the within-step history $H_t$.

**Memory transfer.** Each forecast step is a fresh model interaction with access to $\mathcal{I}_{\leq \tau_t}$. In memory-free mode, nothing from earlier steps is carried forward. In memory-on mode, the notebook from the previous step is inserted into the next prompt, while previous turns, reasoning traces, and tool observations are discarded, so the notebook is the only transferred information (format in Figure 7).

Table 2: **Main Results.** Values average recorded forecasts; smaller ± values are SDs across four rollout means. Event-bootstrap 95% CIs are reported in Appendix C.1. Unusable forecasts use uniform fallback; accuracy uses fractional ties and is reported in percent. Info-$\alpha$ is computed where market probabilities are available. **Bold** and underlined denote the best and second-best models. Superscripts †, ‡, and § mark configurations with $> 5\%$ unusable forecasts. Uniform and market rows are references; the market row covers 790 event–dates with complete market vectors.

| | No tools | | | Tools, memory-free | | | Tools, memory-on | | |
|---|---|---|---|---|---|---|---|---|---|
| Model | Brier↓ | Acc.↑ | Info-$\alpha$ ↑ | Brier↓ | Acc.↑ | Info-$\alpha$ ↑ | Brier↓ | Acc.↑ | Info-$\alpha$ ↑ |
| *Proprietary* | | | | | | | | | |
| GPT-5.6 Sol | **0.650** ±0.004 | **47.03** ±0.5 | **−0.422** ±0.006 | **0.554** ±0.008 | 56.36 ±0.9 | **−0.138** ±0.023 | **0.546** ±0.004 | 57.59 ±1.3 | **−0.104** ±0.017 |
| GPT-5.5 | 0.698 ±0.001 | 43.85 ±0.6 | −0.603 ±0.008 | 0.564 ±0.002 | **57.69** ±0.7 | −0.186 ±0.010 | 0.571 ±0.005 | **57.68** ±1.0 | −0.198 ±0.013 |
| GPT-5.4 | 0.708 ±0.002 | 41.34 ±0.8 | −0.633 ±0.015 | 0.586 ±0.002 | 53.12 ±0.4 | −0.237 ±0.006 | 0.583 ±0.005 | 52.87 ±1.0 | −0.222 ±0.021 |
| Claude Opus 4.8 | 0.725 ±0.002 | 40.30 ±0.8 | −0.643 ±0.004 | 0.589 ±0.006 | 52.63 ±0.8 | −0.254 ±0.023 | 0.605 ±0.008 | 50.94 ±1.1 | −0.300 ±0.019 |
| Claude Opus 4.6 | 0.762 ±0.002 | 35.16 ±0.5 | −0.788 ±0.007 | 0.603 ±0.003 | 52.35 ±0.9 | −0.282 ±0.008 | 0.598 ±0.002 | 53.32 ±0.5 | −0.263 ±0.006 |
| *Open-weight* | | | | | | | | | |
| GLM-5 | 0.731 ±0.006 | 38.54 ±0.6 | −0.686 ±0.016 | 0.625 ±0.004 | 50.02 ±1.0 | −0.372 ±0.024 | 0.632 ±0.002 | 49.96 ±0.4 | −0.398 ±0.017 |
| Qwen3.5-397B | 0.759 ±0.005 | 36.54 ±0.7 | −0.796 ±0.021 | 0.639 ±0.004 | 48.49 ±1.2 | −0.422 ±0.007 | 0.643 ±0.010 | 48.90 ±0.9 | −0.441 ±0.027 |
| Kimi K2.5 | 0.773 ±0.002 | 36.30 ±0.8 | −0.837 ±0.011 | 0.637 ±0.010 | 49.37 ±2.2 | −0.420 ±0.024 | 0.658 ±0.009 | 48.07 ±1.7 | −0.439 ±0.005 |
| MiniMax M2.5 | 0.758 ±0.005 | 35.68 ±0.9 | −0.797 ±0.011 | 0.654‡ ±0.012 | 46.87 ±0.7 | −0.461 ±0.031 | 0.640 ±0.007 | 47.19 ±1.6 | −0.413 ±0.022 |
| DeepSeek-V3.2 | 0.805 ±0.004 | 34.34 ±1.0 | −0.991 ±0.017 | 0.655 ±0.012 | 47.73 ±1.9 | −0.464 ±0.042 | 0.666 ±0.019 | 47.65 ±1.5 | −0.508 ±0.064 |
| gpt-oss-120b | 0.896 ±0.004 | 32.44 ±0.5 | −1.378 ±0.023 | 0.699‡ ±0.010 | 41.95 ±0.6 | −0.594 ±0.020 | 0.696§ ±0.012 | 43.03 ±0.7 | −0.584 ±0.052 |
| Nemotron 3 Super | 0.882† ±0.006 | 30.84 ±0.5 | −1.516 ±0.021 | 0.695‡ ±0.011 | 43.49 ±1.2 | −0.743 ±0.038 | 0.688§ ±0.019 | 44.89 ±2.7 | −0.720 ±0.031 |
| Uniform reference | 0.771 | 22.9 | — | 0.771 | 22.9 | — | 0.771 | 22.9 | — |
| Market reference | 0.498 | 64.5 | — | 0.498 | 64.5 | — | 0.498 | 64.5 | — |

### 4.5 Evaluation and Learning

We evaluate forecasts primarily with the multiclass Brier score (Glenn et al., 1950), and additionally report top-label accuracy, dataset-level calibration error (Guo et al., 2017), and market-relative measures such as Information-$\alpha$. Each episode supports trajectory-level analyses of how beliefs evolve (Appendix B.3). With trajectory logging, each step also retains $H_t$ and, in memory-on mode, the notebook, to which outcome-based feedback can be attached. Evaluation events use these records for benchmarking, and training events supply trajectories and feedback for learning, which we demonstrate with supervised fine-tuning in Section 5.5.

## 5 Experiments

### 5.1 Experimental Setup

We evaluate 12 proprietary and open-weight models on 230 held-out events (797 event–date pairs), with four rollouts per pair. Every model's reported knowledge or training-data cutoff, or otherwise its checkpoint release date, predates the evaluation period (Table 11), and a no-tools diagnostic finds no evidence of widespread outcome recall (Appendix B.1). We compare three settings on the same scheduled tasks. The *no-tools* setting (Figure 5) forecasts without any tools. The *memory-free* setting (Figure 6) uses date-restricted search, article reads, and Python from a fresh context at each date. The *memory-on* setting also passes each rollout's previous belief notebook to the next date, without earlier conversations or raw tool outputs. Configurations and tool budgets are given in Appendix B.2.

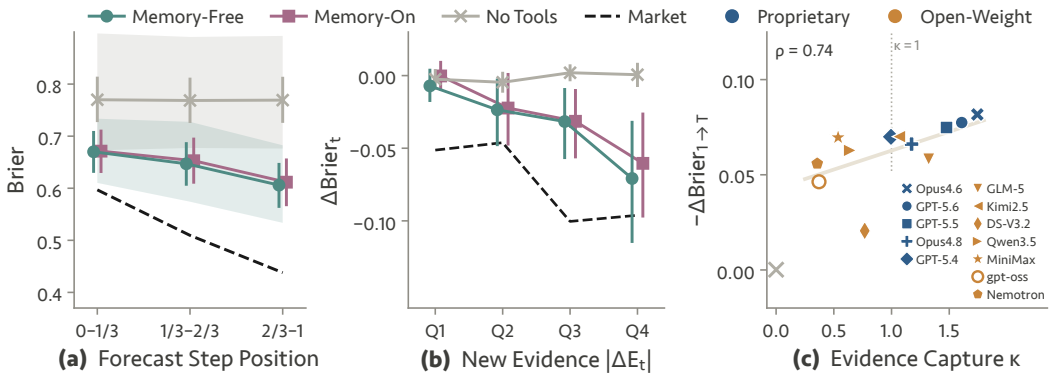


Figure 2: **Forecast improvements track newly recorded evidence.** (a) Brier across early, middle, and late thirds of each event trajectory, with equal event weighting; shading shows the range of model means. (b) Consecutive-step Brier change by new-evidence quartile, using leave-one-model-out evidence indices. (c) Evidence capture $\kappa$ versus first-to-last Brier improvement with memory; correlation is across 12 models, with the no-tools setting shown as a reference. Scores use the same pairing and uniform fallback as Table 2; (a,b) show 95% event-bootstrap CIs. See Appendix B.7 for definitions and sensitivity analyses.

We report multiclass Brier score, top-label accuracy, and Information-$\alpha$, which compares the agent's and the market's log scores on the realized outcome, with positive values favoring the agent. Uniform forecasts and historical market probabilities serve as references. The market may use information outside the news archive, and its probabilities are never shown to the agents. An unusable recorded forecast is replaced by a uniform distribution over its $K$ offered outcomes, giving Brier $1-1/K$ and fractional-tie accuracy $1/K$. Longitudinal averages weight events equally, and all 95% CIs and paired comparisons resample events (Appendix B.3).

### 5.2 Overall Forecasting Performance

**Research tools improve forecasting across model families.** Relative to no tools, memory-free research lowers Brier and improves accuracy for all 12 models, and every paired 95% CI excludes zero (Tables 2 and 20). GPT-5.5, for example, improves from 0.698 to 0.564 in Brier and from 43.85% to 57.69% in accuracy, and the open-weight DeepSeek-V3.2 improves from 0.805 to 0.655 and from 34.34% to 47.73%.

**Proprietary models achieve lower Brier scores with tools.** In both tool-enabled settings, all five proprietary models have lower mean Brier than every open-weight model. The proprietary group is 0.078 lower on average without memory (95% CI $[0.056, 0.103]$), although the closest pair, Claude Opus 4.6 and GLM-5, is not separated by its paired CI (Table 22). GPT-5.6 Sol has the lowest Brier without and with memory (0.554 and 0.546), and its lead over GPT-5.5 is significant only with memory. GLM-5 is the strongest open-weight model (0.625 and 0.632).

**Accuracy and Brier can disagree through calibration.** GPT-5.5 has the highest memory-free accuracy (57.69%) but is underconfident. Its top-ranked outcome receives 0.50 probability on average while being correct on 58% of usable forecasts, the largest top-label calibration error among the 12 models (Appendix C.3). GPT-5.6 Sol is slightly less accurate (56.36%; the paired accuracy-difference CI includes zero), partly because 4.49% of its memory-free runs stop before a final answer, yet it has lower Brier (0.554 versus 0.564) and higher Information-$\alpha$ ($-0.138$ versus $-0.186$). With uniform fallback, all 12 models beat the uniform reference in Brier under both tool-enabled settings. This convention does not imply successful output, since 19.45% of gpt-oss-120b's memory-free reports are unusable (Appendix B.4).

**Memory yields model-dependent changes in forecast quality.** Adding a belief notebook lowers mean Brier for six models and raises it for the other six. None of the six improvements has a 95%

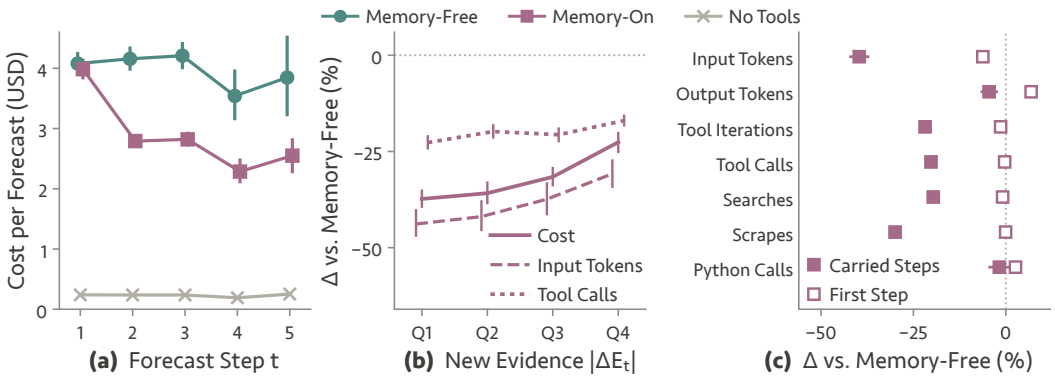


Figure 3: **Memory lowers research cost, mostly after the first forecast.** (a) Mean API cost per forecast step across five proprietary models; later steps cover fewer events (67/230 at step 4; 25/230 at step 5). (b) Memory-on vs. memory-free percentage changes in cost, input tokens, and tool calls for later steps ($t \geq 2$), grouped by new-evidence quartile from Figure 2(b). (c) The same changes at the first versus later steps; notebooks are only available after the first step. Negative values denote reductions. Tokens and tool calls use all 12 models; intervals are 95% event-bootstrap CIs. Per-model results are in Table 15.

CI that excludes zero, whereas memory significantly raises Brier for Claude Opus 4.8 ($+0.016$, $[0.006, 0.025]$) and Kimi K2.5 ($+0.021$, $[0.002, 0.041]$) on all recorded pairs (Table 20). Much of this split comes from output reliability. For GPT-5.6 Sol, the share of forecasts without a usable answer falls from 4.49% to 0.53% with memory, yet on forecasts usable in both settings the paired Brier difference is only $+0.0010$. We therefore separate memory's effects on forecast quality and output reliability from its effect on research cost (Section 5.4).

**The market reference remains ahead across scoring rules.** The market achieves Brier 0.498 and accuracy 64.55%, against the best model scores of 0.546 and 57.69%. Paired on the same event–dates, the Brier gap of the best configuration is 0.050 (95% CI $[0.019, 0.082]$), and the gap's CI lies above zero for every configuration. Information-$\alpha$ is negative for every model and setting, with every 95% CI below zero (Table 21). Because the market may incorporate information outside the agents' archive, it serves as an external reference rather than an information-matched baseline.

### 5.3 Forecasting over Time

**Forecasts improve over time with research tools.** Figure 2(a) compares forecasts at the early, middle, and late stages of each event. Mean Brier decreases from 0.670 to 0.606 with memory-free research and from 0.671 to 0.612 with memory-on research. In contrast, the no-tools baseline remains nearly unchanged near 0.769. All 12 models improve from the first to the last stage with memory-free research, each with a 95% CI below zero (Table 26). The market improves more sharply, from 0.597 to 0.438. The improvement is concentrated in events on which the market also moved toward the outcome (Appendix C.4). These results show that agents benefit from access to dated information as events unfold, even without retaining their previous research.

**Larger improvements coincide with more new evidence.** We estimate new evidence between consecutive dates from dated entries in belief notebooks. For each model, the *new-evidence index* $|\Delta E_t|$ counts evidence that the other models recorded within the interval (Appendix B.6). For memory-free agents, the mean step-to-step Brier change is $-0.007$ in the lowest index quartile and $-0.071$ in the highest (Figure 2(b)). Memory-on agents show a similar pattern, while no-tools forecasts change little in every quartile. Models that record more newly dated evidence also improve more over an episode (Spearman $\rho = 0.74$, Figure 2(c)), although this association holds between models rather than within them (Appendix B.7).

Table 3: **Supervised fine-tuning on Forecast-Dojo trajectories.** Paired SFT−base differences ($\Delta$) are computed before rounding; 95% CIs use event-level bootstrap. All scores use uniform fallback for unusable forecasts; the uniform row is shown for reference. Binary and multi-choice paired differences are $-0.165$ $[-0.290, -0.050]$ and $-0.178$ $[-0.216, -0.141]$, respectively.

| Model | Brier↓ | | | Log loss↓ | Acc.↑ | Usable | Tool calls |
|---|---|---|---|---|---|---|---|
| | All | Binary | Multi-choice | | (%) | (%) | / forecast |
| Student (base) | 0.924 | 0.704 | 0.975 | 2.74 | 34.8 | 100.0 | 1.4 |
| **+ Dojo SFT** | **0.749** | **0.539** | **0.797** | **1.89** | **42.8** | 99.5 | 3.7 |
| Uniform | 0.771 | 0.500 | 0.832 | 1.74 | 22.9 | – | – |
| Paired $\Delta$ | $-0.176$ | $-0.165$ | $-0.178$ | $-0.86$ | $+8.1$ | $-0.5$ | $+2.3$ |
| 95% CI | $[-0.214, -0.139]$ | | | $[-1.03, -0.70]$ | $[+5.1, +11.2]$ | | |

### 5.4 Cost Analysis

**Memory reduces the cost of repeated forecasting.** For the five proprietary models, memory-on execution costs less per forecast than memory-free execution in every case, from 8% less for Claude Opus 4.6 to 33% less for GPT-5.5 (median 24%, all 95% CIs excluding zero, Table 15). GPT-5.5, for example, costs $3.90 instead of $5.85 per forecast, while its paired Brier difference of $+0.007$ has a CI that includes zero. Both settings cost about the same at the first forecast. The saving appears from the second forecast onward, is largest when little new evidence has appeared, and comes with fewer input tokens, tool iterations, and research calls (Figure 3, Appendix B.5). This pattern is consistent with agents reusing earlier findings, and lower research cost is memory's clearest benefit in the evaluated protocols.

### 5.5 Training Proof of Concept

We next test whether `Forecast-Dojo` interactions can support agent training. Qwen3-235B-A22B-Thinking-2507 (Yang et al., 2025a) generates four memory-free trajectories for each of 3,965 event–date tasks from 1,023 training events (August 2025 to February 2026), and we keep one demonstration per task, chosen by a weighted score of outcome-based Brier, accuracy, and tool use (Appendix C.5). The environment's realized-outcome feedback is thus used offline to select demonstrations. We fine-tune Qwen3-30B-A3B-Thinking-2507 (Yang et al., 2025a) for three epochs with next-token cross-entropy on assistant tokens, including reasoning and tool calls, and evaluate the final checkpoint (Appendix C.5).

On the 230 held-out events, the fine-tuned student outperforms the base on all forecasting metrics (Table 3). Brier falls from 0.924 to 0.749 (paired $-0.176$, 95% CI $[-0.214, -0.139]$), with gains on both binary and multi-choice forecasts. Log loss falls from 2.74 to 1.89, and accuracy rises from 34.8% to 42.8% ($+8.1$ points, $[+5.1, +11.2]$). Tool use increases from 1.4 to 3.7 calls per forecast, and 99.5% of reports remain usable. An oracle uniform-shrinkage diagnostic reaches Brier 0.752 for the base and 0.705 for SFT, so much of the raw Brier gain can be matched by rescaling the base probabilities, while the accuracy gain is unaffected because shrinkage preserves outcome rankings (Appendix C.5). Together, these results provide a proof of concept that `Forecast-Dojo` trajectories can support training agents with improved held-out forecasting performance.

## 6 Conclusion

`Forecast-Dojo` replays resolved prediction-market events as fixed sequences of historical forecast states, so agents can be compared on the same scheduled tasks and dated corpus boundary throughout an event's evolution. Research tools improve Brier and accuracy for all 12 models, with larger step-wise gains when peer models record more newly dated evidence. Yet all agents remain behind historical market forecasts, and persistent memory reduces research cost more consistently than it improves forecast quality. The same trajectories also support training, with the demonstrated student improving accuracy from 34.8% to 42.8% on later, held-out events. Appendix D discusses limitations.

### AI USE STATEMENT

We used generative AI tools in two ways in this work. First, large language models are part of the benchmark pipeline. They generate retrieval queries and judge evidence sufficiency when we filter events (Appendix A.2), and a teacher model generates the research trajectories used as fine-tuning data (Section 5.5). The models we evaluate are the objects of study rather than tools. Second, we used AI assistants for writing. They polished parts of the paper, including the abstract, parts of the introduction and conclusion, and several figure captions. They also edited the text for readability, helped survey related work, checked cross-references for consistency, and suggested fixes to the reference list. We have not used generative AI tools to propose hypotheses, develop theoretical frameworks, formulate or prove mathematical claims, design the benchmark or experiments, or implement methods, and translation and qualitative data analysis are not applicable to this work. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

Forecast-Dojo evaluates retrospective prediction-market questions using archived news. Event selection favors active markets with sufficient retrieved coverage, so results need not represent less-covered events or broader public interests. Forecasting scores on these selected historical tasks do not establish reliability for live decisions. Any release of news text or research traces should respect source licenses and account for personal information that may appear in retrieved articles. Forecasting agents could also be used to inform trading on prediction markets. `Forecast-Dojo` evaluates historical, resolved events and does not place or simulate trades.

### REPRODUCIBILITY STATEMENT

Section 4 and Appendix A describe how events, the dated corpus, and forecast schedules are constructed. Appendices B.2 and B.3 give model settings, scoring rules, and the uncertainty procedure, Appendix B.4 covers failures and aggregation, and Appendix C.5 describes the training use case. All prompts are reproduced in Appendix E. Our code is available at `https://anonymous.4open.science/r/Forecast-Dojo-EF88`. Our data is available at `https://huggingface.co/datasets/foye107/Forecast-Dojo`.

## REFERENCES

Anthropic. Claude opus 4.6 system card. `https://www.anthropic.com/system-cards`, February 2026a. Claude Opus 4.6.

Anthropic. Introducing claude opus 4.8. `https://www.anthropic.com/news/claude-opus-4-8`, May 2026b. Accessed 2026-09-22.

Nikhil Chandak, Shashwat Goel, Ameya Prabhu, Moritz Hardt, and Jonas Geiping. Scaling open-ended reasoning to predict the future. *arXiv preprint arXiv:2512.25070*, 2025.

Pu Cheng, Juncheng Liu, and Yunshen Long. Polybench: Benchmarking llm forecasting and trading capabilities on live prediction market data. *arXiv preprint arXiv:2604.14199*, 2026.

Yizhou Chi, Eric Chamoun, Zifeng Ding, and Andreas Vlachos. Worldreasoner: Evaluating whether language model agents forecast events with valid reasoning. *arXiv preprint arXiv:2606.11816*, 2026.

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`.

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

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. *arXiv preprint arXiv:2605.15188*, 2026.

Google Search Central. Article structured data. https://developers.google.com/search/docs/appearance/structured-data/article. Accessed 2026.

Ramanathan V. Guha, Dan Brickley, and Steve Macbeth. Schema.org: evolution of structured data on the web. *Commun. ACM*, 59(2):44–51, 2016. doi: 10.1145/2844544. URL https://doi.org/10.1145/2844544.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017. URL http://proceedings.mlr.press/v70/guo17a.html.

Danny Halawi, Fred Zhang, Yueh-Han Chen, and Jacob Steinhardt. Approaching human-level forecasting with language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/5a5acfd0876c940d81619c1dc60e7748-Abstract-Conference.html.

Scott Jeen, Matthew Aitchison, Maximilian Anthony Hugh Clark, Toby Shevlane, and Ben Day. Reaching the frontier of ai forecasting with reinforcement learning. In *Forecasting as a New Frontier of Intelligence*, 2026.

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 Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 4636–4650. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.ACL-LONG.357. URL https://doi.org/10.18653/v1/2021.acl-long.357.

Hailey Joren, Jianyi Zhang, Chun-Sung Ferng, Da-Cheng Juan, Ankur Taly, and Cyrus Rashtchian. Sufficient context: A new lens on retrieval augmented generation systems. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=Jjr2Odj8DJ.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hervé Jégou, and Tomás Mikolov. Fasttext.zip: Compressing text classification models. *CoRR*, abs/1612.03651, 2016. URL http://arxiv.org/abs/1612.03651.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomás Mikolov. Bag of tricks for efficient text classification. In Mirella Lapata, Phil Blunsom, and Alexander Koller (eds.), *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pp. 427–431. Association for Computational Linguistics, 2017. doi: 10.18653/V1/E17-2068. URL https://doi.org/10.18653/v1/e17-2068.

Ezra Karger, Houtan Bastani, Yueh-Han Chen, Zachary Jacobs, Danny Halawi, Fred Zhang, and Philip Tetlock. Forecastbench: A dynamic benchmark of AI forecasting capabilities. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=lfPkGWXLLf.

Amit Arnold Levy. Reinforcement learning for llm-based event forecasting. *arXiv preprint arXiv:2606.15917*, 2026.

Tom Liptay, Dan Schwarz, Rafael Poyiadzi, Jack Wildman, and Nikos I Bosse. Evaluating strategic reasoning in forecasting agents. *arXiv preprint arXiv:2604.26106*, 2026.

Kevin Murphy. Agentic forecasting using sequential bayesian updating of linguistic beliefs. *arXiv preprint arXiv:2604.18576*, 2026.

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

Daniel Paleka, Abhimanyu Pallavi Sudhir, Alejandro Alvarez, Vineeth Bhat, Adam Shen, Evan Wang, and Florian Tramèr. Consistency checks for language model forecasters. *arXiv preprint arXiv:2412.18544*, 2025.

Guilherme Penedo, Hynek Kydlícek, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin A. Raffel, Leandro von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/370df50ccfdf8bde18f8f9c2d9151bda-Abstract-Datasets_and_Benchmarks_Track.html.

Rushi Qiang, Yuchen Zhuang, Yinghao Li, Dingu Sagar V K, Rongzhi Zhang, Changhao Li, Ian Shu-Hei Wong, Sherry Yang, Percy Liang, Chao Zhang, and Bo Dai. Mle-dojo: Interactive environments for empowering llm agents in machine learning engineering. *arXiv preprint arXiv:2505.07782*, 2025.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, H. Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew J. Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher. *CoRR*, abs/2112.11446, 2021. URL https://arxiv.org/abs/2112.11446.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL https://jmlr.org/papers/v21/20-074.html.

Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, 2009. doi: 10.1561/1500000019. URL https://doi.org/10.1561/1500000019.

Schema.org. datePublished. https://schema.org/datePublished. Accessed 2026.

ByteDance Seed. Futurex-pro: Extending future prediction to high-value vertical domains. *CoRR*, abs/2601.12259, 2026. doi: 10.48550/ARXIV.2601.12259. URL https://doi.org/10.48550/arXiv.2601.12259.

Jack Wildman, Nikos I Bosse, Daniel Hnyk, Peter Mühlbacher, Finn Hambly, Jon Evans, Dan Schwarz, Lawrence Phillips, et al. Bench to the future: A pastcasting benchmark for forecasting agents. *arXiv preprint arXiv:2506.21558*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Qingchuan Yang, Simon Mahns, Sida Li, Anri Gu, Jibang Wu, and Haifeng Xu. Llm-as-a-prophet: Understanding predictive intelligence with prophet arena. *arXiv preprint arXiv:2510.17638*, 2025b.

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

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, Jinpeng Wang, Zaiyuan Wang, Yang Yang, Lingyue Yin, Mingren Yin, Zhenwei Zhu, Tianle Cai, 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 H. Blanchet, Xipeng Qiu, Mengdi Wang, and Wenhao Huang. Futurex: An advanced live benchmark for LLM agents in future prediction. *CoRR*, abs/2508.11987, 2025. doi: 10.48550/ARXIV.2508.11987. URL https://doi.org/10.48550/arXiv.2508.11987.

Jaden Zhang, Gardenia Liu, Oliver Johansson, Hileamlak Yitayew, Kamryn Ohly, and Grace Li. Prediction arena: Benchmarking AI models on real-world prediction markets. *CoRR*, abs/2604.07355, 2026. doi: 10.48550/ARXIV.2604.07355. URL https://doi.org/10.48550/arXiv.2604.07355.

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 (NeurIPS), Datasets and Benchmarks Track*, 2022.

# Appendix for Forecast-Dojo

**A Dataset and Benchmark Details** **14**

A.1 Information Corpus . . . . . 14

A.2 Event Collection and Filtering . . . . . 15

A.3 Forecast-Step Construction . . . . . 17

A.4 Evaluation-Set Composition . . . . . 17

A.5 Environment Details . . . . . 18

**B Experiment Details** **19**

B.1 Knowledge Cutoffs and Checkpoint Dates . . . . . 19

B.2 Models and Execution Protocols . . . . . 19

B.3 Evaluation Metrics . . . . . 20

B.4 Aggregation and Failures . . . . . 21

B.5 Inference Cost and Tool Usage . . . . . 21

B.6 Measuring Newly Recorded Evidence . . . . . 22

B.7 Evidence Capture and Forecast Improvement . . . . . 24

**C Additional Results** **26**

C.1 Statistical Uncertainty and Paired Comparisons . . . . . 26

C.2 Results by Event Type and Scoring Convention . . . . . 26

C.3 Calibration . . . . . 26

C.4 Longitudinal Robustness . . . . . 29

C.5 Training Use Case . . . . . 30

**D Limitations** **32**

**E Prompts** **32**

## A Dataset and Benchmark Details

We describe in detail how we construct our information corpus (§A.1), collect and filter forecasting events (§A.2), and select forecast dates for each task (§A.3).

### A.1 Information Corpus

**Corpus source.** We build the retrieval corpus from the monthly CC-News releases of Common Crawl (Nagel, 2016) from March 2025 through May 2026. For each news page, we extract the title, summary, body text, URL, crawl time, and structured publication metadata.

**Text and quality filtering.** Our filtering pipeline is summarized in Table 4. The first two stages normalize the extracted text and retain English-language documents using fastText (Joulin et al., 2017; 2016), matching the language of the benchmark questions. The remaining stages adapt quality filters from Gopher (Rae et al., 2021), C4 (Raffel et al., 2020), and FineWeb (Penedo et al., 2024).

| Stage | Deployed criterion |
|---|---|
| 1. Cleaning | Normalize whitespace and inline URLs; drop documents with an empty cleaned title or body. |
| 2. Language | Keep documents that fastText identifies as English with confidence $\geq 0.5$. |
| 3. Gopher repetition | Apply duplicate-line, duplicate-paragraph, and repeated $n$-gram checks. |
| 4. Gopher quality | Require 50–100,000 non-symbol tokens, mean word length in $[3, 10]$, bounded symbol, bullet, and ellipsis ratios, alphabetic-word ratio $\geq 0.75$, and at least two stop-word occurrences. |
| 5. C4 quality | Apply C4 line and document cleanup and require at least five sentences. |
| 6. FineWeb quality | Require at least $12\%$ of lines to end with terminal punctuation and at most $80\%$ short lines. |
| 7a. URL deduplication | Remove documents whose canonicalized URL has already been seen. |
| 7b. Title deduplication | Suppress identical normalized titles appearing within a seven-day window, retaining the earliest-published document. |
| 7c. MinHash deduplication | Apply MinHash-LSH over word 5-grams and retain the earliest-published representative of each near-duplicate component. |
| 8. Serving-date window | Retain documents whose resolved publication date falls within March 1, 2025 through May 31, 2026. |

Table 4: Corpus filtering pipeline. The aggregate post-quality count is reported in Table 6a.

**Publication dates and temporal integrity.** We read publication and modification timestamps from structured page metadata (Guha et al., 2016; Schema.org; Google Search Central), such as schema.org and OpenGraph fields, and discard values outside 2010–2030. Each article receives the UTC day of the structured timestamp closest to its crawl time, preferring the later one in case of a tie and never exceeding the crawl time. Articles without a valid structured timestamp receive their crawl day. Across the corpus, the assigned date comes from publication metadata for $58.6\%$ of articles, from modification metadata for $28.3\%$, and from the crawl time for $13.1\%$.

Retrieval at forecast date $\tau_t$ includes articles assigned to $\tau_t$ itself. Because dates are assigned at day precision, they do not show that a page existed at a particular time of day or that an edited page already had its captured content on the assigned date. The leakage audit in Appendix A.2 addresses the violations we observed.

**Duplicate suppression.** We remove repeated coverage by URL matching, suppression of identical titles within a seven-day window, and MinHash near-duplicate detection over word 5-grams, keeping the earliest-published copy in each case. When a URL was captured more than once, we keep a single capture, so a retained page is not necessarily its earliest version.

**Indexing and temporal restriction.** Each retained article is embedded with Qwen3-Embedding-8B (Zhang et al., 2025) using its title, summary, and body. The normalized embeddings are stored in exact inner-product indices, one per month. At forecast date $\tau_t$, search is restricted to

$$\mathcal{I}_{\leq \tau_t} = \{d \in \mathcal{I} : \text{2025-03-01} \leq t(d) \leq \tau_t\}.$$

The date restriction is applied before similarity ranking, and full-article access independently rechecks the same temporal constraint.

### A.2 Event Collection and Filtering

**Event collection and market beliefs.** We construct forecasting questions from resolved Polymarket events. A binary event is a single yes/no market, and a multi-option event is a group of at least two mutually exclusive markets. Other market layouts are excluded. For market leg $k$, let $t_k^{\text{start}}$ and $t_k^{\text{close}}$ denote its recorded start and close times. We define the complete tradeable span as

$$[t_{\text{start}}, t_{\text{close}}] = \left[\min_k t_k^{\text{start}}, \max_k t_k^{\text{close}}\right].$$

Ground-truth labels are derived from terminal YES prices: binary markets resolve to YES at $\geq 0.99$ and to NO at $\leq 0.01$, while multi-option events require exactly one YES-resolved option. We

| Criterion | Binary | Multi-option |
|---|---|---|
| Forecasting horizon | $\geq 8$ days | $\geq 8$ days |
| Active day | $\geq 6$ trades, $\geq 200$ shares | $\geq 5$ trades, $\geq 50$ shares |
| Active-day coverage | $\rho_Q \geq 0.80$ | $\rho_Q \geq 0.70$ |
| Max. inactive run | $S_Q \leq 3$ | $S_Q \leq 5$ |
| Residual uncertainty | $a_Q \geq 0.05$ | $a_Q \geq 0.05$ |
| Temporal signal | $V_Q \geq 10^{-6}$ and $(C_Q \geq 0.20 \vee V_Q \geq 0.50)$ | |

Table 5: Market-side event-selection criteria.

reconstruct the daily market belief $m_{Q,t} \in \Delta^{K-1}$ from the UTC-day mean of YES trade prices. For a multi-option event with priced options $A_t$,

$$m_{Q,t}(k) = \frac{x_{t,k}}{\sum_{j \in A_t} x_{t,j}}, \qquad k \in A_t,$$

with missing options left undefined rather than imputed.

**Market-side filtering.** We first filter events for a meaningful forecasting horizon and a reliable, nontrivial market-belief trajectory. Events must span at least eight UTC calendar days. We then define a day as *active* when the trade count and share volume on the market leg corresponding to the realized outcome exceed the type-specific thresholds in Table 5. Let $\rho_Q$ denote the fraction of active days and $S_Q$ the longest consecutive run of inactive days.

Among events passing this activity filter, we measure task nontriviality from the market probability assigned to the realized outcome, $m_{Q,t}(Y)$. Let $\mathcal{F}_Q$ denote active days with an observed belief, and let $\Delta_t = m_{Q,t}(Y) - m_{Q,t-1}(Y)$ for adjacent calendar days with valid active-day observations. We define

$$a_Q = \frac{1}{|\mathcal{F}_Q|} \sum_{t \in \mathcal{F}_Q} -\log \operatorname{clip}\left(m_{Q,t}(Y), 10^{-6}, 1 - 10^{-6}\right), \quad V_Q = \sum_t |\Delta_t|, \quad C_Q = \frac{\sum_t \Delta_t}{\sum_t |\Delta_t|}.$$

These remove events that are already nearly certain or have little meaningful temporal variation.

**Corpus-side filtering.** Market-side filtering identifies events with usable forecasting trajectories, but does not establish whether the frozen information environment is suitable for forecasting. We therefore apply a complementary corpus-side screen that evaluates both the sufficiency of pre-resolution evidence and potential information leakage.

*Evidence sufficiency.* Following the sufficient-context framework of Joren et al. (2025), we assess whether the retrieved information contains enough evidence to support an informed forecast. We adapt this protocol to historical forecasting by first decomposing each event into targeted information needs. For each event surviving the market-side filters, we use Claude Opus 4.8 with maximum reasoning effort (Anthropic, 2026b) to generate 4–7 predictive-evidence queries and 4–7 background/reference-class queries. Query generation explicitly targets information available before resolution and forbids searches for the realized outcome or post-resolution reports. See the full decomposition prompt in Figure 8.

Sufficiency is evaluated once at the latest eligible forecasting state,

$$\tau^\star = t_{\text{close}} - 2 \text{ days}.$$

The buffer is relative to the recorded market close. It reduces exposure to close-day reporting and adjacent-day timestamp ambiguity, but does not ensure that the deciding event was still unresolved. For each query, we retrieve dense (Qwen3-Embedding-8B) and BM25 (Robertson & Zaragoza, 2009) top-10 results restricted to $[2025\text{-}03\text{-}01, \tau^\star]$ and combine them using reciprocal-rank fusion,

$$\text{RRF}(d) = \sum_q \sum_{r \in \{\text{dense}, \text{BM25}\}} \frac{\mathbf{1}[d \in L_{q,r}]}{60 + \text{rank}_{q,r}(d)}.$$

The 15 highest-ranked unique documents form the evidence set $E_Q$. We then use Claude Opus 4.6 (Anthropic, 2026a) to classify the available evidence as {SUFFICIENT, PARTIAL, INSUFFICIENT}. Only events receiving a SUFFICIENT verdict are eligible for the final benchmark.

(a) Corpus construction

| Stage | Removed | Remaining |
|---|---|---|
| Stages 1–6 | – | 26,533,747 |
| Stage 7a | 194,940 | 26,338,807 |
| Stage 7b | 5,563,879 | 20,774,928 |
| Stage 7c | 1,124,734 | 19,650,194 |
| Stage 8 | 823,254 | **18,826,940** |

(b) Event selection

| Stage | Train | Eval |
|---|---|---|
| Temporal split pool | 13,671 | 13,920 |
| Forecasting horizon | 9,051 | 7,590 |
| Market activity | 1,965 | 423 |
| Task nontriviality | 1,777 | 378 |
| Evidence sufficiency & judge | **1,338** | **230** |

Table 6: Construction waterfalls for the deployed news corpus and forecasting events. Event counts use the final June–March training and March–June evaluation splits.

*Leakage control.* The same judge additionally screens the retrieved evidence for potential outcome leakage. This content-level check complements the metadata-level temporal restriction on retrieval and provides an additional safeguard when the visible content of a page may not be fully reflected by its assigned publication timestamp. The judge does not receive the structured realized outcome, final market prices, or crowd trajectory.

Let $v_Q$ denote the evidence-sufficiency verdict and $\ell_Q$ the leakage flag. At this screening stage, an event is eligible iff

$$v_Q = \textsc{Sufficient} \qquad \text{and} \qquad \ell_Q = \text{false}.$$

See Figure 9 for the judge prompt and Table 6 for event filtering funnels.

### A.3 Forecast-Step Construction

**Temporal split.** We split at the event level using the complete tradeable lifetime. An event enters training iff $t_{\text{start}} \geq$ 2025-06-01 and $t_{\text{close}} <$ 2026-03-01, and enters evaluation iff $t_{\text{start}} \geq$ 2026-03-01 and $t_{\text{close}} <$ 2026-06-01. Events that cross the March 1 boundary are excluded from both splits.

**Number of forecast steps.** For event $Q$, let $D_Q$ be its contiguous UTC daily grid. We reserve a two-day buffer before the recorded close date and define

$$E_Q = \{d \in D_Q : \text{date}(t_{\text{close}}) - d \geq 2\}, \qquad n_Q = |E_Q|.$$

The number of forecast steps is

$$T_Q = \text{clamp}\left(\text{round}\,\sqrt{n_Q}, 3, 10\right).$$

**Date selection.** Within the eligible grid, market movement is

$$b_t = \begin{cases} |m_{Q,t}(\text{YES}) - m_{Q,t-1}(\text{YES})|, & \text{binary}, \\ \frac{1}{2}\sum_k |m_{Q,t}(k) - m_{Q,t-1}(k)|, & \text{multi-option}, \end{cases}$$

with zero assigned when an adjacent belief is unavailable. Corpus activity $c_t$ is the total number of indexed news articles published on calendar day $t$. It is a global news-volume signal rather than an event-specific relevance score. After independently max-normalizing both signals, eligible date $e_j$ receives

$$s_j = 0.7\,\tilde{b}_{e_j} + 0.3\,\tilde{c}_{e_j} + 10^{-3}\frac{j}{n_Q - 1}.$$

We partition the eligible sequence into $T_Q$ contiguous equal-count bins and choose the highest-scoring date from each bin. This preserves temporal coverage while favoring dates with larger belief changes or greater overall news activity.

### A.4 Evaluation-Set Composition

Table 8 summarizes the 230 evaluation events. Sports, crypto, and politics make up about two thirds of them. Most crypto events ask for a Bitcoin or Ethereum price range on a given date, and most three-option events are soccer matches with a draw outcome. Multi-option events offer a median of nine outcomes. Episodes are short: the median event closes nine days after its first forecast date.

Table 9 breaks down Brier scores by domain. Research helps most where the no-tools prior is weakest. In politics, no-tools forecasts are worse than uniform (0.964 versus 0.852), and research lowers Brier by 0.24. In sports, models already hold useful priors, and research adds 0.05. The gap to the market is smallest in sports (0.045) and largest in pop culture (0.28), where many events resolve on streaming charts or box-office tallies that news covers only partly. This is consistent with the market drawing on information outside the archive. The science, health, and weather domain shows an even larger gap (0.46) but has only eight events.

Table 7: Forecast-Dojo task splits. Memory-free and memory-on use the same forecast steps.

| Split | Temporal Range | Events | Binary | Multi-option | Forecast steps |
|---|---|---|---|---|---|
| Train | [2025-06, 2026-03) | 1,338 | 602 | 736 | 5,325 |
| Evaluation | [2026-03, 2026-06) | 230 | 40 | 190 | 797 |

Table 8: Composition of the evaluation split (230 events, 797 forecast dates). Domains use each event's primary domain label.

| **Primary domain** | | **Offered outcomes $K$** | |
|---|---|---|---|
| Sports | 56 | 2 (binary) | 40 |
| Crypto | 55 | 3 | 47 |
| Politics | 40 | 4–6 | 32 |
| Pop culture | 26 | 7–10 | 32 |
| Geopolitics | 20 | 11–20 | 70 |
| Finance | 19 | >20 (max 83) | 9 |
| Science, health, weather | 8 | | |
| Tech | 5 | **Forecast dates per event** | |
| Crime and society | 1 | 3 / 4 / 5 / 6–8 | 163 / 42 / 14 / 11 |
| Days from the first forecast date to close: median 9 (IQR 6–13, max 58). | | | |

Table 9: Brier score by primary domain. Values average the 12 models and four rollouts within each event and then weight events equally, so they differ slightly from the forecast-weighted means in Table 2. The single crime-and-society event is omitted.

| Domain | Events | Uniform | No tools | Memory-free | Memory-on | Market |
|---|---|---|---|---|---|---|
| Sports | 56 | 0.694 | 0.564 | 0.514 | 0.522 | 0.469 |
| Crypto | 55 | 0.836 | 0.868 | 0.748 | 0.744 | 0.638 |
| Politics | 40 | 0.852 | 0.964 | 0.728 | 0.746 | 0.627 |
| Pop culture | 26 | 0.858 | 0.838 | 0.639 | 0.630 | 0.357 |
| Geopolitics | 20 | 0.698 | 0.690 | 0.599 | 0.597 | 0.424 |
| Finance | 19 | 0.641 | 0.561 | 0.465 | 0.463 | 0.393 |
| Science, health, weather | 8 | 0.871 | 0.940 | 0.806 | 0.830 | 0.346 |
| Tech | 5 | 0.723 | 1.079 | 0.804 | 0.789 | 0.677 |

### A.5 Environment Details

**Information boundary.** Search and article reads each enforce the inclusive date boundary. Python runs in an isolated runtime with networking disabled and cannot read benchmark data files, including the task files, which store outcomes and market probabilities, and the corpus and index, which contain articles dated after the forecast date. Date-restricted search and article reads are therefore the agent's only access to corpus articles.

**Memory transfer.** Each memory-free forecast starts a new conversation. In memory-on execution, only the previous `<belief_notebook>` block is passed forward, without earlier conversations, tool outputs, or interpreter state. If a step produces no notebook, the next step receives an empty one.

Table 10: Tools available to the agent. Search and article reads enforce the date restriction, and Python runs in an isolated runtime without network or benchmark-data access.

| Tool | Semantics and permissions |
|---|---|
| `search` | Dense retrieval over the assigned-date window, with default $k = 5$. Returns article IDs, titles, URLs, assigned dates, similarity scores, and short snippets. Candidate dates are restricted before ranking. |
| `scrape` | Returns the article body for an ID returned by search in the same forecast step, after rechecking the date boundary. Bodies are truncated to 8,000 tokens. |
| `python` | Executes code for numerical analysis in an isolated runtime with networking disabled and no access to benchmark data files, including the task files, news corpus, and retrieval index. |

Table 11: Knowledge and release dates of evaluated models. We report an official knowledge or training-data cutoff when available; otherwise, we use the public checkpoint release date.

| Model | Date | Basis |
|---|---|---|
| *Proprietary* | | |
| GPT-5.6 Sol | 2026-02-16 | Cutoff |
| GPT-5.5 | 2025-12-01 | Cutoff |
| GPT-5.4 | 2025-08-31 | Cutoff |
| Claude Opus 4.8 | 2026-01 | Cutoff |
| Claude Opus 4.6 | 2025-05 | Cutoff |
| *Open-weight* | | |
| GLM-5 | 2026-02-11 | Release |
| Qwen3.5-397B | 2026-02-16 | Release |
| Kimi K2.5 | 2026-01-27 | Release |
| MiniMax M2.5 | 2026-02-12 | Release |
| DeepSeek-V3.2 | 2025-12-01 | Release |
| gpt-oss-120b | 2024-06-01 | Cutoff |
| Nemotron 3 Super | 2026-02 | Post-train |

*Note.* "Release" denotes the public checkpoint release date when no official knowledge cutoff is reported.

# B Experiment Details

## B.1 Knowledge Cutoffs and Checkpoint Dates

Table 11 lists each model's reported knowledge cutoff or checkpoint date. All dates precede the evaluation period, which begins on March 1, 2026.

**Diagnostic for outcome recall.** Reported cutoffs may not be exact, so we also check the forecasts directly. Widespread, usable outcome recall would be expected to support no-tools accuracy even when the contemporaneous market favored the wrong outcome. The market favorite was wrong on 35.4% of evaluation dates. On these dates, no-tools accuracy ranges from 11.4% to 16.9% across the 12 models, below the 19.1% of a uniform forecast, while on the other dates it ranges from 41.5% to 63.3%. This holds for GPT-5.6 Sol, whose cutoff is closest to the evaluation period (16.8% versus 63.3%). No-tools Information-$\alpha$ at the first forecast date is negative for every model. These aggregate diagnostics do not exclude partial event memorization or other post-cutoff knowledge.

## B.2 Models and Execution Protocols

Table 12 lists the decoding settings and tool budgets of the 12 model configurations. All runs use the same 230 held-out events, four rollouts per date. Tool runs share a budget of 120 tool iterations and 400 calls per forecast step. No-tool runs remove the tools and the corpus and raise the output cap to 65,536 tokens, since the whole forecast is then produced in a single call.

Settings for the training use case are given in Appendix C.5.

Table 12: **Model Evaluation Configurations.** Max output tokens is the cap per model call and includes reasoning tokens. default denotes the provider's default value. Tool budgets are per forecast step and identical in memory-free and memory-on mode; no-tool runs disable all tools.

| | | Max output tokens | | Temperature / top-$p$ / top-$k$ | | Tool budget |
|---|---|---|---|---|---|---|
| Model | Reasoning | Tools | No tools | Tools | No tools | iters / calls |
| GPT-5.6 Sol | max | 32,768 | 65,536 | default | default | 120 / 400 |
| GPT-5.5 | xhigh | 32,768 | 65,536 | default | default | 120 / 400 |
| GPT-5.4 | xhigh | 32,768 | 65,536 | default | default | 120 / 400 |
| Opus 4.8 | max | 32,768 | 65,536 | default | default | 120 / 400 |
| Opus 4.6 | max | 32,768 | 65,536 | default | default | 120 / 400 |
| GLM-5 | high | 32,768 | 65,536 | 1.0 / 1.0 / default | 1.0 / 1.0 / default | 120 / 400 |
| Kimi K2.5 | high | 32,768 | 65,536 | 1.0 / 1.0 / default | 1.0 / 1.0 / default | 120 / 400 |
| DeepSeek V3.2 | high | 32,768 | 65,536 | 1.0 / 1.0 / default | 1.0 / 1.0 / default | 120 / 400 |
| MiniMax M2.5 | native[a] | 32,768 | 65,536 | 1.0 / 1.0 / default | 1.0 / 1.0 / default | 120 / 400 |
| Nemotron 3 Super | high | 32,768 | 65,536 | 1.0 / 1.0 / default | 1.0 / 1.0 / default | 120 / 400 |
| gpt-oss-120b | high | 32,768 | 65,536 | 1.0 / 1.0 / default | 1.0 / 1.0 / default | 120 / 400 |
| Qwen3.5-397B | native[a] | 32,768 | 65,536 | 1.0 / 1.0 / default | 1.0 / 1.0 / default | 120 / 400 |

[a] The model reasons by default and exposes no effort setting.

### B.3 Evaluation Metrics

**Brier score.** Let $i = (e, t, r)$ index an event, forecast date, and rollout, and let $Y_i$ denote the realized outcome. For a usable probability report, negative entries are clipped to zero and the remaining positive mass is normalized. Reported labels must match the offered outcome labels exactly. Omitted outcomes receive zero probability, and probability placed on a label that matches no offered outcome counts as error. Let $\mathcal{U}_i$ be the union of the reported labels and the truth label. The multiclass Brier score is

$$B_i = \sum_{c \in \mathcal{U}_i} \left(p_{ic} - \mathbf{1}\{c = Y_i\}\right)^2 .$$

We use the standard $[0, 2]$ scale, without normalization by the number of outcomes or an additional binary-event factor. A uniform forecast over $K_i$ offered outcomes has Brier score $1 - 1/K_i$.

**Accuracy.** Let $\mathcal{T}_i = \arg\max_c p_{ic}$ denote the set of outcomes assigned maximal probability. We use fractional-tie accuracy,

$$a_i = \frac{\mathbf{1}\{Y_i \in \mathcal{T}_i\}}{|\mathcal{T}_i|}.$$

Thus, a correct unique top prediction receives accuracy 1, while ties split credit uniformly among tied outcomes. A uniform forecast over $K_i$ outcomes therefore has accuracy $1/K_i$.

**Information-$\alpha$.** We measure improvement over the contemporaneous market belief using

$$\alpha_i = \log \max\{p_i(Y_i), \epsilon\} - \log \max\{p_{\text{market},i}(Y_i), \epsilon\}, \qquad \epsilon = 10^{-3},$$

with natural logarithms; positive values favor the agent. We compute it for every recorded forecast whose market probability for the realized outcome is available, using the uniform fallback for unusable reports. Neither realized outcomes nor market probabilities are provided to the forecasting agent.

**Log loss.** We report $\ell_i = -\log \max\{p_i(Y_i), 10^{-3}\}$ with natural logarithms. An unusable report contributes $\log K_i$.

**Top-label calibration.** For a usable forecast, let $c_i = \max_c p_{ic}$ be its top-ranked probability. We group forecasts into 15 equal-width bins of $c_i$ and report $\text{ECE} = \sum_b \frac{n_b}{n} |\bar{c}_b - \bar{a}_b|$, where $\bar{c}_b$ and $\bar{a}_b$ are the mean top-ranked probability and the mean fractional-tie accuracy in bin $b$.

Table 13: **Failure rate of scheduled forecasts by condition (%) for the 12 default model configurations.** A failure is a missing output, unusable output, or no positive finite probability mass. Each cell covers 3,188 scheduled forecasts (797 event–dates, four rollouts). Recorded unusable outputs are scored as uniform forecasts; missing records are excluded from score means.

| Model | No tools | Memory-free | Memory-on |
|---|---|---|---|
| GPT-5.6 Sol | 0.03 | 4.49 | 0.53 |
| GPT-5.5 | 0.06 | 0.25 | 0.03 |
| GPT-5.4 | 0.00 | 0.09 | 0.00 |
| Opus 4.8 | 0.00 | 0.00 | 0.00 |
| Opus 4.6 | 0.38 | 0.03 | 0.00 |
| GLM-5 | 0.28 | 1.47 | 1.13 |
| Kimi K2.5 | 0.00 | 2.16 | 2.45 |
| DeepSeek V3.2 | 0.25 | 1.76 | 1.88 |
| MiniMax M2.5 | 0.63 | 5.52 | 1.85 |
| Nemotron 3 Super | 9.16 | 18.07 | 11.39 |
| gpt-oss-120b | 0.00 | 19.45 | 13.64 |
| Qwen3.5-397B | 0.03 | 1.10 | 1.16 |

**Uncertainty.** Unless stated otherwise, 95% CIs are percentile intervals from 4,000 bootstrap resamples of the 230 evaluation events (seed 0). To compare two configurations, we average the four rollouts of each configuration at every event–date, take differences on matched event–dates, and resample events. Memory comparisons instead pair forecasts by event, date, and rollout. The $\pm$ values in Table 2 are standard deviations of the four rollout means and describe run-to-run variation, not uncertainty over events.

### B.4 Aggregation and Failures

Not every scheduled forecast produces a valid probability report. An output is unusable if its answer does not follow the required format (Appendix E) or contains no valid positive probability. We replace a recorded unusable report by the uniform distribution over its $K_i$ offered outcomes for all scores: $B_i = 1 - 1/K_i$, $a_i = 1/K_i$, and log loss $\log K_i$. Main-table means average recorded forecasts, not missing records. GPT-5.6 Sol and Nemotron 3 Super have 3,184 and 3,170 recorded memory-on forecasts, respectively; all other configurations have 3,188. Table 13 additionally counts missing records as failures, using all 3,188 scheduled forecasts as its denominator.

The same fallback applies to the main table, longitudinal analyses, and SFT evaluation. Information-$\alpha$ is omitted only when the scalar market probability is unavailable. Imputation does not change whether an output is counted as a failure.

**Causes of unusable reports.** Table 14 splits unusable reports by cause. Early stops are rare overall. They affect 1.27% of memory-free forecasts (0.52% output-token cap, 0.41% tool-iteration limit, 0.31% context overflow, and 0.03% tool-call limit), 0.22% of memory-on forecasts, and 0.03% of no-tools forecasts. The causes differ by model. All of GPT-5.6 Sol's unusable memory-free reports are early stops, which memory largely removes, whereas all of gpt-oss-120b's are finished runs whose reports cannot be scored.

### B.5 Inference Cost and Tool Usage

Table 15 summarizes average inference cost and tool usage per recorded forecast. Dollar columns show proprietary-model provider estimates; NR marks the open-weight models, whose prices depend on the serving provider. Within-model cost reductions in the main text use the five proprietary models with provider-reported prices. Research calls count search, scrape, and Python calls, including calls that return errors. In the memory-on setting, the first forecast of each episode starts without prior memory, while later forecasts receive the notebook produced at the preceding step. These statistics are intended as descriptive resource estimates rather than hardware-normalized efficiency

Table 14: **Unusable forecasts by cause (% of recorded forecasts).** Early: the run stopped before a final answer because of context overflow, the output-token cap, or a tool limit. Format: the run finished but its report could not be scored.

| | No tools | | Memory-free | | Memory-on | |
|---|---|---|---|---|---|---|
| Model | Early | Format | Early | Format | Early | Format |
| GPT-5.6 Sol | 0.00 | 0.03 | 4.49 | 0.00 | 0.41 | 0.00 |
| GPT-5.5 | 0.00 | 0.06 | 0.16 | 0.09 | 0.00 | 0.03 |
| GPT-5.4 | 0.00 | 0.00 | 0.09 | 0.00 | 0.00 | 0.00 |
| Claude Opus 4.8 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Claude Opus 4.6 | 0.38 | 0.00 | 0.00 | 0.03 | 0.00 | 0.00 |
| GLM-5 | 0.00 | 0.28 | 0.00 | 1.47 | 0.00 | 1.13 |
| Qwen3.5-397B | 0.00 | 0.03 | 0.00 | 1.10 | 0.00 | 1.16 |
| Kimi K2.5 | 0.00 | 0.00 | 0.03 | 2.13 | 0.00 | 2.45 |
| MiniMax M2.5 | 0.00 | 0.63 | 0.03 | 5.49 | 0.00 | 1.85 |
| DeepSeek-V3.2 | 0.00 | 0.25 | 0.03 | 1.73 | 0.00 | 1.88 |
| gpt-oss-120b | 0.00 | 0.00 | 0.00 | 19.45 | 0.00 | 13.64 |
| Nemotron 3 Super | 0.00 | 9.16 | 10.35 | 7.72 | 2.37 | 8.52 |

Table 15: **Inference cost and tool usage by model.** Average provider-estimated cost and research calls per recorded forecast. Research calls sum search, scrape, and Python calls, including errors. No-tools forecasts make no research calls.

| | USD per forecast[a] | | | Research calls per forecast | |
|---|---|---|---|---|---|
| Model | No tools | Memory-free | Memory-on | Memory-free | Memory-on |
| GPT-5.6 Sol | 0.18 | 4.45 | 3.38 | 136.6 | 118.3 |
| GPT-5.5 | 0.25 | 5.85 | 3.90 | 70.3 | 56.8 |
| GPT-5.4 | 0.20 | 4.49 | 3.52 | 89.9 | 76.6 |
| Opus 4.8 | 0.25 | 2.74 | 2.02 | 22.9 | 17.0 |
| Opus 4.6 | 0.28 | 2.91 | 2.66 | 33.2 | 29.5 |
| GLM-5 | NR | NR | NR | 24.9 | 22.3 |
| Kimi K2.5 | NR | NR | NR | 24.3 | 20.1 |
| DeepSeek V3.2 | NR | NR | NR | 39.6 | 34.4 |
| MiniMax M2.5 | NR | NR | NR | 16.7 | 12.5 |
| Nemotron 3 Super | NR | NR | NR | 33.9 | 30.6 |
| gpt-oss-120b | NR | NR | NR | 21.7 | 17.2 |
| Qwen3.5-397B | NR | NR | NR | 19.1 | 20.7 |

[a] NR: not reported; the open-weight models are served without a comparable per-forecast price.

comparisons. Changes in Figure 3 use ratios of summed resources over matched model–event–date–rollout records; intervals use 4,000 bootstrap resamples of events.

Table 16 breaks research calls into search, full-article reads, and Python, and reports token use. On matched records, memory lowers cost per forecast by 23.9% $[20.6, 27.3]$ for GPT-5.6 Sol, 33.3% $[31.4, 35.1]$ for GPT-5.5, 21.5% $[19.2, 23.8]$ for GPT-5.4, 26.2% $[23.8, 28.6]$ for Claude Opus 4.8, and 8.3% $[6.2, 10.4]$ for Claude Opus 4.6.

### B.6 Measuring Newly Recorded Evidence

**Longitudinal scores and support.** Figure 2 uses the same scoring as the main table. We divide each event's retained forecast dates into early, middle, and late thirds by relative position $(t-1)/(T_e-1)$. Panel (a) uses the 223 events (224 for the market) with forecasts and market probabilities in all three stages. Each event receives equal weight, and 95% intervals use 4,000 event-level bootstrap resamples.

**Newly dated evidence.** We count newly dated evidence recorded between consecutive forecasts in memory-on notebooks, using those counts as a proxy for evidence availability. For model $k$, rollout

Table 16: **Research calls and tokens per recorded forecast.** Search, read, and Python calls include tool errors. Tokens are provider-reported totals in thousands for memory-free forecasts.

| | Memory-free calls | | | Memory-on calls | | | Tokens (k) | |
|---|---|---|---|---|---|---|---|---|
| Model | Search | Read | Python | Search | Read | Python | Input | Output |
| GPT-5.6 Sol | 96.6 | 27.4 | 12.5 | 86.9 | 22.0 | 9.5 | 7,037 | 25.4 |
| GPT-5.5 | 47.0 | 19.8 | 3.4 | 39.2 | 14.6 | 3.0 | 2,525 | 36.0 |
| GPT-5.4 | 68.3 | 17.4 | 4.2 | 59.3 | 13.5 | 3.8 | 3,365 | 60.8 |
| Claude Opus 4.8 | 12.5 | 7.7 | 2.6 | 9.7 | 4.9 | 2.5 | 411 | 27.4 |
| Claude Opus 4.6 | 19.8 | 10.2 | 3.2 | 17.6 | 8.5 | 3.4 | 502 | 15.7 |
| GLM-5 | 13.9 | 8.0 | 3.0 | 12.7 | 6.7 | 3.0 | 274 | 8.9 |
| Qwen3.5-397B | 11.8 | 6.0 | 1.4 | 12.4 | 5.7 | 2.7 | 171 | 6.0 |
| Kimi K2.5 | 15.3 | 5.5 | 3.5 | 12.3 | 5.1 | 2.7 | 384 | 10.2 |
| MiniMax M2.5 | 9.5 | 3.8 | 3.5 | 7.4 | 2.7 | 2.4 | 273 | 8.3 |
| DeepSeek-V3.2 | 23.6 | 7.3 | 8.6 | 19.5 | 6.2 | 8.8 | 937 | 18.2 |
| gpt-oss-120b | 10.8 | 7.8 | 3.1 | 9.0 | 5.9 | 2.3 | 438 | 8.9 |
| Nemotron 3 Super | 26.9 | 2.8 | 4.3 | 19.4 | 2.0 | 9.2 | 1,686 | 36.7 |

$r$, event $e$, and step $t \geq 2$, let

$$n_{k,r}(e,t) = \sum_{a \in \text{ledger}(M_{k,r}(e,t))} \mathbf{1}\{\tau_{e,t-1} < d(a) \leq \tau_{e,t}\},$$

where $d(a)$ is the evidence date recorded in notebook entry $a$. Thus, an entry counts only when its recorded evidence date falls between the previous and current forecast dates. Earlier evidence discovered late is not counted, and carried entries are not counted again at later steps. We count both active and superseded entries and exclude entries with invalid or future dates.

We first average across the available notebook chains of each model,

$$\text{own}_k(e,t) = \frac{1}{|\mathcal{R}_k(e,t)|} \sum_{r \in \mathcal{R}_k(e,t)} n_{k,r}(e,t).$$

**Leave-one-model-out evidence availability.** To estimate how much new evidence was available at an event-step without using the evaluated model's own notebook, we average the corresponding counts over the set $\mathcal{P}_k(e,t)$ of other models with available notebook counts:

$$A_k(e,t) = \frac{1}{|\mathcal{P}_k(e,t)|} \sum_{j \in \mathcal{P}_k(e,t)} \text{own}_j(e,t).$$

We use $A_k(e,t)$ as the evidence-availability index for model $k$, including when analyzing its memory-free forecasts. This leave-one-model-out construction avoids directly coupling a model's forecast change to its own recording behavior.

For consecutive forecasts, we define

$$\Delta B^A_{k,r}(e,t) = B^A_{k,r}(e,t) - B^A_{k,r}(e,t-1),$$

where negative values indicate improvement. Panel (b) groups consecutive-date transitions by quartiles of $A_k(e,t)$, averages first within events and then across events, and resamples events for confidence intervals.

The index should be interpreted as a proxy for newly available, event-relevant evidence rather than an exhaustive corpus count. It depends on what the other agents record in their notebooks and may miss relevant evidence that no model retrieves.

Table 17 lists the values behind Figure 2(b) together with the market reference. The market improves in every quartile and most in the top two, which suggests that the index tracks periods with informative news. Agents gain little when few new entries are recorded.

Table 17: **Consecutive-step Brier change by new-evidence quartile.** Values underlie Figure 2(b). Quartile cut points are 1.49, 2.45, and 3.98 newly dated entries. Brackets are 95% event-bootstrap CIs, and the last column counts events contributing memory-free transitions.

| Quartile | No tools | Memory-free | Memory-on | Market | Events |
|---|---|---|---|---|---|
| Q1 | $-0.002$ $[-0.010, +0.004]$ | $-0.007$ $[-0.018, +0.005]$ | $-0.000$ $[-0.010, +0.010]$ | $-0.051$ $[-0.095, -0.013]$ | 124 |
| Q2 | $-0.005$ $[-0.012, +0.003]$ | $-0.024$ $[-0.049, -0.002]$ | $-0.022$ $[-0.048, +0.002]$ | $-0.046$ $[-0.086, -0.006]$ | 146 |
| Q3 | $+0.002$ $[-0.004, +0.008]$ | $-0.032$ $[-0.057, -0.009]$ | $-0.031$ $[-0.057, -0.009]$ | $-0.100$ $[-0.145, -0.056]$ | 135 |
| Q4 | $+0.001$ $[-0.008, +0.009]$ | $-0.071$ $[-0.115, -0.031]$ | $-0.060$ $[-0.098, -0.025]$ | $-0.096$ $[-0.140, -0.058]$ | 103 |

### B.7 Evidence Capture and Forecast Improvement

**Evidence capture.** The availability index above measures how much new evidence appears to be available at an event-step. To measure how much of that evidence each model captures, we compare the model's own newly dated entries with the leave-one-model-out availability index.

For event–step pairs $\mathcal{S}_k$ with own and peer counts, the no-intercept slope is

$$\kappa_k = \frac{\sum_{(e,t)\in\mathcal{S}_k} \mathrm{own}_k(e,t) A_k(e,t)}{\sum_{(e,t)\in\mathcal{S}_k} A_k(e,t)^2}.$$

A value of $\kappa_k = 1$ means that the model records newly dated evidence at the peer-average rate; values above or below one indicate higher or lower capture, respectively. For example, $\kappa_k = 1.5$ corresponds to a fitted recording rate 50% above the peer average. Importantly, $\kappa_k$ is a relative rate, not the fraction of an exhaustive evidence set that the model retrieves.

**Episode gain.** We measure how much a model improves over an episode using its scheduled first and last retained memory-on forecasts, pairing recorded endpoints within each rollout:

$$G_k = \frac{1}{|\mathcal{E}_k|} \sum_{e\in\mathcal{E}_k} \frac{1}{|\mathcal{R}_k^{\mathrm{end}}(e)|} \sum_{r\in\mathcal{R}_k^{\mathrm{end}}(e)} \left[ B_{k,r}^{\mathrm{on}}(e,1) - B_{k,r}^{\mathrm{on}}(e,T_e) \right].$$

Positive values indicate improvement. We first average endpoint pairs across rollouts within each event and then average equally across events. Panel (c) compares $\kappa_k$ with $G_k$ across the 12 models and reports their Spearman correlation.

**Interpretation.** Across the 12 models, $\kappa_k$ ranges from 0.36 to 1.74. Under uniform fallback, evidence capture is associated with episode-level improvement (Spearman $\rho = 0.74$, permutation $p = 0.008$ with 12 models), more than search or scrape counts ($\rho = 0.31$ and $0.49$, $p = 0.32$ and $0.10$). The association weakens under other scoring choices. It is 0.64 ($p = 0.028$) when both endpoints must be usable, 0.60 ($p = 0.043$) with typographic label normalization, and 0.14 ($p = 0.65$) when unusable reports receive the worst-case score. It is also mainly a between-model pattern. Within the same model, event, and step, rollouts that record more newly dated entries than their sibling rollouts improve only slightly more (Pearson $r = -0.017$ between the demeaned entry count and the demeaned Brier change, 95% event-bootstrap CI $[-0.036, 0.002]$, negative for 10 of 12 models). Table 18 reports notebook statistics and the per-model values of $\kappa_k$ and $G_k$.

These relationships are correlational. Notebook entries are self-reported, their recorded dates need not always be correct, and evidence missed by all models is invisible to the measure. We therefore interpret $\kappa_k$ as a diagnostic of relative evidence capture, not as a complete or causal measure of information acquisition.

Table 18: **Belief notebooks and evidence capture in the memory-on setting.** Parsed: share of forecasts with a parsed notebook. Entries: median number of ledger entries per notebook. $\kappa_k$ and $G_k$ follow Appendix B.7, with 95% event-bootstrap CIs.

| Model | Parsed (%) | Entries | Capture $\kappa_k$ | Episode gain $G_k$ |
|---|---|---|---|---|
| GPT-5.6 Sol | 99.6 | 17 | 1.61 [1.52, 1.70] | 0.077 [0.048, 0.110] |
| GPT-5.5 | 99.8 | 16 | 1.48 [1.40, 1.55] | 0.075 [0.045, 0.106] |
| GPT-5.4 | 100.0 | 15 | 1.00 [0.96, 1.03] | 0.070 [0.042, 0.100] |
| Claude Opus 4.8 | 99.9 | 9 | 1.17 [1.13, 1.22] | 0.066 [0.035, 0.099] |
| Claude Opus 4.6 | 100.0 | 15 | 1.74 [1.68, 1.81] | 0.082 [0.048, 0.116] |
| GLM-5 | 94.1 | 12 | 1.32 [1.28, 1.37] | 0.058 [0.022, 0.094] |
| Qwen3.5-397B | 97.1 | 9 | 0.64 [0.60, 0.68] | 0.063 [0.030, 0.095] |
| Kimi K2.5 | 95.2 | 10 | 1.07 [1.02, 1.11] | 0.070 [0.032, 0.108] |
| MiniMax M2.5 | 95.1 | 8 | 0.54 [0.50, 0.58] | 0.070 [0.036, 0.104] |
| DeepSeek-V3.2 | 97.6 | 12 | 0.77 [0.73, 0.80] | 0.021 [−0.021, 0.062] |
| gpt-oss-120b | 86.7 | 9 | 0.37 [0.32, 0.44] | 0.046 [0.011, 0.082] |
| Nemotron 3 Super | 80.4 | 7 | 0.36 [0.32, 0.40] | 0.056 [0.025, 0.087] |

# C Additional Results

## C.1 Statistical Uncertainty and Paired Comparisons

Table 19 adds event-level 95% CIs to the main results. With 230 events, these intervals span roughly $\pm 0.05$ in Brier, far wider than the rollout standard deviations in Table 2, so we compare configurations with paired differences. Research tools improve Brier and accuracy for every model with CIs that exclude zero (Table 20). Memory changes are small. On forecasts that are usable in both settings, memory lowers Brier for four models, none significantly, and raises it significantly for Claude Opus 4.8. Every configuration trails the market, and both the Brier gap and Information-$\alpha$ have CIs that exclude zero in all 36 cases (Table 21). Table 22 lists the model contrasts discussed in Section 5.2.

Table 19: **Main results with event-level uncertainty.** Point estimates match Table 2. Brackets give 95% CIs from 4,000 bootstrap resamples of the 230 evaluation events. Market CIs use the 790 event–dates with a complete market vector.

| | Brier↓ | | | Accuracy (%)↑ | | |
|---|---|---|---|---|---|---|
| Model | No tools | Memory-free | Memory-on | No tools | Memory-free | Memory-on |
| GPT-5.6 Sol | 0.650 [0.596, 0.705] | 0.554 [0.505, 0.601] | 0.546 [0.496, 0.595] | 47.03 [40.7, 53.3] | 56.36 [51.1, 61.7] | 57.59 [52.2, 63.0] |
| GPT-5.5 | 0.698 [0.641, 0.755] | 0.564 [0.516, 0.610] | 0.571 [0.520, 0.620] | 43.85 [37.6, 49.9] | 57.69 [52.3, 63.4] | 57.68 [52.2, 63.3] |
| GPT-5.4 | 0.708 [0.659, 0.757] | 0.586 [0.540, 0.633] | 0.583 [0.535, 0.631] | 41.34 [35.2, 47.4] | 53.12 [47.5, 58.7] | 52.87 [47.2, 58.5] |
| Claude Opus 4.8 | 0.725 [0.669, 0.782] | 0.589 [0.539, 0.639] | 0.605 [0.553, 0.657] | 40.30 [34.0, 46.5] | 52.63 [47.0, 58.4] | 50.94 [45.3, 56.6] |
| Claude Opus 4.6 | 0.762 [0.705, 0.818] | 0.603 [0.551, 0.655] | 0.598 [0.545, 0.653] | 35.16 [29.3, 41.1] | 52.35 [46.7, 57.9] | 53.32 [47.9, 58.9] |
| GLM-5 | 0.731 [0.683, 0.778] | 0.625 [0.577, 0.673] | 0.632 [0.580, 0.684] | 38.54 [32.9, 44.1] | 50.02 [44.8, 55.2] | 49.96 [44.6, 55.2] |
| Qwen3.5-397B | 0.759 [0.710, 0.807] | 0.639 [0.591, 0.686] | 0.643 [0.590, 0.695] | 36.54 [31.0, 42.4] | 48.49 [43.1, 53.9] | 48.90 [43.5, 54.4] |
| Kimi K2.5 | 0.773 [0.719, 0.824] | 0.637 [0.592, 0.681] | 0.658 [0.609, 0.706] | 36.30 [30.8, 41.8] | 49.37 [44.3, 54.4] | 48.07 [43.1, 53.2] |
| MiniMax M2.5 | 0.758 [0.709, 0.804] | 0.654 [0.605, 0.702] | 0.640 [0.589, 0.691] | 35.68 [30.5, 41.1] | 46.87 [41.8, 51.9] | 47.19 [41.7, 52.6] |
| DeepSeek-V3.2 | 0.805 [0.748, 0.862] | 0.655 [0.605, 0.705] | 0.666 [0.612, 0.719] | 34.34 [28.9, 40.0] | 47.73 [42.5, 53.1] | 47.65 [42.5, 52.9] |
| gpt-oss-120b | 0.896 [0.825, 0.965] | 0.699 [0.654, 0.745] | 0.696 [0.645, 0.748] | 32.44 [27.0, 38.1] | 41.95 [37.7, 46.1] | 43.03 [38.3, 47.7] |
| Nemotron 3 Super | 0.882 [0.824, 0.940] | 0.695 [0.646, 0.744] | 0.688 [0.634, 0.742] | 30.84 [25.8, 36.0] | 43.49 [38.8, 48.1] | 44.89 [40.0, 49.9] |
| Market reference | | 0.498 [0.451, 0.544] | | | 64.55 [59.5, 69.8] | |

## C.2 Results by Event Type and Scoring Convention

Table 23 adds log loss and splits Brier by event type. Table 24 shows how the treatment of unusable reports changes Brier. Worst-case scoring mainly penalizes the models with high failure rates, while scoring only usable reports removes the reliability component. Scoring with typographic label normalization instead of exact matching changes mean Brier by at most 0.014 for Nemotron 3 Super and by less than 0.005 for every other model.

## C.3 Calibration

Table 25 reports top-label calibration. Research lowers ECE for 9 of the 12 models. GPT-5.5 has the largest memory-free ECE (0.085) because it is underconfident, giving its top-ranked outcome 0.50 probability on average while being correct on 58% of usable forecasts. This helps explain why it has the highest accuracy in Table 2 without having the lowest Brier.

Table 20: **Paired effects of research tools and memory.** Tools: memory-free minus no tools, paired on event–dates after averaging rollouts. Memory: memory-on minus memory-free, paired by event, date, and rollout, on all recorded pairs and on pairs where both forecasts are usable. Negative Brier differences favor the first setting. Brackets are 95% event-bootstrap CIs.

| | Tools minus no tools | | Memory minus memory-free (Brier) | |
|---|---|---|---|---|
| Model | Brier | Accuracy (pp) | All recorded | Jointly usable |
| GPT-5.6 Sol | −0.096 [−0.135, −0.058] | +9.3 [+4.9, +14.1] | −0.008 [−0.019, +0.003] | +0.001 [−0.009, +0.011] |
| GPT-5.5 | −0.134 [−0.186, −0.086] | +13.8 [+8.8, +19.2] | +0.007 [−0.003, +0.021] | +0.008 [−0.002, +0.021] |
| GPT-5.4 | −0.122 [−0.167, −0.079] | +11.8 [+6.5, +17.2] | −0.003 [−0.012, +0.005] | −0.004 [−0.012, +0.005] |
| Claude Opus 4.8 | −0.136 [−0.183, −0.091] | +12.3 [+7.2, +17.5] | +0.016 [+0.006, +0.025] | +0.016 [+0.006, +0.025] |
| Claude Opus 4.6 | −0.160 [−0.210, −0.113] | +17.2 [+11.9, +22.7] | −0.004 [−0.014, +0.006] | −0.004 [−0.014, +0.006] |
| GLM-5 | −0.107 [−0.154, −0.060] | +11.5 [+6.7, +16.4] | +0.007 [−0.009, +0.024] | +0.008 [−0.008, +0.024] |
| Qwen3.5-397B | −0.120 [−0.168, −0.074] | +11.9 [+7.0, +17.0] | +0.003 [−0.013, +0.019] | +0.001 [−0.016, +0.017] |
| Kimi K2.5 | −0.136 [−0.183, −0.091] | +13.1 [+8.5, +17.8] | +0.021 [+0.002, +0.041] | +0.019 [−0.001, +0.039] |
| MiniMax M2.5 | −0.104 [−0.149, −0.062] | +11.2 [+6.7, +15.8] | −0.014 [−0.030, +0.003] | −0.007 [−0.023, +0.009] |
| DeepSeek-V3.2 | −0.151 [−0.210, −0.094] | +13.4 [+8.1, +18.7] | +0.011 [−0.003, +0.025] | +0.010 [−0.005, +0.025] |
| gpt-oss-120b | −0.197 [−0.260, −0.136] | +9.5 [+4.8, +14.4] | −0.003 [−0.024, +0.016] | +0.000 [−0.023, +0.023] |
| Nemotron 3 Super | −0.187 [−0.244, −0.131] | +12.7 [+8.0, +17.3] | −0.007 [−0.027, +0.015] | −0.007 [−0.029, +0.015] |

Table 21: **Comparison with the market reference.** The Brier gap is the model's Brier minus the market's on matched event–dates, so positive values favor the market. Information-$\alpha$ follows Appendix B.3, where negative values favor the market. Brackets are 95% event-bootstrap CIs.

| | Brier gap to market | | | Information-$\alpha$ | | |
|---|---|---|---|---|---|---|
| Model | No tools | Memory-free | Memory-on | No tools | Memory-free | Memory-on |
| GPT-5.6 Sol | 0.154 [0.111, 0.202] | 0.057 [0.029, 0.088] | 0.050 [0.019, 0.082] | −0.422 [−0.559, −0.302] | −0.138 [−0.207, −0.075] | −0.104 [−0.172, −0.042] |
| GPT-5.5 | 0.200 [0.149, 0.253] | 0.066 [0.034, 0.100] | 0.074 [0.038, 0.114] | −0.603 [−0.771, −0.456] | −0.186 [−0.275, −0.107] | −0.198 [−0.300, −0.111] |
| GPT-5.4 | 0.210 [0.162, 0.257] | 0.089 [0.054, 0.125] | 0.085 [0.050, 0.124] | −0.633 [−0.779, −0.489] | −0.237 [−0.341, −0.143] | −0.222 [−0.326, −0.128] |
| Claude Opus 4.8 | 0.227 [0.175, 0.279] | 0.093 [0.057, 0.130] | 0.108 [0.069, 0.149] | −0.643 [−0.799, −0.503] | −0.254 [−0.356, −0.164] | −0.300 [−0.409, −0.200] |
| Claude Opus 4.6 | 0.264 [0.213, 0.318] | 0.105 [0.064, 0.150] | 0.101 [0.059, 0.148] | −0.788 [−0.931, −0.651] | −0.282 [−0.394, −0.183] | −0.263 [−0.384, −0.159] |
| GLM-5 | 0.233 [0.184, 0.283] | 0.128 [0.090, 0.169] | 0.135 [0.095, 0.179] | −0.686 [−0.843, −0.540] | −0.372 [−0.493, −0.264] | −0.398 [−0.538, −0.276] |
| Qwen3.5-397B | 0.262 [0.208, 0.315] | 0.143 [0.102, 0.186] | 0.146 [0.102, 0.194] | −0.796 [−0.955, −0.642] | −0.422 [−0.545, −0.308] | −0.441 [−0.592, −0.305] |
| Kimi K2.5 | 0.275 [0.218, 0.332] | 0.138 [0.101, 0.179] | 0.160 [0.116, 0.205] | −0.837 [−1.017, −0.666] | −0.420 [−0.536, −0.317] | −0.439 [−0.567, −0.323] |
| MiniMax M2.5 | 0.260 [0.210, 0.309] | 0.156 [0.115, 0.199] | 0.143 [0.100, 0.187] | −0.797 [−0.940, −0.658] | −0.461 [−0.591, −0.343] | −0.413 [−0.544, −0.294] |
| DeepSeek-V3.2 | 0.308 [0.246, 0.371] | 0.158 [0.117, 0.204] | 0.169 [0.123, 0.218] | −0.991 [−1.194, −0.798] | −0.464 [−0.600, −0.342] | −0.508 [−0.664, −0.370] |
| gpt-oss-120b | 0.396 [0.323, 0.471] | 0.202 [0.163, 0.245] | 0.198 [0.154, 0.245] | −1.378 [−1.622, −1.136] | −0.594 [−0.739, −0.470] | −0.584 [−0.734, −0.448] |
| Nemotron 3 Super | 0.383 [0.321, 0.446] | 0.199 [0.154, 0.245] | 0.194 [0.148, 0.242] | −1.516 [−1.724, −1.315] | −0.743 [−0.888, −0.606] | −0.720 [−0.885, −0.567] |

Table 22: **Model contrasts discussed in Section 5.2.** Differences use matched event–dates after averaging rollouts. Claude Opus 4.6 and Claude Opus 4.8 are the weakest proprietary models in each setting, and GLM-5 is the strongest open-weight model. Brackets are 95% event-bootstrap CIs.

| Contrast | Setting | Metric | Difference [95% CI] |
|---|---|---|---|
| GPT-5.6 Sol − GPT-5.5 | Memory-free | Brier | $-0.010$ [$-0.031$, $+0.010$] |
| GPT-5.6 Sol − GPT-5.5 | Memory-on | Brier | $-0.025$ [$-0.052$, $-0.002$] |
| GPT-5.5 − GPT-5.6 Sol | Memory-free | Accuracy (pp) | $+1.33$ [$-0.70$, $+3.51$] |
| Claude Opus 4.6 − GLM-5 | Memory-free | Brier | $-0.022$ [$-0.054$, $+0.012$] |
| Claude Opus 4.8 − GLM-5 | Memory-on | Brier | $-0.027$ [$-0.058$, $+0.004$] |
| Proprietary − open-weight mean | Memory-free | Brier | $-0.078$ [$-0.103$, $-0.056$] |
| Proprietary − open-weight mean | Memory-on | Brier | $-0.080$ [$-0.105$, $-0.055$] |

Table 23: **Log loss and Brier by event type.** Log loss uses natural logarithms with a probability floor of $10^{-3}$. The evaluation split has 40 binary and 190 multi-option events, so binary results are noisier. NT, MF, and MO denote no tools, memory-free, and memory-on.

| | Log loss↓ | | | Binary Brier↓ | | | Multi-option Brier↓ | | |
|---|---|---|---|---|---|---|---|---|---|
| Model | NT | MF | MO | NT | MF | MO | NT | MF | MO |
| GPT-5.6 Sol | 1.431 | 1.150 | 1.118 | 0.401 | 0.367 | 0.351 | 0.707 | 0.597 | 0.591 |
| GPT-5.5 | 1.620 | 1.199 | 1.210 | 0.429 | 0.404 | 0.440 | 0.759 | 0.600 | 0.601 |
| GPT-5.4 | 1.647 | 1.250 | 1.235 | 0.443 | 0.418 | 0.398 | 0.769 | 0.624 | 0.625 |
| Claude Opus 4.8 | 1.658 | 1.266 | 1.312 | 0.518 | 0.415 | 0.443 | 0.772 | 0.629 | 0.642 |
| Claude Opus 4.6 | 1.802 | 1.294 | 1.275 | 0.440 | 0.457 | 0.460 | 0.836 | 0.636 | 0.630 |
| GLM-5 | 1.699 | 1.383 | 1.409 | 0.463 | 0.435 | 0.481 | 0.793 | 0.668 | 0.666 |
| Qwen3.5-397B | 1.808 | 1.432 | 1.451 | 0.561 | 0.449 | 0.474 | 0.805 | 0.683 | 0.681 |
| Kimi K2.5 | 1.850 | 1.434 | 1.451 | 0.573 | 0.441 | 0.522 | 0.818 | 0.681 | 0.689 |
| MiniMax M2.5 | 1.811 | 1.474 | 1.424 | 0.479 | 0.410 | 0.407 | 0.821 | 0.710 | 0.693 |
| DeepSeek-V3.2 | 2.001 | 1.473 | 1.519 | 0.647 | 0.492 | 0.502 | 0.841 | 0.692 | 0.703 |
| gpt-oss-120b | 2.400 | 1.607 | 1.595 | 0.682 | 0.525 | 0.550 | 0.945 | 0.739 | 0.729 |
| Nemotron 3 Super | 2.534 | 1.750 | 1.727 | 0.659 | 0.480 | 0.430 | 0.933 | 0.744 | 0.748 |
| Uniform reference | | 1.742 | | | 0.500 | | | 0.832 | |
| Market reference | | 1.023 | | | 0.315 | | | 0.540 | |

Table 24: **Brier under three treatments of unusable reports.** U: uniform fallback, the main convention. W: worst-case scoring with Brier 2. Only: usable reports only.

| | No tools | | | Memory-free | | | Memory-on | | |
|---|---|---|---|---|---|---|---|---|---|
| Model | U | W | Only | U | W | Only | U | W | Only |
| GPT-5.6 Sol | 0.650 | 0.650 | 0.650 | 0.554 | 0.606 | 0.541 | 0.546 | 0.551 | 0.545 |
| GPT-5.5 | 0.698 | 0.699 | 0.698 | 0.564 | 0.567 | 0.563 | 0.571 | 0.572 | 0.571 |
| GPT-5.4 | 0.708 | 0.708 | 0.708 | 0.586 | 0.587 | 0.586 | 0.583 | 0.583 | 0.583 |
| Claude Opus 4.8 | 0.725 | 0.725 | 0.725 | 0.589 | 0.589 | 0.589 | 0.605 | 0.605 | 0.605 |
| Claude Opus 4.6 | 0.762 | 0.767 | 0.762 | 0.603 | 0.603 | 0.603 | 0.598 | 0.598 | 0.598 |
| GLM-5 | 0.731 | 0.735 | 0.732 | 0.625 | 0.643 | 0.622 | 0.632 | 0.646 | 0.630 |
| Qwen3.5-397B | 0.759 | 0.760 | 0.759 | 0.639 | 0.653 | 0.638 | 0.643 | 0.657 | 0.641 |
| Kimi K2.5 | 0.773 | 0.773 | 0.773 | 0.637 | 0.663 | 0.633 | 0.658 | 0.687 | 0.654 |
| MiniMax M2.5 | 0.758 | 0.765 | 0.757 | 0.654 | 0.719 | 0.644 | 0.640 | 0.662 | 0.637 |
| DeepSeek-V3.2 | 0.805 | 0.808 | 0.805 | 0.655 | 0.676 | 0.652 | 0.666 | 0.689 | 0.663 |
| gpt-oss-120b | 0.896 | 0.896 | 0.896 | 0.699 | 0.941 | 0.685 | 0.696 | 0.864 | 0.684 |
| Nemotron 3 Super | 0.882 | 0.991 | 0.889 | 0.695 | 0.919 | 0.680 | 0.688 | 0.823 | 0.679 |

Table 25: **Top-label calibration on usable forecasts.** ECE uses 15 equal-width bins of the top-ranked probability. The last two columns compare the mean top-ranked probability with accuracy for memory-free forecasts, where a mean probability below accuracy indicates underconfidence.

| | ECE↓ | | | Memory-free | |
|---|---|---|---|---|---|
| Model | No tools | Memory-free | Memory-on | Mean top prob. | Accuracy (%) |
| GPT-5.6 Sol | 0.038 | 0.059 | 0.055 | 0.54 | 58.3 |
| GPT-5.5 | 0.081 | 0.085 | 0.095 | 0.50 | 57.8 |
| GPT-5.4 | 0.050 | 0.052 | 0.043 | 0.49 | 53.2 |
| Claude Opus 4.8 | 0.063 | 0.054 | 0.054 | 0.50 | 52.6 |
| Claude Opus 4.6 | 0.111 | 0.070 | 0.068 | 0.49 | 52.4 |
| GLM-5 | 0.071 | 0.051 | 0.063 | 0.52 | 50.5 |
| Qwen3.5-397B | 0.099 | 0.049 | 0.045 | 0.48 | 48.7 |
| Kimi K2.5 | 0.085 | 0.051 | 0.054 | 0.49 | 50.0 |
| MiniMax M2.5 | 0.087 | 0.045 | 0.045 | 0.51 | 48.5 |
| DeepSeek-V3.2 | 0.121 | 0.056 | 0.074 | 0.52 | 48.2 |
| gpt-oss-120b | 0.204 | 0.077 | 0.094 | 0.53 | 46.3 |
| Nemotron 3 Super | 0.191 | 0.083 | 0.076 | 0.54 | 47.9 |

### C.4 Longitudinal Robustness

Table 26 gives the late-minus-early change for each model. With memory-free research, every model improves with a CI below zero. Pooled over models, the change is $-0.064$ $[-0.089, -0.041]$ without memory and $-0.060$ $[-0.085, -0.036]$ with memory, against $-0.001$ $[-0.008, 0.006]$ without tools.

Table 26: **Late-minus-early Brier by model.** Stages follow Figure 2(a), and events are weighted equally. Negative values indicate improvement. Brackets are 95% event-bootstrap CIs.

| Model | No tools | Memory-free | Memory-on |
|---|---|---|---|
| GPT-5.6 Sol | $+0.003$ $[-0.014, +0.022]$ | $-0.077$ $[-0.114, -0.042]$ | $-0.073$ $[-0.105, -0.045]$ |
| GPT-5.5 | $+0.000$ $[-0.013, +0.013]$ | $-0.060$ $[-0.096, -0.025]$ | $-0.070$ $[-0.101, -0.041]$ |
| GPT-5.4 | $-0.002$ $[-0.012, +0.010]$ | $-0.067$ $[-0.097, -0.038]$ | $-0.066$ $[-0.096, -0.039]$ |
| Claude Opus 4.8 | $-0.002$ $[-0.009, +0.006]$ | $-0.070$ $[-0.103, -0.040]$ | $-0.064$ $[-0.096, -0.034]$ |
| Claude Opus 4.6 | $+0.013$ $[+0.003, +0.024]$ | $-0.076$ $[-0.108, -0.047]$ | $-0.080$ $[-0.114, -0.047]$ |
| GLM-5 | $-0.009$ $[-0.023, +0.004]$ | $-0.069$ $[-0.107, -0.034]$ | $-0.056$ $[-0.091, -0.020]$ |
| Qwen3.5-397B | $-0.002$ $[-0.014, +0.008]$ | $-0.072$ $[-0.105, -0.042]$ | $-0.059$ $[-0.091, -0.028]$ |
| Kimi K2.5 | $+0.002$ $[-0.012, +0.017]$ | $-0.050$ $[-0.079, -0.022]$ | $-0.067$ $[-0.105, -0.031]$ |
| MiniMax M2.5 | $-0.009$ $[-0.022, +0.003]$ | $-0.047$ $[-0.081, -0.014]$ | $-0.064$ $[-0.098, -0.031]$ |
| DeepSeek-V3.2 | $+0.002$ $[-0.013, +0.016]$ | $-0.071$ $[-0.108, -0.035]$ | $-0.021$ $[-0.060, +0.019]$ |
| gpt-oss-120b | $-0.005$ $[-0.028, +0.018]$ | $-0.066$ $[-0.101, -0.033]$ | $-0.044$ $[-0.078, -0.010]$ |
| Nemotron 3 Super | $-0.001$ $[-0.028, +0.026]$ | $-0.044$ $[-0.082, -0.004]$ | $-0.052$ $[-0.081, -0.022]$ |
| All models (pooled) | $-0.001$ $[-0.008, +0.006]$ | $-0.064$ $[-0.089, -0.041]$ | $-0.060$ $[-0.085, -0.036]$ |
| Market reference | | $-0.159$ $[-0.198, -0.122]$ | |

**Event selection.** Several event filters use the realized outcome (Table 5), so the evaluation set may favor events whose market belief moves toward the outcome. Table 27 splits events by whether the market's Brier fell between the first and last scheduled dates. The market improved on 175 of 224

events. On these events, memory-free agents improve by 0.084 from the early to the late stage. On the other 49 events, agents stay roughly flat ($+0.005$, 95% CI $[-0.027, 0.038]$) while the market's Brier rises by 0.139. The pooled improvement in Figure 2(a) thus comes from events on which public information moved toward the outcome, and its size depends on how many such events the filters retain.

Table 27: **Stage-wise Brier split by whether the market's Brier fell between the first and last scheduled dates.** Events are weighted equally, and the CI is for the late-minus-early change. Group sizes count market events, and agent rows cover one event fewer in the first group, as in Figure 2(a).

| Setting | Early / middle / late | Late − early [95% CI] |
|---|---|---|
| *Market improved (175 events)* | | |
| No tools | 0.770 / 0.768 / 0.768 | $-0.002$ [$-0.011$, $+0.006$] |
| Memory-free | 0.670 / 0.640 / 0.586 | $-0.084$ [$-0.114$, $-0.055$] |
| Memory-on | 0.672 / 0.645 / 0.588 | $-0.084$ [$-0.114$, $-0.055$] |
| Market | 0.592 / 0.464 / 0.350 | $-0.242$ [$-0.285$, $-0.205$] |
| *Market did not improve (49 events)* | | |
| No tools | 0.769 / 0.770 / 0.772 | $+0.003$ [$-0.009$, $+0.016$] |
| Memory-free | 0.670 / 0.672 / 0.676 | $+0.005$ [$-0.027$, $+0.038$] |
| Memory-on | 0.669 / 0.682 / 0.695 | $+0.027$ [$-0.008$, $+0.063$] |
| Market | 0.613 / 0.674 / 0.752 | $+0.139$ [$+0.097$, $+0.186$] |

### C.5 Training Use Case

**Demonstration data.** We collect demonstrations on training events from August 2025 through February 2026, after the July 2025 release of both the teacher and the student. The teacher, Qwen3-235B-A22B-Thinking-2507, answers each of 3,965 event–date tasks from 1,023 events four times in the memory-free setting of `Forecast-Dojo`. For each task, we score the usable rollouts by a weighted combination of Brier score, accuracy, and number of tool calls, and keep the highest-scoring rollout as the demonstration. Realized outcomes therefore enter training only through this selection step, and the training and evaluation events are disjoint in time.

**Demonstration selection.** A rollout is eligible if it terminates normally, produces a parseable forecast in the required format, makes at least two tool calls, and has a Brier score. Within each task, we min–max normalize the Brier score and the number of tool calls over the eligible rollouts, reversing the Brier score so that lower values score higher, and select the rollout that maximizes

$$S(r) = 0.7\,\widetilde{B}(r) + 0.3\,a(r) + 0.5\,\widetilde{C}(r),$$

where $\widetilde{B}$ is the reversed normalized Brier score, $a$ is accuracy, and $\widetilde{C}$ is the normalized tool-call count. A criterion that is constant within a task is set to one, and ties are broken by rollout index. The tool-call term favors demonstrations that research before answering. Every task has at least one eligible rollout, so the 3,965 tasks yield 3,965 demonstrations.

**Training objective.** Each demonstration is a multi-turn conversation containing the system prompt, the question, the teacher's reasoning and tool calls, the tool outputs, and the final forecast. We fine-tune Qwen3-30B-A3B-Thinking-2507 with next-token cross-entropy on assistant tokens only, which include reasoning, tool calls, and the final forecast. System, question, and tool-output tokens are masked. We train for three epochs and evaluate the final checkpoint. Table 28 lists the optimization settings.

**Evaluation settings.** The base and fine-tuned students use the same memory-free protocol and no-belief prompt, with search, article reads, and Python, a budget of 80 tool iterations and 200 calls per forecast, temperature 0.6, top-$p = 1.0$, no top-$k$ restriction, and a 32,768-token output cap. Scores use the 230 evaluation events and the uniform fallback of Appendix B.3.

**Calibration and the SFT gain.** The base student scores worse than uniform (Brier 0.924 versus 0.771; log loss 2.74 versus 1.74). Table 29 examines how much rescaling its probabilities can

Table 28: Optimization settings for supervised fine-tuning.

| **Setting** | **Value** |
|---|---|
| Student initialization | Qwen3-30B-A3B-Thinking-2507 |
| Optimizer | Adam with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, and decoupled weight decay 0.1 |
| Learning rate | Peak $10^{-5}$ with warmup over the first 10% of training, then cosine decay to $10^{-6}$ |
| Gradient clipping | Global gradient norm 1.0 |
| Batch size | 32 conversations per update |
| Sequence handling | Complete conversations without truncation, packed into micro-batches of up to 49,152 tokens per GPU |
| Precision | BF16 parameters with FP32 gradient accumulation |
| Dropout | 0 |
| Epochs | 3 |
| Checkpoint | Final checkpoint, without early stopping |

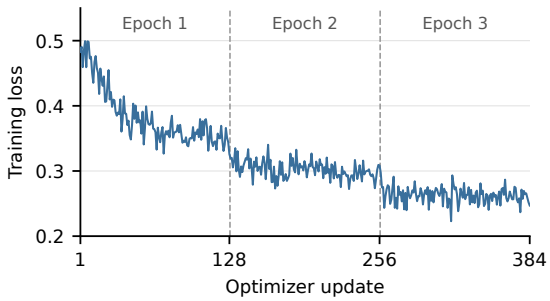


Figure 4: Training loss of supervised fine-tuning, measured as the cross-entropy on assistant tokens at each optimizer update. Dashed lines mark epoch boundaries.

improve these scores. It selects a separate shrinkage weight for each model on the evaluation set, making this an oracle diagnostic rather than a held-out calibration procedure. Base Brier falls to 0.752, close to the raw SFT score of 0.749; shrinking SFT yields 0.705. This shows that rescaling can reproduce much of the raw Brier improvement, but does not identify the fraction caused by calibration. Both selected weights preserve outcome rankings, so the accuracy improvement from 34.8% to 42.8% remains. Table 28 lists the optimization settings, and Figure 4 shows the training loss.

Table 29: Shrinking SFT-evaluation forecasts toward uniform, $p' = (1 - \lambda)p + \lambda u$. The weight $\lambda$ is chosen on the evaluation set, which gives an optimistic bound for recalibration. Shrinkage leaves the ranking of offered outcomes unchanged.

| Model | Brier | Best $\lambda$ | Brier after shrinkage | Accuracy (%) |
|---|---|---|---|---|
| Uniform reference | 0.771 | – | – | 22.9 |
| Student (base) | 0.924 | 0.75 | 0.752 | 34.8 |
| + Dojo SFT | 0.749 | 0.45 | 0.705 | 42.8 |

# D LIMITATIONS

`Forecast-Dojo` inherits the coverage of its sources. The corpus covers only English CC-News, and about two thirds of the evaluation events are sports, crypto, or politics markets, most closing within two weeks of their first forecast date. Several event filters condition on the realized outcome, so the benchmark is not a random sample of Polymarket events. The new-evidence index relies on self-reported notebook entries.

# E PROMPTS

**No-Tools 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. 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 what is known as of the forecast date, 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. Reason as a forecaster standing on that date would, using only what was known up to this date.

### 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. Weigh the evidence systematically

- Decompose the question into the few sub-questions that would most move your estimate, and work through each. Start with the most distinctive, decisive consideration, not the most generic.
- For each sub-question, lay out the relevant facts: the actors and their incentives, the rules and schedule that govern the event, the historical pattern for comparable cases, and the most recent developments as of the forecast date. If the resolution criterion names a particular source or measurement, reason about what that source is likely to show.
- Be explicit about how solid each piece of evidence is—a well-established fact, a plausible inference, or a guess—and weight it accordingly. Do not invent specifics you do not have.

#### 4. Compute what can be computed

A forecast question is a judgment problem with computable parts—settle those parts with explicit arithmetic rather than by feel. Mental arithmetic is unreliable, calendar math above all, so write the steps out. The computations that recur:

- **Extrapolation**: take 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 $\to 1 - \exp(-kt/n)$).
- **Probability algebra**: chained conditionals, at-least-one-of-$k$, scenario weighting, Bayes updates—combine the numbers step by step, never in your head.
- **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 know or can reasonably bound; a guess run through a formula 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.

### OUTPUT FORMAT

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

**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 5: No-tools forecasting-agent system prompt.

## 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 6: Memory-free forecasting-agent system prompt.

## 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 $\to 1 - \exp(-kt/n)$).
- **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 7: Memory-on forecasting-agent system prompt.

## Query Decomposition Prompt

You are an expert forecasting analyst. Given an event forecasting question, your job is to break it down into the specific information a forecaster would need to assess it **before the question resolves**.

### CRITICAL CONSTRAINT

The agent that uses your queries can only see articles published **before the forecast date** (i.e., before the question resolves). Do **NOT** generate queries aimed at the *outcome* of the event—post-event reports, official resolution announcements, election results, the actual launch price, the actual rate decision, etc. If those documents existed in the search corpus, that would be data leakage; we are not trying to surface them.

Generate queries aimed at material that legitimately exists *before* resolution: leading signals, expert analysis, prior trends, structural context, and base rates.

### WHAT YOU PRODUCE

Return a JSON object with exactly two lists of search queries:

- `evidence`: pre-resolution signals that move a forecaster's belief. Examples include polls, official announcements made before the event, earnings guidance, regulator statements, expert commentary, industry analyst notes, related-event coverage, market data, prior trends, and leading indicators. Time-anchor these queries where useful.
- `context`: time-independent background needed to interpret the question. Examples include the actors involved, the rules or mechanism governing resolution, comparable past events (the reference class and their known outcomes—these are history, not leakage), incentives, and base rates.

Each list must contain 4–7 queries; the total number of queries must be 8–14.

### RULES FOR THE QUERIES

- Use proper nouns wherever possible (entity names, locations, institution names). Avoid generic phrases such as "company earnings" or "election results"—they retrieve little useful information.
- For evidence queries, date-anchor when the question is time-sensitive (e.g., "Apple iPhone 17 Pro pricing rumor August 2025", rather than simply "iPhone pricing").
- Aim for diverse phrasings across the two lists. Do not repeat the same query in both angles. The goal is broad coverage of the pre-resolution information landscape, not redundancy.
- Each query should contain 3–10 words. Long queries match worse on dense retrieval and BM25 alike.
- Do **NOT** include the question's title verbatim. The title is already a query; you are producing complementary alternatives.
- Do **NOT** generate queries that aim at the post-resolution answer (e.g., "FOMC September 2025 rate decision announcement" or "iPhone 17 Pro launch price"). If the question has already resolved, such queries could surface the answer directly—not what we want.

### OUTPUT FORMAT

Return **ONLY** a JSON object. No prose, no Markdown fences, and no preamble.

Schema:

```
{
  "evidence": ["q1", "q2", ...],
  "context":  ["q1", "q2", ...]
}
```

Each list must contain 4–7 strings. The total number of queries across both lists should be 8–14.

### EXAMPLE

**Question:** Will the Federal Reserve cut interest rates in September 2025?
**Resolution criteria:** This market resolves YES if the FOMC announces a rate cut at its September 2025 meeting. NO otherwise.
**Output:**

```
{
  "evidence": [
    "Federal Reserve August 2025 CPI inflation report",
    "FOMC July 2025 minutes dot plot",
    "Jerome Powell Jackson Hole 2025 speech",
    "treasury yield curve August 2025",
    "Fed officials hawkish dovish remarks August 2025",
    "labor market jobs report August 2025"
  ],
  "context": [
    "FOMC voting members 2025 composition",
    "Federal Reserve dual mandate inflation employment",
    "Fed rate cut history 2024 pause cycle",
    "FOMC meeting calendar 2025 schedule"
  ]
}
```

Figure 8: Query-decomposition prompt used for corpus-side evidence retrieval.

## Evidence Sufficiency and Leakage Judge Prompt

You are an expert judge evaluating whether a small offline news corpus carries enough signal to forecast an event question. You will be shown:

1. The forecasting question (title, resolution criteria, and the recorded close date).
2. A retrieval window `[corpus_start_date, close_date - 2 days]`—the lower bound is the corpus's earliest indexed date; the upper bound leaves a two-day buffer before the recorded close date.
3. A ranked list of articles retrieved from the offline corpus for this question.

Your job is **not** to forecast the answer. Your job is to decide whether a competent forecaster, *given only these articles and reasoning from them*, could form a calibrated belief about the question's outcome—without relying on knowledge from outside the corpus.

### WHAT YOU MUST PRODUCE

Return a single JSON object with this exact shape. No fences and no prose around it:

```
{
  "verdict": "SUFFICIENT" | "PARTIAL" | "INSUFFICIENT",
  "informativeness": 0 | 1 | 2 | 3,
  "leakage_detected": true | false,
  "leakage_reason": null | "<short string>",
  "confident_lean": null | "YES" | "NO" | "<leg-label>",
  "key_hits": ["<article_id>", ...],
  "reasoning": "<one or two sentences>"
}
```

### HOW TO DECIDE EACH FIELD

#### `verdict` — three buckets

- `SUFFICIENT` — the corpus contains direct or strong indirect evidence about the question's outcome. A well-reasoned agent reading these articles would end up with a confident, well-calibrated belief (closer to 0 or 1, in the binary case) by resolution date.
- `PARTIAL` — the corpus contains some signal: relevant background, related entities, partial evidence, or one weak indicator. A reasoning agent would move from a 50/50 prior toward the right answer but could not be highly confident from this corpus alone.
- `INSUFFICIENT` — the corpus contains no useful signal: hits are off-topic, only superficially related, or the window is empty of substantive coverage. The agent would have no basis to update from the prior.

#### `informativeness` — 0–3 ordinal

A finer-grained score representing how much these articles would move a competent forecaster from a 50/50 prior:

- 0 = no movement (corpus is noise, off-topic, or irrelevant).
- 1 = small movement (some related context, no decisive evidence).
- 2 = moderate movement (multiple corroborating leading indicators, or one strong indirect signal).
- 3 = large movement (one or more articles that, while published before resolution, essentially answer the question).

Map this to `verdict` consistently:

- 0 → `INSUFFICIENT`
- 1 → `PARTIAL`
- 2 or 3 → `SUFFICIENT`

### Recency — does the evidence cover the right time period?

Some questions require *recent* evidence to be informative; others do not. Apply this test before scoring:

1. Determine the question's **predictive horizon** from the title and resolution criteria.
   - Snapshot or count over a fixed window (e.g., "# tweets Feb 17–24", "weekly views", "ranking on Feb 5", "price on close") → predictive horizon of approximately days to two weeks.
   - Trajectory needing fresh data points (e.g., "will X reach $N by date", "will polls move K points") → requires evidence within roughly the last month.
   - One-shot event whose preconditions persist (e.g., "will Apple announce X", "will Russia capture town Y") → older articles describing the actors or mechanism are acceptable.
2. Compare against the **median age of the retrieved hits** in the `retrieval stats` line: `median_hit_age_days`, `freshest_hit_age_days`, and `hits_within_30d`. The retrieval cutoff is `close_date - 2 days`.
3. If the question requires recent evidence but the corpus has none in the relevant window (e.g., `hits_within_30d == 0` for a snapshot question), the verdict is **`INSUFFICIENT`** regardless of how many topically related but stale articles exist. Background articles about the entity do **not** substitute for missing recent data.

When in doubt, a snapshot/count question with no hits in the last approximately two weeks is `INSUFFICIENT`; a one-shot event question with strong older articles can still be `SUFFICIENT`.

### `leakage_detected` — corpus integrity check

Before judging informativeness, verify that the corpus respects the retrieval window. The question's `close_date` is the market's **recorded close date**. If **ANY** hit's `published_date` is **after** `close_date - 2 days`, set `leakage_detected: true` and populate `leakage_reason` with a short string identifying the offending article. Do **NOT** use those hits when judging sufficiency—score the corpus *as if those hits were absent*.

If an article dated before close already states the resolved outcome (e.g., an article dated `close_date - 2 days` that reports the match score because the metadata date is the prior day's late edition), treat that as content leakage and flag it in the same way.

### `confident_lean` — optional directional read

If the in-window corpus gives a clear directional answer:

- For binary questions, set `confident_lean` to `"YES"` or `"NO"`.
- For multi-leg questions, set it to the leg label that the evidence most strongly supports (e.g., `"Manchester City FC"`).
- If the evidence is mixed, weak, or absent, set it to `null`.

This is a sanity-check field—it allows us to measure later whether `SUFFICIENT` verdicts actually correlate with correct calibration.

### `key_hits` — citations

Provide up to five `article_id` values from the retrieved hits that most influenced your verdict. Order them by importance, with the most decisive first. If `verdict == "INSUFFICIENT"`, this can be an empty list.

### `reasoning` — one or two sentences

Provide a short justification for the verdict. Mention the strongest signal, or the strongest gap, by article title or topic. Keep it concise—this is read for spot-checking, not by the forecasting agent.

### Important Rules

- Use **ONLY** the retrieved articles. Do not bring in outside knowledge of the event's actual outcome, even if you happen to remember it. The point of this judge is to score the *corpus*, not your prior.
- Be conservative on `SUFFICIENT`. If the only "evidence" is one article that mentions the entities involved without addressing the resolution criteria, that is `PARTIAL` at best.
- Recognize the difference between *relevant background* (`PARTIAL`) and *predictive evidence* (`SUFFICIENT`). Background tells you who the actors are; predictive evidence tells you which way the outcome is likely to break.
- When in doubt between two adjacent buckets, choose the lower one:

$$\texttt{INSUFFICIENT} < \texttt{PARTIAL} < \texttt{SUFFICIENT}.$$

### Output Format Reminder

Return **ONLY** the JSON object. No Markdown fences, no preamble, and no prose outside the JSON. The object must validate against the schema above.

Figure 9: Evidence-sufficiency and leakage judge prompt used for corpus-side filtering.