TEMPO: Temporal Enforcement via Mode-Separated Policy Optimization for Trustworthy LLM Backtesting

Zeyu Zhang*
Department of Statistics and Data Science
Northwestern University
zeyuzhang2028@u.northwestern.edu

Bradly C. Stadie
Department of Statistics and Data Science
Northwestern University
bstadie@northwestern.edu

Abstract

Backtesting large language models on historical events requires reasoning exclusively from information available before a specified cutoff date. Yet models routinely leak post-cutoff knowledge from pre-training into their reasoning, inflating apparent accuracy and undermining evaluation validity. Prompt-based constraints fail when suppressed content is causally related to the prediction, and knowledge unlearning cannot address this problem because temporal compliance is instance-specific: the same fact may be legitimate evidence for one cutoff date and a violation for another. Rather than erasing knowledge, the model must learn *temporal discipline*—selecting evidence conditioned on each instance's cutoff date. We propose TEMPO (Temporal Enforcement via Mode-separated Policy Optimization), which trains this discipline via two contributions: (1) a two-mode reward where a leakage mode drives post-cutoff claims to zero as a hard prerequisite before a performance mode optimizes task performance; and (2) a GRPO-based training pipeline that enables the model to discover temporally valid reasoning strategies. We prove that training monotonically decreases leakage, converges linearly to a small neighborhood of the leak-free optimum, and improves task performance once compliance is achieved. On three prediction tasks and two models, TEMPO reduces leakage from 2–13% to 0.6–3.7% across all conditions, with task performance improving 6–13% where strong pre-cutoff signals exist and maintained where the prediction task is inherently difficult from valid information alone.

1 Introduction

Large language models are increasingly used in prediction tasks across diverse domains, including finance [Wu et al., 2023, Fu, 2025], law [Shui et al., 2023, Deng et al., 2024], and sports [Mendes-Neves et al., 2024]. Evaluating prediction ability at scale requires *backtesting*: testing on historical events whose outcomes are already known so that accuracy can be measured immediately [Halawi et al., 2024]. Yet because these events have already occurred, their outcomes may be encoded in the model's pre-training data, allowing the model to recall actual outcomes rather than reason from evidence available at the time—inflating apparent accuracy [Paleka et al., 2026]. This *temporal knowledge leakage* differs from traditional data contamination [Carlini et al., 2021]: the model need not reproduce verbatim text but exploits general post-cutoff knowledge encoded in parametric memory [Wallat et al., 2024, Li et al., 2025]. Prompt-based constraints and standard post-training fail to suppress it [Gao et al., 2025, Kocyigit and Yildirim, 2026]. Backtesting validity therefore depends on strict temporal discipline: the model must reason as an observer at time t, with no access to later information.

*Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Existing mitigations—knowledge unlearning [Blanco-Justicia et al., 2025, Gandikota et al., 2025], chronologically consistent pretraining [He et al., 2025], temporal abstention [Zhou et al., 2026], and inference-time verification [Zhang et al., 2026b]—each address partial aspects but none trains the model to determine which evidence is temporally valid for a given cutoff date. Temporal compliance is instance-specific: the same fact may be legitimate for one cutoff and a violation for another, so the target set cannot be enumerated. No existing method trains the model to behaviorally avoid post-cutoff evidence while maintaining prediction quality.

We observe that temporal leakage is not a knowledge-storage problem but a knowledge-use problem: the model does not need to forget post-cutoff facts but must learn *temporal discipline*—conditioning its evidence selection on each instance's cutoff date so that only pre-cutoff information enters its stated evidence and reasoning. To train this discipline, we propose a reinforcement learning method: **T**emporal **E**nforcement via **M**ode-separated **P**olicy **O**ptimization (**TEMPO**). TEMPO introduces a *two-mode reward design*: a group-level mode gate partitions each training step into a *leakage mode* that drives post-cutoff claims to zero as a hard prerequisite, and a *performance mode* that optimizes task performance subject to evidence-coverage constraints once compliance is achieved. Because any leakage invalidates the evaluation, leakage elimination must be a hard gate rather than a soft penalty (Appendix B). TEMPO fine-tunes via Group Relative Policy Optimization (GRPO; Shao et al. 2024), enabling the model to discover temporally valid reasoning strategies and ground its stated reasoning in pre-cutoff fundamentals rather than recalled outcomes. We prove that training under this reward monotonically decreases leakage and, under a regularity condition, converges linearly to a small neighborhood of the leak-free optimum whose size shrinks with the step size. Once compliance is achieved, task performance improves (Theorem 1).

On three prediction tasks (stock ranking, salary estimation, legal outcome prediction) and two model generations, TEMPO reduces leakage from 2–13% to 0.6–3.7% across all conditions. Once leak-free, the training curriculum naturally encourages the model to explore reasoning paths grounded in temporally valid evidence, improving task performance 6–13% on tasks where strong pre-cutoff signals exist and maintaining it on tasks that are inherently difficult to predict from pre-cutoff information alone.

2 Related work

LLM Prediction and Backtesting. LLMs increasingly approach human-level prediction accuracy across forecasting [Halawi et al., 2024, Schoenegger et al., 2024, Karger et al., 2025], finance [Wu et al., 2023, Li et al., 2025], law [Shui et al., 2023], and sports [Mendes-Neves et al., 2024]. Backtesting—the dominant evaluation paradigm [Paleka et al., 2026]—rests on a strict temporal assumption: the model must reason exclusively from pre-cutoff information. When this assumption is violated, back-tested performance conflates genuine prediction ability with temporal leakage. Live forecasting benchmarks such as ForecastBench [Karger et al., 2025], FutureX [Zeng et al., 2025], and MIRAI [Ye et al., 2024] avoid leakage by asking about events still unresolved at evaluation time; for the same reason, they cannot test whether a method prevents it. Our setting is the complementary one, in which outcomes are already in the pretraining data (Appendix D.1).

Data Contamination and Temporal Leakage. Standard contamination mitigations—dynamic benchmarks [Karger et al., 2025], deduplication, membership-inference detection [Carlini et al., 2021, Xu et al., 2025]—target instance-level overlap. Temporal leakage is a harder sub-problem: the model encodes *general post-cutoff knowledge* through parametric memory without verbatim reproduction [Wallat et al., 2024, Zhu et al., 2025], so standard defenses and prompt-based constraints do not apply [Gao et al., 2025, Kocyigit and Yildirim, 2026]. Knowledge unlearning [Blanco-Justicia et al., 2025, Gandikota et al., 2025, Yin et al., 2024, Reisizadeh et al., 2026] cannot enumerate the instance-specific target set and correlated knowledge reconstructs erased content [Wang et al., 2025]; chronologically consistent pretraining [He et al., 2025] and temporal abstention [Zhou et al., 2026] require $O(T)$ checkpoints or teach refusal rather than prediction; inference-time verification [Zhang et al., 2026b] filters claims post-hoc but leaves the model's behavior unchanged. TEMPO is the first method that trains per-instance temporal discipline into the model's behavior.

GRPO and RL for Behavioral Compliance. GRPO [Shao et al., 2024] has become the standard RL algorithm for LLM post-training, validated at scale by DeepSeek-R1 [DeepSeek-AI, 2025] and

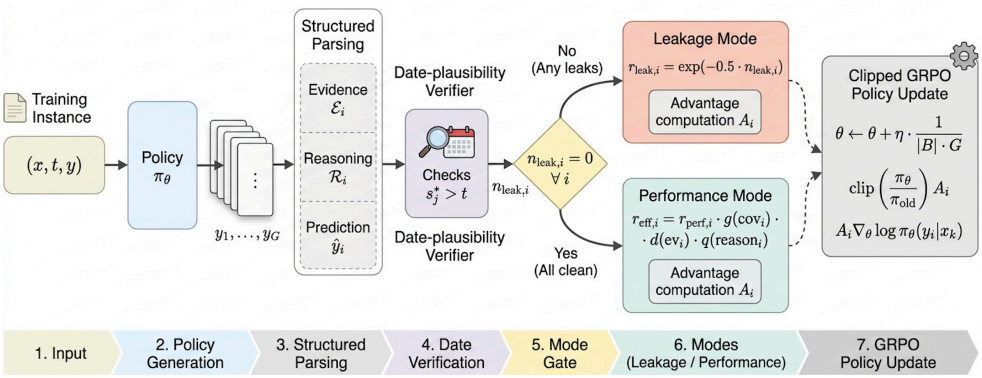


Figure 1: Overview of one TEMPO training step. For each training instance (x, t, y), the policy π_θ generates G completions, each parsed into a structured output with an evidence list, reasoning, and prediction. A date-plausibility verifier counts leaked claims per completion. The core design contribution is the *group-level mode gate*: if any completion in the group contains a leaked claim, the entire group enters *leakage mode* with an exponential penalty reward (Eq. 3); only when all completions are clean does the group enter *performance mode* to optimize task accuracy (Eq. 4). This hard separation ensures leakage elimination is a prerequisite for performance optimization. The resulting rewards drive a clipped GRPO policy update.

stabilized by DAPO [Yu et al., 2025]. When multiple objectives compete, one reward can dominate; MO-GRPO [Ichihara et al., 2025], GDPO [Liu et al., 2026], and constrained GRPO [Girgis et al., 2026] mitigate this but all permit soft trade-offs. RL for behavioral compliance—Safe RLHF [Dai et al., 2024], TruthRL [Wei et al., 2025], CaRR [Zhang et al., 2026a]—learns fixed rules (a claim is always safe or unsafe regardless of context) with soft trade-offs. Two gaps remain: (1) no method enforces a *hard constraint* where any violation invalidates the evaluation, and (2) no method handles *instance-specific rules* where compliance depends on a per-instance variable. TEMPO fills both gaps with a binary mode gate, and we prove convergence to a neighborhood of the leak-free optimum while interleaving instance-specific mode assignments within each batch.

3 Methodology

Temporal discipline—conditioning evidence selection on a per-instance cutoff date—is a behavioral constraint that must be learned through training. This section formalizes the problem (Section 3.1), introduces the two-mode reward that separates leakage elimination from performance optimization (Section 3.2), and presents the GRPO-based training algorithm with convergence guarantees (Section 3.3). Figure 1 illustrates the full pipeline.

3.1 Problem formulation

Let π_θ denote the policy (an LLM parameterized by θ). A training instance is a tuple (x, t, y): a prompt x containing the task description and case-specific input, a *cutoff date* t specifying the last calendar date whose information may legitimately inform the answer, and a ground-truth label y. We operate in the backtesting regime where y is already resolved, so both the task-performance reward and the temporal-leakage penalty are computable at every training step—a property that makes reinforcement learning directly applicable.

The policy produces a structured completion: an evidence list $\mathcal{E} = \{(c_j, s_j)\}_{j=1}^{N}$ of atomic claims with declared source dates, a reasoning segment $\mathcal{R}$ citing evidence by index, and a prediction $\hat{y}$. This structure enables per-claim temporal verification: each evidence item carries a date checkable against the cutoff.

Definition 1 (Temporal leakage)**.** *Claim* c_j leaks *relative to cutoff* t *if its effective source date* $s_j^* > t$*, where* s_j^* *is the declared date* s_j *adjusted by a date-plausibility verifier (Appendix C). The* leaked claim count *for completion* y *at cutoff* t *is*

$$c(y, t) \;=\; \big|\{j : s_j^*(y) > t\}\big|.$$

We write $n_{\text{leak},i} = c(y_i, t)$ for the i-th completion in a group. A completion is clean *if $n_{\text{leak},i} = 0$.*

Claims without a declared date are not sent to the verifier and count as not leaked, so valid but undated evidence incurs no leakage penalty. In practice this case does not arise: none of the 28,801 evidence items in the evaluation outputs lacks a declared date (Appendix F.1).

Following GRPO [Shao et al., 2024], we sample a group of G completions $\{y_1, \ldots, y_G\} \sim \pi_\theta(\cdot \mid x)$ for each training instance. Let $q(\theta; x, t) = \Pr_{y \sim \pi_\theta}[c(y, t) > 0]$ denote the per-instance leakage probability—the chance that a single sampled completion contains at least one leaked claim. The probability that *all* G completions are clean is $p_0 = (1-q)^G$, and the group leakage indicator is $L = \mathbf{1}[\exists\, i : n_{\text{leak},i} > 0]$.

TEMPO maximizes the mode-gated expected reward:

$$J_{\text{TEMPO}}(\theta) \;=\; \mathop{\mathbb{E}}_{(x,t,y)}\big[p_0 \;\mathbb{E}[r_{\text{eff}} \mid L{=}0] \;+\; (1{-}p_0) \;\mathbb{E}[r_{\text{leak}} \mid L{=}1]\big], \tag{1}$$

where r_{leak} and r_{eff} are mode-specific rewards (Section 3.2). Since $p_0 = (1-q)^G$ decays exponentially in G, the objective implicitly prioritizes leakage elimination: any positive q suppresses performance-mode contribution.

3.2 Two-mode reward design

A scalar combination $r_{\text{mix}} = \alpha\, r_{\text{leak}} + (1{-}\alpha)\, r_{\text{perf}}$ fails: performance variance dominates the z-scored advantage, and the leakage signal vanishes. A constrained objective with a fixed multiplier, $r_{\text{perf}} + \lambda, r_{\text{leak}}$, is the same reward up to scale and fails the same way. Since any leakage invalidates the evaluation, we instead make leakage elimination a hard prerequisite, enforced by a group-level mode gate. Details are in Appendix B.

Mode Selection Gate. The group enters *performance mode* if and only if every completion is clean:

$$\text{mode} = \begin{cases} \textsc{Performance} & \text{if } n_{\text{leak},i} = 0 \;\forall\, i \in \{1, \ldots, G\}, \\ \textsc{Leakage} & \text{otherwise.} \end{cases} \tag{2}$$

The gate operates at group level because GRPO computes advantages as within-group contrasts: all completions in a group must be evaluated under the same objective for the gradient to be coherent. Early in training, most groups contain at least one leaked completion and enter leakage mode; as training drives q toward zero, groups progressively transition to performance mode—an emergent curriculum characterized in Section 3.3.

Leakage Mode. When the group enters leakage mode, the reward drives down leaked evidence before competing on task performance. Each completion receives a reward that is strictly decreasing in the number of leaked claims:

$$r_{\text{leak},i} = \exp\big(-0.5 \cdot n_{\text{leak},i}\big). \tag{3}$$

The exponential form provides graded signal: completions with fewer leaked claims receive higher reward, and the gap between clean ($r_{\text{leak}} = 1.0$) and single-leak ($r_{\text{leak}} = 0.61$) completions remains large enough for clear gradient signal even when most completions are already clean.

Performance Mode. Once every completion is clean, the group optimizes task performance. The effective reward is

$$r_{\text{eff},i} = r_{\text{perf},i} \cdot g(\text{cov}_i) \cdot d(\text{ev}_i) \cdot w(\text{reason}_i), \tag{4}$$

where $r_{\text{perf},i} \in [0, 1]$ is a task-specific performance score (Section 4.3), and g, d, w are multiplicative quality gates for evidence citation, diversity, and reasoning depth (Appendix B). These gates prevent degenerate compliance strategies such as producing minimal evidence to avoid leakage.

3.3 TEMPO optimization

We optimize π_θ with GRPO [Shao et al., 2024], which z-scores rewards within each group of G completions to compute advantages without a critic network. A batch-level baseline calibrates advantages across prompts, and a per-token KL penalty (β_{KL}) against the frozen base model prevents excessive drift (masked to content tokens only). The full training loop appears in Algorithm 1 (Appendix C).

Convergence Guarantees. Standard GRPO convergence theory [Shao et al., 2024, Ghadimi and Lan, 2013] applies to single-objective optimization. TEMPO's two-mode design introduces two structural complications that require new analysis: (1) the leakage-mode reward $r_{\text{leak}} = \exp(-0.5\,c)$ is a nonlinear transformation of the leaked claim count—does maximizing it actually reduce leakage? (2) leakage-mode and performance-mode updates interleave within each batch—can performance-mode updates undo leakage progress? The following two propositions address these gaps; once both are established, convergence follows from the standard stochastic descent framework.

Let $V_c(\theta) = \mathbb{E}_{(x,t)}\,\mathbb{E}_{y\sim\pi_\theta}[c(y,t)]$ be the expected leakage cost and $V_r(\theta) = \mathbb{E}_{(x,t)}\,\mathbb{E}_{y\sim\pi_\theta}[r_{\text{eff}}(y) \mid c(y,t){=}0]$ the expected performance reward conditional on clean outputs. We assume V_c is L_c-smooth, V_r is L_r-smooth, and $\mathbb{E}[\|\hat{g}_k\|^2] \le B_g^2$.

Proposition 1 (Gradient alignment)**.** *Let $V_f(\theta) = \mathbb{E}_{(x,t)}\,\mathbb{E}_{y\sim\pi_\theta}[\exp(-0.5\,c(y,t))]$ denote the expected exponential reward. Let $\hat{g}_{\text{leak}}$ be the stochastic GRPO gradient computed from leakage-mode groups, and let $\bar{g}_{\text{leak}}(\theta) = \mathbb{E}[\hat{g}_{\text{leak}} \mid \theta]$ denote its expectation. Under the compatible approximation condition [Kakade, 2001], the following holds whenever $\mathrm{Var}_{\pi_\theta}(c) > 0$:*

$$\langle \bar{g}_{\text{leak}},\ \nabla V_c \rangle \ \le\ -\gamma(\theta)\,\|\nabla V_c\|^2, \tag{5}$$

where $\gamma(\theta) > 0$ is a policy-dependent alignment constant. That is, the GRPO gradient under the exponential leakage reward is a provable descent direction for the expected leakage cost (proof in Appendix A.1).

Proposition 2 (Bounded cross-mode contamination)**.** *Let $\alpha_k \in [0,1]$ be the fraction of groups in leakage mode at step k, and let $\hat{g}_{\text{perf}}$ be the stochastic GRPO gradient from performance-mode groups. The mixed-batch update $\theta_{k+1} = \theta_k + \eta\,(\alpha_k \hat{g}_{\text{leak}} + (1{-}\alpha_k)\hat{g}_{\text{perf}})$ satisfies*

$$\mathbb{E}[V_c(\theta_{k+1}) \mid \theta_k] \ \le\ V_c(\theta_k) - \underbrace{\big(\alpha_k\gamma(\theta_k) - (1{-}\alpha_k)B_g/\|\nabla V_c(\theta_k)\|\big)}_{\textit{effective descent rate}}\ \eta\|\nabla V_c(\theta_k)\|^2 + \tfrac{L_c\eta^2}{2}B_g^2. \tag{6}$$

The effective descent rate is positive whenever the leakage gradient (weighted by α_k) dominates the worst-case cross-mode interference from performance-mode updates (proof in Appendix A.2).

With both gaps filled, convergence follows from the standard stochastic optimization framework:

Theorem 1 (TEMPO convergence)**.** *Under Propositions 1–2, let $\gamma_0 = \inf\{\gamma(\theta) : V_c(\theta) \ge \epsilon_0\} > 0$ for a target threshold ϵ_0, and suppose at least a fraction $\alpha_{\min} > 0$ of groups are in leakage mode whenever $V_c(\theta_k) \ge \epsilon_0$. For step size $\eta \le \gamma_0/(L_c B_g^2)$: (i) leakage decreases at the standard nonconvex rate $\min_{k\le T}\|\nabla V_c\|^2 = O(1/\sqrt{T})$; (ii) performance improves during performance-mode updates by the standard ascent lemma; (iii) under the Polyak–Łojasiewicz condition $\|\nabla V_c\|^2 \ge 2\mu\,V_c$, leakage converges linearly:*

$$\mathbb{E}[V_c(\theta_k)] \ \le\ (1 - \alpha_{\min}\eta\gamma_0\mu)^k\,V_c(\theta_0) \ +\ \frac{L_c\eta B_g^2}{2\alpha_{\min}\gamma_0\mu}. \tag{7}$$

Complete derivations appear in Appendix A. A practical consequence of Theorem 1 is that training does not require manual scheduling of when to switch from leakage elimination to performance optimization—the mode gate handles this automatically:

Corollary 1 (Two-phase curriculum)**.** *Recall that the group-clean probability is $\bar{p}_0(\theta) = \mathbb{E}_{(x,t)}[(1{-}q)^G]$. Since $\bar{p}_0(\theta) \ge (1 - V_c(\theta))^G$ by Markov's inequality, the linear convergence of V_c in Theorem 1(iii) drives $\bar{p}_0$ toward 1, up to the residual in Eq. (7): as leakage decreases, progressively more groups become fully clean and enter performance mode, so training automatically transitions from a leakage-dominated phase to a performance-optimization phase.*

4 Experimental setup

We evaluate TEMPO on three prediction tasks spanning financial markets, professional sports contracts, and federal law, comparing against inference-time baselines under an identical evaluation pipeline.

Table 1: Dataset statistics across three prediction tasks. All training and evaluation cutoff dates predate March 2025, ensuring both models could have encountered the outcomes during pre-training. Control instances postdate February 2026, after both models' knowledge cutoffs, establishing a zero-knowledge performance floor.

Task	Train	Eval	Control	Train period	Eval period
Stock ranking	1,200	100	100	2020-07 – 2024-06	2019-12 – 2020-06
Salary prediction	1,066	137	26	2017-12 – 2024-12	2019-06 – 2024-07
Legal case	894	100	90	2014-10 – 2024-12	2015-01 – 2024-07

4.1 Tasks and data

The three tasks are chosen to cover a spectrum of leakage sensitivities. Stock ranking is the most leakage-sensitive: market returns are highly volatile and difficult to predict from pre-cutoff fundamentals alone, so models with access to post-cutoff knowledge gain a disproportionate advantage. Salary prediction is intermediate: contract values depend on recent player performance and market conditions that may fall within days or weeks of the cutoff, making up-to-date knowledge informative but not as decisive as in stock markets. Legal case prediction is least susceptible: judicial outcomes rely primarily on precedent, statutory reasoning, and procedural history available well before the decision, and similar prior cases provide strong valid reference points. Each instance carries a cutoff date equal to the date its oracle result became publicly determinable; only information available on or before this date is considered temporally valid. Training and evaluation instances never overlap: stock evaluation uses a temporally disjoint window, salary evaluation uses a held-out sport (NBA, absent from training), and legal evaluation is drawn by stratified sampling with verified zero case overlap.

Stock Ranking. Rank five same-sector companies by six-month total return; evaluation covers COVID-19 onset (Dec 2019–Jun 2020).

Salary Prediction. Predict the average annual contract value (AAV) for an athlete who changed teams; training spans NFL/MLB/NHL, evaluation uses NBA.

Legal Case Prediction. Predict the probability the petitioner prevails given case facts and procedural history; training combines Supreme Court and federal appellate cases (2014–2024).

A separate control set per task (answers postdating Feb 2026) establishes a zero-knowledge performance floor. Table 1 summarizes statistics; dataset details are in Appendix D.

4.2 Models and methods

We train two mixture-of-experts models from the Qwen3 family [Yang et al., 2025]: Qwen3-30B-A3B (30B total, 3B active per token, knowledge cutoff approximately March 2025) and Qwen3.5-35B-A3B (35B total, 3B active, cutoff approximately February 2026). Both share the same architecture family but differ in scale and pre-training vintage, allowing us to test whether temporal discipline generalizes across model generations. All methods receive identical data, prompts, and evaluation pipelines.[2]

We compare against three inference-time baselines that form a progression of intervention levels, testing whether each additional mechanism is sufficient to suppress temporal leakage:

Temporal Hint (Base). The unmodified model receives a structured prompt with a temporal constraint anchored to the instance cutoff date. This tests whether instruction alone suffices, without retrieval or verification.

RAG. The base prompt is augmented with pre-cutoff evidence retrieved via a search API with a hard date filter set to the instance cutoff. This tests whether providing external valid evidence reduces reliance on parametric memory.

[2]Code and data are available at `https://github.com/nu-snail/TEMPO`.

TimeSPEC. An inference-time leakage prevention approach [Zhang et al., 2026b] that extends RAG with a post-generation verify–regenerate–aggregate pipeline. This tests whether inference-time claim filtering achieves compliance.

TEMPO differs fundamentally: rather than filtering at inference time, it trains temporal discipline into the model's parameters (Section 3), requiring only a single forward pass at evaluation. Training takes 7–22 hours per model-task combination on the Tinker API (Table 5); evaluation uses greedy decoding, and we release all evaluation outputs and code. Implementation details and hyperparameters for all methods appear in Appendix C; the TimeSPEC pipeline is documented in Appendix E.3.

4.3 Evaluation protocol

All methods are scored by an identical pipeline to ensure fair comparison. We report four complementary metrics (formal definitions in Appendix F).

Overall Leakage Rate (OLR). $\mathrm{OLR} = N^{-1} \sum_j n_{\mathrm{leak},j}/n_{\mathrm{total},j}$, the fraction of claims whose effective source date exceeds the cutoff. A date-plausibility verifier adjusts declared dates for temporal consistency, mirroring the training signal (Section 3.2). Lower is better; 0% means every claim is grounded in pre-cutoff information. Control experiments on post-cutoff instances (where the model lacks parametric knowledge of the outcome) confirm the verifier produces negligible false positives (0.0–0.7% OLR; Appendix G.5).

Decision-Critical Leakage Rate (DCLR). OLR weights every leaked claim equally, so a method could keep few but decisive leaks. The Shapley-weighted DCLR [Zhang et al., 2026b] weights each claim by its influence on the prediction: for an instance with claims $c_1, \ldots, c_n$, Shapley values ϕ_i, and leakage indicators $\ell(c_i) \in \{0, 1\}$, $\mathrm{DCLR} = \sum_{i=1}^{n} |\phi_i|\, \ell(c_i) / \sum_{i=1}^{n} |\phi_i|$, averaged over instances. DCLR above OLR means leakage is concentrated in influential claims.

Task Performance (Perf). Task-specific performance mapped to $[0, 1]$: Spearman rank correlation $(\rho{+}1)/2$ for stock, relative accuracy $\max(0, 1{-}|\hat{y}{-}y|/y)$ for salary, and Brier score complement $1{-}(\hat{p}{-}y)^2$ for legal, where $\hat{y}$ (or $\hat{p}$) denotes the model's prediction and y the ground truth. Performance is interpretable only alongside OLR: high performance at high OLR may reflect contamination rather than genuine prediction ability. Control experiments on post-cutoff instances establish the zero-knowledge floor.

Coverage (Cov). $\mathrm{Cov} = |\{e_k : e_k \text{ cited in reasoning}\}|/|\{e_k\}|$, the fraction of evidence items cited at least once in the reasoning; high coverage reflects a coherent evidence–reasoning–prediction chain.

5 Results and analysis

Table 2 reports the full results, and Figure 2 plots leakage against performance. TEMPO reduces OLR from 2–13% (Temporal Hint) to 0.6–3.7% across all six conditions, while keeping performance comparable to the other leakage-controlled methods and above the zero-knowledge control. Prompt constraints (Temporal Hint) and date-filtered retrieval (RAG) do not solve leakage, and RAG often makes it worse by retrieving post-cutoff content close to the prediction target. TEMPO is also the only method whose DCLR is lower than its OLR in all six conditions, so its remaining leaks have little influence on its predictions. The baselines' DCLR usually matches or exceeds their OLR, most clearly for RAG on Qwen3.5-35B (9.5% vs. 6.8% on stock). Appendices G–I give extended analysis: instance-level leakage and coverage distributions, inference cost, and case studies.

5.1 Per-task comparison

Stock Ranking. Temporal Hint produces 5.4%/12.7% OLR as the model cites post-cutoff closing prices directly. RAG amplifies leakage further (9.8% for Qwen3-30B): even with a hard date filter, financial news discussing post-cutoff dynamics passes through, demonstrating that search cutoffs alone do not prevent contamination. Both TimeSPEC (1.7%/1.5%) and TEMPO (2.5%/2.4%) suppress leakage effectively, but differ sharply in coverage: TimeSPEC's post-hoc filtering breaks the evidence–reasoning chain (73.8%/75.5%), while TEMPO generates clean evidence and reasoning together in one pass (99.3%/95.7%). For Qwen3-30B, TEMPO achieves the highest accuracy among all methods (0.625) while maintaining near-zero leakage. For Qwen3.5-35B, Temporal Hint's higher

Table 2: Evaluation results across three temporal prediction tasks and two models. OLR: overall leakage rate (fraction of claims citing post-cutoff information). DCLR: Shapley-weighted decision-critical leakage rate [Zhang et al., 2026b], which weights each leaked claim by its influence on the prediction. Perf: task-specific prediction accuracy (mapped Spearman ρ for stock, $1-\mathrm{RE}$ for salary, $1-\mathrm{BS}$ for legal; all in $[0, 1]$). Cov: fraction of evidence items cited in reasoning. Temporal Hint, RAG, and TimeSPEC are inference-time methods without parameter updates; TEMPO fine-tunes the model via GRPO. Best value per model block in **bold**. †Control evaluates Temporal Hint on post-cutoff instances where the model lacks parametric knowledge of the answer.

	Stock Ranking				Salary Prediction				Legal Outcome			
Method	OLR↓	DCLR↓	Perf↑	Cov↑	OLR↓	DCLR↓	Perf↑	Cov↑	OLR↓	DCLR↓	Perf↑	Cov↑
Qwen3-30B-A3B												
Temporal Hint	5.4%	5.5%	0.584	94.6%	8.1%	7.5%	0.607	83.0%	2.2%	2.5%	0.757	61.7%
RAG	9.8%	11.5%	0.578	83.2%	18.2%	15.3%	**0.751**	76.0%	3.2%	2.9%	**0.875**	**85.8%**
TimeSPEC	**1.7%**	**1.0%**	0.517	73.8%	**2.6%**	**2.4%**	0.638	71.2%	**0.1%**	**0.2%**	0.824	65.2%
TEMPO (Ours)	2.5%	1.8%	**0.625**	**99.3%**	3.7%	3.2%	0.658	**91.2%**	0.7%	0.3%	0.804	75.7%
Control†	*0.7%*	*0.7%*	*0.476*	*99.3%*	*0.6%*	*0.7%*	*0.454*	*79.5%*	*0.0%*	*0.0%*	*0.707*	*66.2%*
Qwen3.5-35B-A3B												
Temporal Hint	12.7%	13.5%	**0.690**	88.4%	11.9%	12.2%	0.558	80.0%	2.1%	2.1%	0.675	69.0%
RAG	6.8%	9.5%	0.603	90.5%	3.7%	5.0%	**0.778**	84.9%	4.0%	6.0%	**0.890**	**81.4%**
TimeSPEC	**1.5%**	**1.6%**	0.590	75.5%	**0.7%**	**0.5%**	0.656	75.0%	0.8%	0.8%	0.811	73.0%
TEMPO (Ours)	2.4%	2.0%	0.630	**95.7%**	1.7%	1.3%	0.631	**92.0%**	**0.6%**	**0.5%**	0.741	74.1%
Control†	*0.2%*	*0.3%*	*0.497*	*92.5%*	*0.4%*	*0.4%*	*0.549*	*86.2%*	*0.0%*	*0.0%*	*0.647*	*74.3%*

raw accuracy (0.690) comes at 12.7% OLR—reflecting post-cutoff citations rather than genuine prediction. Across both models, TEMPO exceeds the Control floor (0.476/0.497) by 27–31%, confirming genuine prediction from valid pre-cutoff information.

Salary Prediction. RAG achieves the highest accuracy (0.751/0.778) but at 18.2% OLR for Qwen3-30B, as the retriever surfaces actual contract details published after the cutoff. Both TimeSPEC and TEMPO suppress leakage to 0.7–3.7% and achieve comparable honest prediction quality (TimeSPEC 0.638/0.656; TEMPO 0.658/0.631), with both substantially exceeding Control (0.454/0.549). The key difference is that TEMPO improves accuracy over Temporal Hint (0.658 vs. 0.607 for 30B; 0.631 vs. 0.558 for 35B) while suppressing leakage—its performance-mode objective actively trains better reasoning from valid pre-cutoff information—whereas TimeSPEC has no such mechanism. The simultaneous leakage reduction and accuracy gain confirms the two-phase curriculum.

Legal Outcome Prediction. All methods achieve low OLR (0.1–4.0%) because legal reasoning draws on precedent available before decisions; the mode gate operates correctly even when baseline leakage is already near zero. Both TEMPO and TimeSPEC substantially exceed Control (0.707/0.647), confirming genuine prediction from valid precedent. TimeSPEC achieves higher accuracy on this task (0.824/0.811 vs. TEMPO's 0.804/0.741), with a larger gap for Qwen3.5-35B. TEMPO's advantage is coverage (75.7%/74.1% vs. 65.2%/73.0%) and inference cost: it preserves reasoning coherence in a single forward pass without external API calls.

5.2 Temporal discipline and prediction quality

Figure 2 plots each method in the leakage–performance space. Baselines occupy the upper region where leakage is uncontrolled and apparent performance is unreliable. TimeSPEC moves down—leakage suppressed—but without a performance-improvement objective it tends left, and coverage loss from filtering further degrades reasoning quality. TEMPO moves down *and* stays right or improves: its performance-mode objective trains the model to reason better from valid information while preserving 91–99% coverage. Across all six conditions, TEMPO suppresses leakage to 0.6–3.7% while consistently exceeding Control by 14–45%, confirming genuine prediction from valid pre-cutoff evidence. Where baselines appear to outperform TEMPO in raw accuracy (RAG on salary/legal; Temporal Hint on stock 35B), the gap invariably accompanies significantly higher OLR—this "performance" reflects contaminated reasoning and is not trustworthy for backtesting.

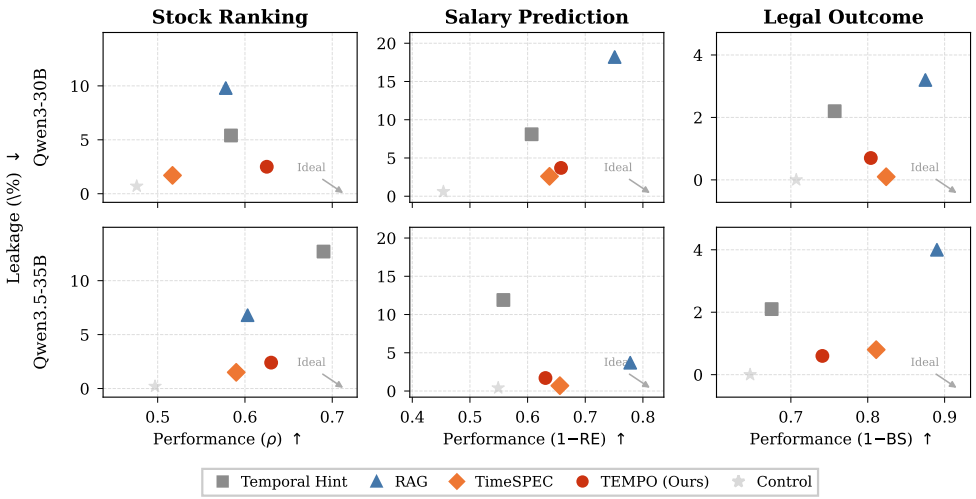


Figure 2: Leakage–performance trade-off across tasks (columns) and models (rows). The ideal region is the lower right (low leakage, high performance). TEMPO (red) consistently occupies this region. Stars denote control experiments on post-cutoff instances.

Table 3: Two-mode transition over training. Performance-mode fraction: proportion of GRPO groups where all completions are leak-free. The mode gate transitions automatically, confirming Corollary 1.

Model	Task	Initial (%)	Final (%)	Step >50%	Total steps
Qwen3-30B	Stock	62.5	100.0	0	213
Qwen3-30B	Salary	50.0	98.8	2	211
Qwen3.5-35B	Stock	12.5	96.2	11	213
Qwen3.5-35B	Salary	0.0	81.2	25	211

On coverage, TEMPO outperforms TimeSPEC by 17–26 percentage points on stock and salary; the gap narrows on legal where fewer claims are filtered. This pattern is what Theorem 1 guarantees: part (i) drives leakage down, part (ii) improves performance conditional on compliance. Leakage plateaus above zero, as part (iii) predicts: training converges to a neighborhood of the leak-free optimum, and measured leakage cannot fall below the verifier's false-positive rate (0.0–0.7% OLR on the control set; Appendix A.5).

5.3 Two-phase curriculum validation

Table 3 validates the emergent two-phase curriculum predicted by Corollary 1 by tracking the fraction of GRPO groups assigned to performance mode over training. On the highest-leakage condition (Qwen3.5-35B salary), training begins with 0% of groups in performance mode—every group contains at least one leaked completion—and crosses 50% at step 25 of 211, reaching 81.2% by training end. Qwen3-30B converges faster, reaching 100% performance mode on stock and 98.8% on salary, because lower baseline leakage means fewer groups start in leakage mode. No manual phase scheduling is required. Legal prediction has near-zero baseline leakage (2.1–2.2% OLR), so the transition effect is less pronounced; the design is validated on tasks where substantial leakage exists to be eliminated.

5.4 Reward-design ablation

Table 4 trains each alternative under the same model, data, budget, and hyperparameters, changing only the reward. The scalar mixture stalls at 4.0% OLR, above TEMPO's 2.5% at equal performance, as predicted in Section 3.2. Performance-only GRPO raises leakage above the untrained model (6.5% vs. 5.4%): its higher performance comes from leaked evidence. Leakage-only training reaches

Table 4: Reward-design ablation on the stock task (Qwen3-30B-A3B). Each alternative is trained with the same model, data, budget, and hyperparameters as TEMPO; only the reward differs. Untrained is Temporal Hint without fine-tuning. Parse: fraction of completions in the required output format.

Reward design	OLR↓	Perf↑	Cov↑	Parse↑
Untrained	5.4%	0.584	94.6%	98%
Scalar mixture (α=0.5)	4.0%	0.621	99.5%	98%
Leakage-only	0.9%	0.497	70.5%	95%
Performance-only	6.5%	0.665	96.4%	99%
Two-mode (TEMPO)	2.5%	0.625	99.3%	100%

the lowest OLR but drops performance to 0.497 and coverage to 70.5%. Only the two-mode gate achieves low leakage without losing performance.

6 Conclusion

Temporal knowledge leakage undermines backtesting validity, yet no existing method trains the model to behaviorally condition its evidence selection on each instance's cutoff date. Therefore, we propose TEMPO, which trains this discipline via a two-mode GRPO reward: a leakage mode drives post-cutoff claims to zero as a hard prerequisite, and a performance mode optimizes prediction quality from valid evidence once compliance is achieved. Across three prediction tasks and two model generations, TEMPO reduces leakage to levels comparable to the best inference-time baseline, while maintaining or improving task performance. Our analysis shows that leakage decreases monotonically during training and converges linearly to a small neighborhood of the leak-free optimum under a regularity condition. Once outputs are compliant, task performance improves, so training shifts from leakage elimination to performance optimization on its own, without a manual schedule. These results show that temporal discipline can be trained into the model itself: it learns to select evidence by each instance's cutoff and produces an auditable pre-cutoff evidence trail. Because no filtering is needed at inference, TEMPO enables trustworthy LLM backtesting at the cost of a single forward pass.

Limitations. TEMPO constrains only the model's outputs: each prediction must be supported by dated, pre-cutoff evidence sufficient to reproduce it. It cannot verify whether the model also relies on memorized outcomes internally. Our memorization probe and reconstruction test (Appendix H) find no evidence of such reliance, but a direct test requires interpretability tools and is left to future work. TEMPO also requires open-weight models and task-specific temporal annotations, and we evaluate it on two Qwen models only. Finally, training and evaluation share one verifier, which the model could in principle learn to evade. Independent judges, a human-labeled gold set, and a manual audit of clean-judged claims show that no such evasion exists (Appendix H.1).

Acknowledgments and Disclosure of Funding

This work received no third-party funding; all training and API costs were covered by the authors. The authors declare no competing interests.

References

Alekh Agarwal, Sham M. Kakade, Jason D. Lee, and Gaurav Mahajan. Optimality and approximation with policy gradient methods in Markov decision processes. In *Conference on Learning Theory (COLT)*, pages 64–66, 2020.

Alberto Blanco-Justicia, Najeeb Jebreel, Benet Manzanares, David Sánchez, Josep Domingo-Ferrer, Guillem Collell, and Kuan Eeik Tan. Digital forgetting in large language models: A survey of unlearning methods. *Artificial Intelligence Review*, 58(90), 2025.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, et al. Extracting training data from large language models. In *USENIX Security Symposium*, 2021.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe RLHF: Safe reinforcement learning from human feedback. In *International Conference on Learning Representations (ICLR)*, 2024.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *Nature*, 645:633–638, 2025.

Chenlong Deng, Kelong Mao, Yuyao Zhang, and Zhicheng Dou. Enabling discriminative reasoning in LLMs for legal judgment prediction. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 784–796, 2024.

Weilong Fu. The new quant: A survey of large language models in financial prediction and trading. *arXiv preprint arXiv:2510.05533*, 2025.

Rohit Gandikota, Sheridan Feucht, Samuel Marks, and David Bau. Erasing conceptual knowledge from language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Xin Gao, Ruiyi Zhang, Daniel Du, Saurabh Mahindre, Sai Ashish Somayajula, and Pengtao Xie. Can prompts rewind time for LLMs? Evaluating the effectiveness of prompted knowledge cutoffs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 20777–20788, 2025.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Roger Girgis, Rodrigue de Schaetzen, Luke Rowe, Azalée Robitaille, Christopher Pal, and Liam Paull. Constrained group relative policy optimization. *arXiv preprint arXiv:2602.05863*, 2026.

Danny Halawi, Fred Zhang, Chen Yueh-Han, and Jacob Steinhardt. Approaching human-level forecasting with language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Songrun He, Linying Lv, Asaf Manela, and Jimmy Wu. Chronologically consistent large language models. *arXiv preprint arXiv:2502.21206*, 2025.

Yuki Ichihara, Yuu Jinnai, Tetsuro Morimura, Mitsuki Sakamoto, Ryota Mitsuhashi, and Eiji Uchibe. MO-GRPO: Mitigating reward hacking of group relative policy optimization on multi-objective problems. *arXiv preprint arXiv:2509.22047*, 2025.

Sham M. Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

Ezra Karger, Houtan Bastani, Chen Yueh-Han, Zachary Jacobs, Danny Halawi, Fred Zhang, and Philip E. Tetlock. ForecastBench: A dynamic benchmark of AI forecasting capabilities. In *International Conference on Learning Representations (ICLR)*, 2025.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 795–811, 2016.

Muhammed Yusuf Kocyigit and Caglar Yildirim. The impact of post-training on data contamination. *arXiv preprint arXiv:2601.06103*, 2026.

Xiangyu Li, Yawen Zeng, Xiaofen Xing, Jin Xu, and Xiangmin Xu. Profit mirage: Revisiting information leakage in LLM-based financial agents. *arXiv preprint arXiv:2510.07920*, 2025.

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

Tiago Mendes-Neves, Luís Meireles, and João Mendes-Moreira. Estimating player performance in different contexts using fine-tuned large events models. *arXiv preprint arXiv:2402.06815*, 2024.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer, 2004.

Daniel Paleka, Shashwat Goel, Jonas Geiping, and Florian Tramèr. Pitfalls in evaluating language model forecasters. In *International Conference on Learning Representations (ICLR)*, 2026.

Hadi Reisizadeh, Jinghan Jia, Zhiqi Bu, Bhanukiran Vinzamuri, Anil Ramakrishna, Kai-Wei Chang, Volkan Cevher, Sijia Liu, and Mingyi Hong. BLUR: A bi-level optimization approach for LLM unlearning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 7043–7058, 2026.

Philipp Schoenegger, Indre Tuminauskaite, Peter S. Park, Rafael Valdece Sousa Bastos, and Philip E. Tetlock. Wisdom of the silicon crowd: LLM ensemble prediction capabilities rival human crowd accuracy. *Science Advances*, 10(45):eadp1528, 2024. doi: 10.1126/sciadv.adp1528.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1889–1897, 2015.

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

Ruihao Shui, Yixin Cao, Xiang Wang, and Tat-Seng Chua. A comprehensive evaluation of large language models on legal judgment prediction. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7337–7348, 2023.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, 2000.

Jonas Wallat, Adam Jatowt, and Avishek Anand. Temporal blind spots in large language models. In *Proceedings of the ACM Web Search and Data Mining Conference (WSDM)*, pages 683–692, 2024.

Wenyu Wang, Mengqi Zhang, Xiaotian Ye, Zhaochun Ren, Pengjie Ren, and Zhumin Chen. UIPE: Enhancing LLM unlearning by removing knowledge related to forgetting targets. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 25212–25227, 2025.

Zhepei Wei, Xiao Yang, Kai Sun, Jiaqi Wang, Rulin Shao, Sean Chen, Mohammad Kachuee, Teja Gollapudi, Tony Liao, Nicolas Scheffer, Rakesh Wanga, Anuj Kumar, Yu Meng, Wen-tau Yih, and Xin Luna Dong. TruthRL: Incentivizing truthful LLMs via reinforcement learning. *arXiv preprint arXiv:2509.25760*, 2025.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. BloombergGPT: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023.

Yuxing Xu, Yunpu Liu, Ling Li, Yutong Guo, Jia Wang, and Tongtong Jia. A survey on data contamination for large language models. *arXiv preprint arXiv:2501.11880*, 2025.

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

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

Xunjian Yin, Jin Jiang, Liming Yang, and Xiaojun Wan. History matters: Temporal knowledge editing in large language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19413–19421, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Zhiyuan Zeng, Jiashuo Liu, Siyuan Chen, Tianci He, Yali Liao, Yixiao Tian, Jinpeng Wang, Zaiyuan Wang, Yang Yang, Lingyue Yin, Mingren Yin, Zhenwei Zhu, Tianle Cai, Zehui Chen, Jiecao Chen, Yantao Du, Xiang Gao, Jiacheng Guo, Liang Hu, Jianpeng Jiao, Xiangsheng Li, Jingkai Liu, Shuang Ni, Zhoufutu Wen, Ge Zhang, Kaiyuan Zhang, Xin Zhou, Jose Blanchet, Xipeng Qiu, Mengdi Wang, and Wenhao Huang. Futurex: An advanced live benchmark for llm agents in future prediction, 2025. URL https://arxiv.org/abs/2508.11987.

Jiajie Zhang, Xin Lv, Ling Feng, Lei Hou, and Juanzi Li. Chaining the evidence: Robust reinforcement learning for deep search agents with citation-aware rubric rewards. *arXiv preprint arXiv:2601.06021*, 2026a.

Zeyu Zhang, Ryan Chen, and Bradly C. Stadie. All leaks count, some count more: Interpretable temporal contamination detection in llm backtesting, 2026b. URL https://arxiv.org/abs/2602.17234.

Xinyu Zhou, Chang Jin, Carsten Eickhoff, Zhijiang Guo, and Seyed Ali Bahrainian. When silence is golden: Can LLMs learn to abstain in temporal QA and beyond? In *International Conference on Learning Representations (ICLR)*, 2026.

Zhiyuan Zhu, Yusheng Liao, Zhe Chen, Yuhao Wang, Yunfeng Guan, Yanfeng Wang, and Yu Wang. EvolveBench: A comprehensive benchmark for assessing temporal awareness in LLMs on evolving knowledge. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 16173–16188, 2025.

Appendix overview. Appendix A gives the convergence proofs. Appendices B and C detail the reward design and the training implementation, including compute. Appendices D–F describe the datasets, baselines, and evaluation metrics. Appendix G reports extended results, and Appendix H tests the validity of the leakage measurement: verifier accuracy, independent judges, memorization, and reconstruction. Appendices I and J give case studies and prompts.

A Convergence analysis

This section provides full proofs for Propositions 1–2, Theorem 1, and Corollary 1 stated in Section 3.3.

Setup. Let $V_c(\theta) = \mathbb{E}_{(x,t)}\mathbb{E}_{y\sim\pi_\theta}[c(y,t)]$ denote the expected leakage cost, $V_f(\theta) = \mathbb{E}_{(x,t)}\mathbb{E}_{y\sim\pi_\theta}[\exp(-0.5\,c(y,t))]$ the expected exponential reward, and $V_r(\theta) = \mathbb{E}_{(x,t)}\mathbb{E}_{y\sim\pi_\theta}[r_{\text{eff}}(y) \mid c(y,t){=}0]$ the expected performance reward conditional on clean outputs. Write $s(y) = \nabla_\theta \log \pi_\theta(y \mid x)$ for the score function and e_y for the standard basis vector at position y. The smoothness and bounded-moment conditions hold for LoRA-parameterized policies with bounded rewards ($r_{\text{leak}} \in (0,1]$, $r_{\text{eff}} \in [0,1]$) and finite sequence length, which bound the score function norms and ensure Lipschitz gradients.

A.1 Proof of Proposition 1: Gradient Alignment

Policy gradient identification. By the policy gradient theorem [Sutton et al., 2000], the REINFORCE gradient of V_f with baseline $\bar{f}$ is

$$\nabla_\theta V_f(\theta) = \mathbb{E}_{\pi_\theta}\big[(\exp(-0.5\,c(y,t)) - \bar{f})\,s(y)\big],$$

where $\bar{f} = \mathbb{E}_{\pi_\theta}[\exp(-0.5\,c)]$. The GRPO z-scoring divides by the within-group sample standard deviation; in expectation over groups of size G, this converges to the population standard deviation $\sigma_f = \sqrt{\text{Var}_{\pi_\theta}(\exp(-0.5\,c))}$, so the expected leakage-mode gradient satisfies $\bar{g}_{\text{leak}}(\theta) = \nabla_\theta V_f(\theta)/\sigma_f + O(1/G)$. Since $\sigma_f > 0$ whenever $\text{Var}_{\pi_\theta}(c) > 0$, this is a positive rescaling of ∇V_f (up to a vanishing $O(1/G)$ bias) that does not affect the sign of any inner product for G sufficiently large; the rescaling factor is absorbed into the alignment constant $\gamma(\theta)$. Similarly, $\nabla V_c = \mathbb{E}_{\pi_\theta}[(c(y,t) - \bar{c})\,s(y)]$.

Overparameterized regime. We establish that the Euclidean inner product $\langle \nabla V_f, \nabla V_c \rangle$ is strictly negative under the compatible approximation condition. In the overparameterized regime [Agarwal et al., 2020], the policy class can represent any distribution over outputs, and the score function takes the form $s(y) = e_y - \pi_\theta$ (one logit per output). This condition is satisfied when the LoRA rank far exceeds the effective dimension of the leakage-count distribution; with rank 32 and typically 5–10 distinct leakage levels per instance, the score vectors span all centered functions of the leakage count. Computing the inner product directly from the score structure, the cross-terms vanish (since $\sum_y \pi(y)(f(y) - \bar{f}) = 0$ and $\sum_y \pi(y)(c(y) - \bar{c}) = 0$), yielding

$$\langle \nabla V_f,\, \nabla V_c \rangle \;=\; \sum_y \pi_\theta(y)^2 \,\big(\exp(-0.5\,c(y)) - \bar{f}\big)\big(c(y) - \bar{c}\big). \tag{8}$$

The sum in (8) equals the covariance $\text{Cov}_Q(\exp(-0.5\,c),\, c)$ under the tilted distribution $Q(y) \propto \pi_\theta(y)^2$. Since $\exp(-0.5\,c)$ is strictly decreasing in c and Q has the same support as π_θ, the Chebyshev covariance inequality gives $\text{Cov}_Q(\exp(-0.5\,c),\, c) < 0$ whenever c takes at least two distinct values under π_θ—precisely the regime where the policy sometimes leaks.

For the LoRA-parameterized LLM, the effective output space for the leakage count c is low-dimensional (typically a handful of distinct integer values $0, 1, 2, \ldots$), while the LoRA rank r is much larger. In this overparameterized regime, the policy can independently adjust the probability of each leakage level, and the softmax analysis applies [Agarwal et al., 2020].

Combining with the policy gradient identification, we obtain $\langle \bar{g}_{\text{leak}}, \nabla V_c \rangle = \langle \nabla V_f, \nabla V_c \rangle < 0$. The alignment constant $\gamma(\theta) = |\langle \nabla V_f, \nabla V_c \rangle| / (\sigma_f \,\|\nabla V_c\|^2)$ is positive whenever $\text{Var}_{\pi_\theta}(c) > 0$, and is bounded below on any sublevel set $\{V_c(\theta) \geq \epsilon\}$ for $\epsilon > 0$. To see this, note that $V_c(\theta) =$

$\mathbb{E}_{(x,t)}\mathbb{E}_y[c(y,t)] \geq \epsilon > 0$ requires some instance (x,t) where $\mathbb{E}_y[c] > 0$; since $c \geq 0$ is integer-valued and π_θ has full support over outputs, this implies $\mathrm{Var}_{\pi_\theta}(c \mid x,t) > 0$ for that instance, giving a strictly negative inner product contribution.

Extension to clipping. The clipped importance ratio $\min(\pi_\theta/\pi_{\mathrm{old}}, 1+\epsilon)$ defines a surrogate objective that matches the true policy gradient to first order at $\theta = \theta_{\mathrm{old}}$ [Schulman et al., 2015]. Within the trust region $D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{old}}) \leq \delta$, the clipped gradient has the same sign of inner product with ∇V_c as the unclipped version, up to an $O(\delta)$ correction that is absorbed into the step-size constraint $\eta \leq \gamma/(L_c B_g^2)$. □

A.2 Proof of Proposition 2: Bounded Cross-Mode Contamination

Batch decomposition. At step k, the batch contains fraction α_k of groups in leakage mode and $(1-\alpha_k)$ in performance mode. The combined stochastic gradient is $\hat{g}_k = \alpha_k\, \hat{g}_{\mathrm{leak},k} + (1-\alpha_k)\, \hat{g}_{\mathrm{perf},k}$, and the policy update is $\theta_{k+1} = \theta_k + \eta\, \hat{g}_k$. By L_c-smoothness of V_c [Nesterov, 2004],

$$V_c(\theta_{k+1}) \leq V_c(\theta_k) + \eta \langle \nabla V_c(\theta_k),\, \hat{g}_k \rangle + \tfrac{L_c \eta^2}{2} \|\hat{g}_k\|^2. \tag{9}$$

Leakage-mode component. By Proposition 1, $\mathbb{E}[\langle \nabla V_c, \hat{g}_{\mathrm{leak},k} \rangle \mid \theta_k] = \langle \nabla V_c, \bar{g}_{\mathrm{leak}} \rangle \leq -\gamma(\theta_k) \|\nabla V_c\|^2$. The leakage-mode contribution to the expected inner product in (9) is therefore at most $-\alpha_k \eta \gamma(\theta_k) \|\nabla V_c\|^2$.

Performance-mode component. The performance-mode gradient $\bar{g}_{\mathrm{perf}} = \mathbb{E}_{y|c=0}[(r_{\mathrm{eff}}(y) - \bar{r}_{\mathrm{eff}})\, s(y)]$ is computed from completions that all satisfy $c(y,t) = 0$. By the bounded second-moment assumption and Cauchy–Schwarz,

$$|(1-\alpha_k)\, \eta\, \mathbb{E}[\langle \nabla V_c,\, \hat{g}_{\mathrm{perf}} \rangle]| \;\leq\; (1-\alpha_k)\, \eta\, B_g\, \|\nabla V_c\|.$$

This bounds the worst-case cross-mode contribution: the performance gradient can project onto ∇V_c, but only up to the norm product. Under the Polyak–Łojasiewicz condition $\|\nabla V_c\|^2 \geq 2\mu V_c$, the leakage-mode descent $\alpha_k \gamma(\theta_k) \|\nabla V_c\|^2$ dominates this term whenever $\|\nabla V_c\| \geq (1-\alpha_k) B_g / (\alpha_k \gamma(\theta_k))$, which is satisfied for V_c above a noise floor proportional to $B_g^2/(\alpha_k^2 \gamma^2 \mu)$.

Combined bound. Taking conditional expectations in (9) and combining the two components:

$$\mathbb{E}[V_c(\theta_{k+1}) \mid \theta_k] \leq V_c(\theta_k) - \big(\alpha_k \gamma(\theta_k) - (1-\alpha_k) B_g / \|\nabla V_c(\theta_k)\|\big)\, \eta \|\nabla V_c(\theta_k)\|^2 + \tfrac{L_c \eta^2}{2} B_g^2. \tag{10}$$

The effective descent rate $\alpha_k \gamma(\theta_k) - (1-\alpha_k) B_g / \|\nabla V_c\|$ is positive whenever $\|\nabla V_c\| > (1-\alpha_k) B_g / (\alpha_k \gamma(\theta_k))$. Under the PL condition ($\|\nabla V_c\|^2 \geq 2\mu V_c$), this holds for all V_c above a threshold proportional to $B_g^2/(\alpha_k^2 \gamma^2 \mu)$, ensuring net descent throughout the leakage phase. □

A.3 Proof of Theorem 1: Convergence

Given Propositions 1–2, the convergence proof follows the standard smooth stochastic optimization framework. We state the key steps for completeness, noting that the algebraic machinery is due to Ghadimi and Lan [2013] for the nonconvex rate and Karimi et al. [2016] for the PL linear rate.

Part (i): Nonconvex rate. Let $\gamma_0 = \inf\{\gamma(\theta) : V_c(\theta) \geq \epsilon_0\}$ for target accuracy $\epsilon_0 > 0$. On the sublevel set $\{V_c \geq \epsilon_0\}$, the PL condition gives $\|\nabla V_c\| \geq \sqrt{2\mu\epsilon_0}$, so the effective rate in (10) satisfies $\alpha_{\min}\gamma_0 - (1-\alpha_{\min}) B_g / \sqrt{2\mu\epsilon_0} > 0$ for $\epsilon_0 \geq 2(1-\alpha_{\min})^2 B_g^2 / (\mu \alpha_{\min}^2 \gamma_0^2)$. The condition $\alpha_k \geq \alpha_{\min} > 0$ holds throughout the leakage phase by construction: while $V_c \geq \epsilon_0$, Markov's inequality guarantees $\bar{q} \geq \epsilon_0 / c_{\max} > 0$ (where $c_{\max}$ bounds the maximum leak count), ensuring a positive fraction of groups contain leaked completions and enter leakage mode. Summing (10) over $k = 0, \ldots, K-1$:

$$\frac{1}{K} \sum_{k=0}^{K-1} \mathbb{E}\|\nabla V_c(\theta_k)\|^2 \leq \frac{V_c(\theta_0)}{\alpha_{\min}\, \eta \gamma_0 K} + \frac{L_c \eta B_g^2}{2\alpha_{\min}\gamma_0}.$$

Setting $\eta = O(1/\sqrt{K})$ gives $\min_{k \leq K} \|\nabla V_c(\theta_k)\|^2 = O(1/\sqrt{K})$, the standard nonconvex SGD rate [Ghadimi and Lan, 2013].

Part (ii): Performance improvement. In performance mode, all completions satisfy $c = 0$ and the GRPO gradient maximizes $V_r(\theta) = \mathbb{E}[r_{\text{eff}}(y) \mid c(y,t){=}0]$ directly. By the policy gradient theorem [Sutton et al., 2000] applied to the conditional distribution over clean completions, the expected gradient is $\bar{g}_{\text{perf}} = \nabla_\theta V_r(\theta)/\sigma_r + O(1/G)$, where σ_r is the within-group standard deviation of r_{eff} among clean completions. Since $\sigma_r > 0$ for non-degenerate policies, this is proportional to ∇V_r and the standard ascent lemma under L_r-smoothness gives

$$\mathbb{E}[V_r(\theta_{k+1}) \mid \theta_k] \geq V_r(\theta_k) + \eta\|\nabla V_r(\theta_k)\|^2 - \tfrac{L_r\eta^2}{2}B_g^2.$$

Part (iii): PL linear convergence. We invoke the Polyak–Łojasiewicz (PL) condition $\|\nabla V_c(\theta)\|^2 \geq 2\mu\, V_c(\theta)$. This condition is natural for the leakage landscape: $V_c \geq 0$ has a unique global minimum at zero (leaked claim counts are non-negative), and the LoRA parameterization is heavily overparameterized relative to the effective output dimension, precluding spurious local minima; all six training runs converge to near-zero leakage, confirming this empirically. Under PL, the cross-mode term satisfies $(1{-}\alpha_k)B_g/\|\nabla V_c\| \leq (1{-}\alpha_{\min})B_g/\sqrt{2\mu V_c}$. For $V_c \geq \epsilon_0 = 2(1{-}\alpha_{\min})^2 B_g^2/(\mu\alpha_{\min}^2\gamma_0^2)$, the effective rate in (10) is at least $\alpha_{\min}\gamma_0/2 > 0$, giving

$$\mathbb{E}[V_c(\theta_{k+1}) \mid \theta_k] \leq (1 - \alpha_{\min}\eta\gamma_0\mu)\, V_c(\theta_k) + \tfrac{L_c\eta^2}{2}B_g^2.$$

Unrolling the recursion [Karimi et al., 2016]:

$$\mathbb{E}[V_c(\theta_k)] \leq (1 - \alpha_{\min}\eta\gamma_0\mu)^k\, V_c(\theta_0) + \frac{L_c\eta B_g^2}{2\alpha_{\min}\gamma_0\mu}.$$

The first term decays geometrically; the second is a noise floor that shrinks with the learning rate. Since $V_c \geq 0$ with global minimum at 0, this establishes linear convergence to a neighborhood of the leak-free regime. □

A.4 Proof of Corollary 1: Two-Phase Curriculum

Markov's inequality gives $q(\theta; x, t) = \Pr[c \geq 1] \leq \mathbb{E}_{y\sim\pi_\theta}[c(y,t)]$, so $\bar{q}(\theta) \leq V_c(\theta)$. Since $(1{-}q)^G$ is convex, Jensen's inequality yields $\bar{p}_0 = \mathbb{E}[(1{-}q)^G] \geq (1{-}\bar{q})^G \geq (1{-}V_c)^G$. As $V_c(\theta_k)$ decreases to the residual of Eq. (7) (Theorem 1(iii)), $\bar{p}_0(\theta_k)$ rises toward 1, up to this residual: almost all groups end in performance mode. Combined with Theorem 1(ii), this establishes the two-phase curriculum: training automatically transitions from a leakage-dominated phase to a performance-optimization phase without manual scheduling. The transition is monotone because V_c is non-increasing in expectation, so $(1{-}V_c)^G$ is non-decreasing.

A.5 Why leakage does not reach zero

Theorem 1(iii) predicts this: leakage converges to a neighborhood whose size is the residual term in Eq. (7), not to zero. Two further effects set the observed floor. First, the mode gate is self-limiting by construction: a fully clean group is assigned to performance mode and receives no leakage-reduction signal, so the pressure on leakage diminishes as more groups become clean (Table 3). Second, leakage is measured through the verifier, so even a leak-free model cannot score below the verifier's false-positive rate, which the control set bounds at 0.0–0.7% OLR (Table 2). TEMPO's 0.6–3.7% lies within about three percentage points of this floor. The bound also suggests a way to lower the residual: it grows linearly with the step size η, so a decaying step size would tighten it in theory, at the cost of slower training; we leave an empirical test to future work.

B Reward design: justification and details

This section provides formal justification for the two-mode reward design and the full specification of the quality gates, format failure penalty, and overlong penalty shaping referenced in Section 3.2.

Setup. Let π_θ be the policy, $c(y,t) \geq 0$ the integer-valued leaked claim count, and $q(\theta; x, t) = \Pr_{y\sim\pi_\theta}[c(y,t) > 0]$ the per-instance leakage probability. For a group of G i.i.d. completions, define the group-clean probability $p_0(\theta; x, t) = (1-q)^G$ and its expectation $\bar{p}_0(\theta) = \mathbb{E}_{(x,t)}[p_0]$.

B.1 Gradient coherence: why scalar mixing fails

Proposition 3 (Gradient coherence)**.** *Consider the naively mixed reward* $R_{\mathrm{mix},i} = \alpha\, r_{\mathrm{leak},i} + (1{-}\alpha)\, r_{\mathrm{perf},i}$*, with* S_{perf}*,* S_{leak}*,* S_{mix} *the within-group standard deviations of* r_{perf}*,* r_{leak}*,* R_{mix}*, and* $A_i^m = (r_{m,i} - \bar{r}_m)/S_m$ *the single-objective z-scored advantages. The GRPO advantages decompose as*

$$A_i^{\mathrm{mix}} = \underbrace{\tfrac{(1-\alpha)\, S_{\mathrm{perf}}}{S_{\mathrm{mix}}}}_{w_{\mathrm{perf}}}\, A_i^{\mathrm{perf}} + \underbrace{\tfrac{\alpha\, S_{\mathrm{leak}}}{S_{\mathrm{mix}}}}_{w_{\mathrm{leak}}}\, A_i^{\mathrm{leak}}. \tag{11}$$

Let $g_m = G^{-1} \sum_i A_i^m \nabla_\theta \log \pi_\theta(y_i \mid x)$ *be the GRPO policy gradient under reward* r_m*.* (i) *When* $S_{\mathrm{perf}} \gg S_{\mathrm{leak}}$*,* $w_{\mathrm{leak}}/w_{\mathrm{perf}} \to 0$ (scale dominance). (ii) *There exist policy states where* $\langle g_{\mathrm{leak}}, g_{\mathrm{perf}} \rangle < 0$ (gradient conflict)*. The mode gate eliminates both pathologies.*

Proof. Expanding $R_{\mathrm{mix},i} - \bar{R}_{\mathrm{mix}} = \alpha(r_{\mathrm{leak},i} - \bar{r}_{\mathrm{leak}}) + (1{-}\alpha)(r_{\mathrm{perf},i} - \bar{r}_{\mathrm{perf}})$ and normalizing yields the decomposition (11), so the GRPO gradient decomposes as $g_{\mathrm{mix}} = w_{\mathrm{perf}}\, g_{\mathrm{perf}} + w_{\mathrm{leak}}\, g_{\mathrm{leak}}$. The ratio $w_{\mathrm{leak}}/w_{\mathrm{perf}} = \alpha S_{\mathrm{leak}}/((1{-}\alpha)S_{\mathrm{perf}}) \to 0$ when $S_{\mathrm{perf}} \gg S_{\mathrm{leak}}$, establishing (i).

For (ii), consider a policy producing compliant completions ($c = 0$, low performance r_{low}) and leaked completions ($c > 0$, high performance $r_{\mathrm{high}} > r_{\mathrm{low}}$). Under r_{leak} the compliant type receives positive advantage; under r_{perf} the leaked type does. The gradients push probability mass in opposing directions, giving $\langle g_{\mathrm{leak}}, g_{\mathrm{perf}} \rangle < 0$.

Under the mode gate, $g_{\mathrm{TEMPO}} = \mathbf{1}[L{=}0]\, g_{\mathrm{perf}} + \mathbf{1}[L{=}1]\, g_{\mathrm{leak}}$, so each update optimizes a single objective with homogeneous advantages, eliminating both pathologies. □

B.2 Exact penalty property

Remark 1 (Exact penalty)**.** *The mode gate implements an exact penalty without tunable dual variables. The expected per-group reward decomposes as* $J_{\mathrm{TEMPO}} = p_0\, \mathbb{E}[r_{\mathrm{eff}} \mid L{=}0] + (1{-}p_0)\, \mathbb{E}[r_{\mathrm{leak}} \mid L{=}1]$*. For any* $q > 0$*, the opportunity cost relative to the feasible optimum (*$q = 0$*) is* $J_{\mathrm{TEMPO}}|_{q=0} - J_{\mathrm{TEMPO}} \geq (1{-}p_0)\big(\mathbb{E}[r_{\mathrm{eff}} \mid L{=}0] - \mathbb{E}[r_{\mathrm{leak}} \mid L{=}1]\big) > 0$*, since* $\mathbb{E}[r_{\mathrm{eff}} \mid L{=}0] > \mathbb{E}[r_{\mathrm{leak}} \mid L{=}1]$ *for any non-trivial policy and* $1 - p_0 = 1 - (1{-}q)^G \to 1$ *as* $G \to \infty$*. No dual variable requires tuning; at feasibility (*$q = 0$*),* $J_{\mathrm{TEMPO}} = \mathbb{E}[r_{\mathrm{eff}}]$*, recovering pure performance optimization.*

B.3 Reward hacking resistance

Proposition 4 (Reward hacking resistance)**.** *Accepting leakage to boost performance is strictly suboptimal under the mode gate. Under a Lagrangian formulation* $V_r - \mu V_c$*, any finite* μ *allows the policy to trade* Δc *leakage for* $\Delta r > \mu \Delta c$ *performance. Under the mode gate, the marginal value of any performance improvement is multiplied by* $(1{-}q)^G$*, which decays exponentially in* G *for* $q > 0$*; no finite performance gain can compensate.*

Proof. A policy with $q > 0$ accesses performance mode with probability $(1{-}q)^G$, which decreases exponentially in G. The marginal benefit of any performance improvement is multiplied by this factor. For G large, $(1{-}q)^G \to 0$: no finite performance gain can compensate for the exponentially vanishing access to performance mode. Therefore accepting leakage is strictly suboptimal for any $q > 0$. □

B.4 Quality gates

Three multiplicative quality gates attenuate the effective reward in performance mode (Equation 4). Each gate maps a per-completion signal to a bounded range.

Coverage Gate. The coverage gate rewards completions whose reasoning cites a large fraction of their evidence list:

$$g(\mathrm{cov}) = \max(\mathrm{cov},\ c_{\mathrm{floor}})\,, \tag{12}$$

where $\mathrm{cov} = |\{e_k : e_k \text{ cited in reasoning}\}|/|\{e_k\}|$ is the coverage fraction (Section 4.3) and $c_{\text{floor}} = 0.20$. The floor ensures that completions with zero coverage still receive 20% of the performance reward, preserving gradient signal for improving citation alongside accuracy.

Evidence Diversity Gate. The evidence diversity gate penalizes template collapse—the tendency for the policy to converge on a fixed, minimal number of evidence items:

$$d(\mathrm{count}) = \min\left(\frac{\mathrm{count}}{e_{\text{target}}},\ 1\right), \tag{13}$$

where count is the number of distinct evidence items and $e_{\text{target}} = 8$. A completion producing 4 items receives $d = 0.5$, halving its quality credit. The target is calibrated to the base model's natural output: median evidence count at initialization is 7–9 items across tasks.

Reasoning Quality Gate. The reasoning quality gate penalizes shallow reasoning that provides no analytical depth beyond restating evidence:

$$w(\mathrm{words}) = \min\left(\frac{\mathrm{words}}{w_{\text{target}}},\ 1\right), \tag{14}$$

where words is the word count of the reasoning paragraph and $w_{\text{target}} = 120$. The target is set below the base model's median reasoning length (150–170 words), so the gate is transparent for outputs of normal length and activates only when reasoning degrades below a minimum acceptable depth.

Interaction During Training. The effective performance-mode reward is $r_{\text{eff}} = r_{\text{perf}} \cdot g(\mathrm{cov}) \cdot d(\mathrm{ev}) \cdot w(\mathrm{reason})$. The evidence diversity and reasoning quality gates provide direct gradient signal against evidence count collapse and reasoning degradation. In leakage mode, the quality bonus uses d but omits w: reasoning length in leakage mode is less informative because the model may produce shorter reasoning when actively avoiding leaked claims, which is a valid strategy.

Leakage-Mode Bonuses. Clean completions ($n_{\text{leak},i} = 0$) within a group that is still in leakage mode receive small additive bonuses to encourage maintaining evidence quality:

$$b_{\text{cov},i} = w_{\text{cov}} \cdot g(\mathrm{cov}_i) \cdot \mathbf{1}[\text{clean}_i], \qquad b_{\text{qual},i} = w_{\text{qual}} \cdot d(\mathrm{ev}_i) \cdot \mathbf{1}[\text{clean}_i], \tag{15}$$

where $w_{\text{cov}} = 0.05$ and $w_{\text{qual}} = 0.02$. The total leakage-mode reward for completion i is:

$$r_{\text{leak,total},i} = r_{\text{leak},i} + b_{\text{cov},i} + b_{\text{qual},i}. \tag{16}$$

GRPO computes within-group z-scored advantages from $r_{\text{leak,total}}$. Because b_{cov} and b_{qual} are small (≤ 0.07 combined) relative to the leakage reward spread (r_{leak} ranges from $\exp(-0.5 \cdot 10) \approx 0.007$ for heavily leaked completions to 1.0 for clean ones), leakage reduction dominates the advantage ranking while the bonuses provide a secondary signal encouraging clean completions to maintain evidence quality.

B.5 Format failure penalty

The format penalty is computed dynamically from group statistics to ensure parse failures always receive an advantage strictly below the worst valid completion, preventing the optimizer from rewarding ill-formed outputs. The penalty adapts to three cases depending on the number of valid completions n_{valid} in the group.

When $n_{\text{valid}} = 0$, all advantages are set to zero and the group is skipped (no gradient contribution). When $n_{\text{valid}} = 1$, the single valid completion receives advantage zero and all parse failures receive a fixed negative floor $-m$, where $m = 1.0$ is the format margin. When $n_{\text{valid}} \geq 2$, the penalty is derived from the group's advantage distribution:

$$f_{\text{pen}} = \max\big(\min(\mathbf{a}) - m,\ -\alpha \cdot \max(\mathbf{a})\big), \tag{17}$$

where $\mathbf{a}$ is the vector of valid-completion advantages, $m = 1.0$ is the format margin, and $\alpha = 5.0$ is the maximum negative-to-positive ratio. The ratio cap $-\alpha \cdot \max(\mathbf{a})$ prevents the format penalty from dominating the advantage distribution when valid completions receive small positive advantages, which would otherwise cause the optimizer to spend most of its gradient budget on format compliance rather than leakage and accuracy.

B.6 Overlong penalty shaping

Parse-failure completions that are very long receive an attenuated penalty, because these outputs typically contain useful reasoning that happens to violate the JSON schema at the end rather than being entirely degenerate:

$$A_i \leftarrow A_i \cdot \max\big(1 - \delta \cdot \ell_i,\ s_{\text{floor}}\big), \tag{18}$$

where $\ell_i = \min(1, |\text{text}_i|/L_{\max})$ is the normalized text length, $\delta = 0.5$ is the decay rate, $L_{\max} = 32{,}000$ characters, and $s_{\text{floor}} = 0.3$ is the minimum scaling factor. A maximum-length parse failure receives 30% of the base format penalty, while short parse failures receive the full penalty.

B.7 Design rationale and evolution

The final reward design (v4) emerged through four iterations, each addressing failure modes observed in its predecessor.

V1 (Weighted Linear Combination). The initial design combined leakage and performance rewards into a single scalar advantage with cosine-annealed weights (w_{leak} from 0.95 to 0.80). Coverage was logged but did not enter the reward. Within 50 training steps, coverage collapsed from 88% to 53%: the model learned to fabricate vague, safe-sounding evidence with pre-cutoff dates while minimizing the number of citable items. The parse-failure rate spiked from 2% to over 90% by step 200, because malformed outputs received a single blended penalty that was too weak to override the leakage-reduction gradient.

V2 (Coverage-Gated Two-Mode). To address coverage collapse, v2 introduced the two-mode structure (Section 3.2): performance-mode reward was made contingent on a coverage gate $g(\text{cov})$, and a small coverage bonus was added in leakage mode. This stabilized coverage but exposed a new failure: on a different model (gpt-oss-20b, salary task), training degenerated from step 35 and halted at step 63. Root causes were (a) mode-specific weight coefficients that dampened the gradient signal, (b) an unbounded format penalty that reached 11:1 to 99:1 negative-to-positive ratios, and (c) a uniform penalty for near-miss parse failures (long, mostly correct outputs with a trailing JSON error) versus fully degenerate outputs.

V3 (Degeneration-Resistant). v3 removed the mode-specific weight coefficients, introduced the ratio-capped format penalty (Equation 17, $\alpha = 5.0$), and added overlong penalty shaping (Equation 18) to distinguish near-miss from degenerate failures. Temperature was reduced from 0.8 to 0.6 and the PPO clip high was raised from 1.4 to 2.0. Training no longer degenerated, but a new failure emerged on Qwen3.5-35B: template collapse, where evidence count dropped from 7–9 to 3–5 items and reasoning length from 150+ to 60–80 words over training. The v3 reward had no mechanism to penalize this quality degradation because r_{perf} depended only on prediction accuracy.

V4 (Quality-Gated, Final). v4 added the multiplicative quality gates—evidence diversity d (Equation 13), reasoning quality w (Equation 14)—and the leakage-mode quality bonus (Equation 15) to the performance-mode and leakage-mode rewards respectively. A masked per-token KL penalty was added to constrain policy drift. These additions eliminated template collapse: evidence count and reasoning length remained within one standard deviation of the base model's distribution across all six training runs.

Component-Level Rationale. The exponential leakage reward (Equation 3) was chosen over a binary clean/leaked indicator because the exponential form provides graded signal: reducing leaked claims from many to fewer yields a larger reward change than a binary switch, creating a stronger gradient early in training when most completions contain multiple leaked claims.

The two-mode advantage structure separates conflicting optimization objectives. In multi-objective GRPO, naively combining leakage and prediction rewards into a single advantage can cause one signal to dominate depending on their relative scales [Ichihara et al., 2025]. The mode split ensures each group optimizes a single objective, with the strict binary condition ($n_{\text{leak},i} = 0\ \forall\, i$; Equation 2) acting as a hard gate between the two.

The quality gate thresholds ($e_{\text{target}} = 8$, $w_{\text{target}} = 120$) are calibrated from the base model's output distribution. Setting the evidence target at the base model's median output ensures the gate penalizes only deterioration below baseline, not failure to exceed an aspirational standard.

C TEMPO implementation details

C.1 Training algorithm

Algorithm 1 details the TEMPO training loop. At each step, the policy generates G completions per prompt in the batch. Each completion is parsed into its structured components (evidence, reasoning, prediction), and the date-plausibility verifier counts leaked claims. The group-level mode gate then determines whether each prompt's group enters leakage mode or performance mode: if any completion in the group contains a leaked claim, the entire group receives leakage-mode rewards (exponential penalty on leak count); only when all G completions are clean does the group enter performance mode, where rewards combine task accuracy with quality gates for coverage, evidence diversity, and reasoning depth. Advantages are z-scored within each group and calibrated across prompts via a batch-level baseline. A masked KL penalty constrains drift from the base model on content tokens only. The clipped GRPO update then adjusts the policy parameters.

Algorithm 1 TEMPO training loop. Each step samples completions, assigns mode-specific rewards via the group-level gate (Eq. 2), and updates the policy with clipped GRPO and a masked KL penalty.

Require: Dataset $\mathcal{D} = \{(x_k, t_k, y_k)\}$, base policy π_{ref}, group size G, batch size B, steps T, learning rate η

Initialize $\pi_\theta \leftarrow \pi_{\text{ref}}$ with LoRA adapters
for step $= 1$ **to** T **do**
  Sample batch $\mathcal{B} = \{(x_k, t_k, y_k)\}_{k=1}^{B}$ from $\mathcal{D}$
  for each $(x, t, y) \in \mathcal{B}$ **do**
    Sample G completions $\{y_1, \ldots, y_G\} \sim \pi_\theta(\cdot \mid x)$
    Parse each $y_i \to (\mathcal{E}_i, \mathcal{R}_i, \hat{y}_i)$
    Compute $n_{\text{leak},i} = |\{j : s^*_{ij} > t\}|$ ▷ via date-plausibility verifier
    if $n_{\text{leak},i} = 0 \; \forall \, i$ **then** ▷ Performance mode (Eq. 2)
      $r_{\text{eff},i} = r_{\text{perf},i} \cdot g(\text{cov}_i) \cdot d(\text{ev}_i) \cdot w(\text{reason}_i)$
      $A_i \leftarrow \text{zscore}(r_{\text{eff}}) + \lambda \, (r_{\text{eff},i} - \bar{r}^{\text{batch}}_{\text{eff}})$
    else ▷ Leakage mode
      $r_{\text{leak},i} = \exp(-0.5 \cdot n_{\text{leak},i})$
      $A_i \leftarrow \text{zscore}(r_{\text{leak}}) + \lambda \, (r_{\text{leak},i} - \bar{r}^{\text{batch}}_{\text{leak}})$
    end if
  end for
  Apply KL penalty: $A_i \leftarrow A_i + \beta_{\text{KL}} \, \mathbf{m}_i \odot (\bar{\delta} - \delta_i)$ ▷ $\delta_i = \log \frac{\pi_\theta}{\pi_{\text{ref}}}$, $\bar{\delta}$ = batch mean
  $\theta \leftarrow \theta + \eta \cdot \frac{1}{|\mathcal{B}| \, G} \sum_{k,i} \text{clip}\big(\frac{\pi_\theta}{\pi_{\text{old}}}\big) \, A_i \, \nabla_\theta \log \pi_\theta(y_i \mid x_k)$
end for
return π_θ

C.2 Training pipeline

Each training step samples G=12 completions per prompt at temperature 0.6 with a maximum budget of 8,192 tokens. A `<think>` prefill is injected so the model emits a chain-of-thought before the structured JSON answer. The system prompt enforces a three-step procedure—evidence with source dates, reasoning with inline citations, prediction—and specifies a JSON output schema. Task-specific WRONG/RIGHT examples in the prompt clarify the distinction between announcement dates and period-end dates to reduce source-date errors.

Completions are scored through a four-phase reward pipeline. Phase 1 parses the JSON output via three fallback strategies (direct parse, markdown fence extraction, greedy regex) and validates against a Pydantic schema; parse failures receive zero rewards. Phase 2 extracts claims from the

evidence list and deduplicates by lowercased text. Phase 3 sends claims in chunks of 80 to a date-plausibility verifier (Gemini 3 Flash Preview)[3] that checks whether each declared source date is temporally consistent with the claim type; the leakage verdict is determined deterministically by comparing the effective date against the instance cutoff. Phase 4 computes coverage (fraction of evidence items cited) and task-specific prediction accuracy programmatically.

Training instances are shuffled randomly per epoch with a deterministic seed (`data_seed + epoch`), and batches of $B{=}8$ prompts are drawn sequentially from the shuffled order.

Validation runs at 15 evenly-spaced steps across the training run using greedy decoding ($G{=}1$, temperature 0.0) with the same scoring pipeline as training. The checkpoint with the lowest validation OLR is saved as the best checkpoint; a final checkpoint is saved unconditionally at the end of training.

Model Adaptation. Both models are fine-tuned via low-rank adaptation (LoRA) with rank 32. Alpha, target modules, and weight merging are managed by the Tinker training API,[4] which handles adapter injection into all attention projection layers. The optimizer is Adam with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\varepsilon = 10^{-8}$, and a constant learning rate of 2×10^{-5}.

Compute. We use no local GPUs. All training and sampling run on the Tinker managed fine-tuning API, which hosts the base models and trains the LoRA adapters server-side with four concurrent workers; judge and search calls run through OpenRouter (Gemini 3 Flash, Perplexity Sonar). Reproducibility depends on the API endpoints and hyperparameters (Table 6). Table 5 reports the measured wall-clock time per training run (batch size 8, group size 12), about 81 hours in total. Both models have since been retired from Tinker for fine-tuning, so the reward-design ablation (Section 5.4) covers only the stock task; our code, configurations, and saved outputs document all runs, but re-training either model requires another provider.

Table 5: Wall-clock training time per run on the Tinker API.

Model	Stock	Salary	Legal
Qwen3-30B	7.6 h	10.4 h	6.7 h
Qwen3.5-35B	19.5 h	21.8 h	14.9 h

C.3 Stability techniques

Two mechanisms stabilize GRPO training.

Asymmetric clipping. Following DAPO [Yu et al., 2025], we use asymmetric PPO clipping with $\epsilon_{\text{low}} = 0.1$ (clip ratio lower bound $1{-}\epsilon_{\text{low}} = 0.9$) and $\epsilon_{\text{high}} = 1.0$ (clip ratio upper bound $1{+}\epsilon_{\text{high}} = 2.0$). The wider upper bound encourages exploration of high-advantage completions (particularly those that reduce leakage count), while the tighter lower bound prevents catastrophic probability collapse on previously favored outputs.

Masked KL penalty. A per-token KL penalty against the frozen base model π_{ref} constrains policy drift:

$$A_i \leftarrow A_i + \beta_{\text{KL}}\, \mathbf{m}_i \odot (\bar{\delta} - \delta_i), \tag{19}$$

where $\delta_i = \log \pi_\theta(y_i \mid x) - \log \pi_{\text{ref}}(y_i \mid x)$ is the per-token log-ratio, $\bar{\delta}$ is its batch mean, and $\mathbf{m}_i$ is a binary mask that excludes structured output tokens (JSON delimiters, field names) from the penalty. The coefficient $\beta_{\text{KL}} = 0.05$ (Table 6) is set low enough to preserve adaptation speed while preventing the mode-collapse failure mode observed in early experiments (Appendix B.7).

C.4 Inference configuration

At evaluation time, all methods use greedy decoding (temperature 0.0) with a single completion per instance ($G{=}1$) and a maximum budget of 8,192 tokens. The scoring pipeline is identical to the

[3] Accessed via OpenRouter: `https://openrouter.ai/api/v1`

[4] `https://pypi.org/project/tinker/`

Table 6: GRPO training hyperparameters. Both models use identical settings except for total training steps, which vary by dataset size. Key choices: LoRA rank 32, learning rate 2×10^{-5}, group size G=12, KL coefficient β_{KL}=0.05.

Parameter	**Value**
Architecture	
Adapter	LoRA, rank 32
Maximum tokens	8,192
GRPO	
Group size G	12
Batch size B	8
Epochs	1
Learning rate	2×10^{-5}
Sampling temperature	0.6
Loss function	PPO (clipped)
Clip range (low / high)	0.9 / 2.0
KL penalty coefficient	0.05
Advantage z-clip	5.0
Reward weights	
Leakage weight schedule w_{leak}	cosine $0.95 \rightarrow 0.80$
Leakage decay λ_{base}	0.5
Coverage target / floor	0.80 / 0.20
Leak-mode coverage weight $w_{\mathrm{cov,leak}}$	0.05
Format margin	1.0
Quality gates	
Evidence target	8 items
Reasoning target	120 words
Quality leak-mode weight	0.02
Data	
Validation fraction	10%
Difficulty tiers	3 (hard / medium / easy)
Validation checkpoints	15
Data seed	42
Judges	
Date-plausibility judge (leakage)	Gemini 3 Flash Preview
Search model	Perplexity Sonar

training reward pipeline: parse, extract claims, verify leakage, compute coverage and task-specific accuracy. This ensures that training and evaluation metrics are directly comparable. The base model, best checkpoint (lowest validation OLR), and final checkpoint are evaluated for each model-task combination; the main table reports whichever checkpoint achieved the best combination of low OLR and high parse rate.

D Dataset construction

Every instance is designed so that the ground-truth answer was publicly available before the earlier model's knowledge cutoff (approximately March 2025 for Qwen3-30B-A3B). The control sets require answers published after the later model's cutoff (approximately February 2026 for Qwen3.5-35B-A3B), establishing a genuine-prediction baseline where the model lacks the outcome in parametric memory. All per-instance cutoff dates are day-level precise (YYYY-MM-DD), sourced from official databases rather than approximated.

Stock Ranking. Adjusted closing prices are retrieved from Yahoo Finance via the `yfinance` Python library.[5] The company universe consists of S&P 500 constituents grouped by GICS sector into pools of approximately 100 tickers each. Four six-month training windows capture distinct macroeconomic regimes: W1 (January–June 2022; Russia-Ukraine conflict, energy shock), W2

[5] `https://pypi.org/project/yfinance/`

(March–August 2023; SVB collapse, NVIDIA AI surge), W3 (July–December 2020; COVID-19 recovery, vaccine rollout), and W4 (January–June 2024; AI momentum, rate-cut expectations). Each window generates 300 questions for a total of 1,200 training instances. Within each window, 80% of questions sample companies from sectors most affected by the corresponding macroeconomic event, and the remaining 20% draw from other sectors, concentrating signal where return dispersion is highest. Five companies are sampled without replacement per question, with deduplication enforcing unique groups (random seed 42). The evaluation window covers December 2019 to June 2020 (COVID-19 onset, 100 instances), temporally disjoint from all training windows. The control window covers February–March 2026 (100 instances), using real market data that neither model encountered during pre-training (random seed 99). Information Technology is overrepresented in training because it appears as an affected sector in three of four windows; the 80/20 sampling strategy accepts this in exchange for higher signal density.

Salary Prediction. Free-agent contract data are scraped from Spotrac,[6] the standard public reference for professional sports contracts. Three sports are included for training: NFL, MLB, and NHL. Only players who changed teams with AAV above sport-specific thresholds ($5M for NFL and MLB, $3M for NHL) are retained, filtering out minimum-salary signings with negligible variation. The raw Spotrac tables provide the contract year but not the exact announcement date; precise signing dates (YYYY-MM-DD) are added via the Perplexity `sonar` search-enabled model through OpenRouter, querying batches of 15 players per request. Fallback dates for failed lookups use known free-agency opening dates: NFL league-specific dates per year (e.g., 2024-03-13), NHL July 1, and MLB December 1 of the prior year. All entries with signing dates on or after March 1, 2025 are excluded; the 15 MLB 2025 entries that remain were signed in November–December 2024, consistent with the MLB off-season calendar. The training set spans late 2017–2024 across three sports, totaling 1,066 instances (524 NFL, 374 MLB, 168 NHL). Evaluation uses 137 NBA free-agent contracts from 2019–2024—a held-out sport not seen during training. The control set comprises 26 transactions from after February 1, 2026: 13 MLB late signings, 9 NHL trade-deadline acquisitions, and 4 NFL franchise tags and pre-free-agency trades, compiled from FOX Sports, ESPN, and NHL.com.

Legal Case Prediction. U.S. Supreme Court cases from October Terms 2014–2023 (decisions spanning 2015–2024) are retrieved from the Oyez API,[7] maintained by Cornell LII and Chicago-Kent College of Law. The outcome is determined from the `winning_party` field, mapped to PETITIONER or RESPONDENT via name matching against the listed parties; cases without a clear binary disposition are excluded, yielding 497 SCOTUS cases. Federal appellate opinions from the 1st through 11th Circuits, D.C. Circuit, and Federal Circuit (2017–2024) are retrieved from the CourtListener REST API v4,[8] sorted by citation count within each year to prioritize substantive, well-cited opinions. For each candidate, the opinion PDF is downloaded and the first 12 pages are extracted via `pdfplumber` (capped at 8,000 characters). A structured extraction prompt sent to Gemini 2.0 Flash (temperature 0) requests the parties, factual background, legal holding, disposition (PETITIONER if reversed/vacated, RESPONDENT if affirmed), and panel vote split; cases where the model cannot determine a clear binary outcome are excluded, yielding 397 appellate cases. The combined 894 training cases and 100-instance evaluation set are split by stratified sampling: approximately 10 cases per decision year (2014–2024) with outcome balance enforced within each year stratum. Three additional cases are removed from training to eliminate case-name overlap arising from different proceedings sharing a party name. The training set exhibits a PETITIONER skew (63.5% vs. 36.5%) because the Oyez data is biased toward cases where the Supreme Court grants certiorari to reverse; the evaluation set is intentionally balanced (46% PETITIONER, 54% RESPONDENT). The control set consists of 90 federal appellate decisions filed between February 25 and March 6, 2026, extracted with the same CourtListener and Gemini pipeline.

Quality Assurance. Automated verification (via `scripts/verify_dataset.py`) confirms that all 3,713 instances across every split conform to the temporal design. All cutoff dates are in YYYY-MM-DD format with zero violations: no training or evaluation date falls on or after March 1, 2025, and no control date falls before February 1, 2026. Cross-split integrity checks confirm zero case-name overlap between legal training and evaluation sets, zero player-year overlap between salary

[6] `https://www.spotrac.com`

[7] `https://api.oyez.org`

[8] `https://www.courtlistener.com/api/rest/v4/`

splits, and temporal disjointness between stock training and evaluation windows. No single decision year exceeds 14% of the legal training set, and salary training spans 2018–2025 across three sports with calendar-month diversity from sport-specific free-agency schedules.

D.1 Why not existing forecasting benchmarks

A leakage test set needs outcomes that resolved before both model cutoffs (March 2025 and February 2026). Live forecasting benchmarks are built for the opposite goal: they ask about unresolved events, so every fact is pre-cutoff by construction and leakage should be zero for every method. Their already-resolved questions do not fit either as shown in Table 7. A clean score on a barely-crawled recent outcome cannot be told apart from the outcome being absent, which is why we build our tasks on outcomes resolved between 2014 and 2024 (Appendix D). Our post-cutoff control sets play the role that unresolved questions play in these benchmarks.

Table 7: Existing forecasting benchmarks as leakage test sets.

Benchmark (launch)	Resolved before March 2025	Usable as leakage test set?
FutureX (Aug 2025)	none; launched after the cutoff	no instances exist
ForecastBench (Jul 2024)	$\sim$8 months; 89% templated indicators	too recent to be well represented
MIRAI (test set Nov 2023)	705 GDELT queries from one month	agentic relational-coding format

E Baseline implementations

The three baselines form a progressive intervention hierarchy, each adding one mechanism on top of the previous to test whether it suffices to suppress temporal leakage. **Temporal Hint** relies solely on prompt-based temporal constraints—no retrieval and no verification—testing how the model's forecasting behavior changes when it is given temporal instructions from the prompt alone. **RAG** adds adaptive retrieval with a hard date-cutoff filter, providing the model with focused pre-cutoff evidence to test how external valid information, even with a date filter, will affect the leakage and performance of the model's forecasting behavior. **TimeSPEC** adds post-generation verification on top of RAG, using a supervisor to extract and verify claims and filter out leaked content, testing whether inference-time claim filtering achieves temporal compliance. This progression isolates the contribution of each mechanism: if prompt constraints alone fail, does retrieval help? If retrieval introduces contamination, does verification fix it? The results (Section 5) show that none of these mechanisms is sufficient, motivating TEMPO's parametric approach.

E.1 Temporal Hint (base agent)

The Temporal Hint baseline is the simplest configuration and serves as the foundation for all other methods. The evaluation model receives the task-specific system prompt and a user prompt containing the instance description, the reference (cutoff) date, and explicit temporal instructions: the model must reason using only information available before the cutoff date. No retrieval, no post-generation verification, and no parameter updates are applied. The system prompt includes structured output formatting instructions that define the evidence list, reasoning paragraph, and prediction fields. All other methods inherit this prompt structure verbatim; differences arise solely from retrieval augmentation (RAG), post-hoc verification (TimeSPEC), or parameter updates (TEMPO).

E.2 RAG baseline

The RAG baseline augments the base model with retrieved pre-cutoff evidence at inference time without any parameter updates. For each evaluation instance, Gemini 3 Flash Preview receives the user prompt and cutoff date and generates a set of task-adaptive search queries: 7–12 for stock ranking (which benefits from diverse financial signals) and 3–6 for salary and legal tasks. Queries are constructed to target distinct aspects of the information needed, with the year included to anchor temporal relevance.

Retrieval uses the Perplexity Search API[9] with a hard `search_before_date_filter` set to the instance cutoff date, returning up to 5 results per query. Results are deduplicated by URL across all queries for the same instance. The deduplicated results are truncated to a 3,000-token budget (estimated at 4 characters per token), selecting results in relevance order until the budget is exhausted.

Retrieved passages are formatted as numbered plain-text entries (title and snippet) and prepended to the user prompt. Following the convention of retrieval-augmented generation methods, no temporal metadata, validity framing, or behavioral guidance is included in the context block—the model must determine temporal relevance from content alone, analogous to how the base and RL-trained variants rely on parametric memory. The system prompt remains identical to the base method. Scoring uses the same evaluation pipeline as all other methods.

E.3 TimeSPEC baseline

TimeSPEC adapts the five-phase inference architecture from Zhang et al. [2026b] to the current experimental setting. The original paper introduced this pipeline using Claude for all generation phases; this implementation substitutes Qwen (via the Tinker API) for generation, regeneration, and aggregation, Gemini 3 Flash Preview for supervision and verification, and the Perplexity Search API for retrieval. The core architecture is otherwise unchanged.

Phase 0–1: Retrieval and Generation. These phases are identical to the RAG baseline described above: Gemini generates task-adaptive search queries, Perplexity retrieves results with a hard date filter set to the instance cutoff, and the deduplicated context is prepended to the user prompt. The system prompt and generation model are byte-for-byte identical to the RAG condition. Reusing the same retrieval and generation phases ensures that any difference between RAG and TimeSPEC is attributable solely to the post-generation verification pipeline.

Phase 2: Supervisor (Claim Extraction and Verification). The supervisor extracts atomic claims from the model's evidence list and reasoning text via a single Gemini call using the `CLAIM_EXTRACTION_PROMPT`. Each claim is classified into the A1–A5/B1–B2 temporal taxonomy from Zhang et al. [2026b]: A1–A3 are temporally verifiable assertions (discrete events, state measurements, publications) whose availability date can be determined via search; A4 (outcome) and A5 (consequential) claims are deterministically leaked by taxonomy definition, as they describe the very result the model is predicting or its direct consequences; B1 (background knowledge) and B2 (definitional) claims are deterministically clean, as they describe time-invariant facts. This taxonomy-based classification reduces the number of claims requiring search verification, as only A1–A3 claims proceed to the search step.

For each A1–A3 claim, a search query is constructed directly from the claim text (e.g., "When was this first publicly reported: {claim}") without an additional LLM call. The Perplexity Search API retrieves results *without* a date filter, in contrast to the RAG retrieval phase which uses a hard date filter. The absence of a date filter is essential: the verifier must determine when the information first became publicly available, which requires access to the full temporal record. All (claim, search result) pairs for one instance are judged in a single batch Gemini call using the `TIMESPEC_CLAIM_VERIFY_PROMPT`, which produces a deterministic verdict: `event_date` $>$ `cutoff_date` $\Rightarrow$ `leaked`.

Phase 3: Regenerator (Conditional). Regeneration runs only when the supervisor detects leaked claims. The system prompt is byte-for-byte identical to the Base and RAG conditions. A constraint block is appended to the user prompt listing claims to avoid as a plain numbered list—no temporal language (e.g., "post-cutoff"), no corrected values, no corrected dates, and no explanation of why each claim should be avoided. This design ensures the generation model receives no information about the temporal verification process; it sees only a list of facts it must not use. At most one regeneration cycle is performed (`timespec_max_regen_cycles=1`).

Phase 4: Resupervisor (Conditional). If regeneration occurred, the resupervisor applies the same claim extraction and verification pipeline as Phase 2 to the regenerated output. Claims that appeared

[9] `https://docs.perplexity.ai/api-reference/sonar-search`

in the original output and were already verified in Phase 2 reuse their cached verdicts, avoiding redundant API calls. Only newly introduced claims require fresh verification.

Phase 5: Aggregator. The aggregator always runs, including when no leaked claims were found, to enforce a closed-world constraint. The system prompt is the original task system prompt extended with a single constraint paragraph: the model must select evidence *exclusively* from a provided list of validated facts and must not introduce any facts, numbers, events, or entities not present in that list. Validated claims—all non-leaked Group A claims plus all Group B claims—are presented as plain-text facts *without* source dates, category labels, or any temporal metadata. The model must infer its own `source_date` fields from the fact text alone, identical to how all other baselines produce source dates. This ensures the aggregator's information surface is restricted to verified-clean evidence and the standard task prompt. The aggregator output is scored through the identical evaluation pipeline as all baselines.

E.4 Fairness of comparison

For the comparison across methods to be reliable, we must ensure that no method gains an unfair informational advantage at generation time. Table 8 verifies this: all four methods share the same parametric memory and task prompt; RAG and TimeSPEC additionally receive retrieved pre-cutoff passages, but no method exposes verification search results or temporal metadata to the generation model. Crucially, the evaluation pipeline (leakage detection, performance scoring, coverage computation) is identical for all methods, so any performance difference reflects the method's behavior rather than measurement bias.

Table 8: Information surface at generation time. All methods share parametric memory and task prompt. RAG and TimeSPEC add retrieved passages; no method exposes verification results or temporal metadata to the generator. The identical evaluation pipeline ensures fair comparison.

Information source	Temp. Hint	RAG	TimeSPEC	TEMPO
Parametric memory	✓	✓	✓	✓
Task prompt + cutoff date	✓	✓	✓	✓
Retrieved pre-cutoff passages	—	✓	✓	—
Verification search results	—	—	—	—
Temporal metadata on claims	—	—	—	—
Identical evaluation pipeline	✓	✓	✓	✓

The fairness constraint is most delicate for TimeSPEC, which runs an internal verification pipeline that could leak information to the generation model if not carefully isolated. Four properties ensure this does not occur: (1) Qwen never sees search results from the verification phase—these are internal to the Gemini supervisor; (2) the regeneration constraint lists claims to avoid as plain text, with no temporal language, corrected values, or explanation of why each claim was flagged; (3) the aggregator receives validated facts without source dates, category labels, or metadata; (4) the claim extraction prompt uses only fields derivable from the user prompt. RAG satisfies the constraint trivially (only pre-cutoff passages, no verification metadata), and Temporal Hint adds nothing beyond the prompt. TEMPO's information surface is identical to Temporal Hint at inference because all temporal discipline is encoded in the model's parameters during training.

F Evaluation protocol and metric definitions

This section formally defines the evaluation metrics introduced in Section 4.3 and describes the scoring pipeline. All metrics are computed deterministically from a single model completion per instance.

F.1 Leakage detection pipeline

The structured output format requires the model to produce an evidence list in which each item carries a `source_date` field. Claims with declared source dates are batched in chunks of 80 and submitted to the date-plausibility verifier (Gemini 3 Flash Preview). The verifier assesses whether

each declared date is temporally plausible given the claim type: for example, full-year financial results cannot predate the fiscal year end, and quarterly earnings appear 1–2 months after quarter close. If the declared date is implausible, the verifier returns a corrected date. The leakage verdict is deterministic:

$$\text{leaked}(c_j) = \mathbf{1}[\text{effective_date}(c_j) > t_{\text{ref}}]\,, \tag{20}$$

where effective_date(c_j) is the verifier-corrected date if the original was implausible, and the declared date otherwise. Claims without a declared source date default to not-leaked. This is a conservative choice that may undercount leakage but avoids penalizing the model for omitting dates on genuinely pre-cutoff facts. Undated items still count toward the evidence-diversity gate and citation coverage; the only date-dependent term is a 0.10-weighted component of the format score, which does not enter the training advantage of parseable completions.

F.2 Overall leakage rate (OLR)

Let $n_{\text{leak},j}$ denote the number of leaked claims and $n_{\text{total},j}$ the total number of claims with declared source dates for instance j. The instance-level leakage rate is $\ell_j = n_{\text{leak},j}/\max(n_{\text{total},j}, 1)$. The Overall Leakage Rate (OLR) averages across all N instances:

$$\text{OLR} = \frac{1}{N}\sum_{j=1}^{N} \ell_j. \tag{21}$$

OLR $= 0$ indicates that no instance contains post-cutoff evidence; lower is better.

F.3 Decision-critical leakage rate (DCLR)

For instance j with evidence claims $c_1, \ldots, c_n$, let $\ell(c_i) \in \{0, 1\}$ be the verifier's leakage verdict and ϕ_i the Shapley value of claim c_i for the prediction. Following Zhang et al. [2026b],

$$\text{DCLR}_j = \frac{\sum_{i=1}^{n} |\phi_i|\, \ell(c_i)}{\sum_{i=1}^{n} |\phi_i|}, \qquad \text{DCLR} = \frac{1}{N}\sum_{j=1}^{N} \text{DCLR}_j. \tag{22}$$

The value of a claim subset is how closely a closed-world model, given only those claims, reproduces the original prediction; the corresponding base model serves as this evaluator. Shapley values are computed exactly for instances with at most eight claims and with Leverage SHAP otherwise. Leakage verdicts are those of the verifier above, and an instance without leaked claims has $\text{DCLR}_j = 0$.

F.4 Task performance (Perf)

The performance metric is task-specific and mapped to $[0, 1]$:

- **Stock ranking**: $(\rho + 1)/2$, where ρ is the Spearman rank correlation between the predicted and ground-truth rankings of the five stocks by return over the prediction window.
- **Salary prediction**: $\max(0,\, 1 - |\hat{y} - y|/y)$, where $\hat{y}$ is the predicted annual salary and y the ground truth.
- **Legal outcome prediction**: $1 - (\hat{p} - y)^2$, the Brier score complement comparing the predicted probability $\hat{p}$ to the binary outcome y.

All three metrics are bounded in $[0, 1]$ with higher values indicating better prediction quality. A performance score strictly above the control experiment (Section G.5) confirms that the model leverages pre-cutoff knowledge rather than guessing.

F.5 Coverage

Coverage measures the fraction of evidence items in the evidence list that are explicitly cited in the reasoning paragraph via bracket markers (e.g., `[1]`, `[3]`):

$$\text{Coverage} = \frac{|\{e_k : e_k \text{ cited in reasoning}\}|}{|\{e_k\}|}. \tag{23}$$

High coverage indicates that the model's reasoning is grounded in the evidence it generates. Low coverage suggests the model produces evidence items as decoration without integrating them into its reasoning chain—a failure mode that undermines interpretability and faithfulness. Instances with zero evidence items or parse failures receive coverage 0.

G Extended evaluation results

This section provides empirical evidence beyond the main results table, including inference cost analysis, instance-level leakage and coverage distributions, training dynamics, and metric validation via control experiments.

G.1 Inference efficiency

Table 9: Inference cost per instance (LLM calls and web searches). TEMPO and Temporal Hint each require a single forward pass with zero external queries, while RAG adds 2 LLM calls with 3–12 web searches and TimeSPEC requires 5+ LLM calls with $\sim$10 searches for multi-phase verification.

Method	LLM calls	Searches	Description
Temporal Hint	1	0	Single forward pass with temporal prompt
RAG	2	3–12	Query gen. (Gemini) + retrieval (Perplexity) + forward pass
TimeSPEC	5+	$\sim$10	5-phase pipeline + per-claim verification
TEMPO (Ours)	1	0	Single forward pass, identical to base

Table 9 quantifies the per-instance inference cost of each method. TEMPO requires exactly one forward pass—identical to the unmodified Temporal Hint baseline—because temporal discipline is encoded in the model's parameters rather than enforced by an external pipeline. RAG adds a query-generation call and web retrieval on top of the base forward pass. TimeSPEC builds on RAG with a multi-phase verification pipeline. In the typical case ($\sim$10 Group A claims, no regeneration needed), TimeSPEC requires 3 LLM calls (generator, supervisor, aggregator) and $\sim$10 Perplexity searches (one per A1–A3 claim). When regeneration triggers, the cost increases to 5 LLM calls (adding regenerator and resupervisor) plus additional searches for newly introduced claims. TEMPO therefore matches TimeSPEC's leakage reduction (Table 2) at a fraction of the inference cost, with no external API dependencies at inference time.

G.2 Instance-level leakage distribution

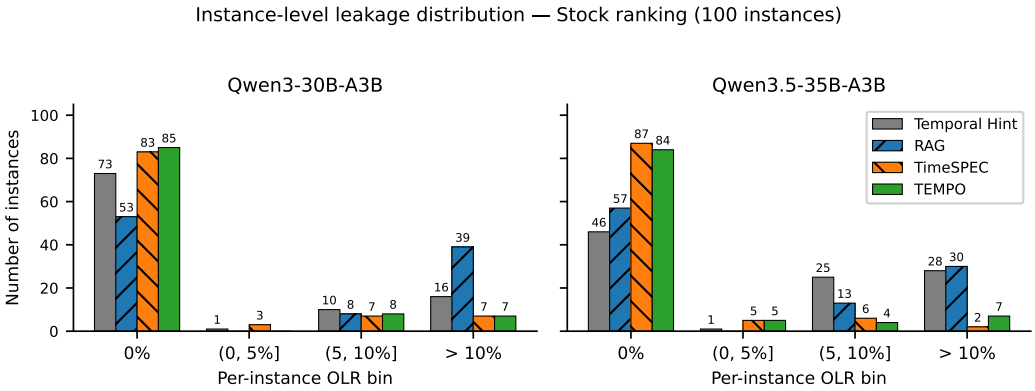


Figure 3: Instance-level OLR distribution on stock ranking (100 instances per condition). TEMPO concentrates instances at zero leakage while reducing the heavy-leakage tail. RAG amplifies leakage via retrieved post-cutoff content.

Figure 3 disaggregates the mean OLR reported in the main table into per-instance leakage bins on the stock ranking task—the most leakage-sensitive of the three tasks, where pre-training data encodes pandemic-era market dynamics that directly answer the evaluation questions. For Qwen3-30B, TEMPO produces 85 zero-leakage instances out of 100 (versus 73 for Temporal Hint), and critically eliminates the $(0, 5\%]$ bin entirely: residual leakage, when it occurs, stems from a small number of instances with moderate contamination rather than widespread low-level leakage. The improvement is more pronounced for Qwen3.5-35B, where zero-leakage instances rise from 46 (Temporal Hint)

to 84 (TEMPO), and the >10% tail shrinks from 28 to 7 instances. TimeSPEC achieves a similar concentration at zero leakage (comparable to TEMPO's distribution), confirming that both methods effectively suppress leakage at the instance level; the key difference between them is not leakage suppression but coverage (Section G.3). RAG amplifies leakage: 39 and 30 instances exceed 10% OLR for the two models, respectively, because retrieved financial news provides additional surface for post-cutoff information.

Salary and legal tasks exhibit lower baseline leakage (8.1%/11.9% and 2.2%/2.1% OLR), so the per-instance distribution is already concentrated near zero. On these tasks, TEMPO's improvement manifests primarily as mean OLR reduction rather than a dramatic distributional shift, consistent with the main results.

G.3 Evidence faithfulness

Table 10: Coverage distribution across methods and tasks. Mean coverage (%) and the fraction of instances exceeding 90% and 80% coverage thresholds. Best per model-task block in **bold**.

		Stock			Salary			Legal		
Model	Method	Mean↑	≥90↑	≥80↑	Mean↑	≥90↑	≥80↑	Mean↑	≥90↑	≥80↑
30B	Temporal Hint	94.6	89	93	83.0	50	73	61.7	7	23
	RAG	83.2	48	65	76.0	36	59	**85.8**	**51**	**76**
	TimeSPEC	73.8	27	49	71.2	21	42	65.2	15	29
	TEMPO	**99.3**	**98**	**99**	**91.2**	**80**	**87**	75.7	15	35
35B	Temporal Hint	88.4	62	78	80.0	39	72	69.0	16	44
	RAG	90.5	64	91	84.9	42	76	**81.4**	30	**69**
	TimeSPEC	75.5	14	41	75.0	23	49	73.0	26	50
	TEMPO	**95.7**	**94**	**94**	**92.0**	**74**	**94**	74.1	**41**	48

Table 10 reports the fraction of instances exceeding 90% and 80% coverage thresholds, disaggregating the mean coverage in the main table to verify that TEMPO's advantage is not an artifact of a few high-coverage outliers. On stock ranking, 98% (30B) and 91% (35B) of TEMPO instances exceed 90% coverage, versus 27% and 14% for TimeSPEC. On salary prediction, 80% and 74% of TEMPO instances exceed 90% coverage, versus 21% and 23% for TimeSPEC. These gaps confirm that TEMPO's evidence–reasoning link is preserved at the instance level: the model cites virtually every evidence item it generates because the evidence and reasoning are produced in a single coherent forward pass, whereas TimeSPEC's post-hoc regeneration decouples the two.

Legal coverage is lower across all methods because legal reasoning draws on procedural history and statutory interpretation that is harder to decompose into discrete citable evidence items. TEMPO and TimeSPEC achieve comparable mean coverage on this task (65–73%), with TEMPO showing a higher fraction above the 90% threshold for Qwen3.5-35B (41% versus 26%).

G.4 Validation trajectory

Table 11 quantifies the training validation trajectory—metrics computed at intermediate checkpoints during training using greedy decoding on the validation split—from the base model (step 0) through the final training checkpoint. Coverage increases by 3–13 percentage points on stock and salary for both models, confirming that TEMPO preserves and often improves evidence citation. OLR drops by 35.6 percentage points for Qwen3-30B on stock (the highest-leakage condition) and by 0.8–7.4 points on other conditions where baseline leakage is lower. Prediction accuracy remains stable across training for all six model-task combinations.

G.5 Control experiment validation

To validate that the OLR metric measures genuine temporal contamination rather than false positives, we evaluate the Temporal Hint baseline on control instances whose answers were published after February 2026—after both models' knowledge cutoffs. On these instances, the model lacks

Table 11: Validation metrics: base model (step 0) vs. final training checkpoint and their change (Δ). Negative OLR change indicates improvement.

		OLR (%)			Performance			Coverage (%)		
Model	Task	Base	Final	Δ	Base	Final	Δ	Base	Final	Δ
30B	Stock	37.2	1.7	-35.6	0.486	0.495	$\approx$0	96.0	99.1	+3.1
30B	Salary	4.3	3.5	-0.8	0.634	0.687	+0.053	80.4	92.9	+12.5
30B	Legal	1.6	1.7	+0.2	0.728	0.740	$\approx$0	65.8	71.6	+5.8
35B	Stock	8.2	0.8	-7.4	0.560	0.550	$\approx$0	90.9	97.3	+6.4
35B	Salary	8.8	4.4	-4.4	0.536	0.686	+0.150	82.5	94.4	+11.9
35B	Legal	2.3	2.8	+0.5	0.765	0.746	$\approx$0	70.3	66.4	-3.9

parametric knowledge of the answer, so any detected leakage represents verifier noise rather than actual post-cutoff knowledge.

Table 12: Control experiment results on post-cutoff instances (answers published after February 2026). Near-zero OLR across all conditions (0.0–0.7%) confirms the leakage metric produces negligible false positives when the model lacks post-cutoff knowledge. Prediction accuracy near chance on stock and salary establishes the zero-knowledge performance floor.

Model	Task	OLR (%)	Perf	Cov (%)	n
30B	Stock	0.7	0.476	99.3	100
30B	Salary	0.6	0.454	79.5	26
30B	Legal	0.0	0.707	66.2	90
35B	Stock	0.2	0.497	92.5	100
35B	Salary	0.4	0.549	86.2	26
35B	Legal	0.0	0.647	74.3	90

Table 12 reports the results. Control OLR is near zero across all six conditions (0.0–0.7%), confirming that the leakage metric is well-calibrated: when the model genuinely lacks post-cutoff knowledge, the verifier produces negligible false positives. Control prediction accuracy is near chance for stock (0.48–0.50) and salary (0.45–0.55), confirming the model cannot predict outcomes absent from its parameters. Legal control accuracy is higher (0.65–0.71), consistent with legal prediction relying on precedent and procedural reasoning available before the cutoff.

H Validity of the leakage measurement

This section tests whether the leakage results can be trusted. Section H.1 validates the verifier itself. TEMPO constrains the evidence a model states, not its internal computation (Section 6); Sections H.2 and H.3 check for the behavior that hidden use of memorized outcomes would produce.

H.1 Verifier validation

The same verifier is used during training and evaluation, so a model could in principle learn to evade it. We test the verifier in four ways.

Gold-label test. We construct 120 factual claims about real, publicly datable events, each annotated by humans with its true event date and the resulting leak label for a given cutoff (64 leaked, 56 clean). The claims span nine claim categories and include 30 boundary pairs, in which the same fact is leaked under one cutoff and clean under another. The paper's verifier, with the same prompt, judge, and decision rule, runs under two conditions: honest declared dates, and an adversarial condition in which the 64 leaked claims are re-declared with fabricated pre-cutoff dates. The adversarial condition is the evasion case: a post-cutoff fact stamped with an early date.

The verifier rejected every fabricated date and recovered the true event date, typically to within a day, so the verdict follows the corrected date rather than the model's declaration (Table 13).

Table 13: Verifier accuracy on 120 human-labeled claims.

Condition	Result
Honest declared dates	120/120 correct; 0 false positives
Adversarial: 64 leaked claims re-dated before the cutoff	64/64 caught

Independent judges. We re-score all 6,569 evidence claims from the Qwen3.5-35B stock evaluations with two judges from other providers (Claude and Kimi), using the paper's verifier prompt and decision rule verbatim. Per-claim agreement with the training judge (Gemini) is 93–99% across methods.

Table 14: OLR (%) on stock (Qwen3.5-35B-A3B) under the training judge and two independent judges.

Method	Gemini (paper)	Claude	Kimi
Temporal Hint	12.7	5.9	6.6
RAG	6.8	5.1	5.2
TimeSPEC	1.5	1.4	1.0
TEMPO	2.4	0.1	0.3

Both independent judges reproduce the method ordering (Table 14). Both also score TEMPO lower than its training judge does, which is the opposite of what judge-specific evasion would produce. The judges disagree most on Temporal Hint (12.7% vs. 5.9–6.6%), on borderline claims; this suggests the baselines may leak somewhat less than Table 2 reports, while TEMPO's rate and the ordering are stable under every judge.

Sensitivity to label noise. We flip each claim's leakage label independently with probability up to 20% and recompute OLR over 1,000 resamples. The conclusion that TEMPO leaks less than Temporal Hint holds in 100% of resamples, and less than RAG in 98.5%.

Human audit of clean-judged claims. The remaining risk is a leak phrased so that every LLM judge misses it. We manually audit 60 randomly sampled TEMPO claims (Qwen3.5-35B, stock) that all three judges passed as clean, checking each stated fact's true event date against the instance cutoff. Three of the 60 are false negatives, all of the same type: a pre-cutoff fact stated with a company's later name or ticker (e.g., "Meta Platforms" for Facebook in 2019, or "Coterra" for the pre-merger Cabot Oil & Gas). None of the 60 contains post-cutoff outcome information such as prices, earnings, or events, the kind of leak that could inflate a backtest. All three are names given in the task input, which lists companies under their current S&P 500 names and tickers. A check of every Qwen3.5-35B stock evidence claim with the same judge confirms the pattern: all 162 TEMPO claims flagged for a later name or ticker reuse the input name, compared with 7 (Temporal Hint), 9 (RAG), and 0 (TimeSPEC) claims whose anachronistic name originates from the model. These names identify companies but carry no outcome information; future versions of the dataset will use names as of each cutoff.

H.2 Outcome-recall probe

Setup. We ask each untrained base model to state the realized outcome directly, with no cutoff instruction: rank the five tickers by actual return (stock) or give the actual contract value (salary), using greedy decoding. An instance counts as *memorized* if the recall matches the truth, with Spearman $\rho \geq 0.6$ for stock or relative error within 10% for salary. We validate the probe on the control set, whose outcomes postdate pretraining, so memorization there is impossible. Qwen3.5-35B-A3B is queried through OpenRouter because Tinker no longer serves this model.

The base models memorize outcomes. On stock, recall on the evaluation set is well above the chance level of the control set, and Qwen3.5-35B declines to recall control outcomes while confidently recalling evaluation outcomes (Table 15). The larger model memorizes more, consistent with its higher baseline OLR in Table 2.

Table 15: Outcome-recall probe on the untrained base models. Recall: mean Spearman ρ between recalled and true ranking (stock) or median relative error of the recalled value (salary). Memorized: share of instances meeting the memorization threshold.

Model	Task	Eval: recall	Eval: memorized	Control set
Qwen3-30B	Stock	$\rho = 0.142$	28.4%	$\rho = 0.003$ (chance)
Qwen3.5-35B	Stock	$\rho = 0.257$	42.0%	declines 94% of questions
Qwen3-30B	Salary	median error 51.5%	8.0%	12.5% flagged
Qwen3.5-35B	Salary	median error 33.3%	18.0%	declines all questions

TEMPO's accuracy does not concentrate on memorized instances. If TEMPO exploited this memory, its accuracy would be higher on memorized instances than familiarity alone explains. The benchmark is not zero: even on the control set the probe flags about 20% of instances, because outcomes of stable, well-known companies are guessable and easy instances score higher for any method. We therefore compare TEMPO's accuracy gap against a null gap measured on the control set, where only familiarity can act (Table 16).

Table 16: Accuracy on instances flagged as memorized versus not flagged (stock). The null row evaluates the untrained model on the control set, where memorization is impossible.

	Acc., memorized	Acc., not memorized	Gap
TEMPO, Qwen3-30B (eval)	0.781 (n=27)	0.574 (n=68)	0.208
TEMPO, Qwen3.5-35B (eval)	0.754 (n=42)	0.541 (n=58)	0.213
Null: untrained, control set	0.611 (n=19)	0.444 (n=77)	0.166

TEMPO's gap exceeds the null by less than 0.05, and the 95% bootstrap confidence interval of the difference includes zero (30B: $[-0.114, 0.198]$; 35B: $[-0.118, 0.208]$), so familiarity fully explains it. The conclusion holds at a stricter threshold ($\rho \geq 0.8$), where the null rises to 0.228 and TEMPO's gaps are 0.226 (30B) and 0.167 (35B). On salary, Qwen3-30B's memorized rate (8.0%) is below the control false-positive rate (12.5%), and Qwen3.5-35B's gap is 0.071 with a confidence interval that includes zero. With about 100 instances per condition, the test cannot separate methods from one another; Temporal Hint's difference from the null also includes zero. It does rule out a large, hidden accuracy advantage on memorized outcomes.

H.3 Closed-world reconstruction test

Setup. If a model decided with hidden knowledge and then selected evidence to justify the answer, a reader given only that evidence could not reproduce the prediction. We test this directly: a fresh evaluator (the corresponding base model) receives only the task framing and the evidence list of each saved completion, is instructed to use nothing else, and re-derives the prediction. The gap is the original accuracy minus the rebuilt accuracy, with paired bootstrap 95% confidence intervals; Corr is the per-instance correlation between original and rebuilt accuracy.

Table 17: Closed-world reconstruction. Orig. re-scores the saved completions that have an extractable evidence list, so its levels differ slightly from Table 2; the test is paired per completion, and the gap is the measured quantity.

Model	Task	Method	Orig.	Rebuilt	Gap	95% CI	Corr
Qwen3.5-35B	Stock	Temporal Hint	0.696	0.705	-0.008	$[-0.033, 0.017]$	0.85
		TEMPO	0.649	0.637	0.013	$[-0.023, 0.051]$	0.74
	Salary	Temporal Hint	0.584	0.581	0.003	$[-0.026, 0.033]$	0.85
		TEMPO	0.637	0.632	0.005	$[-0.017, 0.030]$	0.87
Qwen3-30B	Stock	Temporal Hint	0.596	0.562	0.033	$[0.003, 0.065]$	0.81
		TEMPO	0.636	0.573	0.064	$[0.027, 0.102]$	0.70
	Salary	Temporal Hint	0.621	0.602	0.018	$[-0.018, 0.053]$	0.70
		TEMPO	0.559	0.567	-0.009	$[-0.045, 0.027]$	0.59

The stated evidence reproduces the predictions. All gaps are at most 0.064, and six of eight confidence intervals include zero (Table 17). The two exceptions are both Qwen3-30B stock, for Temporal Hint as well as TEMPO, which points to the weaker 30B rebuilder rather than to TEMPO. Correlations of 0.59–0.87 show that the rebuilt predictions follow the supplied evidence rather than the evaluator's own priors. Because the evaluator is instructed rather than knowledge-free, we read the test comparatively: any residual evaluator knowledge affects both methods alike, and TEMPO's gap never exceeds Temporal Hint's by more than 0.031. The test establishes that the stated evidence is *sufficient* to reproduce each prediction, not that the model's internal reasoning is causally faithful to it (Section 6).

No decide-then-justify signature. Justifying a decision after the fact decouples evidence from reasoning, which shows up as low citation coverage. This signature appears in TimeSPEC's verify-then-regenerate pipeline, where only 14–27% of stock and salary instances exceed 90% coverage, but not in TEMPO, where 74–98% do (Table 10).

I Case studies

This section presents detailed four-method comparisons on selected instances from each task. For each instance, we show representative evidence items (with dates and leakage annotations), reasoning excerpts, the final prediction alongside the ground truth, and the task-specific performance score. Instances are selected to illustrate how temporal leakage propagates through the evidence–reasoning–prediction chain and how each method handles it differently.

Each case study is organized as follows: a brief instance description, followed by one block per method showing key metrics (OLR, Performance, Coverage), evidence excerpts, reasoning, and prediction. Analytical prose after each set of blocks traces (1) which evidence items are leaked and how they enter the reasoning, (2) whether the reasoning chain is grounded in the cited evidence (coverage), and (3) how leakage or its absence affects prediction calibration. Evidence excerpts are shown in reduced font size for compactness. Each evidence item is prefixed by a parenthesized date indicating when the stated information became publicly available, as determined by our date-plausibility verifier.

I.1 Stock Ranking

We examine how each method handles stock ranking for five energy-sector companies (Marathon Petroleum, Targa Resources, Chevron, Halliburton, Occidental Petroleum) with a cutoff date of December 1, 2019. The task requires ranking these companies by predicted 6-month stock return using only information available before the cutoff. The true ranking over the evaluation period (December 2019–May 2020) was CVX $>$ MPC $>$ HAL $>$ TRGP $>$ OXY, reflecting the impact of the COVID-19 pandemic on energy markets—information entirely unavailable at the cutoff.

Temporal Hint — OLR: 25.0%, Perf: 0.65, Coverage: 50.0%

Evidence (20 items):
[16] (2020-05-29) Marathon Petroleum's stock price closed at $61.45 on 2020-05-29. [LEAKED]
[17] (2020-05-29) Targa Resources' stock price closed at $25.10 on 2020-05-29. [LEAKED]
[18] (2020-05-29) Chevron Corporation's stock price closed at $125.30 on 2020-05-29. [LEAKED]
[19] (2020-05-29) Halliburton's stock price closed at $37.10 on 2020-05-29. [LEAKED]
[20] (2020-05-29) Occidental Petroleum's stock price closed at $62.10 on 2020-05-29. [LEAKED]
[1] (2019-10-31) Marathon Petroleum reported Q3 2019 revenue of $23.1 billion, up 12% year-over-year.
[2] (2019-11-07) Targa Resources reported Q3 2019 revenue of $1.8 billion, up 15% year-over-year.
[11] (2019-11-29) Marathon Petroleum's stock price closed at $57.32 on 2019-11-29.
[. . . 12 more items]

Reasoning:
Marathon Petroleum (MPC) showed strong revenue growth of 12% in Q3 2019 [1], and its stock price increased from $57.32 on 2019-11-29 to $61.45 on 2020-05-29 [16], indicating positive momentum. Targa Resources (TRGP) also had strong revenue growth of 15% in Q3 2019 [2], and its stock price increased from $23.45 to $25.10 over the evaluation period [17]. Chevron Corporation (CVX) reported

10% revenue growth in Q3 2019 [3], and its stock price rose from $120.15 to $125.30 [18], showing steady performance. Occidental Petroleum (OXY) experienced a 14% revenue decline in Q3 2019 [5], but its stock price increased from $58.75 to $62.10 [20], suggesting some recovery. Halliburton (HAL) had a significant revenue decline of 11% in Q3 2019 [4], and its stock price increased from $34.22 to $37.10 [19], but the decline in revenue suggests weaker fundamentals.

Predicted ranking: MPC > TRGP > CVX > OXY > HAL *True ranking:* CVX > MPC > HAL > TRGP > OXY

RAG — OLR: 41.7%, Perf: 1.00, Coverage: 58.3%

Evidence (12 items):
[1] (2025-11-01) Targa Resources' Capital Expenditures fell 1894.31% to $671.8 million in Q3 2025 from the same period last year [...] [LEAKED]
[3] (2020-01-21) Halliburton's Completion and Production revenue was $3,506 million in 2019. [LEAKED]
[7] (2020-03-01) Marathon Petroleum's Dividend Yield was 3.12% in 2018 and 3.52% in 2019. [LEAKED]
[8] (2019-12-31) In 2019, the average price of Brent crude oil was $64 per barrel. [LEAKED]
[12] (2025-11-01) Targa Resources' Capital Expenditures (Quarter) stood at $183.5 million in 2021, then soared by 182.78% to $518.9 million [...] [LEAKED]
[2] (2018-03-05) Chevron Corp. expects 900,000 b/d from Permian by yearend 2023, up from 650,000 b/d by yearend 2022.
[5] (2019-02-07) Marathon Petroleum Corp. reported fourth-quarter 2018 earnings of $951 million, or $1.35 per diluted share.
[...5 more items]

Reasoning:
Chevron (CVX) is expected to have strong stock returns due to its significant production growth in the Permian basin, with expectations of 900,000 b/d by yearend 2023 [2], and a 3–4%/year compound production growth rate through 2023 [10]. Marathon Petroleum (MPC) also shows strong potential, with a 15% dividend increase in 2019 [6], and a history of significant capital returns to shareholders [6]. Halliburton (HAL) has shown stable revenue in 2019, with $3,506 million in Completion and Production revenue [3]. Targa Resources (TRGP) has experienced significant fluctuations in capital expenditures, with a sharp decline in 2025 [1], which may indicate reduced investment and potential underperformance. Occidental Petroleum (OXY) is not mentioned in the provided evidence, so its performance cannot be evaluated.

Predicted ranking: CVX > MPC > HAL > TRGP > OXY *True ranking:* CVX > MPC > HAL > TRGP > OXY

TimeSPEC — OLR: 0.0%, Perf: 0.90, Coverage: 42.9%

Evidence (14 items):
[1] (2019-01-01) Chevron Corporation announced a 2019 organic capital and exploratory spending program of $20 billion.
[2] (2019-01-31) Marathon Petroleum Corp. reported fourth-quarter 2018 earnings of $951 million.
[4] (2019-01-28) Marathon Petroleum Corp. announced a 15 percent increase in the quarterly dividend on January 28, 2019.
[6] (2019-01-31) Marathon Petroleum Corp. reported adjusted EBITDA of $4.1 billion for the fourth quarter of 2018.
[10] (2019-01-01) The EIA expects global oil markets to be tightest in the second and third quarters of 2019.
[14] (2019-01-01) The EIA expects diesel fuel margins to increase to an average of 53 cents/gal by the fourth quarter of 2019.
[...8 more items]

Reasoning:
Marathon Petroleum Corp. (MPC) showed strong financial performance in 2018, with $951 million in fourth-quarter earnings [2] and $4.1 billion in adjusted EBITDA [6]. The company also increased its dividend by 15% in January 2019 [4], indicating confidence in its financial position. The EIA projected that diesel fuel margins would rise to 53 cents/gal by the end of 2019 [14], which could benefit refining companies like MPC. Chevron Corporation (CVX) announced a $20 billion capital and exploratory

spending program for 2019 [1], suggesting long-term investment. The EIA also forecast tight global oil markets in the second and third quarters of 2019 [10]. However, there is no direct evidence of specific financial performance for Targa Resources (TRGP), Halliburton (HAL), or Occidental Petroleum (OXY). Based on the available evidence, MPC appears to have the strongest financial position and growth potential, followed by CVX.

Predicted ranking: MPC > CVX > TRGP > HAL > OXY *True ranking:* CVX > MPC > HAL > TRGP > OXY

TEMPO (Ours) — OLR: 0.0%, Perf: 0.55, Coverage: 100.0%

Evidence (10 items):
[1] (2019-10-31) Marathon Petroleum reported Q3 2019 revenue of $25.8 billion, representing a 12% year-over-year increase.
[2] (2019-11-07) Targa Resources reported Q3 2019 revenue of $1.2 billion, representing a 5% year-over-year increase.
[3] (2019-11-01) Chevron Corporation reported Q3 2019 revenue of $32.5 billion, representing a 7% year-over-year increase.
[4] (2019-10-31) Halliburton reported Q3 2019 revenue of $5.1 billion, representing a 3% year-over-year decrease.
[5] (2019-11-04) Occidental Petroleum reported Q3 2019 revenue of $8.9 billion, representing a 10% year-over-year increase.
[6] (2019-04-01) Marathon Petroleum's stock price increased by 15% in the first quarter of 2019.
[7] (2019-04-01) Targa Resources' stock price increased by 8% in the first quarter of 2019.
[8] (2019-04-01) Chevron Corporation's stock price increased by 10% in the first quarter of 2019.
[9] (2019-04-01) Halliburton's stock price decreased by 5% in the first quarter of 2019.
[10] (2019-04-01) Occidental Petroleum's stock price increased by 12% in the first quarter of 2019.

Reasoning:
Marathon Petroleum's 12% year-over-year revenue growth [1] and 15% stock price increase in the first quarter of 2019 [6] indicate strong financial performance and investor confidence. Targa Resources reported a 5% year-over-year revenue growth [2] and an 8% stock price increase [7], suggesting moderate growth. Chevron Corporation's 7% year-over-year revenue growth [3] and 10% stock price increase [8] reflect steady performance. Halliburton experienced a 3% year-over-year revenue decline [4] and a 5% stock price decrease [9], indicating weaker performance. Occidental Petroleum reported a 10% year-over-year revenue growth [5] and a 12% stock price increase [10], showing strong momentum.

Predicted ranking: MPC > OXY > CVX > TRGP > HAL *True ranking:* CVX > MPC > HAL > TRGP > OXY

Leakage analysis. The Temporal Hint model cites the exact post-cutoff closing prices for all five companies—dated May 29, 2020, six months after the cutoff—and uses them directly in its reasoning to compute return differentials (e.g., "its stock price increased from $57.32 on 2019-11-29 to $61.45 on 2020-05-29 [16], indicating positive momentum"). This constitutes circular reasoning: the model ranks stocks by their actual post-cutoff returns rather than forecasting from pre-cutoff fundamentals. Notably, the pre-cutoff closing prices (items [11]–[15], dated November 29, 2019) are not themselves leaked—the leakage lies exclusively in items [16]–[20]. RAG's five leaked items span years into the future: Targa Resources capital expenditure figures from 2021 and Q3 2025 (items [12] and [1], November 2025), Halliburton's full-year 2019 revenue of $3,506M (item [3], January 2020—only available after the annual report), Marathon Petroleum's 2019 dividend yield (item [7], March 2020), and the full-year 2019 Brent crude average of $64/bbl (item [8], December 2019). Both leaking methods produce rankings shaped by hindsight rather than pre-cutoff analysis.

Prediction accuracy. The true ranking is CVX > MPC > HAL > TRGP > OXY. RAG achieves a perfect Spearman $\rho = 1.0$ (Perf $= 1.00$), but this accuracy is contaminated: the model retrieves data from years into the future and effectively "looks up" the answer. The Temporal Hint model's leaked closing prices lead to Perf $= 0.65$ ($\rho = 0.30$); despite knowing the actual May 2020 prices, it overweights absolute price changes over relative returns, misranking CVX. TimeSPEC (Perf $= 0.90$, $\rho = 0.80$) places MPC and CVX in the top two positions based entirely on pre-cutoff Q3 2018 earnings. TEMPO (Perf $= 0.55$, $\rho = 0.10$) correctly identifies MPC as a top performer from

Q3 2019 revenue growth but overweights Occidental Petroleum's revenue recovery, producing a less accurate ranking—without any post-cutoff information.

Evidence faithfulness. TEMPO's 100% coverage indicates that every one of its 10 evidence items is referenced in the reasoning chain: the model cites specific revenue figures and stock price changes for all five companies, and each citation directly supports a claim in the ranking rationale. TimeSPEC's 42.9% coverage reveals that fewer than half of its 14 evidence items are referenced; the verify–regenerate pipeline removes leaked items but the regenerated reasoning does not fully re-anchor to the surviving evidence. The Temporal Hint model's 50% coverage is similarly low despite having 20 evidence items—the reasoning engages only with the leaked closing prices and a subset of the Q3 revenue data, using the former as shortcuts.

Synthesis. TEMPO is the only method that simultaneously achieves zero leakage and complete evidence–reasoning faithfulness, even though its ranking accuracy is lower. RAG's perfect ranking is an artifact of temporal contamination, not superior analysis. TimeSPEC achieves leakage-free status but at the cost of evidence coherence.

I.2 Salary Prediction

We examine how each method handles salary prediction for Patty Mills (cutoff 2021-08-02). Mills played for the San Antonio Spurs in the 2020–2021 season, averaging 10.8 points per game as a veteran bench guard. On August 10, 2021—eight days after the cutoff—he signed a 2-year, $12.07M contract (AAV $6,037,250) with the Brooklyn Nets. This instance is particularly revealing because the signing event falls just days after the boundary, creating a narrow window where parametric knowledge of the Nets signing contaminates the base model's output.

Temporal Hint — OLR: 33.3%, Perf: 0.18, Coverage: 71.4%

Evidence (7 items; 6 unique claims):
[2] (2021-08-10) Patty Mills was a key role player for the Brooklyn Nets during the 2020-2021 season, playing in 56 games and starting 10 of them. [LEAKED]
[5] (2022-07-06) In the 2020 NBA free agency, guards like Lonnie Walker IV signed a 4-year/$52 million deal with the San Antonio Spurs. [LEAKED]
[1] (2021-05-17) Patty Mills averaged 8.5 points per game and 2.8 assists per game during the 2020-2021 NBA season.
[3] (2021-03-14) Patty Mills had a career-high 28.5 points per game in a single game on March 13, 2021, against the New York Knicks.
[4] (2019-07-07) In the 2020 NBA free agency, guards like D'Angelo Russell signed a 4-year/$117 million deal with the Brooklyn Nets.
[6] (2019-07-11) In the 2020 NBA free agency, guards like Tyus Jones signed a 3-year/$24 million deal with the Minnesota Timberwolves.
[7] (2019-07-07) In the 2020 NBA free agency, guards like D'Angelo Russell signed a 4-year/$117 million deal with the Brooklyn Nets.

Reasoning:
Patty Mills had a solid 2020-2021 season, averaging 8.5 points and 2.8 assists per game while playing a key role for the Brooklyn Nets [1]. His performance, including a career-high 28.5 points in a single game [3], showed his ability to contribute in high-pressure situations. However, he is not a primary ball-handler or a high-volume scorer, which limits his market value. Comparing him to other guards who signed deals in the 2020 free agency, such as D'Angelo Russell (4-year/$117M) [4], Lonnie Walker IV (4-year/$52M) [5], and Tyus Jones (3-year/$24M) [6], it is reasonable to expect that Mills would receive a contract in the range of $10–12 million per year.

Predicted salary: $11,000,000 *Actual AAV:* $6,037,250

RAG — OLR: 73.3%, Perf: 0.43, Coverage: 30.0%

Evidence (30 items; 22 leaked, 8 clean):
[7] (2021-08-10) On 10 August 2021, Mills signed with the Brooklyn Nets. [LEAKED]
[16] (2022-07-10) On 10 July 2022, Mills re-signed with the Nets on a two-year, $14.5 million contract. [LEAKED]

[2] (2021-08-07) In 2021, Mills led the Australian Boomers to their first-ever Olympic medal at the Tokyo 2020 Olympics. [LEAKED]
[20] (2024-07-26) Mills represented Australia once again in the 2024 Paris Olympics. [LEAKED]
[23] (2024-07-01) Patty Mills's basic salary for the 2024/25 NBA season is $6,802,950. [LEAKED]
[8] (2021-10-19) On 19 October, Mills made his Nets debut, scoring 21 points on 7-of-7 shooting from deep in a 127–104 loss to the Milwaukee Bucks. [LEAKED]
[14] (2021-12-25) On the NBA Christmas game, Mills tied his career high with 34 points on 8-for-13 from three in a 122–115 win over the Los Angeles Lakers. [LEAKED]
[. . . 15 more leaked items]
[1] (2021-08-01) Patty Mills became a strong contributor off the bench and helped the Spurs win the 2014 NBA championship.
[29] (2021-08-01) The mid-level exception for 2021/22 was projected to be $9,536,000 for standard MLE.
[. . . 6 more clean items]

Reasoning:
Patty Mills has established himself as a reliable and high-performing point guard, particularly known for his three-point shooting and clutch performances. His 2021 season with the Brooklyn Nets was exceptional, with multiple high-scoring games, including a career-high 34 points on Christmas Day and a 10-of-10 three-point shooting performance [8][10][14]. His Olympic success, including leading Australia to its first-ever Olympic medal [2][3][18], further enhances his market value. Mills' recent contract with the Nets was a two-year, $14.5 million deal [16], which suggests that teams are willing to invest in his abilities. Given the 2021 free agency market and the mid-level exception of $9.5 million [29], it is reasonable to expect that Mills could command a contract in the $9–10 million range.

Predicted salary: $9,500,000 *Actual AAV:* $6,037,250

TimeSPEC — OLR: 0.0%, Perf: 0.91, Coverage: 92.3%

Evidence (13 items):
[7] (2021-05-31) Patty Mills averaged 10.8 points, 1.7 rebounds, and 2.4 assists per game in 68 games during the 2020-21 season.
[8] (2021-05-31) Patty Mills shot 37.5% from the 3-point line during the 2020-21 season.
[3] (2020-11-10) The projected NBA salary cap for the 2021 season was $112,414,200.
[5] (2020-11-10) The mid-level exception for NBA teams with cap space started at $4.7 million in the 2020 offseason.
[6] (2020-11-10) The mid-level exception for NBA luxury tax teams started at $5.7 million.
[13] (2021-08-02) The non-taxpayer mid-level exception was approximately $9.3 million in 2021.
[9] (2021-01-01) Patty Mills turns 33 years old in August 2021.
[12] (2014-06-16) Patty Mills won an NBA title with the San Antonio Spurs in 2014.
[. . . 5 more items]

Reasoning:
Patty Mills is a veteran player with a career-high scoring average of 11.6 points per game [10] and a 37.5% 3-point shooting percentage [8]. He has also won an NBA title [12] and led the Australian Boomers to a notable victory [11]. However, he is 33 years old [9], which may affect his market value. His recent performance in the 2020-21 season was solid, averaging 10.8 points, 1.7 rebounds, and 2.4 assists per game [7]. The NBA salary cap for 2021 was $112.4 million [3], and the luxury tax threshold was $136.6 million [4]. The mid-level exception for teams with cap space was $4.7 million [5], and for luxury tax teams, it was $5.7 million [6]. The non-taxpayer mid-level exception was $9.3 million [13]. Given his experience, performance, and age, Mills is likely to receive a contract in the range of the mid-level exception. His age and recent performance may limit the contract to around $5–6 million per year.

Predicted salary: $5,500,000 *Actual AAV:* $6,037,250

TEMPO (Ours) — OLR: 10.0%, Perf: 0.99, Coverage: 100.0%

Evidence (10 items):
[7] (2021-08-11) Patty Mills was 33 years old during the 2021 NBA free agency period. [LEAKED]
[1] (2020-10-01) Patty Mills was a guard for the San Antonio Spurs in the 2020-2021 NBA season.
[2] (2021-05-31) Patty Mills averaged 8.3 points per game and 2.1 assists per game during the 2020-2021 NBA season.

[3] (2021-08-02) Patty Mills was a restricted free agent in the 2021 NBA free agency period.
[5] (2021-08-02) The NBA salary cap for the 2021-2022 season was set at $109.2 million.
[8] (2021-05-31) Patty Mills had a career average of 7.9 points per game and 2.0 assists per game over his NBA career.
[9] (2016-06-13) Patty Mills was a 2-time NBA champion with the San Antonio Spurs in 2014 and 2016.
[10] (2021-05-31) Patty Mills was a key backup guard for the San Antonio Spurs during the 2020-2021 season.
[. . . 2 more items]

Reasoning:
Patty Mills was a guard for the San Antonio Spurs in the 2020-2021 NBA season [1]. He averaged 8.3 points per game and 2.1 assists per game [2]. He was a restricted free agent in 2021 [3], and the Spurs had the right to match any offer sheet [4]. The NBA salary cap for the 2021-2022 season was set at $109.2 million [5], and the luxury tax threshold was set at $132.6 million [6]. Patty Mills was 33 years old [7]. His career average was 7.9 points per game and 2.0 assists per game [8]. He was a 2-time NBA champion with the Spurs [9], and he was a key backup guard during the 2020-2021 season [10]. Based on his performance, experience, and the salary cap constraints, it is reasonable to predict an AAV in the range of $5–7 million.

Predicted salary: $6,000,000 *Actual AAV:* $6,037,250

Leakage analysis. The Temporal Hint model produces two leaked claims out of six unique (OLR $= 33.3\%$). The most consequential is evidence [2] (dated 2021-08-10): "Patty Mills was a key role player for the Brooklyn Nets during the 2020-2021 season." Mills was on the San Antonio Spurs during 2020–2021; his Nets signing occurred on August 10, 2021—eight days after the cutoff. The second leaked claim is evidence [5] (dated 2022-07-06), which fabricates a 4-year/$52M deal for Lonnie Walker IV—a contract that never existed. Walker actually signed with the Los Angeles Lakers in July 2022. The model's reasoning absorbs both leaks: the Nets reference inflates Mills' perceived market value by positioning him on a contending team, while the fabricated comparable contract distorts the salary benchmarking. RAG amplifies leakage catastrophically: 22 of 30 evidence items are flagged, spanning August 2021 to July 2024. Leaked content includes the Nets signing on August 10 (item [7]), Mills' debut on October 19 scoring 21 points on 7-of-7 from three (item [8]), his 2022 re-signing at $14.5M (item [16]), his 2024 Olympic participation (item [20]), and his 2024/25 salary of $6.8M (item [23]). Only 8 of 30 items contain genuinely pre-cutoff information. TEMPO's OLR of 10% reflects a single claim—evidence [7] (dated 2021-08-11), "Patty Mills was 33 years old during the 2021 NBA free agency period." Mills' birthday is August 11, 1988; the age 33 was not determinable until his birthday—nine days after the cutoff. All other nine claims are clean.

Prediction accuracy. The actual AAV is $6,037,250. The Temporal Hint model predicts $11M—82% above actual—because the leaked Nets context inflates Mills' perceived market value. RAG predicts $9.5M (57% above), anchored by the leaked $14.5M re-signing contract and inflated by knowledge of Mills' exceptional 2021–2022 Nets season. The clean methods bracket the true value closely: TimeSPEC predicts $5.5M (9% below) and TEMPO predicts $6M (within 1% of actual). TEMPO's near-perfect prediction (Perf $= 0.99$) demonstrates that the model retains useful pre-cutoff knowledge—career stats, salary cap data, mid-level exception thresholds—sufficient for accurate salary estimation without any post-cutoff information.

Evidence faithfulness. TEMPO's 100% coverage means every one of its 10 evidence items—Spurs career stats, free-agent status, average guard salary, age—is directly referenced in the reasoning chain. The evidence→reasoning→prediction process is fully transparent. TimeSPEC achieves 92.3% coverage: of its 13 evidence items, one goes unreferenced, indicating a minor gap in evidence grounding introduced by the supervisor's filtering. RAG's 30.0% coverage is the most striking: the retriever floods 30 items into the context, but the reasoning references only about 9. The remaining 21 items—including multiple leaked claims about Mills' Nets career—are decorative padding that does not ground the prediction.

Synthesis. TEMPO is the only method that simultaneously achieves near-zero leakage, a calibrated prediction (within 1% of actual), and complete evidence–reasoning faithfulness. TimeSPEC

matches on leakage and prediction calibration but sacrifices minor evidence coherence. The leaking methods produce predictions further from the true value despite having access to more information—their predictions are inflated by post-cutoff context that distorts the salary comparison basis.

I.3 Legal Outcome Prediction

We examine how each method handles legal outcome prediction for *Carol Luther v. Nancy Berryhill* (cutoff 2018-06-04). The case concerns whether the ALJ erred in the handling of a Social Security disability claim. The true outcome is that the petitioner prevails. This instance illustrates how leaked procedural information—court rulings, certiorari petitions, and appellate decisions that occurred after the cutoff—can fundamentally alter the model's reasoning chain and prediction.

Temporal Hint — OLR: 50.0%, Perf: 0.51, Coverage: 50.0%

Evidence (6 items):
[1] (2018-10-01) Carol Ann Luther filed a petition for a writ of certiorari with the Supreme Court of the United States on October 1, 2018, seeking review of a decision by the U.S. Court of Appeals [...] [LEAKED]
[2] (2018-08-28) The U.S. Court of Appeals for the Eighth Circuit affirmed the decision of the U.S. District Court for the District of Minnesota on August 28, 2018, which denied Carol Ann Luther's disability benefits. [LEAKED]
[3] (2018-06-29) The U.S. District Court for the District of Minnesota ruled on June 29, 2018, that Carol Ann Luther was not entitled to disability benefits. [LEAKED]
[4] (1935-08-14) The Social Security Act requires that an individual be unable to engage in any substantial gainful activity due to a medically determinable physical or mental impairment [...]
[5] (1971-05-03) In Richardson v. Perales (1971), the Supreme Court held that the Secretary of Health, Education, and Welfare must provide a full and fair hearing to a claimant [...]
[6] (1986-06-13) In Bowen v. City of New York (1986), the Supreme Court held that the Secretary of Health and Human Services must apply the same standard of review [...]

Reasoning:
The evidence shows that Carol Ann Luther's claim for disability benefits was denied by the U.S. District Court on June 29, 2018, and that this decision was affirmed by the Eighth Circuit on August 28, 2018. The Social Security Act requires that an individual be unable to engage in any substantial gainful activity [4]. The Supreme Court has previously held that the Secretary must apply the same standard of review to disability claims [6]. The Eighth Circuit's decision to affirm the district court's ruling suggests that the court found the Secretary's decision to be supported by substantial evidence. However, the Supreme Court has also emphasized the importance of providing a full and fair hearing [5]. Given the precedents, the Court may scrutinize the Secretary's decision more closely. However, the fact that the lower courts have already affirmed the decision suggests that the petitioner's chances may be limited.

P(petitioner wins): 0.3 *True outcome:* Petitioner wins

RAG — OLR: 20.0%, Perf: 0.88, Coverage: 80.0%

Evidence (5 items):
[4] (2018-06-20) In Lucia v. Securities and Exchange Commission, the Supreme Court held that SEC ALJs are subject to the Appointments Clause, as they exercise significant authority. [LEAKED]
[1] (2018-05-21) The U.S. Supreme Court granted certiorari in Virginia Uranium, Inc., et al. v. Warren, John, et al. and Culbertson, Richard A. v. Berryhill [...]
[2] (2018-04-01) In Luther v. Berryhill, No. 16-55987 (9th Cir. 2018), the Ninth Circuit reversed the denial of disability insurance benefits and supplemental security income for the claimant.
[3] (2017-01-23) Nancy A. Berryhill was appointed as the Acting Commissioner of Social Security under the Vacancies Reform Act, but her appointment was not confirmed by the Senate.
[5] (2017-04-07) Lea Lynch v. Nancy Berryhill, Case No. 17-35308, was filed on April 7, 2017, and involved a Social Security disability claim against Berryhill.

Reasoning:
The evidence shows that Nancy A. Berryhill served as the Acting Commissioner of Social Security under the Vacancies Reform Act, but her appointment was not confirmed by the Senate [3]. This raises questions about the legality of her continued role in administrative proceedings. In Luther v. Berryhill [2], the Ninth Circuit found that an ALJ erred by not adequately addressing a claimant's

VA disability rating, suggesting that administrative decisions involving Berryhill may be subject to scrutiny. The Supreme Court's decision in Lucia v. SEC [4] established that certain ALJs are subject to the Appointments Clause, which could imply that Berryhill's role may also be subject to constitutional scrutiny. The case involving Lea Lynch [5] further demonstrates that Berryhill's role was already under legal challenge. Based on these precedents and the legal uncertainties, the petitioner has a reasonable chance of success, though the outcome is not certain.

P(petitioner wins): 0.65 *True outcome:* Petitioner wins

TimeSPEC — OLR: 0.0%, Perf: 0.94, Coverage: 30.8%

Evidence (13 items):
[5] (2018-01-01) The Decision Review Board (DRB) found that the grievant's decision was "faulty" because it did not adequately evaluate the claimant's condition.
[7] (2018-01-01) Article 5, Section 1 of the parties' agreement acknowledges that ALJs are engaged in duties requiring consistent exercise of discretion, knowledge, and judgment.
[8] (2018-01-01) Article 5, Section 2 of the parties' agreement provides that all judges shall be treated fair and equitably.
[9] (2018-01-01) The Arbitrator determined that the counseling memorandum issued to the grievant operated as discipline.
[11] (2018-01-01) The Arbitrator established a standard that counseling memoranda should only be issued when an ALJ's actions are so egregious that they interfere with the rights of the claimant.
[12] (2018-01-01) The Agency argues that the award is contrary to law because the Arbitrator's interpretation affects management's right to direct employees.
[. . . 7 more items]

Reasoning:
The key issue is whether the Agency violated the parties' agreement by issuing a counseling memorandum to the grievant, an ALJ. The Arbitrator determined that the counseling memorandum operated as discipline and established a standard that such memoranda should only be issued when an ALJ's actions are so egregious that they interfere with the rights of the claimant [11]. The Agency argues that this interpretation is contrary to law because it infringes on management's right to direct employees [12]. However, the parties' agreement explicitly states that ALJs are engaged in duties requiring consistent exercise of discretion [7], and that all judges shall be treated fairly and equitably [8]. The Arbitrator's interpretation appears grounded in the agreement's language. The Agency's argument that the Arbitrator inserted a provision not in the agreement is not clearly supported by the evidence.

P(petitioner wins): 0.75 *True outcome:* Petitioner wins

TEMPO (Ours) — OLR: 0.0%, Perf: 0.64, Coverage: 42.9%

Evidence (7 items):
[1] (2018-01-01) The case involves a challenge to the Social Security Administration's (SSA) interpretation of the term "disability" under the Social Security Act.
[2] (2018-01-01) The petitioner, Carol Ann Luther, argues that the SSA's definition of disability is too narrow and does not properly account for her condition.
[3] (2018-01-01) The respondent, Nancy Berryhill, represents the SSA and defends the agency's interpretation of the disability standard.
[4] (2018-01-01) The Social Security Act defines disability as the inability to engage in any substantial gainful activity by reason of any medically determinable physical or mental impairment.
[5] (1986-06-13) In Bowen v. City of New York (1986), the Supreme Court held that the SSA's interpretation of the Social Security Act is entitled to deference.
[6] (2003-11-12) In Barnhart v. Thomas (2003), the Supreme Court reaffirmed the Chevron deference applicable to agency interpretations of the Social Security Act.
[7] (2010-06-21) In Astrue v. Ratliff (2010), the Supreme Court held that the SSA's interpretation of the term "disability" is entitled to deference.

Reasoning:
The petitioner argues that the SSA's definition of disability is too narrow [2], while the respondent defends the agency's interpretation [3]. The Supreme Court has consistently applied Chevron deference to the SSA's interpretations [5], [6], [7]. This precedent suggests that the Court is likely to defer to the SSA's interpretation. However, the petitioner's argument could still persuade the Court if it finds the SSA's interpretation to be overly restrictive. The Court's willingness to apply Chevron deference

in previous cases suggests that the respondent has a strong position, but the petitioner's argument may challenge this deference if the agency's interpretation is found unreasonable.

P(petitioner wins): 0.4 *True outcome:* Petitioner wins

Leakage analysis. The Temporal Hint model's reasoning is built entirely around leaked procedural events: the district court ruling (June 29, 2018), the Eighth Circuit affirmance (August 28, 2018), and the certiorari petition (October 1, 2018)—all dated after the June 4 cutoff. These events reveal the case's appellate trajectory: denial, affirmance, and Supreme Court review. The model uses this trajectory to reason that "the lower courts have already affirmed the decision," leading to a low 0.3 probability for the petitioner—a prediction grounded not in legal analysis but in knowledge of what actually happened. RAG introduces a single leaked item: the *Lucia v. SEC* decision (June 20, 2018), which established that SEC ALJs are subject to the Appointments Clause. While the remaining four RAG items are pre-cutoff, this post-cutoff precedent strengthens the reasoning that ALJ decisions may face constitutional scrutiny.

Prediction accuracy. The true outcome is that the petitioner wins. The Temporal Hint model's leaked knowledge leads to the least accurate prediction ($P = 0.3$, Perf $= 0.51$): knowing that lower courts denied the claim causes the model to underestimate the petitioner's chances, missing the eventual reversal. RAG ($P = 0.65$, Perf $= 0.88$) and TimeSPEC ($P = 0.75$, Perf $= 0.94$) both predict in the petitioner's favor, though with different evidence bases. TEMPO ($P = 0.4$, Perf $= 0.64$) reasons from pre-cutoff constitutional principles and Chevron deference doctrine, arriving at an uncertain prediction that reflects genuine legal ambiguity rather than leaked procedural knowledge.

Evidence faithfulness. RAG achieves 80.0% coverage (4 of 5 items referenced), the highest among all methods for this instance, because its compact evidence set is well-integrated into the reasoning. TimeSPEC's 30.8% coverage is notably low: of 13 evidence items about ALJ procedural standards, only 4 are referenced in the reasoning. The verify–regenerate pipeline produces a large evidence set but the regenerated reasoning does not fully anchor to it. TEMPO's 42.9% coverage (3 of 7 items) reflects a reasoning chain that engages with the core constitutional arguments but leaves some precedent items unreferenced. The Temporal Hint model's 50% coverage (3 of 6 items) shows that the reasoning engages primarily with the three leaked procedural events, treating the pre-cutoff legal precedents as secondary.

Synthesis. Legal outcome prediction presents a different dynamic: the Temporal Hint model's leaked knowledge of the appellate trajectory actually *hurts* its prediction by anchoring it to the denial rather than the reversal. TEMPO produces the most epistemically honest prediction—uncertain but grounded in pre-cutoff legal doctrine—without any leaked information. TimeSPEC achieves the highest accuracy among clean methods, but with substantially lower evidence–reasoning coherence than TEMPO.

J Prompt specifications

We provide key prompts verbatim for reproducibility. Template variables are shown as `{variable}`. All prompts listed below are extracted directly from the codebase; minor formatting adjustments (line breaks, indentation) are made for readability.

J.1 Task Prompts (Base, TEMPO, RAG, TimeSPEC)

The same system–user prompt pair is shared across Base, TEMPO, RAG generation, and the TimeSPEC generator. We show the Stock prompt in full; Salary and Legal follow an identical structure with task-specific adaptations noted below. All methods produce a structured JSON output with an `evidence` array (each item containing `id`, `fact`, and `source_date`), a `reasoning` string, and a task-specific prediction field (`ranking` for stock, `predicted_salary` for salary, `probability_petitioner` for legal).

Stock ranking — System prompt

You are an expert stock analyst. Your task is to rank a set of companies by their expected stock return over a specified evaluation period.

STRICT TEMPORAL CONSTRAINT: You may ONLY use information that was publicly available BEFORE `{cutoff_date}`. Do NOT use any knowledge of events, earnings reports, market movements, announcements, or developments from on or after `{cutoff_date}`. If uncertain whether a piece of information predates the cutoff, omit it.

EVIDENCE INTEGRITY: Evidence must be factually genuine. Do not fabricate numbers, dates, revenue figures, or events. Each evidence item must reflect a real, publicly known fact. If you are uncertain about a specific number, state the fact qualitatively rather than inventing a precise figure.

You MUST follow this three-step procedure:
STEP 1 — EVIDENCE: List every verifiable fact you will use. Each fact must be concrete and specific; each must have been publicly known before `{cutoff_date}`. For EACH fact, include a `source_date` field (YYYY-MM-DD) indicating when this information was FIRST PUBLICLY ANNOUNCED (earnings release date, press release date, SEC filing date). This must be the ANNOUNCEMENT date, NOT the period end. Write each fact as a plain statement; do not include citation markers.
STEP 2 — REASONING: Write a reasoning paragraph that cites evidence by number (e.g. [1], [3]). You MUST cite every evidence item at least once, INLINE next to the analysis it supports. Do NOT introduce any new facts not listed in your evidence.
STEP 3 — PREDICTION: Give your final ranking based solely on the reasoning above.

OUTPUT FORMAT — a single JSON object, no markdown fences:
`{"evidence": [...], "reasoning": "...", "ranking": ["BEST", ..., "WORST"]}`

Stock ranking — User prompt

Sector: `{sector}`
Evaluation period: `{start_date}` to `{end_date}`
Companies to rank:
- `{ticker}` (`{company_name}`), ...

Salary and Legal Variants. Both follow the identical three-step structure (temporal constraint, evidence integrity, evidence→reasoning→prediction) with task-specific role descriptions. *Salary:* "You are an expert sports salary analyst"; predicts a free agent's AAV in USD; user prompt includes `{sport}`, `{year}`, `{player_name}`, and optional `{status}` (e.g., unrestricted). *Legal:* "You are an expert Supreme Court analyst"; predicts the probability the petitioner wins; user prompt includes `{case_name}`, `{term}`, `{petitioner}`, `{respondent}`, and optional `{summary}`.

J.2 RAG query generation

RAG uses the same system–user task prompt as Base, with retrieved passages prepended to the user prompt as numbered plain text. Query generation is a single Gemini Flash call with the following prompt (not a system–user pair):

RAG query generation (single prompt)

You are generating web search queries to gather relevant evidence for a task. All evidence must predate the cutoff date. You will receive the same task description that an analyst would receive. Generate search queries that would help you find useful information to answer this task.

CUTOFF DATE: `{cutoff_date}`
TASK DESCRIPTION: `{user_prompt}`

Generate between `{min_queries}` and `{max_queries}` search queries. Each query should target a DISTINCT aspect of the information you need. Include the year in each query to anchor temporal relevance.
Output as JSON array: `[{"query": "..."}, ...]`

Query ranges: stock 7–12; salary and legal 3–6.

J.3 TimeSPEC prompts

Claim Extraction. A single Gemini call extracts atomic factual claims from the completion's evidence and reasoning for temporal verification.

TimeSPEC claim extraction (single prompt)

Task: Extract ALL factual claims from the rationale below.
Requirements: (1) Comprehensive—every factual assertion; (2) Faithful—only what is stated; (3) Self-contained—each claim understandable alone; (4) Atomic—one verifiable fact per claim.

Context:
Prediction task: `{task_description}`
Target event: `{event_description}`
Knowledge cutoff: `{cutoff_date}`

Claim Categories:
Group A (temporally verifiable): A1 discrete event, A2 state/measurement, A3 publication/announcement, A4 outcome (always leaked), A5 consequential (always leaked).
Group B (non-temporal): B1 background, B2 definitional.
Exclude predictions, judgments, speculation.

Rationale: `{rationale}`

Output JSON: `{"claims": [{"claim_id": 1, "text": "...", "original_span": "...", "temporal_ref": "...", "category": "A1", "category_reason": "..."}, ...]}`

Temporal Verification. One Gemini call per chunk of claims, each paired with web-search results retrieved without a date filter. The prompt determines when each fact was first publicly available and labels it as leaked or clean relative to the cutoff (full prompt in the released codebase).

Regenerator. System prompt is identical to the task system prompt. User prompt is the RAG-augmented user text with an appended constraint: `"CONSTRAINT -- do NOT use the following facts in your analysis: 1. {claim_text} ... Rewrite your analysis without these facts."`

Aggregator. System prompt appends a `CLOSED-WORLD CONSTRAINT`: "For STEP 1, you MUST select evidence exclusively from the VALIDATED FACTS list below. Do NOT introduce any facts not present in that list." User prompt is "VALIDATED FACTS: 1. `{fact_text}` ..." followed by the original task description. Validated facts are plain text (no source dates or metadata).

J.4 Evaluation: Date-Plausibility Verifier

The shared evaluation pipeline uses the following single prompt (via Gemini Flash, 80 claims per batch) to detect temporal leakage by verifying whether each evidence item's declared `source_date` is plausible.

Date-plausibility verifier (single prompt)

You are a temporal audit verifier for financial, sports, and legal analysis.
Cutoff date: `{cutoff}`

For each evidence item you are given: the factual claim text and the declared_source_date (YYYY-MM-DD). Your task: verify whether the declared_source_date is PLAUSIBLE for this type of information.

VERIFICATION RULES:
- Full-year financial results for year Y are first published in Jan–Feb of Y+1.
- Quarterly results are published 1–2 months after quarter end.
- Stock prices are available on the trading date itself.

- Acquisition/merger: use press release or 8-K filing date.
- Contract signings: use announced signing date.
- Court decisions: use decision or opinion release date.
- SEC 10-K: typically 60–90 days after fiscal year end.
- If declared date is too early, set `plausible=false` and provide earliest realistic `corrected_date`. Otherwise set `plausible=true` and `corrected_date=null`.

ITEMS: `{items_json}`
Output a JSON array: `[{"index": 0, "plausible": true, "corrected_date": null, "reason": "..."}]`

When the date-plausibility check returns `plausible=false`, the `corrected_date` replaces the declared date for leakage scoring. A claim is marked leaked if its effective date (corrected or declared) falls after the cutoff.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction claim that TEMPO reduces OLR from 2–13% to 0.6–3.7% across all six conditions while maintaining or improving task performance (Table 2). The convergence claims (monotone leakage decrease, linear convergence to a neighborhood of the leak-free optimum, and performance improvement once compliant) are stated with their assumptions in Theorem 1 (Section 3.3). The introduction scopes the claims to the evidence the model states, and the Limitations paragraph (Section 6) lists what TEMPO does not guarantee.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The Limitations paragraph (Section 6) states that TEMPO constrains the evidence and rationale the model states, not its internal computation. TEMPO requires open-weight models and task-specific temporal annotations, and is evaluated on two Qwen models. Training and evaluation also share one verifier, and we validate it in Appendix H.1. Section 5 and Appendix A.5 explain why leakage plateaus above zero, and Section 5.1 reports where TimeSPEC outperforms TEMPO (legal accuracy).

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Propositions 1–2, Theorem 1, and Corollary 1 are stated with explicit assumptions (smoothness, bounded moments, compatible approximation, PL condition) in Section 3.3. Full proofs appear in Appendix A with all assumptions restated. The PL condition is justified both theoretically (unique global minimum, overparameterization) and empirically (all six runs converge). Appendix A.5 relates the theorem's residual term to the leakage floor observed in experiments.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All hyperparameters are listed in Table 6 (Appendix C). The training algorithm is provided in Algorithm 1 (Appendix C). Dataset construction details, data sources, and quality checks are in Appendix D. Evaluation uses greedy decoding, and we release all evaluation outputs and code, so every reported metric can be recomputed. Both base models have since been retired from the Tinker training API, so re-training requires another provider (Appendix C). Full prompts are in Appendix J.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code and data are available at `https://github.com/nu-snail/TEMPO`. The repository includes training scripts, evaluation pipelines, dataset files, and instructions for reproducing all reported results.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Section 4 describes tasks, data splits (with train/eval/control separation logic), models, baselines, and evaluation metrics. Appendix C provides the full hyperparameter table (optimizer, learning rate, batch size, group size, clipping parameters, KL coefficient), stability techniques, and inference configuration.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Evaluation uses greedy decoding, so outputs are deterministic. Metrics are averaged over 100–137 evaluation instances per condition, and consistent patterns across six conditions (two models, three tasks) support robustness. The memorization and reconstruction tests report 95% bootstrap confidence intervals, and a label-noise simulation tests the robustness of the method ranking to verifier errors (Appendix H). Training variance across seeds is not reported, since each condition was trained once.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4.2 states that training takes 7–22 hours per model-task combination. Appendix C describes the setup (the Tinker managed fine-tuning API with four concurrent workers, no local GPUs, and judge and search calls through OpenRouter) and reports the wall-clock time of every training run (Table 5, about 81 hours in total).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses publicly available data (Yahoo Finance, Spotrac, Oyez, CourtListener) and open-weight models. No human subjects are involved. The work aims to improve evaluation integrity by preventing temporal contamination in backtesting.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The positive impact is more trustworthy LLM backtesting. The main risk is that TEMPO's clean evidence could make predictions look more legitimate while the model still draws on memorized outcomes internally. The Limitations paragraph (Section 6) states this risk, and Appendix H tests for it with a memorization probe and a reconstruction test, which find no sign of such reliance.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We release code, datasets built from publicly available financial, sports, and legal information, and evaluation outputs. We release no trained model weights. None of these pose a high risk for misuse.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Data sources are credited with URLs and footnotes (Yahoo Finance/yfinance, Spotrac, Oyez API, CourtListener API) in Appendix D. The base models (Qwen3 family) are cited. External APIs (Perplexity, OpenRouter, Gemini) are identified with URLs. The independent judges used for verifier validation (Claude, Kimi) are identified in Appendix H.1.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The repository includes dataset files with documentation, training scripts with configuration files, and a README describing how to reproduce results. Dataset construction methodology is detailed in Appendix D with quality assurance checks. The repository also includes the 120-claim human-labeled verifier benchmark (Appendix H.1).

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This work does not involve crowdsourcing or human subjects research.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This work does not involve human subjects research. The human labels for the verifier benchmark and audit were produced by the authors.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are central to the methodology: (1) Qwen3 models are the policy being trained (Sections 3–4); (2) Gemini 3 Flash Preview serves as the date-plausibility verifier in both training and evaluation (Section 3.1, Appendix F.1); (3) Gemini is used for RAG query generation and TimeSPEC claim verification (Appendix E); (4) Claude and Kimi serve as independent judges, and Gemini checks company names, in the verifier validation (Appendix H.1); (5) the base Qwen models serve as closed-world evaluators for DCLR and the reconstruction test (Appendices F.3 and H.3). All LLM usage is documented with model names and access methods