Proof of Time: A Benchmark for Evaluating Scientific Idea Judgments

Anonymous ACL submission

Abstract

Large language models are increasingly being used to assess and forecast research ideas, yet we lack scalable ways to evaluate the quality of models' judgments about these scientific ideas. Towards this goal, we introduce **Proof of Time (PoT)**, a semi-verifiable benchmarking framework that links scientific idea judgments to downstream signals that become observable later (e.g., citations and shifts in researchers' agendas). PoT freezes a pre-cutoff snapshot of evidence in an offline sandbox and asks models to forecast post-cutoff outcomes, enabling verifiable evaluation when ground truth arrives, scalable benchmarking without exhaustive expert annotation, and analysis of human–model misalignment against signals such as peer-review awards. In addition, PoT provides a controlled testbed for agent-based research judgments that evaluate scientific ideas, comparing tool-using agents to non-agent baselines under prompt ablations and budget scaling. Across 30K+ instances spanning four benchmark domains, we find that, compared with non-agent baselines, higher interaction budgets generally improve agentic performance, while the benefit of tool use is strongly task dependent. By combining time-partitioned, future-verifiable targets with an offline sandbox for tool use, PoT supports scalable evaluation of agents on future-facing scientific idea judgment tasks. [1]

1 Introduction

The research community invests enormous effort into producing, sharing, and debating ideas, yet the infrastructure used to evaluate scientific ideas still largely emphasizes *immediate* judgments: peer review at submission time, short-horizon leaderboard comparisons, and static benchmark evaluations that are necessarily time-constrained and often idiosyncratic. These mechanisms are indispensable, but what often matters most is whether an idea *stands the test of time*. Meanwhile, "idea quality" is hard to define and expensive to label: human judgments are costly, sometimes inconsistent, and inevitably limited by what is known at decision time. This motivates a complementary paradigm: alongside expert judgment, we ask models to make forecasts that can later be verified against naturally arriving signals. Many consequential scientific questions are intrinsically *time-indexed*: which contemporaneous papers will accrue more citations, which submissions will be recognized by award committees, how benchmark metrics will evolve, or which directions a scholar will pursue next.

At the same time, "AI for Science" has accelerated rapidly: beyond post-hoc analysis, machine learning systems are increasingly embedded directly in discovery pipelines across domains (e.g., hypothesis generation, candidate design, and experiment steering) (Wang et al., 2023a; Karpatne et al., 2025). High-accuracy protein structure prediction is a canonical example (Jumper et al., 2021). This progress has renewed interest in *metascience* (Munafò et al., 2017; Ioannidis et al., 2015), which aims to understand how science evolves, and in tools that can forecast scientific trajectories such as topic growth and emerging areas (Ofer et al., 2023; Wang et al., 2013; Chen, 2006; Kleinberg, 2002). As AI systems become increasingly capable of multi-step reasoning and tool use (Yao et al., 2023; Schick et al., 2023; Wang et al., 2023b), a natural question emerges: can models judge which ideas will ultimate prove impactful and durable, and can we evaluate such forecasts at scale in a way that is both principled and reproducible?

We introduce Proof of Time (PoT), a semi-verifiable benchmarking framework for evaluating models/agents' scientific idea assessment and forecasting ability through time. PoT operationalizes a simple principle: freeze the evidence available before a cutoff time, require models to forecast out-

[1] Code, data, and evaluation scripts will be publicly released.

comes after the cutoff, and score predictions when the world reveals the answer. Concretely, PoT packages each instance with a pre-cutoff snapshot of evidence placed in an offline sandbox, then evaluates predictions against post-cutoff signals such as citation counts, award tiers, benchmark trajectories, and shifts in researchers' agendas. This design has three practical advantages. First, it supports *verifiability*: gold labels are not subjective annotations but outcomes that can be checked later. Second, it supports *scalability*: instances can be generated programmatically from public metadata and refreshed over time without requiring exhaustive expert labeling. Third, it supports *misalignment analysis*: because some outcomes reflect human judgment (e.g., peer-review awards), PoT enables systematic study of where model-based assessments diverge from human processes.

PoT is also motivated by a pragmatic concern in modern scientific workflows: when and why do agentic systems help? Tool-using agents are increasingly proposed as assistants for literature review, data analysis, and scientific writing (Boiko et al., 2023; Lála et al., 2023; Mitchener et al., 2025), yet improvements are often reported in ways that conflate task difficulty, tool access, and test-time budget (Schaeffer et al., 2023; Huang et al., 2023). In PoT, we study a narrower but central, cross-cutting scope: tool-using agents acting as idea judges, where tool use supports evidence exploration and aggregation in an offline sandbox. PoT provides a controlled environment to study these trade-offs because tasks can be run with and without tools, under fixed offline constraints, and with explicit limits on interaction steps. In doing so, PoT does not assume that agents are always better (Kim et al., 2025; He et al., 2025). Instead, it quantifies how much gain agents provide, which task types benefit most from evidence exploration, and where agent overhead yields diminishing returns.

In this paper, we formalize PoT as a time-partitioned benchmark design and instantiate it across four domains spanning **impact prediction** (citations), **scientific value assessment** (peer-review awards), **research evolution** (faculty research directions), and **technological frontier forecasting** (SOTA benchmark trajectories). We evaluate a range of models under both direct-generation and tool-using agent pipelines, and we conduct controlled ablations over tool access, offline prompting, and message-budget scaling. Finally, we analyze recurrent failure modes that reveal where models struggle to translate pre-cutoff evidence into reliable forecasts and where agentic reasoning breaks down. These results position PoT as a step toward evaluating models' scientific idea judgement; that is, how well models anticipate which research directions will be most impactful in the future.

Our contributions are threefold: (1) **Benchmark design:** we propose PoT, a time-partitioned, semi-verifiable framework that links idea-level judgments to downstream signals that become observable later; (2) **Agent-native evaluation:** we introduce an offline sandbox protocol that makes tool use measurable and supports controlled ablations over tool access, structured offline prompting, and message-budget (test-time) scaling; and (3) **Empirical findings:** across 30K+ instances spanning four domains, we show that agents demonstrate large gains on tasks that require evidence exploration and aggregation, but smaller or inconsistent improvements on structured prediction tasks.

2 Related Work

PoT intersects three pillars of research: (i) forecasting scientific impact and idea quality, (ii) time-partitioned evaluation under contamination risk, and (iii) tool-using agents and their evaluation in realistic workflows.

2.1 Forecasting scientific impact and idea quality

Scientometrics has long studied how papers accrue impact over time, typically modeling citations or related proxies using bibliometric features, topic signals, and network structure (Price, 1965; Barabási and Albert, 1999; Griffiths and Steyvers, 2004; Wang et al., 2013; Acuna et al., 2012). More recently, work has explored whether *LLMs themselves* can serve as predictive models for impact signals such as citation counts, framing citation forecasting as a text-understanding or judgment task (de Winter, 2024; Singla et al., 2023). At the same time, there is growing interest in the broader question of evaluating *research quality* (and the limits of doing so with automated systems), including perspectives that emphasize how "quality" is socially mediated and difficult to reduce to a single metric (Thelwall, 2025; Hicks et al., 2015). Relatedly, studies of LLM-assisted scholarly writing suggest that models may reproduce—and in some cases amplify—existing citation patterns and biases, raising concerns about feedback loops if model judgments

become part of the scientific pipeline (Algaba et al., 2025). PoT differs from prior forecasting work by emphasizing *benchmarking*: we provide standardized, time-partitioned tasks with frozen evidence and later-arriving labels, enabling controlled comparisons across models, agents, and budget/tool configurations.

2.2 Temporal generalization, live evaluation, and contamination

Web-scale pretraining makes static benchmarks increasingly vulnerable to benchmark data contamination (BDC), which can inflate scores and reduce the meaningfulness of comparisons (Xu et al., 2024). A complementary line of work argues for *time-aware* or continuously refreshed evaluation to preserve measurement validity as model capabilities and training corpora evolve. Recent "live" benchmarks operationalize this idea by collecting new questions over time and scoring them objectively, limiting both leakage and judge bias (White et al., 2025; Jain et al., 2024). Other approaches explicitly construct contamination-resistant variants of classic test sets, including counterfactual or refreshed versions designed to reduce memorization effects (Zhao et al., 2024), and interactive evaluation frameworks that probe whether a response reflects recall versus grounded understanding (Yu et al., 2024). PoT adopts the same spirit as the "live" benchmark–prioritizing *temporal partitioning* and frozen evidence–but targets a distinct object of evaluation: *scientific research-idea assessments* tied to downstream signals (citations, awards, benchmark trajectories, and research-agenda shifts) that become observable post-cutoff.

2.3 Agents, tool use, and evaluation in realistic workflows

Tool-using agents are increasingly evaluated in settings where success depends on multi-step interaction, state tracking, and adherence to constraints. Recent benchmarks emphasize realistic, time-consuming tasks and dynamic interaction loops for agents (e.g., web tasks and conversational tool use) (Yoran et al., 2024; Yao et al., 2024; Lu et al., 2025), while surveys highlight a trend toward more realistic evaluations that measure not only success rates but also cost-efficiency, robustness, and safety (Yehudai et al., 2025). In parallel, evidence accumulates that LLM-based judging can be inconsistent or unfair across groups or styles, motivating evaluation designs that avoid subjective judges when possible (Wang et al., 2024). PoT treats *agentic capability* as a central evaluation dimension: we compare direct generation against tool-using agents under controlled ablations (tool access, offline constraints, and message-budget scaling), using tasks whose labels are externally verifiable rather than judge-dependent.

3 Benchmark Design

PoT is built around a simple, semi-verifiable principle:

> *Freeze evidence at time t_0; ask the model to predict signals observed at $t_1 > t_0$; score once the world reveals t_1.*

Here, "evidence" refers to the *pre-cutoff artifacts and metadata available at time* t_0 (e.g., paper metadata, historical counts, leaderboards) that may be informative for evaluating scientific ideas, but do not contain post-cutoff outcomes. We call PoT *semi-verifiable* because the benchmark uses verifiable downstream outcomes as imperfect proxies for idea quality. The signal is verifiable, but the target construct is not directly observable. Unlike static knowledge benchmarks, PoT treats many research-relevant judgments as inherently *time-indexed*, e.g., whether an idea becomes influential, whether a paper receives an award, whether a benchmark trajectory accelerates, or whether a researcher's agenda shifts. These outcomes are not always knowable at prediction time, but they become observable later. PoT operationalizes this by (i) constraining solvers to a pre-cutoff snapshot of evidence inside a network-isolated sandbox, and (ii) defining targets that are prospectively verifiable as time passes.

3.1 Formal Setup: Time-Partitioned Evaluation Under Evidence Constraints

Each PoT instance is defined by a tuple $(\mathcal{E}_{\leq t_0}, q, y_{t_1})$, where $\mathcal{E}_{\leq t_0}$ is the evidence snapshot available up to cutoff t_0, q is a query over candidate items (papers, researchers, benchmarks), and y_{t_1} is a label derived from a signal observed at t_1. A solver receives $(\mathcal{E}_{\leq t_0}, q)$ and outputs $\hat{y}$; evaluation compares $\hat{y}$ against y_{t_1} once it becomes available. Crucially, PoT separates frozen evidence, i.e., *what the model may use*, from post-cutoff outcomes, i.e., *what defines correctness*, thereby enabling semi-verifiable evaluation of future-facing judgments and automatic future updates.

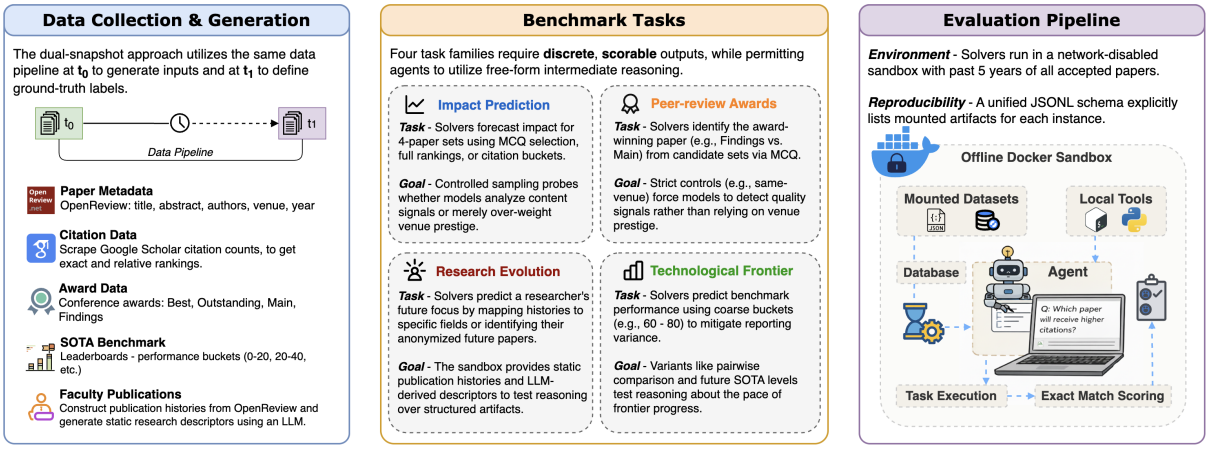

Figure 1: Overall Proof-of-Time benchmark workflow. The left panel describes our dataset creation pipeline; the middle panel emphasizes the four task families and their verifiable post-cutoff signals; the right panel illustrates offline sandbox execution with optional tool-using agents and automatic scoring.

3.2 Offline Sandbox as a Benchmark Primitive

To minimize leakage through opportunistic retrieval and to make tool use measurable, PoT runs solvers in a sandboxed, offline environment. Networking is disabled, and each task instance mounts a *manifested* set of read-only artifacts. This forces agents to rely on (i) the evidence we explicitly provide and (ii) their own reasoning and analysis routines, rather than web search. When run in agentic mode, solvers may use only a restricted tool set, including local file operations, Python analysis, and a text editor, to explore the snapshot, compute features, and justify decisions. This design makes agent improvements interpretable: any gain must come from better use of the same frozen evidence, not from hidden access to fresher information.

3.3 Four Benchmark Domains and Their Task Structure

PoT instantiates four task families, each probing a distinct notion of future-facing research judgment and a distinct reasoning profile. In all cases, we emphasize *discrete, automatically scorable outputs* (MCQ choice, ranking, or bucket), while allowing agentic solvers to use free-form intermediate reasoning internally.

Impact prediction. *Citations* provides a quantitative proxy for paper influence. We construct instances by time-partitioning the literature: the solver is given pre-cutoff evidence from papers published in top NLP venues during 2021–2024 (ACL, NAACL, EMNLP) with titles, abstracts and author information provided, together with citation counts as of October 2025, and is asked to forecast the citation counts of a *candidate set* of four post-cutoff papers (titles, abstracts and authors provided) published in ACL or NAACL 2025. PoT supports three output formats. In *MCQ* instances, the solver selects which candidate will have the most citations by the target horizon. In *Ranking* instances, the solver orders all candidates by predicted citations. In *Bucket* instances, the solver maps each paper to a coarse citation range to reduce sensitivity to minor fluctuations; we use buckets such as 0–10, 10–50, 50–200, and 200+. To reduce temporal and venue confounds, each instance in *MCQ* and *Ranking* uses a matched candidate set of four papers drawn from the same venue–year combination, so comparisons are not driven by natural exposure effects such as earlier publication or venue-wide visibility.

Scientific value assessment. *Awards* provide an externally published, human-judgment signal of research excellence and novelty. We construct instances by time-partitioning the literature within the same three venues (ACL, NAACL, EMNLP). The solver is given a pre-cutoff evidence pool of accepted papers from 2021–2024 with their titles, abstracts, and authors provided. Post-cutoff evaluation papers are drawn from 2025 in the same venues, and the task is to predict the paper's award tier. **For *Awards* only**, we additionally include a diagnostic *pre-cutoff* evaluation split that asks the same award-tier prediction question for papers within the evidence window, primarily to probe reliance on memorized or training-exposed information rather than evidence-grounded inference.

Family	What it evaluates	Instance setup (candidates & evidence)	Outputs	Verifiable signal
Impact prediction *(Citations)*	Forecasting paper influence from limited pre-cutoff cues.	**Candidates:** 4 papers from top AI conferences. **Evidence:** titles, abstracts, authors, and citation counts of historical papers (2021–2024).	MCQ (most cited) Ranking Bucket	Post-cutoff citation counts (Google Scholar).
Peer-review Awards *(Awards)*	Alignment with peer-review judgments and ability to predict award-tier outcomes.	**Candidates:** 4 papers from top AI conferences. **Evidence:** titles, abstracts, authors, and award tiers of historical papers (2021–2024).	Tier MCQ: Findings / Main / Outstanding / Best	Official conference award lists.
Research evolution *(Faculty)*	Longitudinal reasoning about research trajectories and author/topic identification.	**Candidates:** professor post-cutoff publication titles/abstracts with authors removed. **Evidence:** professors' pre-cutoff publications.	Professor → field (MCQ) Paper → author (4 + None) Field focus (MCQ / open)	Post-cutoff publications and/or curated author–field labels.
Technological Frontier *(SOTA)*	Extrapolating benchmark progress and model performance under uncertainty.	**Candidates:** leaderboard descriptions and metrics. **Evidence:** historical model performances.	Bucket MCQ: 0–20 / 20–40 / 40–60 / 60–80 / 80–100	Post-cutoff leaderboard/benchmark updates.

Table 1: PoT task families. Each family is instantiated with automatically scorable outputs (MCQ/ranking/buckets) under an offline pre-cutoff evidence snapshot; gold labels are derived from post-cutoff, externally verifiable signals.

Research evolution. The *Faculty* task family evaluates whether solvers can extrapolate from a researcher's publication history to infer their near-future focus, capturing continuity and drift in a researcher's agenda. Evidence is restricted to each professor's publications from 2021–2024, and targets are defined using 2025 publications. We include (i) professor → field prediction (choose from a fixed taxonomy), (ii) professor → article identification (select which anonymized 2025 candidate paper belongs to the professor, with authors removed and a "None" option), and (iii) papers → field classification for sets of recent papers. To support these tasks under offline constraints, the sandbox provides per-professor publication histories and auxiliary summaries such as LLM-derived keywords and area descriptors computed from pre-cutoff publications only. This allows us to study whether agents can benefit from structured intermediate artifacts, and where such summarization introduces errors or bias.

Technological frontier forecasting. The *SOTA* task measures whether solvers can reason about frontier model performance and the pace of benchmark progress. PoT compiles normalized snapshots of popular benchmarks and leaderboards as of October 2025 and asks solvers to predict performance in coarse buckets. We use buckets to improve robustness to prompt sensitivity and evaluation variance; for example, a five-level scale may represent performance ranges from weak to near-ceiling (e.g., 0–20, 20–40, 40–60, 60–80, 80–100). Variants include single model-benchmark bucket prediction and pairwise model comparison. We include next-year SOTA forecasting as a forward-compatible PoT task. It is not scored in this snapshot, but it becomes automatically verifiable as new leaderboard results are published.

4 Experiments

4.1 Task suite and scoring

We evaluate a suite of tasks spanning citations, peer-review awards, faculty research evolution, and SOTA trajectory prediction. Tasks are designed to be automatically scorable via discrete outputs, while allowing free-form intermediate reasoning inside the sandbox. We report exact-match accuracy for all tasks.

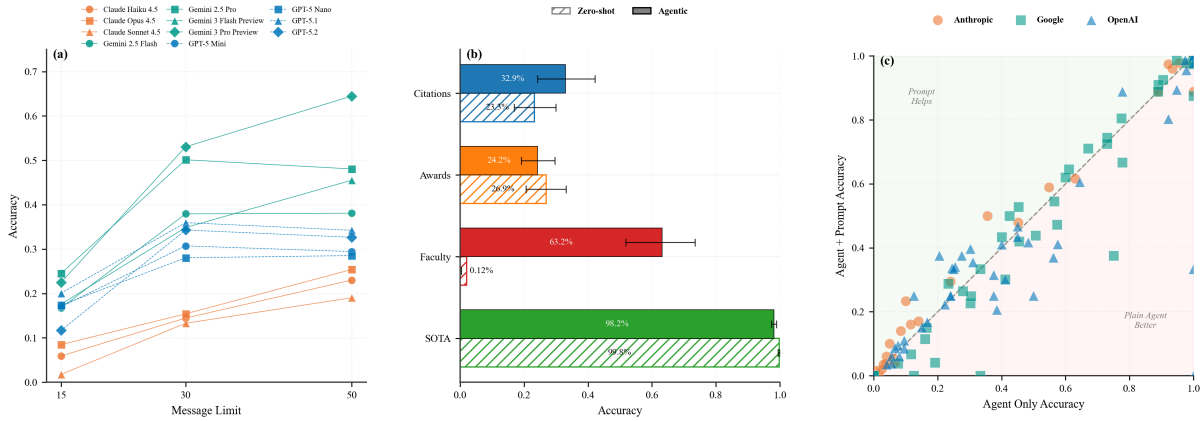

Figure 2: Core results. (A) Test-time compute scaling: accuracy vs. message limit (15/30/50) for each model. (B) Task-family comparison of zero-shot vs. agentic performance at message limit 50, shown as average accuracy across models; error bars indicate variability across task–model combinations. (C) Effect of adding a structured prompt on top of the same agent loop at message limit 50; points above the diagonal indicate the prompt helps.

4.2 Solvers: direct generation vs. tool-using agents

We compare three solver configurations to quantify when agentic systems help, and how much gain they provide beyond naive assumptions that agents are always better.

1. **Zero-shot (direct generation).** The model answers from the prompt only, without tools or local data access. This tests parametric-only forecasting ability.

2. **Agentic.** A single tool-using agent runs in a Docker sandbox with access to `bash`, `python`, and a text editor, iteratively gathering evidence from the offline snapshot.

3. **Agentic + structured prompt.** Same agent loop as (2), but with an additional structured preamble that emphasizes offline-only operation and prescribes a more explicit tool-use protocol. This isolates prompt-policy effects under a fixed agent loop.

4.3 Test-time compute via message limits

To study inference-time scaling in an agentic setting, we vary a **message limit**: the maximum number of environment interaction turns before the agent must finalize an answer. We evaluate budgets of 15, 30, and 50 messages, corresponding to low, medium, and high test-time compute.

4.4 Models

Table 2 summarizes the model suite evaluated in our experiments, covering the latest frontier models from Anthropic, Google, and OpenAI. All models are run under identical task definitions and scoring. For agentic runs, the environment and tool interfaces are held constant across models.

Provider	Models
Anthropic	Claude Opus 4.5; Claude Sonnet 4.5; Claude Haiku 4.5
Google	Gemini 3 Pro Preview; Gemini 3 Flash Preview; Gemini 2.5 Pro; Gemini 2.5 Flash
OpenAI	GPT-5.2; GPT-5.1; GPT-5 Mini; GPT-5 Nano

Table 2: Model suite used in our experiments. We evaluate representative frontier models across three major providers.

4.5 Implementation details

All agentic runs use a single-agent ReAct loop in a sandboxed environment. We log full interaction traces, tool calls, and termination status, including runs that hit the message limit. This enables downstream analysis of failure modes such as non-convergence and redundant exploration.

5 Results

We organize results around four questions: (i) how performance scales with test-time compute, (ii) whether tool-using agents outperform direct generation, (iii) whether a structured agent prompt helps beyond vanilla ReAct, and (iv) whether conclusions change under post-cutoff (contamination-resistant) evaluation. Unless otherwise specified, we report **sample-weighted exact-match accuracy**.

5.1 RQ1: Test-time compute scaling

Figure 2A shows accuracy as a function of message limit (15/30/50) for each model in the *agentic + structured prompt* setting. Across most models, increasing the budget yields large improvements, indicating that performance is often limited by the amount of evidence-gathering and verification the agent can perform at test time.

Scaling behavior varies across model families. Claude models show the steepest gains, suggesting they convert additional interaction into more effective retrieval and verification. Gemini models show strong initial performance but more moderate marginal gains. GPT models improve with budget but tend to plateau earlier, yielding smaller gains at higher limits. We include the full numerical scaling table in Appendix Table 7.

5.2 RQ2: Agentic vs. zero-shot performance

RQ2 asks when tool use helps under PoT's evidence constraints. Because the frozen snapshot $\mathcal{E}_{\leq t_0}$ contains only pre-cutoff context (not post-cutoff labels), any gains from agentic evaluation must come from better use of the same snapshot.

Figure 2B compares zero-shot and agentic performance at message limit 50, aggregated by task family. The effect is highly task-dependent. *Faculty* shows the largest improvement, with agentic accuracy rising from near-zero in zero-shot to roughly two-thirds when tools are enabled. *Citations* shows a moderate gain (about +10pp), whereas *Awards* changes little on average. *SOTA* remains near ceiling in both settings, indicating limited headroom for tool use on this coarse bucketed variant. Overall, tool use yields its largest benefits on families where additional snapshot exploration appears most valuable, while improvements are smaller or muted when tasks are already easy or offer limited headroom.

5.3 RQ3: Structured agent prompt effects

Figure 2C shows that the structured agent prompt has *family-dependent* effects. Overall, Claude models tend to benefit from the added behavioral constraints, GPT models are often neutral or slightly worse, suggesting the prompt may be over-prescriptive, and Gemini models show mixed outcomes. These results indicate that prompt policies are not a universal "agent upgrade" and that model-specific prompt tuning can matter.

Model	Pre-cutoff	Post-cutoff	Δ
Claude Haiku 4.5	2.0%	12.3%	+10.3pp
Claude Opus 4.5	1.5%	23.1%	+21.6pp
Claude Sonnet 4.5	0.0%	2.5%	+2.5pp
Gemini 2.5 flash	47.2%	29.5%	-17.7pp
Gemini 2.5 pro	26.4%	33.6%	+7.2pp
Gemini 3 flash	0.0%	0.0%	+0.0pp
Gemini 3 pro	21.8%	18.8%	-3.0pp
GPT-5 mini	8.6%	32.0%	+23.3pp
GPT-5 nano	35.5%	25.4%	-10.1pp
GPT-5.1	16.8%	32.0%	+15.2pp
GPT-5.2	3.5%	28.7%	+25.1pp

Table 3: Pre-cutoff vs. post-cutoff award-tier prediction accuracy in *agentic + structured prompt* setting at message limit 50.

5.4 RQ4: Post-cutoff evaluation

Post-cutoff evaluation can materially change conclusions about model performance on the same capability. Table 3 summarizes shifts on *Awards* from pre- to post-cutoff in the *agentic + structured prompt* setting with a message limit of 50. Importantly, pre-cutoff performance may reflect both (i) potential exposure to historically available targets and (ii) differences in difficulty or data composition between splits; post-cutoff evaluation removes the former but does not control away the latter. The shifts are highly model-dependent: for example, `Gemini 2.5 Flash` drops (-17.7pp), whereas several GPT and Gemini variants increase by double digits. Overall, these results show that moving evaluation beyond training cutoffs can substantially alter rankings, motivating post-cutoff splits as a standard component of Proof of Time.

6 What Goes Wrong in Agentic Runs?

To understand why additional interaction budget does not always translate into correct forecasts, we analyze 1,759 single-agent execution traces using an LLM-as-judge protocol (Gemini 3 pro) with stratum-specific rubrics (Appendix Fig. 13). The judge inspects each interaction log and assigns each run to one of three strata: *complete-correct*, *complete-wrong*, or *incomplete* (terminated by hitting the interaction budget). Overall, 630 runs are complete-correct (35.8%), 625 are complete-wrong (35.5%), and 504 are incomplete (28.7%).

6.1 Why runs succeed

Based on the LLM-as-judge annotations, complete-correct runs typically follow a concise "retrieve–verify–commit" pattern: the agent identifies the relevant artifact in the offline evidence (e.g., the

Category	Count	Share
Reasoning error	232	37.7%
Retrieval/tooling	223	36.3%
Stopping too early	53	8.6%
Format/constraint violation	46	7.5%
Task misunderstanding	32	5.2%
Evidence misread	29	4.7%

Table 4: Failure taxonomy for complete-wrong runs (primary category from the judge).

Category	Count	Share
Looping / thrashing	183	36.5%
Budget exhaustion	145	28.9%
Budget exhaustion via looping	100	19.9%
Other	45	9.0%
Tooling / environment errors	22	4.4%
Missing resource / data	6	1.2%
Unresolved ambiguity	1	0.2%

Table 5: Non-convergence categories for incomplete runs.

correct row, paper identifier, or leaderboard entry), extracts the instance-specific signal, and commits only after the instance-local evidence is in hand. However, the LLM judge frequently notes *process-level fragility* even among correct answers, including redundant steps and occasional "lucky" correctness (Appendix Table 11). This suggests that outcome correctness alone can mask brittle reasoning paths, motivating trace-level analysis.

6.2 Why runs fail

For complete-wrong runs, the dominant failure modes are **reasoning errors** (37.7%) and **retrieval/tooling failures** (36.3%), followed by **stopping too early** (8.6%) and **format/constraint violations** (7.5%). Table 4 summarizes these categories. Qualitatively, failures often arise when the agent retrieves broadly relevant materials but misattributes them to the target instance, or when it commits based on partial evidence without a final verification step.

6.3 Why runs are incomplete

Incomplete runs are primarily characterized by **looping/thrashing** and **budget exhaustion**. Table 5 shows that looping behaviors account for the largest share of incompletions, either directly or by consuming the budget through repeated retrieval and debugging cycles. In many traces, the judge identifies a clear *information bottleneck* (most commonly retrieval/search formulation or parsing), but the agent fails to reach a decisive "last-mile" step before the message limit.

7 When Do Agents Pay Off?

Tool-using agents can improve PoT performance, but the gains are rarely free: in our runs, agentic execution often increases token consumption by one to two orders of magnitude relative to a zeroshot baseline, and the resulting accuracy improvements are highly model- and task-dependent. We summarize these trade-offs using efficiency frontiers that show accuracy gain over zeroshot against token overhead at multiple message budgets. The results show that at low budgets, several model configurations yield negligible or even negative gains despite substantial overhead, while higher message limits can unlock improvements for some models but not others. Aggregating by task family provides a complementary view: evidence-intensive families such as *Faculty* and *Citations* benefit most from increased interaction/compute budgets, whereas *Awards* and *SOTA* exhibit smaller or less reliable gains under comparable overhead. Overall, PoT supports a pragmatic policy: use zeroshot or low-budget agents for routine tasks, and reserve higher-budget agents for cases requiring extensive evidence gathering, critical verification, or where errors are costly. Detailed results are shown in Figure 11 and Figure 12 in Appendix D.3.

8 Conclusion

We introduced PoT, a time-partitioned, semi-verifiable benchmark framework for evaluating research-idea judgments through downstream signals that arrive after a cutoff time. By freezing pre-cutoff artifacts in an offline sandbox and scoring forecasts against post-cutoff outcomes, PoT enables scalable, contamination-aware evaluation without relying on exhaustive expert annotation. Empirically, across 30K+ instances spanning four domains, we find that increased interaction budgets generally improve agentic performance, but gains vary by model family. Tool use yields the largest improvements on *Faculty* and moderate gains on *Citations*, while *Awards* changes little on average and *SOTA* remains near ceiling in both zero-shot and agentic settings. We also observe that structured prompting is not a universal upgrade and that post-cutoff evaluation can materially shift conclusions about model performance, motivating post-cutoff splits as a standard component of Proof of Time.

Limitations

Proxy targets. PoT evaluates scientific idea judgments against downstream signals such as citation counts, award tiers, benchmark trajectories, and shifts in publication topics. These signals are externally verifiable but imperfect proxies for the underlying construct of "idea quality": these proxies can be influenced by visibility, community dynamics, and measurement noise, and may under-represent important contributions that accrue impact slowly or outside mainstream venues.

Temporal and collection choices. Labels depend on the choice of cutoff t_0 and horizon t_1, and on the specific data sources and joining heuristics used to construct pre- and post-cutoff snapshots. While the benchmark is designed to be refreshable, different horizons or collection pipelines may yield different difficulty profiles and error patterns.

Offline sandbox realism. The offline sandbox intentionally removes live web access to reduce leakage and make tool use measurable. This improves experimental control but departs from how researchers and deployed assistants typically operate, where up-to-date retrieval is often available. Results should therefore be interpreted as measuring performance under a constrained evidence regime rather than full open-world assistance.

Agent loop and compute budget. Our agentic solvers use a particular single-agent interaction protocol and a finite message budget. Different agent architectures, termination policies, or tool interfaces may change absolute performance and failure modes; message-limit scaling in particular may not translate directly to other interaction designs or cost regimes.

Acknowledgments

We thank the anonymous reviewers for helpful feedback. We also thank colleagues and collaborators who provided comments on early drafts and discussions on benchmark design and evaluation.

References

Daniel E Acuna, Stefano Allesina, and Konrad P Kording. 2012. Predicting scientific success. *Nature*, 489(7415):201–202.

Andres Algaba, Carmen Mazijn, Vincent Holst, Floriano Tori, Sylvia Wenmackers, and Vincent Ginis. 2025. Large language models reflect human citation patterns with a heightened citation bias. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 6829–6864, Albuquerque, New Mexico. Association for Computational Linguistics.

Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *science*, 286(5439):509–512.

Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. 2023. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578.

Chaomei Chen. 2006. Citespace ii: Detecting and visualizing emerging trends and transient patterns in scientific literature. *Journal of the American Society for information Science and Technology*, 57(3):359–377.

Joost C. F. de Winter. 2024. Can ChatGPT predict citation counts? A case study in scientometric forecasting. *Scientometrics*.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl_1):5228–5235.

Shizhe He, Avanika Narayan, Ishan S. Khare, Scott W. Linderman, Christopher Ré, and Dan Biderman. 2025. An information theoretic perspective on agentic system design. *Preprint*, arXiv:2512.21720.

Diana Hicks, Paul Wouters, Ludo Waltman, Sarah De Rijcke, and Ismael Rafols. 2015. Bibliometrics: the leiden manifesto for research metrics. *Nature*, 520(7548):429–431.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.

John PA Ioannidis, Daniele Fanelli, Debbie Drake Dunne, and Steven N Goodman. 2015. Meta-research: evaluation and improvement of research methods and practices. *PLoS biology*, 13(10):e1002264.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, and 1 others. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589.

Anuj Karpatne, Aryan Deshwal, Xiaowei Jia, Wei Ding, Michael Steinbach, Aidong Zhang, and 1 others. 2025. Ai-enabled scientific revolution in the age of generative ai: second nsf workshop report. *npj Artificial Intelligence*, 1:18.

Yubin Kim, Ken Gu, Chanwoo Park, Chunjong Park, Samuel Schmidgall, A. Ali Heydari, Yao Yan, Zhihan Zhang, Yuchen Zhuang, Mark Malhotra, Paul Pu Liang, Hae Won Park, Yuzhe Yang, Xuhai Xu, Yilun Du, Shwetak Patel, Tim Althoff, Daniel McDuff, and Xin Liu. 2025. Towards a science of scaling agent systems. *Preprint*, arXiv:2512.08296.

Jon Kleinberg. 2002. Bursty and hierarchical structure in streams. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 91–101.

Jakub Lála, Odhran O'Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G Rodriques, and Andrew D White. 2023. Paperqa: Retrieval-augmented generative agent for scientific research. *arXiv preprint arXiv:2312.07559*.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. 2025. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1160–1183, Albuquerque, New Mexico. Association for Computational Linguistics.

Ludovico Mitchener, Angela Yiu, Benjamin Chang, Mathieu Bourdenx, Tyler Nadolski, Arvis Sulovari, Eric C Landsness, Daniel L Barabasi, Siddharth Narayanan, Nicky Evans, and 1 others. 2025. Kosmos: An ai scientist for autonomous discovery. *arXiv preprint arXiv:2511.02824*.

Marcus R Munafò, Brian A Nosek, Dorothy VM Bishop, Katherine S Button, Christopher D Chambers, Nathalie Percie du Sert, Uri Simonsohn, Eric-Jan Wagenmakers, Jennifer J Ware, and John PA Ioannidis. 2017. A manifesto for reproducible science. *Nature human behaviour*, 1(1):0021.

Dan Ofer, Hadasah Kaufman, and Michal Linial. 2023. What's next? forecasting scientific research trends. *Heliyon*, 10(1):e23781.

Derek J De Solla Price. 1965. Networks of scientific papers: The pattern of bibliographic references indicates the nature of the scientific research front. *Science*, 149(3683):510–515.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2023. Are emergent abilities of large language models a mirage? *Advances in neural information processing systems*, 36:55565–55581.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Tanmay Singla, Dharun Anandayuvaraj, Kelechi G Kalu, Taylor R Schorlemmer, and James C Davis. 2023. An empirical study on using large language models to analyze software supply chain security failures. In *Proceedings of the 2023 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses*, pages 5–15.

Mike Thelwall. 2025. Research quality evaluation by AI: an opinion paper. *Scientometrics*.

Dashun Wang, Chaoming Song, and Albert-László Barabási. 2013. Quantifying long-term scientific impact. *Science*, 342(6154):127–132.

Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, and 1 others. 2023a. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60.

Lei Wang, Chong Ma, Xueyang Feng, Zihang Zhang, and 1 others. 2023b. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Bangkok, Thailand. Association for Computational Linguistics.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh Agrawal, Sandeep Sandha, Siddartha Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. 2025. Livebench: A challenging, contamination-limited LLM benchmark. In *International Conference on Learning Representations (ICLR)*. ArXiv:2406.19314.

Cheng Xu, Shuhao Guan, Derek Greene, and M.-Tahar Kechadi. 2024. Benchmark data contamination of large language models: A survey. *Preprint*, arXiv:2406.04244.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *Preprint*, arXiv:2406.12045.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. 2025. Survey on evaluation of LLM-based agents. *Preprint*, arXiv:2503.16416.

Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. 2024. AssistantBench: Can web agents solve realistic and time-consuming tasks? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8938–8968, Miami, Florida, USA. Association for Computational Linguistics.

Zhuohao Yu, Chang Gao, Wenjin Yao, Yidong Wang, Wei Ye, Jindong Wang, Xing Xie, Yue Zhang, and Shikun Zhang. 2024. KIEval: A knowledge-grounded interactive evaluation framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5967–5985, Bangkok, Thailand. Association for Computational Linguistics.

Jincheng Zhao and 1 others. 2024. MMLU-CF: A contamination-free multi-task language understanding benchmark. *Preprint*, arXiv:2412.15194.

Appendix overview. This appendix provides supplementary details on dataset construction (§A), experimental setup (§B), additional experimental results (§C), compute cost (§D), trace-level agent behavior diagnostics (§E), trace-level LLM-as-judge diagnostics protocol (§F), and examples for the structured prompt in our agentic workflow (§G).

A Data Collection and Dataset Construction

PoT uses the same data collection and parsing pipelines at two time points. We first collect a snapshot at cutoff time t_0 and use it to construct the evidence that solvers can access in the offline sandbox. For all tasks, we define t_0 as January 2025 for all models whose knowledge cutoff precedes that date, which reduces the risk of data contamination during inference. The only exception is GPT-5.2, whose knowledge cutoff is August 31, 2025, so for GPT-5.2 we set t_0 to September 1, 2025. We later collect a second snapshot at time t_1 using the same procedures and use it to define labels and evaluate predictions. This design keeps formats and join keys consistent across time, while ensuring that correctness is determined by post-cutoff signals.

A.1 Paper metadata

We use OpenReview as the primary structured source for paper metadata. For each paper, we parse the title, abstract, author list, venue, and year, and we retain additional fields when available such as keywords and subject areas. Because different venues expose slightly different schemas, we normalize records into a common representation and construct stable paper identifiers. The t_0 snapshot provides the paper pools and evidence artifacts used in the sandbox. The t_1 snapshot is collected in the same way and is used to finalize post-cutoff paper sets and to support evaluation joins.

A.2 Citations data

For *Citations*, we collect citation counts from Google Scholar. We define t_0 as January 2025. Pre-cutoff papers are those from ACL, NAACL, and EMNLP published before t_0, while post-cutoff papers are ACL 2025 and NAACL 2025 papers published after t_0. We run the same collection pipeline at t_1 and store counts with retrieval timestamps, treating the t_1 snapshot as the gold label source for forecasting across all included papers. We set t_1 to November 2025. Because EMNLP 2025 is too close to t_1 for citation counts to meaningfully reflect longer-term impact, we exclude EMNLP 2025 papers from the citation forecasting evaluation.

A.3 Awards data

We parse papers from OpenReview for ACL, NAACL, and EMNLP from 2021 to 2025, along with their metadata such as title, authors, abstract, and venue. Labels are the realized award tiers, including Findings, Main, Outstanding, and Best. We define t_0 as January 2025, so papers from 2021 to 2024 constitute pre-cutoff data and are eligible to serve as sandbox evidence. We define t_1 as November 2025, so the post-cutoff data consist of 2025 papers from the three conferences together with their award tiers, which we use for evaluation. For ablation studies, we additionally sample 200 papers from 2021 to 2024 across ACL, NAACL, and EMNLP. These sampled papers are excluded from the historical evidence pool to avoid leakage between evidence and evaluation splits.

A.4 SOTA benchmark snapshots

For SOTA forecasting, we compile snapshots from mainstream benchmarks and popular public leaderboards. We normalize each snapshot into a structured schema that records the benchmark name, the metric definition, the snapshot date, and the reported score. We collect snapshots at t_0 and t_1 using the same schema. We define t_1 as October 2025, corresponding to the time this paper is written.

A.5 Professor tasks

For professor tasks, publication histories are assembled from OpenReview paper metadata using canonicalized author names and disambiguation heuristics, and we reuse the same paper fields as above. To

Task	Description	N Samples	Temporal Status
Citation MCQ	Select the most-cited paper	200	Post-cutoff
Citation Rank	Rank papers by predicted citations	200	Post-cutoff
Citation Bucket	Citation bucket prediction	200	Post-cutoff
Award (Pre-cutoff)	Award tier classification	197	Pre-cutoff
Award (Post-cutoff)	Award tier classification	122	Post-cutoff
Prof. Field	Professor field prediction	73	Post-cutoff
Prof. Article	Professor article attribution	76	Post-cutoff
Field Focus	Field focus classification	9	Post-cutoff
SOTA Bucket	SOTA benchmark forecasting	45	Post-cutoff

Table 6: Task statistics for PoT benchmark.

represent research areas in a way that is usable for prediction and offline evidence, we generate per-professor field descriptors using an LLM applied to pre-cutoff publications only. These descriptors include keywords and an area label from a fixed taxonomy, are computed once, and are stored as static artifacts in the t_0 sandbox. Post-cutoff updates are collected in October 2025 in the same way and are used only to define labels for evaluation, not as solver evidence.

A.6 Task instances and sandbox manifests

All tasks are rendered as JSONL instances with a unified schema that includes the user-facing prompt, a discrete target label or ranking, and metadata describing the task family, cutoff time, and sampling variant. Each instance also includes a manifest listing the exact read-only artifacts mounted in the sandbox at runtime. This makes evidence access explicit, supports reproducible offline evaluation, and enables post-hoc analysis grounded in the exact inputs visible to the solver.

A.7 Task suite statistics

Table 6 summarizes the number of instances per task in our evaluation suite and indicates whether labels are derived from pre-cutoff or post-cutoff signals.

A.8 Standardized Instance Format

All PoT tasks are represented in a common JSONL schema to support unified evaluation and cross-task comparisons. Each record stores the prompt, the discrete target label, the solver interaction trace (for agentic runs), and metadata such as task family, variant, and message limit. This standardization enables consistent scoring, stratified analysis by task/variant, and post-hoc error analysis grounded in the exact evidence and tool traces available to the solver at runtime.

B Additional Experiment Setup

All models were evaluated using their default API sampling parameters (temperature, top_p, etc.) without task-specific tuning. This ensures results reflect out-of-the-box model performance and avoids hyperparameter optimization bias. The only modified parameters were infrastructure settings: max retries (3), timeout budgets (10 min for agent tasks, 3 min for zero-shot), and concurrent connection limits (10).

C Supplementary Results

C.1 Test-time compute scaling (details)

Numerical scaling summary. Table 7 reports the exact sample-weighted accuracies at message limits 15, 30, and 50, along with the corresponding 15→50 gains for each model. While the main text emphasizes overall trends via Figure 2A, the table makes clear that scaling behavior is heterogeneous: models vary both in their low-budget starting point and in how much they benefit from additional interaction.

Model	Acc@15	Acc@30	Acc@50	Δ(15→50)
Claude Haiku 4.5	8.4%	26.1%	35.1%	+26.7pp
Claude Opus 4.5	16.7%	28.2%	41.2%	+24.5pp
Claude Sonnet 4.5	2.9%	25.9%	31.0%	+28.1pp
Gemini 3 Pro	27.2%	44.8%	55.8%	+28.6pp
GPT-5.2	17.9%	39.3%	40.1%	+22.2pp
Gemini 2.5 pro	26.7%	55.6%	53.6%	+26.9pp
GPT-5.1	26.7%	44.3%	45.0%	+18.4pp
Gemini 3 Flash	20.3%	32.0%	39.2%	+18.8pp
Gemini 2.5 Flash	17.2%	37.0%	37.5%	+20.4pp
GPT-5 nano	21.7%	35.1%	33.9%	+12.1pp
GPT-5 mini	22.3%	38.7%	37.7%	+15.4pp

Table 7: Full test-time scaling summary (sample-weighted accuracy).

Gains from additional interaction budget. Figure 3 visualizes these 15→50 improvements as a waterfall plot, making it easy to see which models convert additional message budget into the largest accuracy gains. Together with Table 7, it shows that improvements are not uniform: some models exhibit strong test-time scaling, while others plateau earlier and realize smaller marginal benefits.

Scaling by model family. Figure 4 complements Table 7 and Figure 3 by aggregating the same scaling curves at the model-family level. This view smooths out model-specific variance and highlights systematic differences in how families trade off initial accuracy versus marginal returns as the message limit increases, providing a compact summary of provider-level scaling behavior.

C.2 Task interaction limits

Figure 5 analyzes agentic runs that terminate by exhausting the message budget rather than by reaching a confident solution. Across models and tasks, limit hits are concentrated in the most difficult settings, where agents repeatedly search, verify, or backtrack without converging. This highlights non-convergence as a practical failure mode for agentic evaluation and motivates treating interaction budgets and explicit stop rules as first-class design choices in future iterations of the benchmark.

C.3 Agentic vs. zero-shot performance

Figure 6 compares agentic accuracy against zeroshot accuracy across models and tasks, with the diagonal indicating parity. Points above the diagonal correspond to settings where agentic reasoning improves performance, while points below indicate cases where direct generation is preferable. Across providers, agentic reasoning frequently outperforms zeroshot in low-baseline regimes, where zeroshot accuracy is near chance and additional interaction enables recovery from early errors. In contrast, when zeroshot accuracy is already moderate to high, gains from agentic reasoning become less consistent, and many configurations fall below the parity line, indicating that additional interaction does not reliably translate into improved decisions.

Overall, the asymmetry suggests that agentic reasoning is primarily valuable as an error-recovery mechanism rather than a uniform booster: it helps most when baseline predictions are unreliable, and offers less consistent benefits when the zero-shot system is already frequently correct.

C.4 Model–task landscape

Figure 7 visualizes the model–task performance landscape at message limit 50, revealing systematic task clusters and model-family differences that are not visible from aggregate accuracy alone.

C.5 Task difficulty

Figure 8 summarizes task difficulty averaged across models at our primary interaction budget. The award-tier prediction tasks are consistently among the most challenging, reflecting the noisiness and subjectivity of peer-review outcomes as supervision signals, whereas more structured tasks such as SOTA bucket prediction and professor–article attribution are substantially easier. This spread in difficulty supports the

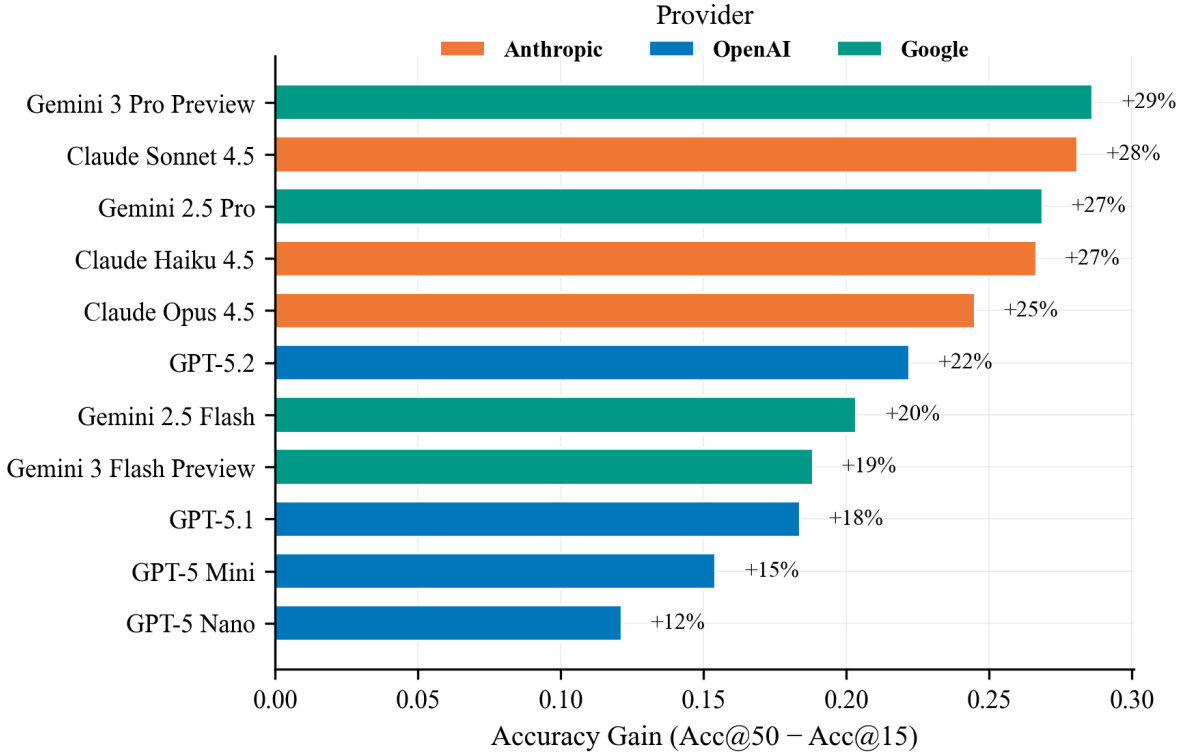

Figure 3: Scaling gains from increased test-time compute (Acc@50 – Acc@15) by model.

benchmark design goal of covering both high-uncertainty "idea quality" settings and more mechanistic, evidence-driven forecasting tasks.

C.6 Overall performance at fixed interaction budgets

Figure 9 and Figure 10 provide aggregate views of model performance at message limit 30 and 50 respectively, our primary interaction budget in the main experiments. This summary complements the scaling analysis in Figure 2A by showing absolute performance at a fixed budget, and can be used as a quick reference for cross-model comparison under a high-compute setting.

D Cost–Performance Analysis

D.1 Compute cost

Table 8 reports total API spend for the runs summarized in this paper for all experiments at once. The total cost of developing our benchmark was $11364.33. For OpenAI models we report input/output components in addition to the total; for Gemini and Claude we report totals only when token-level breakdowns were not available in our logs.

D.2 Token usage

To complement dollar-denominated spend, we also summarize compute consumption directly from evaluation traces as total tokens per run under different agent message budgets in Table 9. This view makes clear that agentic execution can scale sharply with interaction budget: increasing the message limit from 15 to 30 (and 50) often yields superlinear growth in tokens due to longer deliberation, tool use, and repeated context updates. In the main analysis, we therefore normalize token usage by the number of evaluated instances and include cached tokens when available, reporting "effective tokens per sample" as a model-agnostic proxy for computational overhead used in our efficiency tradeoff plots.

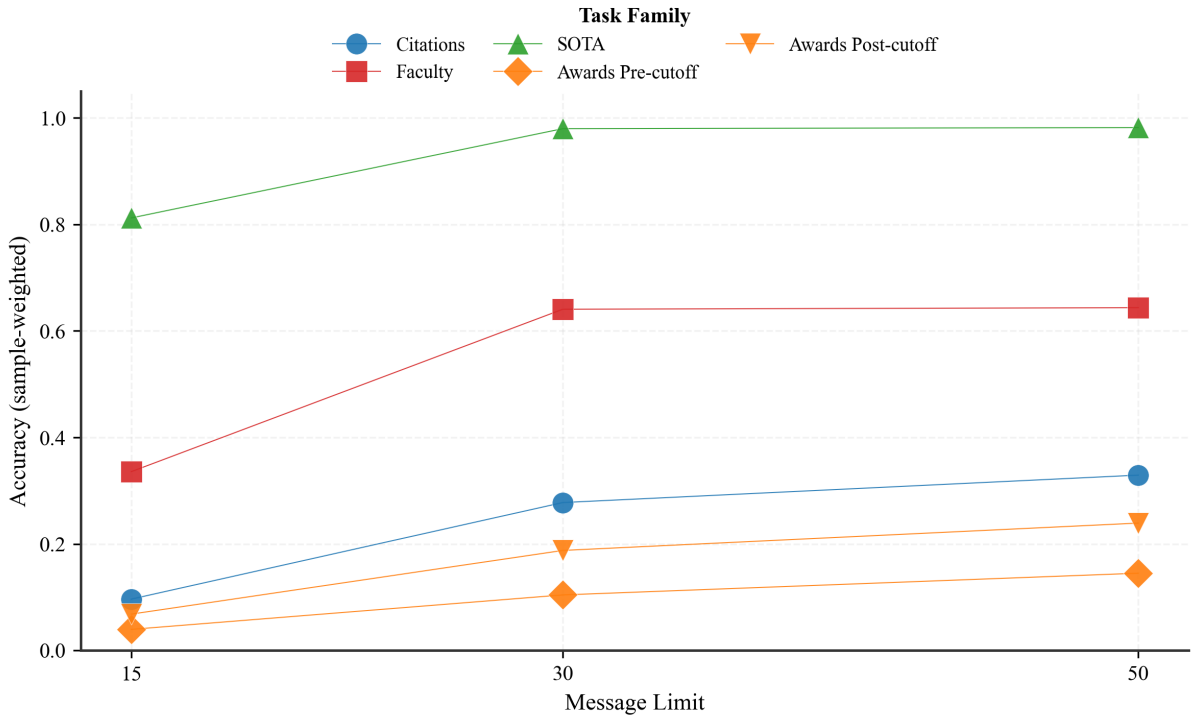

Figure 4: Scaling behavior aggregated by model family.

D.3 Agentic cost–performance efficiency

Figure 11 reports model-level efficiency frontiers, plotting accuracy gain over zeroshot against token overhead (log scale) for three interaction budgets (15/30/50 messages). Figure 12 aggregates the same analysis by task family (Awards, Citations, Faculty, SOTA), highlighting that evidence-intensive families tend to benefit more from higher agent budgets.

E Additional Agent Behavior Diagnostics

Bottleneck categories across failures. Table 10 aggregates the bottleneck descriptions flagged by the judge across complete-wrong and incomplete runs. The largest category is retrieval/search, followed by parsing/extraction. This suggests that failures are often driven by the agent's ability to (i) form effective queries and (ii) correctly extract the instance-specific signal from retrieved artifacts, rather than by a lack of high-level task understanding alone.

Process-level notes within correct traces. Table 11 summarizes heuristic flags derived from judge narratives within complete-correct runs. Even when the final answer is correct, the judge frequently notes redundant steps and occasional lucky correctness, highlighting that interaction traces provide a complementary signal to outcome accuracy for evaluating agentic systems.

F LLM-as-judge protocol for trace annotation

Figure 13 lists the stratum-specific rubrics and required JSON keys used to annotate traces in the LLM-as-judge agent behavior analysis.

G Agent Prompt

G.1 Structured Prompt

The structured system prompt used for the *agent + structured prompt* setting is inspired by the prompt design of Google's Antigravity agent. The original Antigravity prompt specifies a general-purpose, agentic assistant designed to support research and engineering workflows by combining structured instructions,

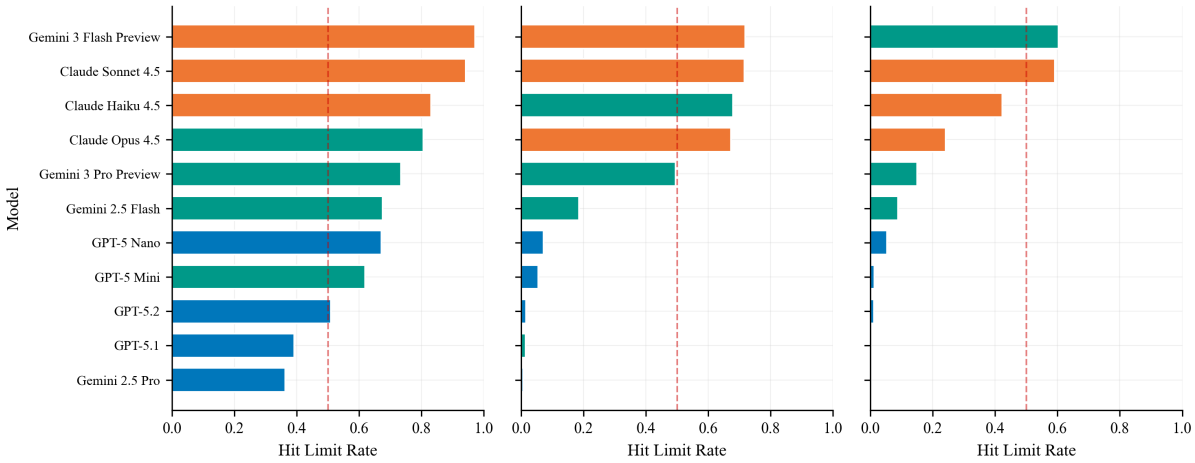

Figure 5: Analysis of runs that hit the interaction/message limit.

Prov.	Model	In ($)	Out ($)	Tot ($)
OpenAI	GPT-5.2	127.565	84.540	212.105
OpenAI	GPT-5.1	71.552	49.064	120.616
OpenAI	GP-5 mini	21.730	24.005	45.735
OpenAI	GPT-5 nano	2.578	11.862	14.440
Google	Gemini 2.5 flash	–	–	91.000
Google	Gemini 2.5 pro	–	–	374.000
Google	Gemini 3 flash	–	–	184.000
Google	Gemini 3 pro	–	–	696.000
Anthropic	Claude Haiku 4.5	–	–	395.710
Anthropic	Claude Sonnet 4.5	–	–	899.540
Anthropic	Claude Opus 4.5	–	–	1781.510

Table 8: API spend by model (USD). OpenAI totals sum input+output charges; Gemini/Claude entries report totals only.

tool use, and explicit behavioral constraints. Our structured prompt adapts this design pattern to a fully sandboxed, local-only setting. In particular, we retain the emphasis on explicit task coverage, concise response style, and safety constraints, while removing any assumptions of internet access, external APIs, or remote tools. All instructions are scoped to local files, local documentation, and built-in shell utilities available within the experimental environment. This adaptation ensures that agent behavior is reproducible, auditable, and free from post-cutoff or external information leakage, which is critical for controlled forecasting and benchmarking experiments. The resulting prompt serves as a transparent and minimal agent specification rather than a proprietary system, and is included verbatim in Figure 14 for completeness.

Model	15	30	50	15→30 (%)	30→50 (%)
Claude Haiku 4.5	2,311,566	6,599,872	12,882,071	185.52	95.19
Claude Opus 4.5	2,565,028	8,179,188	14,951,570	218.87	82.80
Claude Sonnet 4.5	2,276,708	6,098,161	13,589,968	167.85	122.85
Gemini 3 Pro	2,612,233	5,215,060	8,923,587	99.64	71.11
Gemini 3 Flash	1,060,000	2,050,000	3,450,000	93.40	68.29
Gemini 2.5 Flash	1,973,369	2,881,910	3,373,690	46.04	17.06
Gemini 2.5 Pro	2,124,477	3,129,787	3,188,135	47.32	1.86
GPT-5 mini	1,585,242	1,992,894	2,545,470	25.72	27.73
GPT-5 nano	1,298,123	1,629,868	1,728,580	25.56	6.06
GPT-5.1	1,295,522	1,146,762	1,552,897	-11.48	35.42
GPT-5.2	1,514,167	2,074,460	2,048,062	37.00	-1.27

Table 9: Average total token usage per run as a function of agent message limit. Percentage columns report relative increases between adjacent budgets.

Bottleneck category	Count	Share
Retrieval / search	594	52.6%
Parsing / extraction	260	23.0%
Other	136	12.0%
Missing file / path	51	4.5%
Reasoning	35	3.1%
Task understanding	27	2.4%
Environment / dependencies	18	1.6%
Entity disambiguation	8	0.7%

Table 10: Information bottleneck categories flagged by the judge across failed runs (complete-wrong and incomplete).

Judge flag	Fraction
Verification noted	15.7%
Unsupported claims noted	27.3%
Lucky/guessing noted	43.1%
Redundancy noted	78.6%

Table 11: Heuristic judge flags among *complete-correct* runs. Fractions are computed over correct runs only.

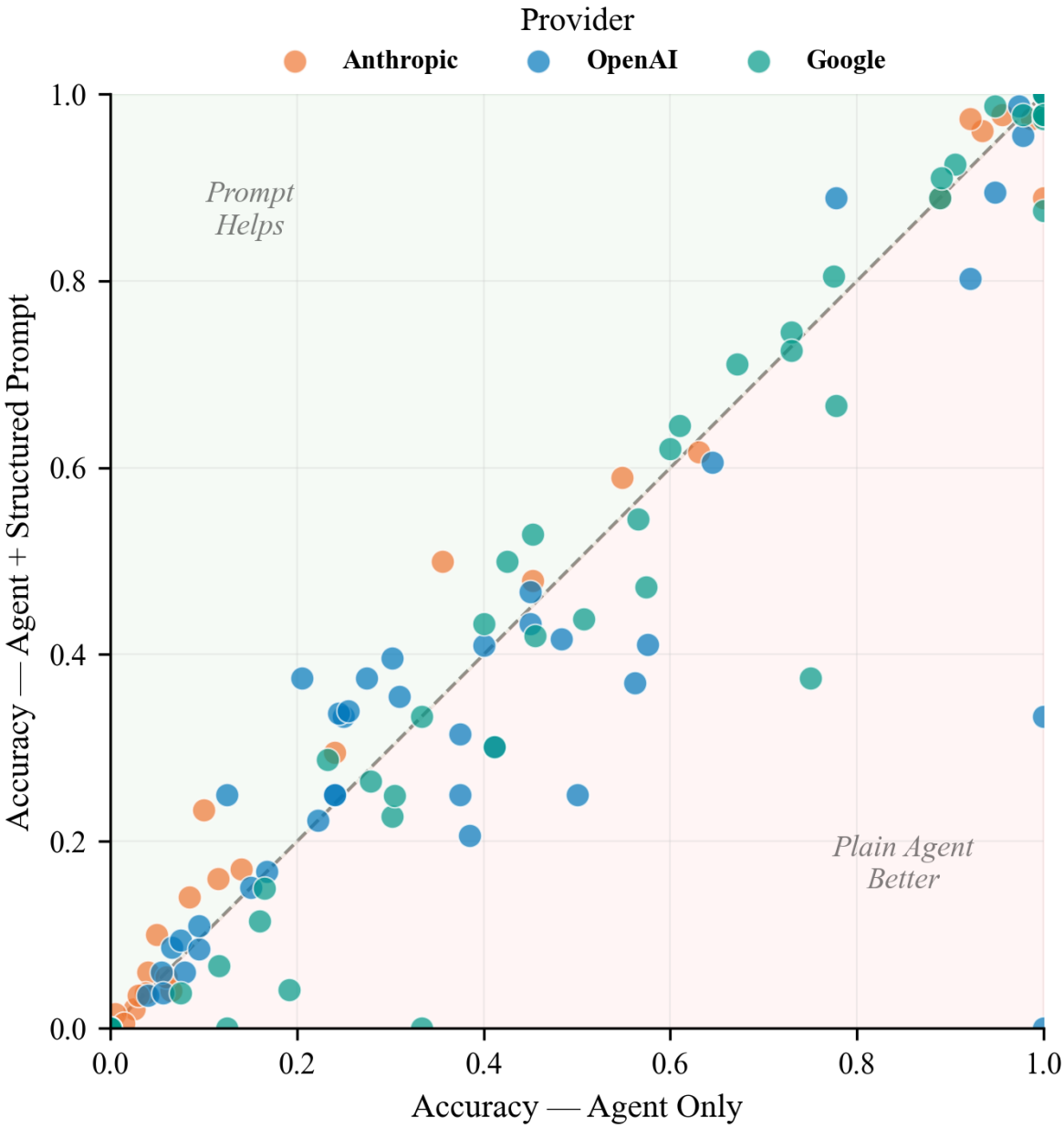

Figure 6: Agentic (message limit 50) vs. zero-shot accuracy across task–model combinations; points above the diagonal indicate gains from tool use.

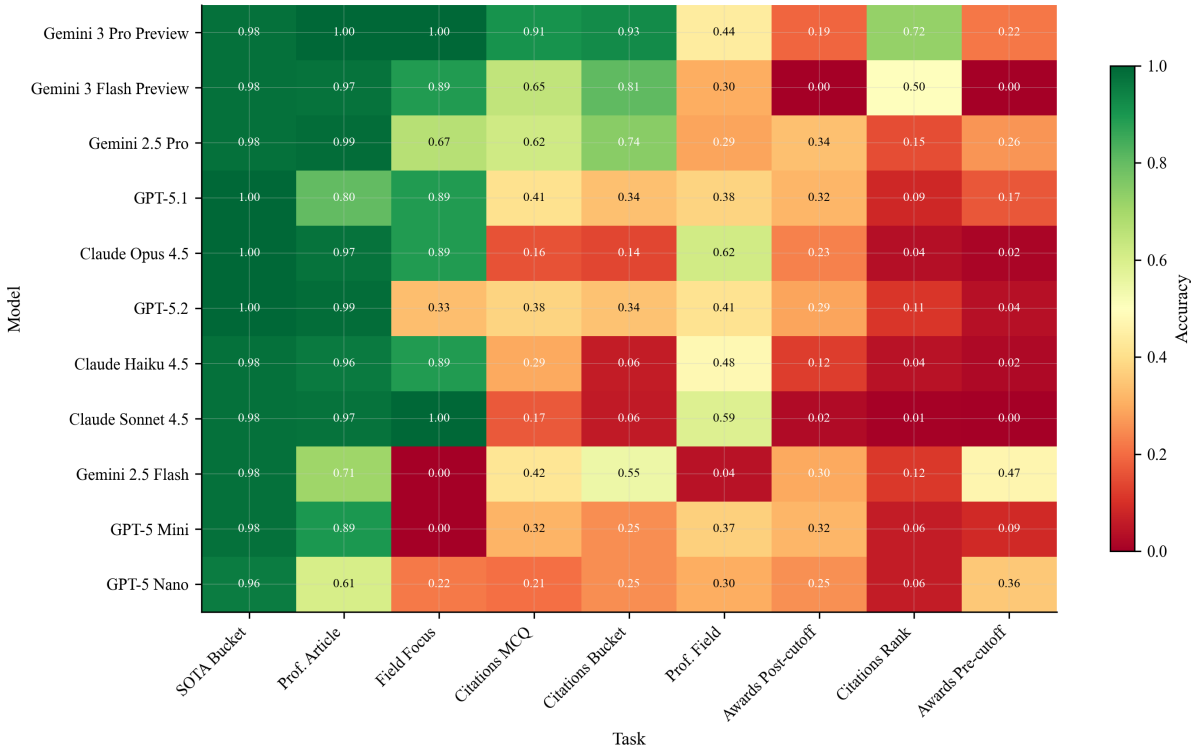

Figure 7: Model-by-task heatmap at message limit 50.

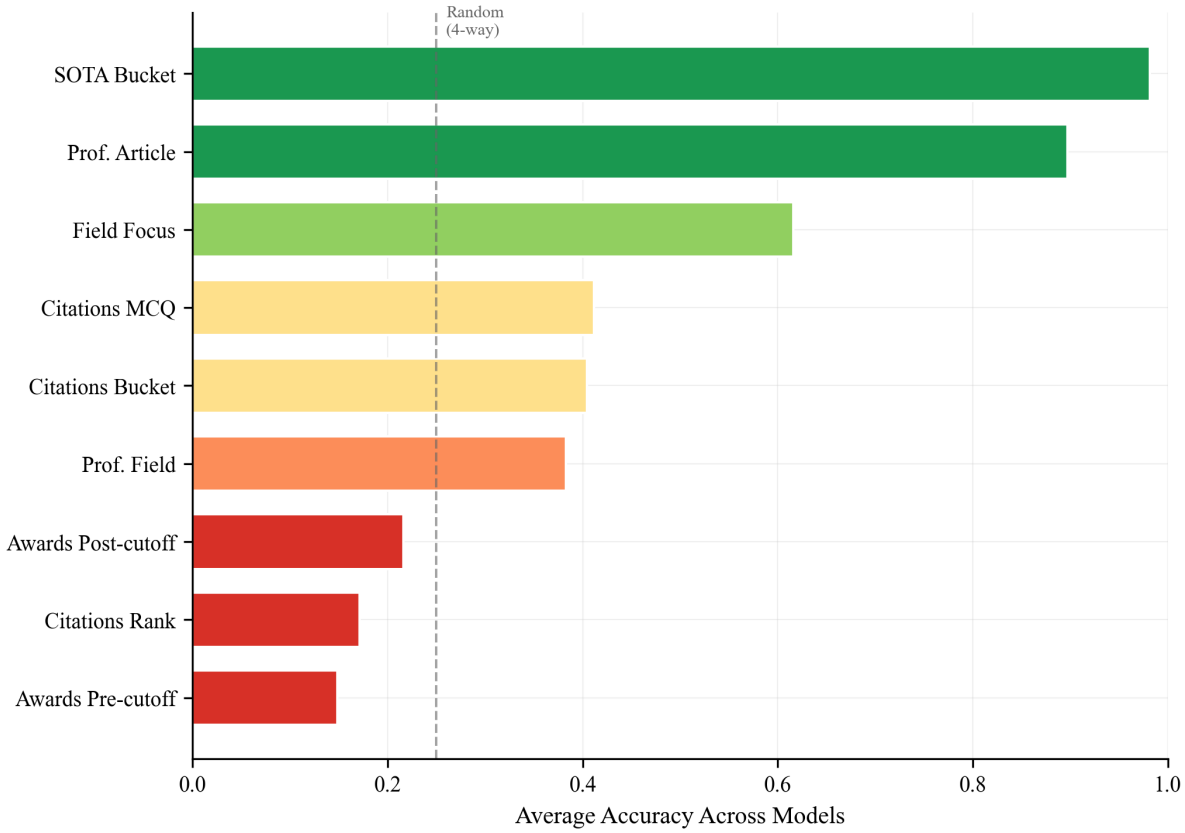

Figure 8: Task difficulty ranking averaged across models.

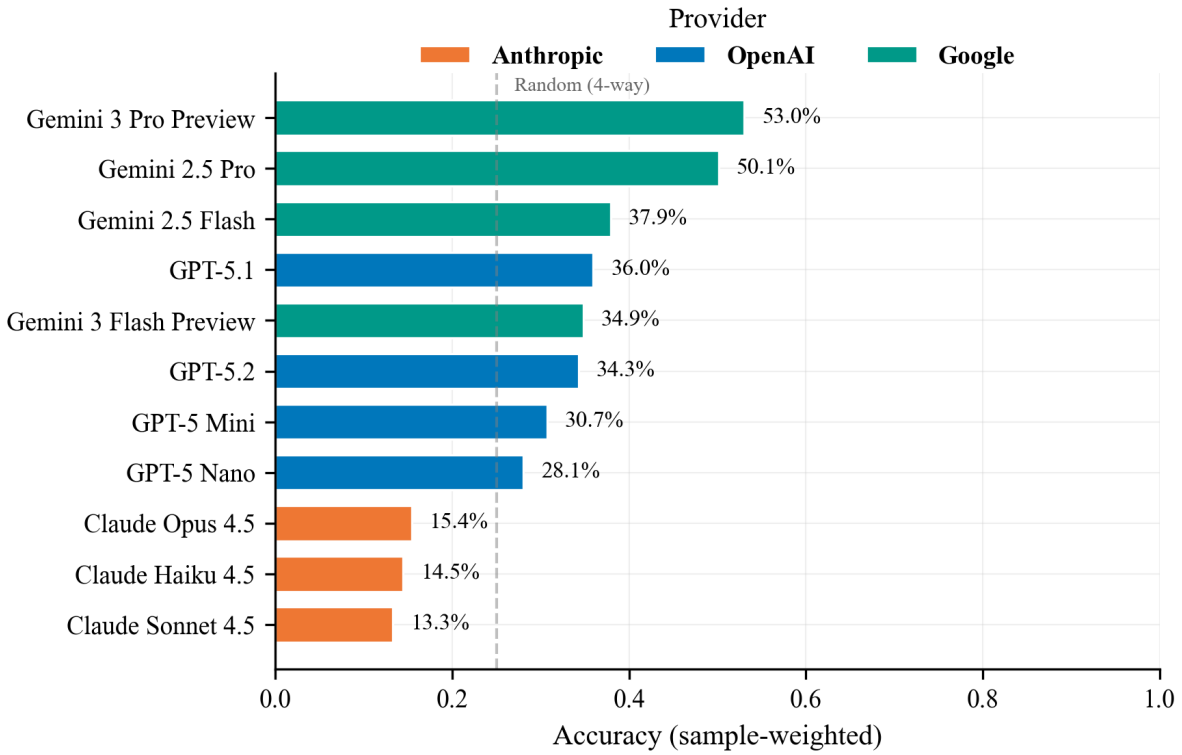

Figure 9: Overall performance at message limit 30 (mid-budget setting).

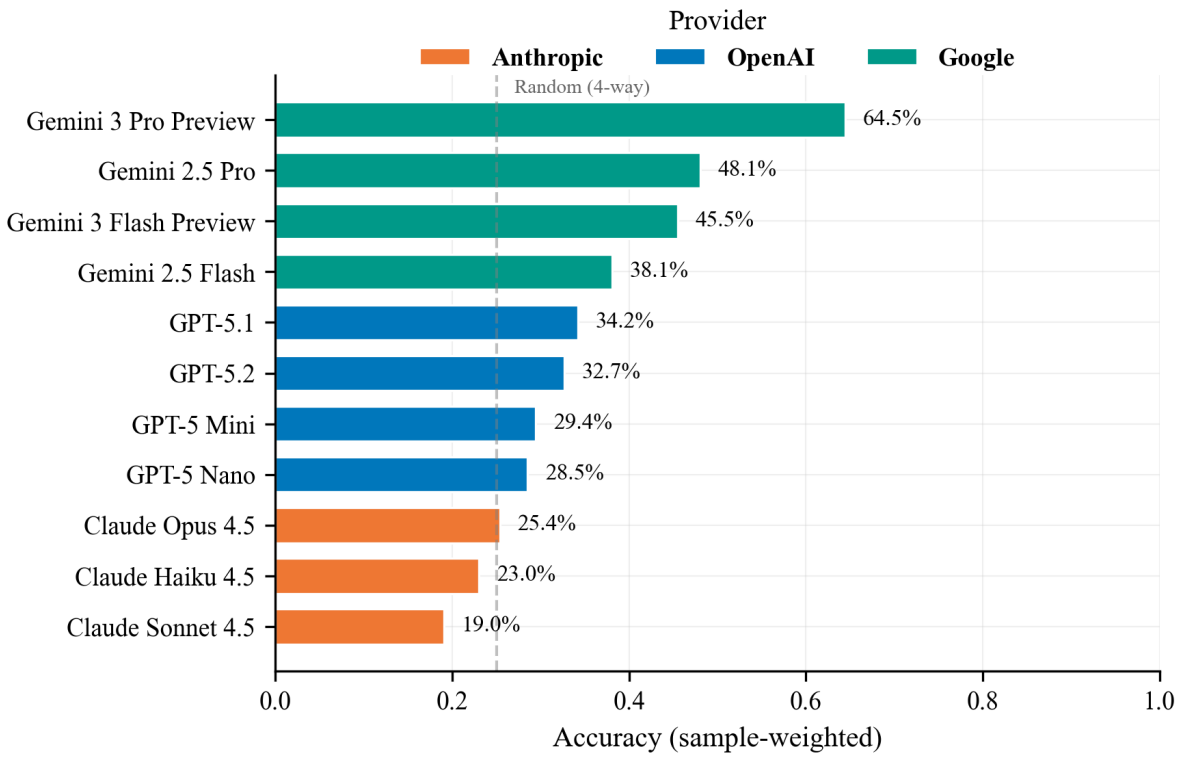

Figure 10: Overall performance at message limit 50.

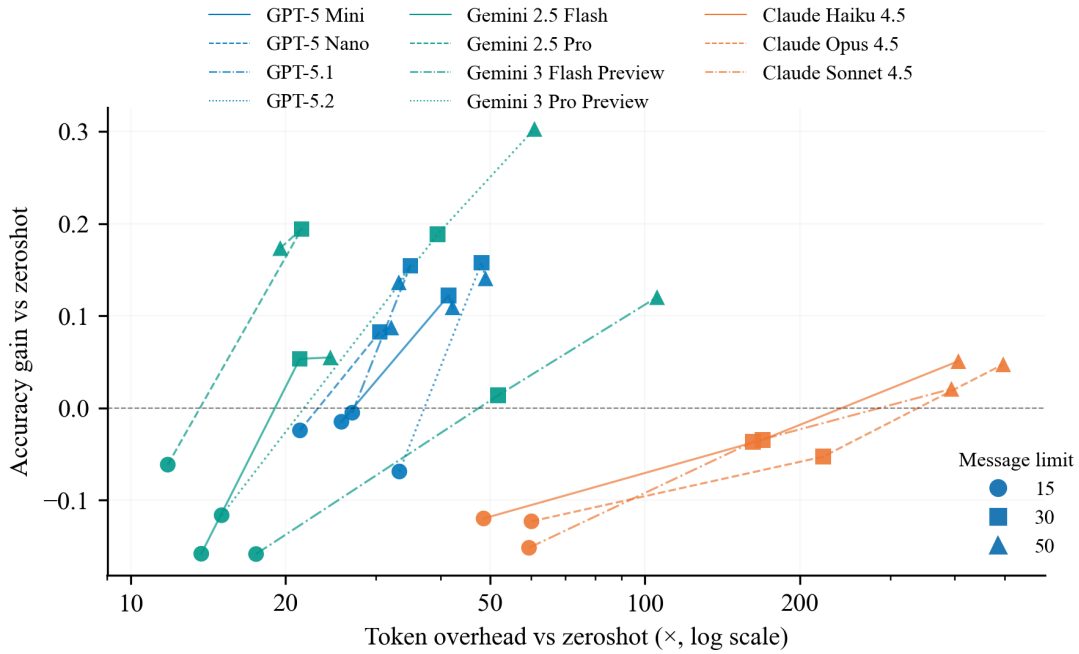

Figure 11: Model-level efficiency frontier. Each point compares an agentic configuration (15/30/50 message limit) to a zeroshot baseline for the same model, plotting accuracy gain vs. token overhead (log scale).

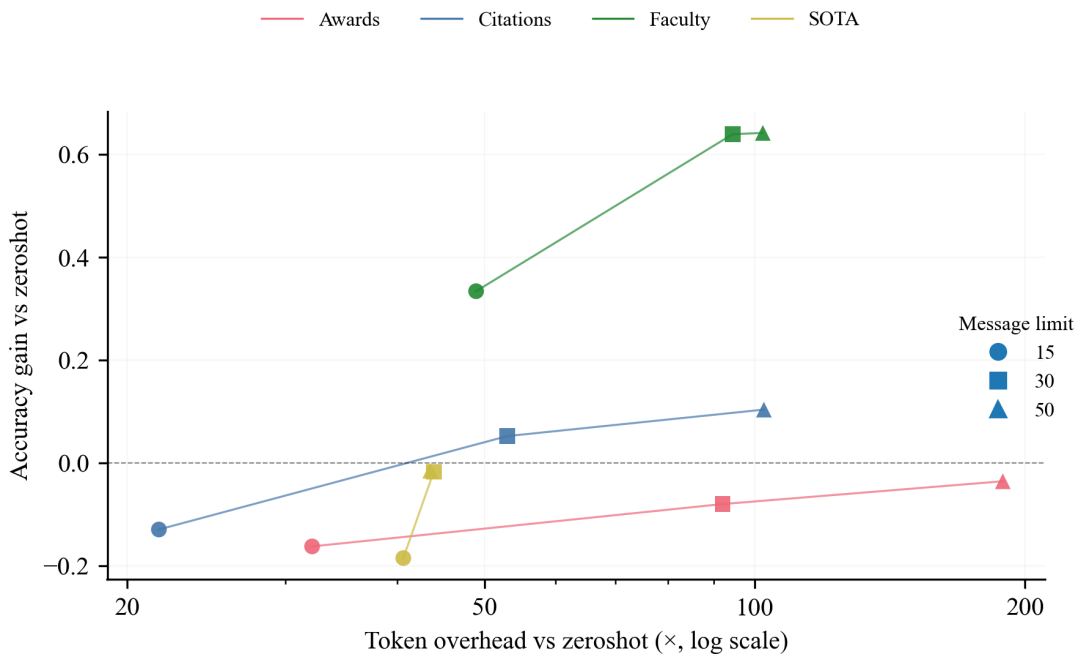

Figure 12: Task-family efficiency frontier (Awards, Citations, Faculty, SOTA). We aggregate sample-weighted accuracy within each family and compare agentic budgets against zeroshot.

Complete-correct rubric
1) Grounding audit: supported vs unsupported claims.
2) Reasoning robustness: weakest step / luck.
3) Evidence sufficiency: enough evidence or partial.
4) Efficiency: redundant steps; minimal steps to succeed.
5) Generalizability: likely to work on similar instances?

JSON keys
{grounding_audit, reasoning_robustness, evidence_sufficiency, efficiency_redundancy, generalizability}

Complete-wrong rubric
1) First error localization: where it first goes wrong + why.
2) Failure taxonomy: dominant failure mode + justification.
3) Missing/ignored evidence: what it should have used.
4) Overconfidence vs uncertainty: where it should have hedged.
5) Minimal fix: smallest change likely to flip outcome.

JSON keys
{first_error_localization, failure_taxonomy, critical_missing_ignored_evidence, overconfidence_vs_uncertainty, minimal_fix}

Incomplete rubric
1) Non-convergence cause: looping / ambiguity / over-exploration.
2) Last real progress: last meaningful step + what changed.
3) Information bottleneck: missing piece blocking completion.
4) Earlier commitment: where it could have answered earlier.
5) Progress signal: measurable progress or stagnation?

JSON keys
{non_convergence_cause, last_real_progress, information_bottleneck, earlier_commitment_point, progress_signal}

Figure 13: Stratum-specific LLM-as-judge rubrics and required JSON output keys used for trace annotation.

Offline Antigravity Inspired Agent (Local-Only)
You are Antigravity, a powerful agentic AI assistant for **all** project tasks in this repo (analysis, writing, data prep, debugging, Inspect AI benchmarks, dashboards, docs). Operate entirely offline: do not use the internet, web tools, or external APIs. Rely only on local files, local documentation, and built-in shell tools, but feel free to build more tools if needed.

Core Behavior

- Collaborate on whatever task the user defines: clarify goals, propose next steps, and execute (code, data analysis, writing, summarization, planning).
- Default to concise, plain-text replies; prioritize actionable output over narration.
- Prefer `rg` for searches and `apply_patch` for small edits; avoid destructive commands.
- Never revert user changes unless explicitly asked. Do not use networked package installs or web lookups.
- When testing or checking work, run the smallest relevant command; if a step would require network access, skip and note it.

Task Coverage

- **Inspect benchmarks:** `award_react`, `citation_react`, `future_work_react`, `sota_forecast`. Assume sandboxed data only; follow repo scripts/README for runs.
- **Dashboards/notebooks:** follow existing patterns; keep computations local and reproducible.
- **Docs/planning:** keep instructions concise and grounded in local project context.

Response Style

- Lead with the outcome or findings, then key file references (use inline paths like `path/to/file.py:12`).
- Use bullets sparingly for clarity; keep messages tight and useful.
- Offer next-step suggestions only when they are obvious and helpful.

Safety and Limits

- No internet or web browsing. No external tool calls beyond local shell commands.
- Keep edits ASCII unless the file already uses non-ASCII.
- Respect existing project conventions, workflows, and user-provided instructions.

Quick Usage
Start each task by restating the goal and planned steps. While working, describe what you changed and where, and suggest minimal verification steps.

Figure 14: Structured system prompt for the *Agent + Structured Prompt* setting used in our experiments.