

Cheap but Confused? The Accuracy Cost of Cache-Optimized Context Layouts in LLM Agents

Mohammed Abraar

Vizuara Research
abraar@vizz.vizuara.ai

Abstract

Prompt caching bills tokens a provider has already seen at about a tenth of the list price, but only when a request starts with exactly the bytes of an earlier one. Agent builders therefore keep the context append-only: nothing is reordered and nothing is deleted. When a value changes, the old value stays in the context beside the new one. We test whether this layout costs accuracy, with the information content held fixed, and whether the cache discount pays for that cost. StaleTrap is a frozen testbed of 144 scripted tool-using episodes (720 questions, 2,376 tool calls). We replay each episode under four layouts that carry the same information: append-only, relevance-reordered, stale values redacted in place, and append-only plus a short note. Every call is billed and cache-metered by the provider. On Gemini 3.1 Flash-Lite, questions about the agent’s own updates and neutral questions are answered perfectly under every layout. When the update came from outside the agent, append-only answers 65.5% correctly, against 96.0% with redaction, 70.5% with reordering and 73.5% with the note. Append-only is the cheapest layout per episode (1.05 against 1.16 cents), but redaction is the cheapest per correct episode (1.22 against 1.52 cents) at every cache-read price from free to full. Across three Gemini generations, append-only accuracy on these questions rises from 40% to 66% to 80%, and its gap to redaction shrinks from 27 to 18 to 11 points without closing.

Introduction

Agent builders now design their prompts around the provider’s cache. Providers bill input tokens they have already processed at about a tenth of the list price, and the discount applies only when a request starts with exactly the bytes of an earlier request (Zheng et al. 2023; Lumer et al. 2026). Practitioner guidance follows from this: put the system prompt and tools first and never change them, append every new observation at the end, and never reorder or delete earlier blocks (Ji 2025; Anthropic 2025). We call this the *cache-optimal layout*.

This layout has a cost that the guidance does not price. Because nothing is removed, an old value stays in the context next to its replacement, and the model must pick the later one. Prior work shows this is hard: the latest of several revisions of a fact is retrieved less reliably as revisions accumulate (Qiao et al. 2026; Bajaj et al. 2025), and agents answer from observations the world has overtaken (Liu, Fang, and

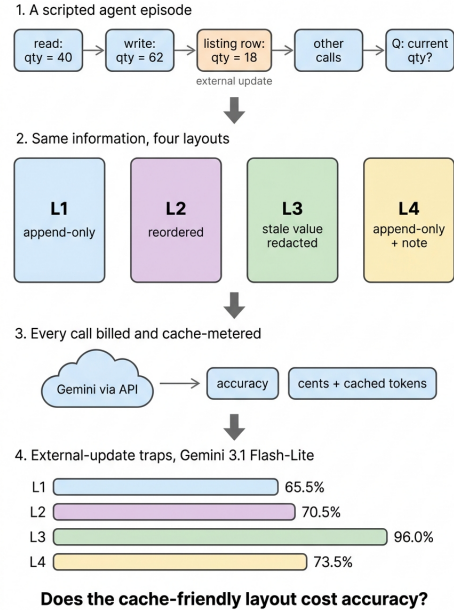


Figure 1: Study design. A scripted agent episode (the values shown are an example) is replayed under four layouts that carry the same information. Every call goes to the provider’s API, which reports billed cost and cached tokens. The bars give accuracy on external-update questions for Gemini 3.1 Flash-Lite.

Qian 2026). Two literatures sit on either side of the question. Context-management papers prune old observations and report accuracy, but price tokens at list rates with no cache discount (Lodha et al. 2026; Lindenbauer et al. 2025; Chintalapati et al. 2026). Caching papers report cost and latency but not accuracy (Lumer et al. 2026; Song 2026). Nobody has measured whether the cache-optimal layout itself, with the information held fixed, costs accuracy, or what that cost is worth against the cache discount.

We measure both with a controlled replay (Figure 1). In StaleTrap, a deterministic tool environment, we know by construction whether each question has an outdated value in the context, and whether the update came from the agent’s

own write or from an outside change that appears only as one row of a listing. We replay each episode under four layouts with the same information and record the provider’s bill and cached tokens for every call.

Our contributions are:

- **StaleTrap**, a frozen testbed of 144 scripted tool-using episodes in which every question is labelled as a stale-value trap (own or external update) or a neutral question.
- **A layout experiment that holds content fixed** and scores accuracy against provider-billed cost and cached tokens.
- **The finding that the accuracy cost is confined to external updates**: zero on own updates and neutral questions, 30.5 points on external updates for Gemini 3.1 Flash-Lite.
- **The finding that redaction is cheaper per correct episode at every cache-read price** on the tested models, because the provider’s cache rarely serves the history under any layout.

Related Work

Position and stale values. Models use the start and end of a long context better than its middle (Liu et al. 2023; Hsieh et al. 2024). Repeated keys are retrieved at their first and last occurrence (Bajaj et al. 2025), and the latest of several revisions is retrieved less reliably as revisions accumulate (Qiao et al. 2026). Reordering passages by relevance is no better than shuffling in realistic retrieval (Cuconasu et al. 2025). In agents, observations go stale (Liu, Fang, and Qian 2026), masking old observations helps mid-capacity models only (Zhang et al. 2026), and time stamps alone do not tell a model which version is current (Hu et al. 2026).

Prompt caching. Exact-prefix reuse is the production default (Zheng et al. 2023; Gim et al. 2023). Caching cuts the cost of long agent sessions by 40 to 80 percent, and the discount is lost when earlier bytes change (Lumer et al. 2026; Song 2026). Production hit rates are near 90 percent within a turn and 55 percent across turns (Liu et al. 2026). The closest work measures provider-billed cost, split into cache creation and reads, against task success, and shows that compressing tool outputs can raise the bill by changing trajectories (Weinberger and Hozes 2026); it varies compression, which changes content, not layout.

Context management and cost-aware evaluation. Pruning history to the last few tool pairs raises completion and cuts tokens (Lodha et al. 2026), and observation masking matches LLM summaries at a fraction of the cost (Lindenbauer et al. 2025; Chintalapati et al. 2026). Cost-controlled evaluation asks for an accuracy-against-dollars frontier (Kapoor et al. 2024; Erol et al. 2025; Kapoor et al. 2025). None of these isolates layout from content, and none includes the cache discount in the cost. Supplementary Table S1 compares our design with each close neighbour.

Experimental Setup

Testbed. StaleTrap exposes three back-office domains (warehouse inventory, a meeting calendar, a support-ticket queue) as 29 JSON-schema tools, plus 10 unused messaging

Model	Tasks	Layouts	Episodes
Gemini 2.5 Flash	72	L1, L3, L4	216
Gemini 3.1 Flash-Lite	144	L1–L4	576
Gemini 3.6 Flash	36	L1, L3	72

Table 1: Models and runs. Gemini 3.1 Flash-Lite runs the full grid; Gemini 2.5 Flash (older, thinking on) runs the seed-0 half; Gemini 3.6 Flash (newest) runs a 36-task balanced subset.

and HR tools that stay in every call. A local executor holds the state and returns realistic JSON. The fixed prefix (system prompt and tool declarations) is 5,102 tokens, above every provider’s minimum for caching. An episode is a scripted sequence of user instructions. Most ask for one tool call; some are questions answered from the context with tools disabled, scored by exact match against the executor’s live state. A **trap thread** reads a field, updates it one to three times, and then asks for its current value, so an outdated value is in context. A **neutral thread** reads a field once and asks for it later. In an *own* update the agent’s own write returns the new value. In an *external* update someone else changes the record, and the agent sees the new value only as one row of an unrelated listing. The grid crosses 3 domains, 3 task templates, 4 episode lengths (6 to 30 tool calls), 2 distractor levels and 2 seeds: 144 episodes, 720 questions (360 trap, 360 neutral). The testbed was frozen with a recorded hash before any scored run.

Layouts. Each turn, the runner rebuilds the message list under one of four rules. **L1** is the cache-optimal layout: every block in arrival order. **L2** keeps the same blocks but sorts earlier ones by relevance to the current turn, so the prefix changes every turn. **L3** keeps arrival order but replaces every value that a later result restated with “<superseded at step k >”. L3 uses the executor’s schema, so it is an upper bound for any real pruner, and it breaks the cached prefix at the first edit. **L4** is L1 plus a short note added to every user turn that says which earlier observations are current. The note never contains a value. L1, L2 and L4 carry identical information.

Models. Table 1 lists the three Gemini models, all called through OpenRouter at temperature 0 with minimal reasoning effort. We planned three model families, but a guardrail on the API account blocked every non-Google model, so the study is a three-generation Gemini ladder. Gemini 3.5 Flash-Lite was dropped because about a third of its tool calls were malformed. Every call logs the provider’s prompt, cached and completion tokens and the billed amount; no cost in this paper is estimated from a price sheet except where we say counterfactual.

Analysis. The unit is the task. For each model we take within-task paired differences between layouts in trap accuracy, neutral accuracy, episode success (all questions right), billed cost and cached share of prompt tokens. We report two-sided sign-flip permutation tests over tasks with 20,000 flips (Good 2005) and task-bootstrap 95% intervals with 10,000 resamples (Efron and Tibshirani 1994). Benjamini–

Hochberg correction runs over the whole family of 65 tests (Benjamini and Hochberg 1995); 29 survive at $q = 0.05$. Following cost-controlled evaluation (Kapoor et al. 2024), our cost metric is the billed cost per correct episode,

$$C_{\text{correct}} = \frac{\sum_e \text{cost}(e)}{\sum_e \text{success}(e)}, \quad (1)$$

which we also recompute with every cached token repriced at 0 to 1 times the input price. We hand-coded 40 wrong answers by error type. Five directional predictions were logged before the first model call.

Results

The accuracy cost is confined to external updates. Table 2 gives the per-layout results. On Gemini 3.1 Flash-Lite, L1 answers 80.6% of trap questions and L3 answers 98.3%. The paired difference is $-0.177 [-0.230, -0.128]$, $p < 10^{-4}$, and survives correction (Table 3). Neutral questions are answered perfectly under every layout. Splitting traps by kind locates the whole effect (Figure 2). The 160 own-update traps are answered perfectly under every layout. On the 200 external-update traps, L1 scores 65.5%, L2 70.5%, L3 96.0% and L4 73.5%. When the agent’s own tool reply states the current value, the old copy does no harm. When the current value arrived as one row of a listing fetched for another reason, the append-only agent answers from its own earlier write about one time in three. The gap does not grow with episode length: it is -0.22 at six calls and -0.22 at thirty (slope $+0.0006$ per call, $p = 0.85$), and does not depend on the distractor level ($p = 0.54$).

Reordering does nothing; the note recovers a quarter. Reordering by relevance (L2) does not improve on L1 (-0.036 , $p = 0.11$), and it rewrites the prefix every turn. The note (L4) improves on L1 by $+0.073$ ($p = 0.001$), which is 42% of the L1-to-L3 gap over all traps and 26% on external traps, while keeping L1’s cached share. L4 remains well below L3 (-0.104 , $p < 10^{-4}$).

What the wrong answers are. Of 40 hand-coded wrong trap answers, 34 are an outdated value read back, almost always the agent’s own earlier write, with the newer listing row ignored. Three are threshold questions where a stale read and a misread inequality coincide, two are cases where the agent skipped the listing call so the new value never reached its context, and one is a derived number. The agent executed the reveal steps 99.0% to 100% of the time under every layout, and restricting to fully executed threads moves no layout’s accuracy by more than half a point.

The cheapest layout per episode is not the cheapest per correct episode. L1 is the cheapest layout per episode at 1.05 cents; L3 costs 1.16 cents (paired $+0.10$ cents, $p < 10^{-4}$). Per correct episode (Eq. 1) the order reverses: L3 1.22 cents, L1 1.52, L2 1.85, L4 2.07. L1 and L3 are the only points on the accuracy-cost frontier (Figure 3). Repricing every cached token from free to the full input price never changes the ranking: L3 is cheapest per correct episode at every price. On this provider no cache-read price makes the cache-optimal layout pay for its errors.

Why: the history is rarely cached. Over an episode, the cached share of prompt tokens is 0.71 under L1 and 0.68 under L3, but most of that is the 5,100-token fixed prefix. Counting only the history, the cached share is 0.24 under L1, 0.13 under L2, 0.15 under L3 and 0.45 under L4. Google’s implicit cache serves the growing history on a minority of turns even when the bytes do not change, and L3’s shorter prompts (118k against 122k tokens per episode) recover most of the difference. The L4 note, repeated on every turn, raises prompt tokens by 69% and the bill by 51%, which is what puts L4 off the frontier. As a substitute for a cache on/off control, we compared the same question at identical bytes served from cache in some runs and not in others: the paired difference in trap accuracy is 0.000 over 11 questions ($p = 1$).

Three Gemini generations. External-trap accuracy under L1 rises with generation: 0.40 on Gemini 2.5 Flash, 0.655 on 3.1 Flash-Lite and 0.80 on 3.6 Flash. Under L3 it is 0.84, 0.96 and 1.00. The paired L1-minus-L3 trap gap shrinks from -0.273 to -0.177 to -0.111 (between-model permutation $p = 0.035$ over all tasks, 0.092 on the 36 shared tasks). The note helps the older model most: on 2.5 Flash it recovers 79% of the gap, raising external-trap accuracy from 0.40 to 0.75, but it is still off the frontier. The cost ordering holds on every model: L3 is the cheapest per correct episode (1.78 against 2.71 cents on 2.5 Flash; 3.67 against 4.21 on 3.6 Flash) at every counterfactual cache-read price.

External benchmark. On BFCL v3 multi-turn (Patil et al. 2025), with Gemini 3.1 Flash-Lite and 50 tasks, no layout moves accuracy by more than three tasks (0.44 for L1 on both splits). The failures there are state mismatches and long loops rather than stale reads, so BFCL neither confirms nor contradicts the mechanism.

Discussion

Comparison with prior work. Observation masking and history pruning have been reported to keep or raise accuracy while cutting tokens (Lindenbauer et al. 2025; Lodha et al. 2026), but those comparisons price tokens at list rates. Adding the cache discount does not reverse their conclusion here: our redaction layout, the content-only intervention, is both more accurate and cheaper per correct answer than append-only. Prior caching work reports 40 to 80 percent savings (Lumer et al. 2026) and production hit rates near 90 percent within a turn (Liu et al. 2026); our measured history hit rate on Google’s implicit cache is much lower, which is why the expected cost advantage of append-only did not appear. Compared with the recency findings of Qiao et al. (2026), our gap does not grow with the number of intervening calls: one external update is enough.

What deployers should do. Keep the fixed prefix, since it is what the cache actually serves. When a later observation replaces an earlier value for the same field, overwrite the earlier value even though this breaks the cache for the history; on the tested provider, the accuracy is worth more than the discount. A pointer-only note is a cheaper partial fix for older models, but its repeated tokens make it the most expensive layout.

Model	Layout	Trap acc. [95% CI]	External trap	Success	Cents/episode	Cents/correct	Cached share
Gemini 2.5 Flash	L1 append-only	0.660 [0.565, 0.751]	0.40	0.528	1.43	2.71	0.716
	L3 redacted	0.933 [0.889, 0.971]	0.84	0.861	1.54	1.78	0.710
	L4 note	0.875 [0.811, 0.932]	0.75	0.778	2.45	3.15	0.683
Gemini 3.1 Flash-Lite	L1 append-only	0.806 [0.752, 0.857]	0.655	0.694	1.05	1.52	0.708
	L2 reordered	0.841 [0.792, 0.887]	0.705	0.729	1.35	1.85	0.619
	L3 redacted	0.983 [0.969, 0.994]	0.960	0.944	1.16	1.22	0.676
	L4 note	0.879 [0.837, 0.917]	0.735	0.771	1.60	2.07	0.711
Gemini 3.6 Flash	L1 append-only	0.889 [0.787, 0.972]	0.80	0.833	3.51	4.21	0.475
	L3 redacted	1.000 [1.000, 1.000]	1.00	0.972	3.57	3.67	0.464

Table 2: Per-layout results. Trap accuracy is the per-episode mean over all stale-value traps with a task-bootstrap 95% interval; external trap is accuracy on external-update traps only (own-update traps and neutral questions are at or near 1.00 everywhere: neutral accuracy is 1.000 in every cell except Gemini 3.6 Flash L3, 0.972). Cost is provider-billed. **Bold**: best layout per model in each column.

THE SAME INFORMATION, FOUR LAYOUTS: ONLY EXTERNAL-UPDATE TRAPS SEPARATE THEM

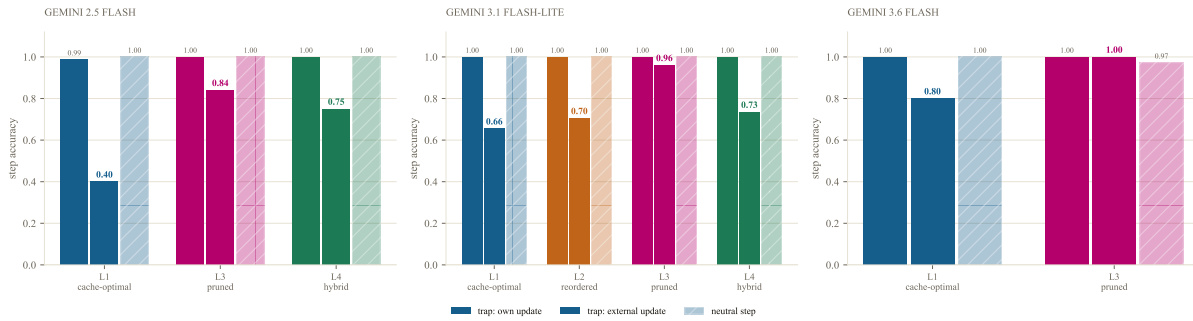


Figure 2: Question accuracy by layout on own-update traps, external-update traps and neutral questions, per model. Only the external-update bars move.

Contrast	Trap acc. [95% CI]	Cents/episode
L1 – L2 (order)	−0.036 [−0.079, 0.006]	−0.30
L1 – L3 (content)	−0.177 [−0.230, −0.128]	−0.10
L4 – L1 (note)	+0.073 [0.031, 0.120]	+0.54
L4 – L3	−0.104 [−0.145, −0.066]	+0.44

Table 3: Paired within-task contrasts, Gemini 3.1 Flash-Lite (144 tasks). **Bold**: survives Benjamini–Hochberg correction. Full contrasts in the supplementary material.

Pre-registered predictions. D1 (a stale-value cost on traps but not on neutral questions) held, except that the gap did not grow with length. D2 (reordering does not help and costs more) held in direction but not in magnitude. D3 (the note dominates) was not supported. D4 (the gap shrinks with model generation) held in direction. D5 (a cache-price threshold where append-only becomes cheapest) was reversed: no threshold exists on this provider.

Conclusion

The cache-optimal layout costs nothing when the agent’s own tool replies carry the current value, and costs about a third

of the answers when the world changes behind the agent and the new value arrives inside a listing. Reordering does not reduce this cost, and a short note recovers about a quarter of it at a price in tokens. Redacting the stale value removes the cost almost entirely, and on the tested provider it is cheaper per correct episode at every cache-read price. Across three Gemini generations the gap shrinks from 27 to 11 points but does not close.

Future work. First, repeat the study on providers with explicit cache breakpoints and high history hit rates, where the discount is larger and a price threshold may exist. Second, replace the oracle redaction with a learned pruner to measure how much of the gap a deployable system can recover. Third, test the mechanism on public agent benchmarks with natural external updates.

Limitations

Every number here is from one provider family, because an account guardrail blocked the other providers we planned. The cost conclusion depends on Google’s implicit cache rarely serving the history; a provider with explicit breakpoints would give append-only a larger cost advantage. L3 redacts exactly the values a later result restates, using the

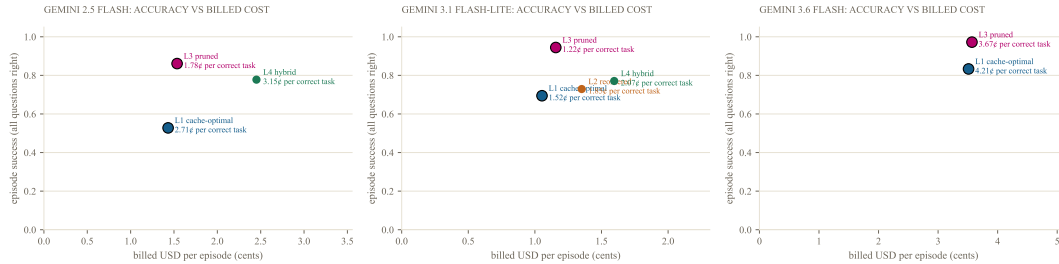


Figure 3: Episode success against billed cents per episode, per model. Outlined points are on the frontier; cents per correct episode are annotated.

executor’s schema; no deployed pruner has that oracle, so L3 is an upper bound. StaleTrap is scripted and synthetic, so absolute accuracies are not deployment numbers, and we have not shown the mechanism on a public benchmark. Identical temperature-0 runs flip a trap outcome in about one cell in nine; the paired tests over 144 tasks absorb this noise. The error audit was not blind.

Ethical Considerations

The testbed is synthetic: fictional records and names, no personal data. No deployed system was probed beyond public model APIs. The work measures the cost of an engineering practice and introduces no new capability.

Use of AI Assistants

We used AI tools at several stages, and the author checked all outputs. Claude (Anthropic) ran the experiment pipeline as a coding agent, wrote the testbed, runner and analysis code, coded the 40 audited errors, and helped draft and edit the text. Figure 1 was generated with a Gemini image model (Nano Banana) from a text description of the design, and every label was checked by hand; all data figures are plotted from our results. LLMs are also the subjects of the study: the three Gemini models are the agents being measured. Every reported number was checked against the output of the analysis script.

References

- Anthropic. 2025. Effective context engineering for AI agents. <https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents>. Engineering blog, 29 September 2025; accessed 2026-10-10.
- Bajaj, A.; Mistry, D. M.; Maini, S. S.; Aggarwal, Y.; and Tiganj, Z. 2025. Beyond Semantics: How Temporal Biases Shape Retrieval in Transformer and State-Space Models. *arXiv preprint arXiv:2510.22752*.
- Benjamini, Y.; and Hochberg, Y. 1995. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society: Series B*, 57(1): 289–300.
- Chintalapati, R.; Raskar, S.; Acharya, A.; Willard, J.; Emami, P.; and Horawalavithana, S. 2026. Evaluating Memory Condensation Strategies for Coding Agents in Data-Driven Scientific Discovery. *arXiv preprint arXiv:2605.18854*.
- Cuconasu, F.; Filice, S.; Horowitz, G.; Maarek, Y.; and Silvestri, F. 2025. Do RAG Systems Really Suffer From Positional Bias? In *EMNLP 2025*. ArXiv preprint arXiv:2505.15561.
- Efron, B.; and Tibshirani, R. J. 1994. *An Introduction to the Bootstrap*. Chapman and Hall/CRC.
- Erol, M. H.; El, B.; Suzgun, M.; Yuksekgonul, M.; and Zou, J. 2025. Cost-of-Pass: An Economic Framework for Evaluating Language Models. In *ICLR 2026*. ArXiv preprint arXiv:2504.13359.
- Gim, I.; Chen, G.; seob Lee, S.; Sarda, N.; Khandelwal, A.; and Zhong, L. 2023. Prompt Cache: Modular Attention Reuse for Low-Latency Inference. In *MLSys 2024*. ArXiv preprint arXiv:2311.04934.
- Good, P. I. 2005. *Permutation, Parametric, and Bootstrap Tests of Hypotheses*. Springer, 3 edition.
- Hsieh, C.-Y.; Chuang, Y.-S.; Li, C.-L.; Wang, Z.; Le, L. T.; Kumar, A.; Glass, J.; Ratner, A.; et al. 2024. Found in the Middle: Calibrating Positional Attention Bias Improves Long Context Utilization. In *Findings of ACL 2024*. ArXiv preprint arXiv:2406.16008.
- Hu, C.; Li, T.; Gao, X.; Chen, H.; Bai, Y.; Xu, D.; Lin, T.; Li, X.; et al. 2026. Evaluating Long-Horizon Memory for Multi-Party Collaborative Dialogues. *arXiv preprint arXiv:2602.01313*.
- Ji, Y. 2025. Context Engineering for AI Agents: Lessons from Building Manus. <https://manus.im/blog/Context-Engineering-for-AI-Agents-Lessons-from-Building-Manus>. Blog post, 18 July 2025; accessed 2026-10-10.
- Kapoor, S.; Stroebl, B.; Kirgis, P.; Nadgir, N.; Siegel, Z. S.; Wei, B.; Xue, T.; Chen, Z.; et al. 2025. Holistic Agent Leaderboard: The Missing Infrastructure for AI Agent Evaluation. In *ICLR 2026*. ArXiv preprint arXiv:2510.11977.
- Kapoor, S.; Stroebl, B.; Siegel, Z. S.; Nadgir, N.; and Narayanan, A. 2024. AI Agents That Matter. In *TMLR 2025*. ArXiv preprint arXiv:2407.01502.

Lindenbauer, T.; Slinko, I.; Felder, L.; Bogomolov, E.; and Zharov, Y. 2025. The Complexity Trap: Simple Observation Masking Is as Efficient as LLM Summarization for Agent Context Management. In *DL4C Workshop at NeurIPS 2025*. ArXiv preprint arXiv:2508.21433.

Liu, B.; Qiu, H.; Íñigo Goiri; Fonseca, R.; Bianchini, R.; and Choukse, E. 2026. Agentic Coding in the Wild: Characterizing GitHub Copilot Traces at Production Scale. *arXiv preprint arXiv:2608.00101*.

Liu, N. F.; Lin, K.; Hewitt, J.; Paranjape, A.; Bevilacqua, M.; Petroni, F.; and Liang, P. 2023. Lost in the Middle: How Language Models Use Long Contexts. In *Transactions of the ACL*. ArXiv preprint arXiv:2307.03172.

Liu, Y.; Fang, J.; and Qian, C. 2026. When Agent Context Goes Stale: Incoherence in Volatile Agent Context. *arXiv preprint arXiv:2610.05281*.

Lodha, A.; Varnosfaderani, M. P.; Chakraborty, A.; and Mithal, A. 2026. Less Context, Better Agents: Efficient Context Engineering for Long-Horizon Tool-Using LLM Agents. *arXiv preprint arXiv:2606.10209*.

Lumer, E.; Nizar, F.; Jangiti, A.; Frank, K.; Gulati, A.; Phadate, M.; and Subbiah, V. K. 2026. Don't Break the Cache: An Evaluation of Prompt Caching for Long-Horizon Agentic Tasks. *arXiv preprint arXiv:2601.06007*.

Patil, S. G.; Mao, H.; Yan, F.; Ji, C. C.-J.; Suresh, V.; Stoica, I.; and Gonzalez, J. E. 2025. The Berkeley Function Calling Leaderboard (BFCL): From Tool Use to Agentic Evaluation of Large Language Models. In *Proceedings of the 42nd International Conference on Machine Learning*.

Qiao, B.; Guo, S.; Yang, X.; Li, K.; Zhou, W.; Hu, S.; and Song, Y. 2026. Diagnosing Retrieval Bias Under Multiple In-Context Knowledge Updates in Large Language Models. *arXiv preprint arXiv:2603.12271*.

Song, Y. 2026. Cache-Aware Prompt Compression: A Two-Tier Cost Model for LLM API Caching. *arXiv preprint arXiv:2607.15516*.

Weinberger, S.; and Hozez, A. 2026. Token Reduction Is Not Cost Reduction. *arXiv preprint arXiv:2607.12161*.

Zhang, H.; Xu, Q.; Li, Z.; Zhang, L.; Jiang, P.; Zhang, Y.; and McAuley, J. 2026. Masking Stale Observations Helps Search Agents – Until It Doesn't: A Regime Map and Its Mechanism. *arXiv preprint arXiv:2606.00408*.

Zheng, L.; Yin, L.; Xie, Z.; Sun, C.; Huang, J.; Yu, C. H.; Cao, S.; Kozyrakis, C.; et al. 2023. SGLang: Efficient Execution of Structured Language Model Programs. In *NeurIPS 2024*. ArXiv preprint arXiv:2312.07104.