Do LLMs Benefit From Their Own Words?

Jenny Y. Huang[1,2], Leshem Choshen[1,2,3], Wei Sun[2,3], Omar Khattab[1], Ramón Fernandez Astudillo[2,3], Mehul Damani[1], Tamara Broderick[1,2]*, Jacob Andreas[1,2]*
[1]Department of Electrical Engineering and Computer Science, MIT
[2]MIT–IBM Watson AI Lab [3]IBM Research
jhuang9@mit.edu

Abstract

In multi-turn conversations, large language models typically condition on the full conversation history: both past user prompts and assistant responses. We revisit this design choice by comparing full-context prompting to four alternative, substantially-reduced context configurations. Analyzing in-the-wild multi-turn conversations across three open reasoning and one state-of-the-art model, we find that response quality is largely preserved under aggressive context filtering: replacing all prior assistant turns with one-sentence summaries or keeping only the most recent user–assistant exchange often matches storing full context in performance while using roughly 8× less context. To understand this result, we observe that a substantial fraction of user turns (36.4%) in multi-turn conversations are self-contained and that many follow-up turns can be addressed by seeing only the immediately preceding user–assistant exchange. Furthermore, we find that when models condition on their own past responses, this can lead to *context pollution*, a phenomenon in which reasoning errors, hallucinations, or stylistic artifacts propagate across turns. Motivated by these findings, we design a context-filtering approach that selectively omits the assistant-side history. Taken together, these findings suggest moving away from storing full dialogue transcripts and instead retaining only what is relevant.

1 Introduction

As large language models (LLMs) are deployed in increasingly complex multi-turn conversations, context management becomes an important challenge. Long contexts increase computational costs (Eyuboglu et al., 2026; Dao et al., 2022; Xiao et al., 2025), slow inference speeds, and can impair a model's capacity to attend to relevant information (Liu et al., 2024; Lee et al., 2026). In response, production systems like Claude Code, Codex, and Cursor have adopted context engineering strategies. These methods include discarding or externalizing tool outputs and reasoning traces (Cursor, 2026; Anthropic, 2025b), summarizing early conversation turns when context limits are

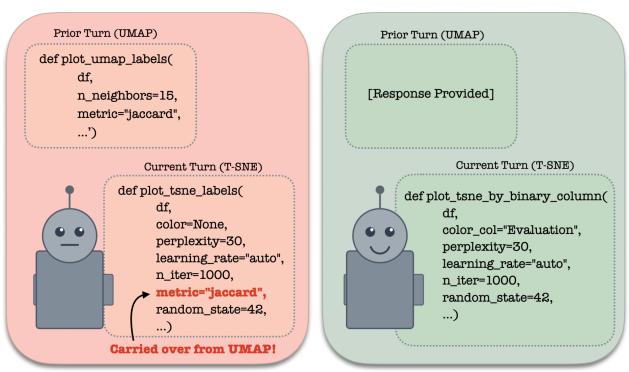

Figure 1: Real example of overrelying on irrelevant context. In Turn 2, the user requests UMAP clustering code. In Turn 5, the user says, "use t-SNE instead." Left: When the previous assistant response remains in context, the model incorrectly carries over the Jaccard metric from UMAP into the t-SNE implementation, introducing a bug. Right: Without the previous response in context, the model generates correct t-SNE code.

*Equal advising.

reached, and allowing users to manually clear or compact conversation histories (Anthropic, 2025b). Despite increasing efforts to compress and prune the older segments of conversation history, one key idea remains largely unexamined: that conditioning on the *full* dialogue transcript, *especially the assistant's own past responses*, is a necessary and beneficial design choice for maintaining downstream response quality.

An earlier line of works on conversational question answering (ConvQA) (Qu et al., 2019; Do et al., 2022; Zaib et al., 2023) found that only a select subset of past conversation turns are useful for answering current turns and that storing irrelevant turns in context can actually degrade downstream performance. The work, however, analyzed human–human, rather than human-LLM, conversations (e.g., QuAC, CoQA). Unlike human responses, LLM-generated responses contain long, speculative, and error-prone reasoning traces. Such LLM-generated content may introduce unique sources of distraction, and even state-of-the-art models tend to be distracted by irrelevant information in context (Hong et al., 2025). A recent line of work on multi-turn LLM conversations (Laban et al., 2026) documented substantial performance degradation attributable to the model "taking a wrong turn" early on in a conversation. Later work (Khalid et al., 2025) proposed to address this concern by consolidating past user turns into a single prompt when context degradation is detected. Both studies, however, relied on a narrow simulation setup in which user turns are shards of a single instruction revealed incrementally (see properties 1-5 in Appendix B of Laban et al., 2026). This design fails to capture various dynamics found in real-world multi-turn conversations, such as mid-conversation topic shifts and self-contained user follow-ups (see Fig. 3, left). Taken together, these gaps make it unclear whether preserving past LLM-generated responses is beneficial, neutral, or harmful in real-world human–AI conversations, an empirical question we set out to address.

Standard multi-turn prompting assumes that models should condition on the full conversation history. We challenge this assumption by comparing full-context prompting against four *substantially*-reduced context configurations: keeping only the last user–assistant exchange preceding the current turn, keeping only the past *user* turns, keeping the past user turns and a one-sentence summary of each assistant turn, or deleting the entire preceding conversation history. We analyze in-the-wild, multi-turn conversations (Zhao et al., 2024; Don-Yehiya et al., 2025) across three open reasoning and one frontier model. We find, to our surprise, that models can often *maintain comparable response quality in a multi-turn conversation given substantially less context.*

To investigate this finding, we analyze both the structure of in-the-wild multi-turn conversations and the effects of conditioning on prior model responses, making two key observations:

(1) Multi-turn dependence is not inherent in real-world multi-turn conversations (see Section 3). Using a language model to classify turns based on their dependence on prior turns, we observe that a substantive fraction (36.4%)[1] of turns in real-world multi-turn conversations are largely self-contained. The composition of real-world multi-turn chats in our analysis suggests that careful, generous filtering of past assistant responses may be justified.

(2) Models tend to over-condition on their past responses, resulting in *context pollution*: a phenomenon in which earlier model-generated outputs introduce errors, hallucinations, or stylistic artifacts that propagate into subsequent turns (see Fig. 1 and Section 4). This tendency for models to over-condition on their earlier responses has been observed as well in past studies (Laban et al., 2026; Sinha et al., 2026; Li et al., 2025; Hong et al., 2025; Lee et al., 2026) and is a behavior that does not seem to mitigate with model size (Sinha et al., 2026). We characterize different forms of context pollution in Section 4 and provide 11 illustrative examples in Section A.22.

Motivated by these findings, we develop an adaptive assistant-response-omission strategy that trains a classifier to predict, for a given user prompt and conversation history, whether the LLM should condition on its past responses (Section 5).

Together, our findings call into question the design choice of storing full dialogue transcripts in context.

[1]See details of this experiment in Section A.7

1.1 Context Management

Single-turn prompt compression. A line of past works studies prompt compression in a single-query setting (Khattab et al., 2021; Wang et al., 2023; Xu et al., 2024; Chuang et al., 2024; Chirkova et al., 2025). Many of these works focus on retrieval-augmented generation (RAG), where retrieved documents are filtered or compressed before being provided to the model. Additional single-turn compression methods, such as Lingua-2 (Pan et al., 2024) and PENCIL (Yang et al., 2025b), remove redundant reasoning traces and are likewise evaluated at the level of individual queries. In this work, we are interested in condensing conversation histories of multi-turn conversations.

Multi-turn context editing. Prior works on context editing within multi-turn conversations (Jiang et al., 2023; Li, 2023) treat the *full* conversation history as a gold standard, evaluating proposed compression methods against full-context-conditioned responses, using similarity metrics like BERTScore and BLEU/ROUGE scores. These works do not question the validity of full-context prompting as the gold standard. A notable exception is ERGO (Khalid et al., 2025), which proposes rewriting the prior user turns into a single prompt while omitting past assistant responses. Their results show that prompt consolidation and assistant-response omission can outperform full-context conditioning on multi-turn math and coding. However, their findings are made exclusively on synthetic conversations.

Context management in agentic systems. Beyond the chat setting, agentic systems continue to develop new context management strategies daily. Production systems like Claude Code (Anthropic, 2025b), Codex (Lopopolo, 2026), and others (Cursor, 2026; Liu et al., 2025; Qwen Team, 2025a; Gao et al., 2025), clear or compress tool outputs and thinking traces and perform truncation when context windows become saturated. A growing ecosystem of memory tools focuses on storing reusable information or content that is most difficult to reconstruct (e.g., tool outputs, bug fixes, or global constraints) in external files for later reference (Anthropic, 2025a; AI DevKit, 2024). Some works suggest outsourcing the role of context management to a smaller LLM (Xiao et al., 2025; Lindenbauer et al., 2025), while others (Liu et al., 2026) suggest training the model to manage its own context.

We identify two gaps in the current literature. First, there is a lack of *evaluation on real-world multi-turn conversational data*. Second, both research and deployed systems often treat the storage of full conversations as a default design choice. In this study, using real-world chat logs on Wildchat (Zhao et al., 2024) and ShareLM (Don-Yehiya et al., 2025), we find that a substantial portion of conversation histories are rarely referenced in later turns and that the response quality of multi-turn chats can be maintained at a comparable level with minimal context.

2 Experiment Setup

Datasets and Models. To evaluate whether retaining the full conversation history provides measurable benefits to downstream response quality, we conduct a controlled experiment across four LLMs spanning a range of model sizes: Qwen3-4B (Yang et al., 2025a), DeepSeek-R1-Distill-Llama-8B (DeepSeek-AI, 2025), GPT-OSS-20B (Agarwal et al., 2025), and GPT-5.2 (OpenAI, 2025a). We use real-world multi-turn conversations drawn from `allenai/WildChat-4.8M` (Zhao et al., 2024) and `shachardon/ShareLM` (Don-Yehiya et al., 2025). Our analysis includes conversations in the domains of math, coding, science, business/finance, education, and creative writing. See Appendix A.1 for additional details on the datasets and our data filtering process.

Generating Responses. For each model, we generate responses under four different context configurations: **Full Context (FC)**, in which the model is prompted with the entire conversation history (both past user prompts and assistant responses); **Assistant-Omitted (AO)** context, in which the model is prompted with only the content of prior *user* turns, while assistant responses are replaced with the placeholder `[Response provided]` to maintain the alternating user/assistant turn structure; **Last-Turn-Only (LT)**, in which the model is prompted with only the most recent user-assistant exchange; **Summarized (SUM)**, in which each assistant response is replaced with a one-sentence summary of that response, given by the same model being evaluated prompted with: `Summarize the following assistant response in exactly one sentence:{response}`; and **Delete-Everything (DE)**, in which the model receives only the current user prompt with no prior history. See Appendix A.3 for the prompt templates for all four context configurations. For all conversations, we retain the historical user prompts and assistant responses from the data as the conversation history. In the appendix, we present an ablation study where the model's own generated responses are fed back into the context for subsequent rounds (see Fig. 9).

Evaluating Responses. To evaluate responses, we use GPT-5 as an LLM-judge. For each conversation round starting from round 2, where the context configurations begin to diverge, the LLM-judge receives both the full context and the alternative context responses alongside the full conversation history (all past user and assistant turns). It then selects a winner (or declares a tie) for each of two evaluation dimensions: *response quality* (Quality), whether a response clearly and accurately addressed the user's current prompt, and *task adherence* (On-Topic), whether the response remains focused on the conversation thread. We randomize response ordering for each comparison to reduce position bias. Since we set out to investigate the impact of distraction from accumulated assistant responses, one natural concern is that the LLM-judge itself may be susceptible to distraction. To address this concern, we supplement the full-context LLM-judge with a variant that receives only the prior user turns during evaluation (see Section A.4). In our main-text experiments, the judge is given both the thinking trace and the final response to perform its evaluation.[2] In the appendix, we provide an additional experiment in which the judge compares only on final responses (see Section A.12). We provide the full judge prompts in Section A.4.

3 Response Quality Can Be Maintained Under Much Less Context

Across all models, response quality is often maintained under a sparse subset of the conversation history. Figure 2 presents these results, and Section A.6 gives a more quantitative comparison of context omission strategies. Out of the strategies we tested, storing *one-sentence summaries* of assistant turns (Fig. 2, top row) is the most effective, often exceeding the performance of storing full responses (FC). We find that when responses from prior turns are retained in full, this can lead models to rehash their earlier responses, rather than addressing the current prompt. Replacing those turns with summaries yields shorter, more focused responses, reducing median response length by roughly 25% across the four models. Section A.9 gives two examples of this behavior.

Storing the *last turn only* (LT) (Fig. 2, second row) is often sufficient for maintaining performance, on both the Quality and On-Topic dimensions, compared to storing the full conversation (blue bars). We find that a substantial fraction of user turns (40%) reference *at most* the user–assistant exchange *directly preceding* the current turn (Fig. 4 (left)). See Section A.19 for more details on this finding.

Storing the *user turns only* (AO) (Fig. 2, third row), results in modest Quality degradation but comparable On-Topic win rates. In ablations where we (i) condition on assistant responses from the generator model[3] (Figure 9) or (ii) present the judge model with only the user turns to make its judgements (Figure 10), we find that the AO context often performs *comparably* to full context. Unsurprisingly, removing the entire conversation history, delete-everything (DE), results in substantial performance degradation (Fig. 2 (bottom row)).

Notably, we find that many user requests in the middle of multi-turn chats (36.4% across our dataset) can be addressed based on the *current prompt alone*; see more details of this finding in Section A.7. And for prompts that are explicit follow-ups to previous conversation turns,

[2] With the exception of GPT-5.2, whose responses do not expose thinking traces.

[3] Presently, we use the historical assistant responses from the dataset.

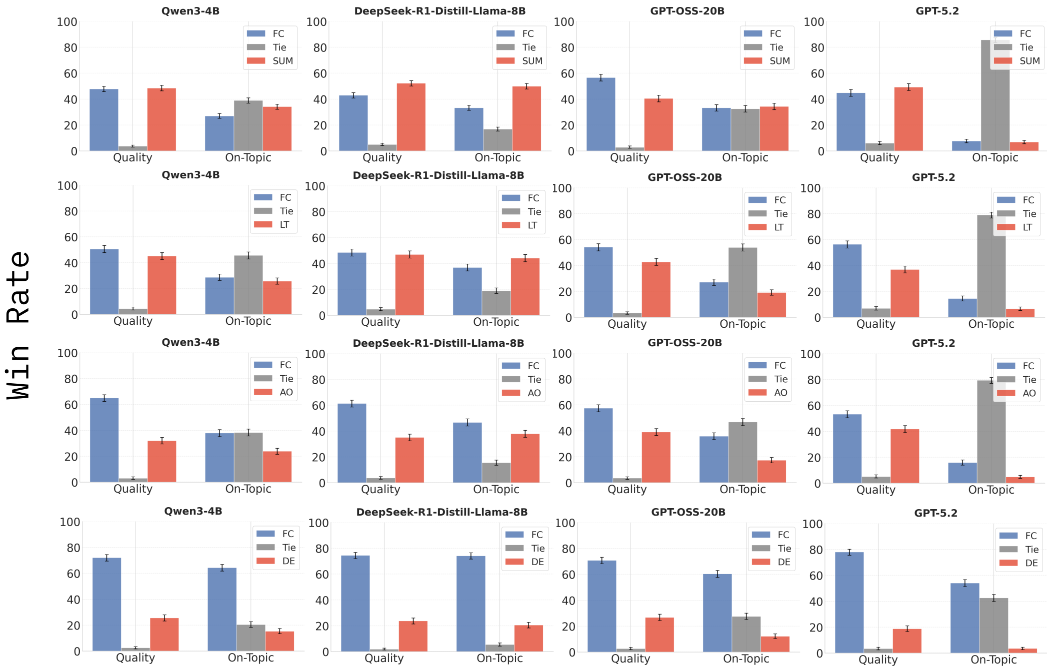

Figure 2: Pairwise win rates between Full Context (FC) prompting and three reduced-context configurations across four models (Qwen3-4B, DeepSeek-R1-Distill-Llama-8B, GPT-OSS-20B, and GPT-5.2), evaluated on the Quality and On-Topic dimensions. **Top row:** FC vs. summarized assistant history (SUM), where each prior assistant response is replaced with a one-sentence summary. **Second row:** FC vs. Last Turn Only (LT), which retains only the most recent user–assistant round. **Third row:** FC vs. Assistant Omitted (AO), which retains all user turns but replaces assistant responses with a placeholder. **Bottom row:** FC vs. Delete Everything (DE), which provides only the current user prompt. Error bars indicate binomial proportion 95% confidence intervals.

many prompts provide concrete-enough instructions to be answered *from scratch* based on the past user turns alone (see Figure 3).[4]

In the left and right conversations (Fig. 3), the follow-up prompt either contains concrete-enough instruction for the model to generate a response *from scratch* without seeing any prior assistant turns (left) or references only the prior *user* turns (right). Thus, a model can plausibly respond without seeing any of the assistant-side history. Fig. 3 (middle) presents a case where a past assistant response is referenced and thus necessary for answering the follow-up prompt. Even in this case, however, storing only the referenced response is sufficient. These findings challenge the assumption that uniformly storing assistant-side history is necessary. Furthermore, we identify three prominent conversation structures in our datasets: (1) an initial primary prompt (e.g., a reading passage) followed by queries about that prompt; (2) a sequence of loosely related, standalone queries; and (3) a cohesive thread characterized by iterative user feedback (e.g., fixing a buggy piece of code).[5] Each conversation structure entails a different degree of reliance on storing past assistant responses; for representative conversations belonging to each category, see Section A.16.

Alternative context configurations offer substantial compute savings. Despite achieving comparable response quality, the reduced-context configurations *significantly* decrease the

[4]We manually inspect 50 full-length conversations across WildChat and ShareLM to reach this conclusion.

[5]These categories were identified through a manual reading of 50 random conversations on WildChat and ShareLM.

Organizing Storage Drives. User asks the assistant for advice on how to set up storage drives.

[User Turn 1]

Provide a design for a disk topology for a NAS built on TrueNAS Scale, as well as a dataset layout [user-inserted specifications]

[User Turn 2]

1. Let's not get this granular. 2. Far less than 0.5TB/yr. 3. Some irreplaceable data [. . .] 4. Primarily existing hardware. 5. Main goal: prevent data loss.

Adding In-text Citations. User shares a written passage and asks the assistant to add citations.

[User Turn 6]

[user-provided text] Please do in-text citations and give me references for the above information.

[User Turn 7]

The second one is not working.

[User Turn 8]

This reference is also not good.

Analyzing a Reading Passage. User shares a reading passage and asks the assistant questions about the passage.

[User Turn 1]

What's your take on the following text? [user-inserted reading passage]

[User Turn 3]

What influences or inspirations do you notice in the text?

Figure 3: Three example conversations on WildChat illustrating different types of follow-ups. **Left:** the user provides feedback that is concrete enough that the model can respond from scratch using the previous user turn and the updated specifications alone. **Middle:** the follow-ups reference specific parts in past *assistant* turns, making it necessary to see the referenced assistant turn. **Right:** the follow-up references a prior *user* turn without concrete feedback; no assistant history is necessary.

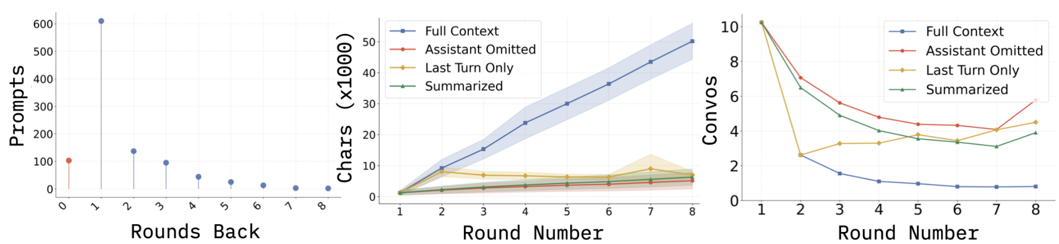

Figure 4: **Left:** Distribution of conversation rounds referenced by user prompts from round 2 onward, across 150 WildChat conversations. The X-axis represents the number of rounds back from the current round that a given prompt references. The red stem at 0 indicates prompts that are self-contained (no prior rounds referenced). **Middle:** Mean cumulative context length across conversations on WildChat as a function of conversation round, comparing Full Context, Assistant Omitted, Last Turn Only, and Summarized context. Shaded band shows ± 2 standard errors. **Right:** The number of conversations that can be answered using a 5K-token budget as a function of conversation round.

amount of compute spent across multi-turn conversations. Figure 4 (center) plots the cumulative context length in characters as a function of conversation length.[6] While the Full Context increases linearly as the conversation expands, the AO, LT, and SUM contexts remain relatively constant. Figure 4 (right) shows the number of conversations that are answerable within a 5k-token budget as a function of conversation length. For a chat of length 8 rounds, the AO, LT, and SUM contexts can successfully answer 4–6 conversations on average, whereas FC can answer only one.

4 LLMs Tend to Over-Condition on Their Past Responses

There is no shortage of anecdotal evidence online suggesting that LLM response quality degrades over the course of multi-turn chats. Users report that responses tend to get "less

[6]These plots are created based on GPT-5.2 generations.

precise, more repetitive, and sometimes subtly wrong"[7] as the chat progresses. Others note that conversations gradually "losing a clear direction," with assistants responses feeling like "a mix of adjustments instead of one clear goal."[8] Some even reach the conclusion: "If you want the maximum quality in your responses, you only can get 3–5 responses before you need to start over."[7] We indeed find instances on WildChat where the user abandons a conversation in favor of a new chat (see Section A.22.6).

In this section, we investigate multi-turn performance degradation through the concept of *context pollution*:[9] the tendency for models to over-condition on their prior responses, leading to errors, hallucinations, and stylistic artifacts that cascade across turns. To avoid quality decline in multi-turn conversations, future work could move toward *multi-turn* reinforcement learning (RL). We hope that the context pollution examples we present (see Section A.22) will be useful for designing future annotation rubrics for human annotators and LLM-judges to assign quality scores to conversation turns.

Next, we show that multi-turn math performance can degrade when past responses are maintained in context. Then, we provide a characterization of different forms of context pollution in multi-turn conversations. Finally, we discuss a promising design for reducing context pollution.

4.1 Multi-Turn Math Performance Degrades When Past Responses Are Retained

Laban et al. (2026) investigated the impact of transforming single-turn math word problems from GSM8K into multi-turn, underspecified dialogues where problem details are disclosed incrementally through a sharding process. Their study revealed a ubiquitous and substantial performance drop across 15 state-of-the-art LLMs when transitioning from the single to multi-turn setting. The researchers primarily attributed this degradation to the models' tendency to make premature assumptions regarding missing information and attempt final solutions before the user had revealed all necessary shards of the instruction.

Building on these insights, we explore whether multi-turn performance degradation is recoverable by simply removing past assistant responses from the context. As illustrated in Figure 5, we find that while the FC responses align with the performance reported in Laban et al. (2026, Table 1), performance partially recovers when past model responses are omitted.

When we inspect instances where models recovered under AO context, we find that models often become distracted or confused by their earlier responses. For instance, a detail invented in an earlier turn may cascade across several conversation turns, ultimately leading to an incorrect answer (see Section A.22.9); a flawed reasoning trace that is carried across several rounds can lead the model to incorrectly conclude that a "problem is impossible" (see Section A.22.11); an earlier response can lead the model to misinterpret the intended task and recompute the problem from scratch, rather than accumulating the solution across turns (see Section A.22.10).

4.2 Storing Past Assistant Responses Creates Context Pollution

In this section, we identify and characterize the ways in which LLMs over-condition on their past responses. To measure the prevalence of context pollution, we use Claude Haiku 4.5 to classify 150 conversations on WildChat into one of five categories (see Table 1) or as no pollution (see Section A.18 for the full LLM annotation prompt).[10] We find that context pollution occurs in roughly 14% of the conversation turns in our dataset.[11]

[7] https://www.reddit.com/r/ChatGPT/comments/1qngpqa/does_chatgpt_quietly_get_worse_in_long/

[8] https://www.reddit.com/r/ChatGPT/comments/1ppbbho/llm_gets_dumber_response_quality_degrades_over/

[9] https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents

[10] The authors manually reviewed a subset of the flagged instances to verify that they were genuine cases of context pollution. Select examples are presented in Section A.18.

[11] This experiment was conducted using the frontier model, GPT-5.2.

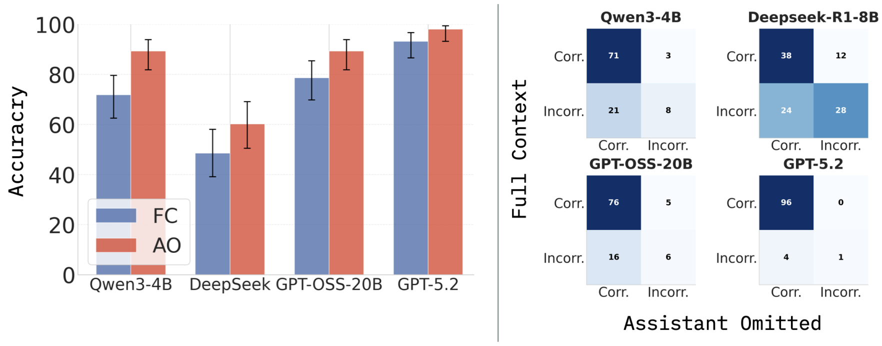

Figure 5: **Left:** Accuracy (%) on multi-turn GSM8K under Full Context (FC) and Assistant Omitted (AO) context. **Right:** Confusion matrices showing the number of problems answered correctly across models and context configurations.

The most common form of context pollution we identified is *intent override* (Table 1). In this setting, the model persists with the content or style of an earlier response, despite a user's attempt to correct or provide new details and specifications. For instance, when a user asks Qwen3-4B to "reflect" on a piece of code, the model instead generates another *version* of the code, rather than providing a *reflection* on the code (see Section A.22.4). In another instance, when a user introduces a new location to a storyline, the model incorrectly confuses it with a previous location that was present in context.

The remaining examples we present include: parameters from earlier code (e.g., UMAP-specific arguments) incorrectly leaking into later turns (t-SNE; Section A.22.1); a hallucinated fact about a novel being carried across turns (Section A.22.2); a specific parameterization of model temperature (Section A.22.5) being misinterpreted in a later turn; and a pathological reasoning pattern being amplified across turns (Section A.22.7).

Types of Context Pollution

Intent Override: The assistant follows a narrative, assumption, or task structure from an earlier response instead of adapting to the current user instruction, leading it to miss or ignore details in the latest instruction.
Code Carryover: Code from an earlier response is reused, even though it is incorrect, or syntax from a previously used programming language appears after the user has requested a different language.
Factual Hallucination Carryover: A false fact introduced in an earlier response is repeated or treated as true in a later response.
Reasoning Trace Carryover: Verbose reasoning patterns or stylistic artifacts from an earlier response are repeated or amplified in a later response.
Formula Carryover: A mathematical formula introduced in an earlier response is reused incorrectly in a later response.

Disentangling Context Pollution From Generic Long-Context Degradation. To test whether context pollution stems from past assistant responses specifically or from *any* additional text in context, we replace the assistant-side context with length-matched neutral filler.

Starting from the GPT-5.2 pollution annotations in Table 1, we analyze turns that were flagged for the three most prevalent categories: Intent Override, Code Carryover, and Factual Hallucination Carryover. For each flagged turn, we replace every prior assistant turn in history with length-matched filler text while leaving the user turns unchanged. We then regenerate GPT-5.2's response under the modified filler-text history and run the same pollution judge (see Section A.18) on the new response. For Intent Override and Factual Hallucination Carryover, the filler text we use is drawn from random Wikipedia

Category	Assistant-response History		Filler History	
	Count	Conf.	Count	Conf.
Intent Override	39	0.74	0	0.96
Incorrect Code Carryover	30	0.77	0	0.95
Factual Hallucination Carryover	27	0.75	0	0.95
Reasoning Trace Carryover	6	–	–	–
Formula Carryover	1	–	–	–
Overall prevalence (assistant-response history)				
Turn-level	104 / 729 turns (14.3%)			
Conversation-level	59 / 150 convos (39.3%)			

Table 1: Prevalence of context pollution instances across 729 responses generated by GPT-5.2 on WildChat conversations, under the original history and a history in which assistant-side responses are replaced with length-matched filler text. Confidence values indicate the LLM annotator's average confidence across instances in that category.

passages, `DragonLLM/Clean-Wikipedia-English-Articles` (Foly et al., 2025); for Code Carryover, the filler text we use is drawn from random Python code snippets, `bigcode/the-stack` (Kocetkov et al., 2023).

Across all tested pollution categories, we observe that every single originally-flagged instance of context pollution flipped from being present to absent under the new filler text. This reveals that context pollution is driven more by the semantic content of past assistant responses (it being a sensical prior assistant response), rather than by general long-context degradation. In fact, an interesting response we came across when spot-checking the length-matched neutral filler generations was one case where GPT-5.2 generated: "Also, something clearly went wrong in my earlier replies (they were unrelated garbage text). Sorry about that." The fact that GPT-5.2 claimed "ownership" over the prior assistant-side responses suggests that the model may treat past assistant responses as privileged and may plausibly choose to rely on them as a good starting point to inform downstream generation.

Keeping The Context Window Clean. To reduce context pollution, conversational agents could move toward *summarization-and-retrieval* frameworks. In Fig. 2 (top row), we find that storing one-sentence summaries often outperforms storing the full context. Rather than retaining the full conversation history, systems could maintain lightweight identifiers of past turns (e.g., short summaries or prefixes) and use these as retrieval keys, loading the relevant content only when a follow-up requires it (Anthropic, 2025c; Zhang et al., 2025; Wang et al., 2023; Packer et al., 2023). Such a summarize-and-retrieve framework may more closely align with the way humans engage in dialogue: each turn updates a conversational state (Lewis, 1979), rather than replaying the entire raw transcript before responding each time.

5 Adaptive Assistant Response Omission

In this section, we introduce a context filter that decides, on a per-turn basis, whether to retain (FC) or omit (AO) the assistant-side history, trading off response quality and token cost. We run this experiment using GPT-5.2, in the setting where the model's own responses are fed back into context.

Given a user prompt and the conversation history leading up to that prompt, we train a binary classifier to predict the LLM-judge's winner on the quality dimension. We label turns where the LLM-judge prefers the full-context response as one, and all other turns as zero.[12] The features we use include round number, cumulative context lengths, prompt category,

[12]We label ties as zero, reflecting cases where retaining assistant-side history confers no quality benefit despite the added compute cost.

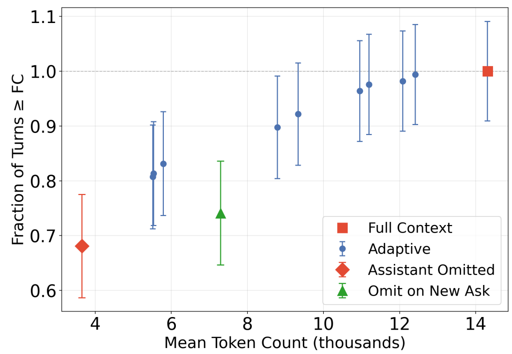

Figure 6: Ratio of adaptive wins to full-context-only wins vs. mean token count (in thousands) for operating points obtained by sweeping the decision rule's threshold τ and cost weight λ. The green triangle marks a heuristic that omits assistant responses on all turns categorized as "New Asks."

and dense embeddings of the user prompt and conversation history from a pretrained text embedding model,[13] reduced to twenty dimensions each using PCA.

We combine the classifier's predicted probability $\hat{P}(\text{FC} \succ \text{AO})$ with a penalty on token cost. Let l_i denote the number of tokens in the assistant-side history up to turn i, and let $\bar{l}$ denote the mean of l_i over all turns in the dataset. We define the *relative history length*, $\hat{\Delta}_i = l_i/\bar{l}$, and score each turn as $\hat{S}_i = \hat{P}(\text{FC} \succ \text{AO}) - \lambda \cdot \hat{\Delta}_i$, selecting full context if and only if $\hat{S}_i \geq \tau$.

Our adaptive approach (see Fig. 6) is able to retain over 99% of the performance of selecting full context for all turns (the red square) while using only an average of 87% of the total token cost. Applying the classifier trained on GPT-5.2 outputs to other LLM backbones in our study, we find that the context filter transfers nicely across different model backbones (see Section A.14). This generalization suggests that whether assistant history should be retained depends more on properties of the conversation itself than on model-specific behavior.

6 Discussion

In this work, we analyze real-world multi-turn conversations and uncover a surprising finding: *conditioning on full conversation histories is largely excessive.* Across all models in our study, we find that reduced-context configurations often perform comparably to conditioning on the full history, while substantially saving compute. We find that dependencies in real-world multi-turn conversations may be shorter in horizon than commonly assumed. In many cases, we also find that follow-up queries contain sufficient instruction to allow models to respond *from scratch*, without seeing their own prior responses. Beyond token savings, we find that storing shorter contexts also reduces distractions. In multi-turn math, removing the assistant-side history can partially *recover* performance, suggesting that premature model responses may interfere with downstream response quality.

We note that our evaluation relies on an LLM-as-judge framework. While the GPT-5 judge achieves $\geq$ 90% alignment with human annotations (Section A.5), future work should extend this analysis to a larger-scale human evaluation.

Our findings point to two distinct areas for future research. First, context management systems may benefit from making *content relevancy*-based filtering decisions, rather than relying on compaction after exceeding a certain length threshold. Second, *multi-turn* reinforcement learning may allow models to avoid some of the behavioral pathologies that we recover in Section 4. Overall, we hope that these findings inspire a shift away from conditioning on full dialogue transcripts toward curating cleaner, more focused contexts.

[13] We use OpenAI's `text-embedding-3-large` to align with the GPT-5 model family we use for response generation.

Acknowledgments

This work was supported by the MIT–IBM Computing Research Lab and the Amazon AI Research Innovation Fellowship. We are grateful to Mitchell Gordon and Koren Lazar for helpful feedback and discussions.

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

AI DevKit. Memory. https://ai-devkit.com/docs/6-memory/, 2024. Accessed: 2026-03-17.

Anthropic. Memory tool. https://platform.claude.com/docs/en/agents-and-tools/tool-use/memory-tool, 2025a. Accessed: 2026-03-17.

Anthropic. Context Editing and Tool Result Clearing. https://platform.claude.com/docs/en/build-with-claude/context-editing, 2025b. Anthropic Claude Developer Documentation. Accessed: 2026-02-05.

Anthropic. Effective context engineering for AI agents. https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents, 2025c. Accessed: 2026-03-20.

Nadezhda Chirkova, Thibault Formal, Vassilina Nikoulina, and Stéphane Clinchant. Provence: Efficient and robust context pruning for retrieval-augmented generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yu-Neng Chuang, Tianwei Xing, Chia-Yuan Chang, Zirui Liu, Xun Chen, and Xia Hu. Learning to compress prompt in natural language formats. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7756–7767, 2024.

Cursor. Dynamic context discovery. https://cursor.com/blog/dynamic-context-discovery, 2026. Cursor blog post. Accessed: 2026-02-07.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

DeepSeek-AI. DeepSeek-R1-Distill-Llama-8B. https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B, 2025.

Xuan Long Do, Bowei Zou, Liangming Pan, Nancy Chen, Shafiq Joty, and Aiti Aw. CoHS-CQG: Context and history selection for conversational question generation. In *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 580–591, 2022.

Shachar Don-Yehiya, Leshem Choshen, and Omri Abend. The ShareLM collection and plugin: Contributing human-model chats for the benefit of the community. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 167–177, 2025.

Sabri Eyuboglu, Ryan Saul Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Ruoyu Liu, Atri Rudra, James Zou, Azalia Mirhoseini, and Christopher Re. Cartridges: Lightweight and general-purpose long context representations via self-study. In *The Fourteenth International Conference on Learning Representations*, 2026.

S. Foly, J. Liu, J.-G. Barthelemy, G. Caillaut, R. Qader, M. Nakhle, and A. Sadoune. Clean-Wikipedia-English-Articles. https://huggingface.co/datasets/DragonLLM/Clean-Wikipedia-English-Articles, 2025. Dataset.

Pengfei Gao, Zhao Tian, Xiangxin Meng, Xinchen Wang, Ruida Hu, Yuanan Xiao, Yizhou Liu, Zhao Zhang, Junjie Chen, Cuiyun Gao, Yun Lin, Yingfei Xiong, Chao Peng, and Xia Liu. Trae agent: An LLM-based agent for software engineering with test-time scaling. *arXiv preprint arXiv:2507.23370*, 2025.

Kelly Hong, Anton Troynikov, and Jeff Huber. Context rot: How increasing input tokens impacts LLM performance. Technical report, Chroma, July 2025. URL https://research.trychroma.com/context-rot.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13381, 2023.

Haziq Mohammad Khalid, Athikash Jeyaganthan, Timothy Do, Yicheng Fu, Vasu Sharma, Sean O'Brien, and Kevin Zhu. ERGO: Entropy-guided resetting for generation optimization in multi-turn language models. In *Proceedings of the 2nd Workshop on Uncertainty-Aware NLP*, pp. 273–286, 2025.

Omar Khattab, Christopher Potts, and Matei Zaharia. Baleen: Robust multi-hop reasoning at scale via condensed retrieval. *Advances in Neural Information Processing Systems*, 34: 27670–27682, 2021.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Yacine Jernite, Margaret Mitchell, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro Von Werra, and Harm de Vries. The Stack: 3 TB of permissively licensed source code. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=pxpbTdUEpD.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs get lost in multi-turn conversation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=VKGTGGcwl6.

Seongyun Lee, Yongrae Jo, Minju Seo, Moontae Lee, and Minjoon Seo. Lost in the noise: How reasoning models fail with contextual distractors. *arXiv preprint arXiv:2601.07226*, 2026.

David Lewis. Scorekeeping in a language game. *Journal of Philosophical Logic*, 8(1):339–359, 1979. doi: 10.1007/BF00258436. S2CID: 206773185.

Mufei Li, Dongqi Fu, Limei Wang, Si Zhang, Hanqing Zeng, Kaan Sancak, Ruizhong Qiu, Haoyu Peter Wang, Xiaoxin He, Xavier Bresson, Yinglong Xia, Chonglin Sun, and Pan Li. Haystack engineering: Context engineering meets the long-context challenge in large language models. In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*, 2025.

Yucheng Li. Unlocking context constraints of LLMs: Enhancing context efficiency of LLMs with self-information-based content filtering. *arXiv preprint arXiv:2304.12102*, 2023.

Tobias Lindenbauer, Igor Slinko, Ludwig Felder, Egor Bogomolov, and Yaroslav Zharov. The complexity trap: Simple observation masking is as efficient as LLM summarization for agent context management. In *Fourth Deep Learning for Code Workshop at NeurIPS 2025*, 2025.

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

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Xiaoyuan Liu, Tian Liang, Dongyang Ma, Deyu Zhou, Haitao Mi, Pinjia He, and Yan Wang. The pensieve paradigm: Stateful language models with learned memory management. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=GymjF88oGQ.

Ryan Lopopolo. Harness engineering: leveraging Codex in an agent-first world. OpenAI Blog, February 2026. URL https://openai.com/index/harness-engineering/. Accessed: 2026-08-07.

OpenAI. Introducing GPT-5.2. https://openai.com/index/introducing-gpt-5-2/, 2025a. Accessed: 2026-02-09.

OpenAI. Latest models guide. https://developers.openai.com/api/docs/guides/latest-model, 2025b. Developer documentation, accessed February 2026.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. MemGPT: towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560v2*, 2023.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, Vicky H Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981, 2024.

Chen Qu, Liu Yang, Minghui Qiu, Yongfeng Zhang, Cen Chen, W Bruce Croft, and Mohit Iyyer. Attentive history selection for conversational question answering. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1391–1400, 2019.

Qwen Team. Qwen 3.5. https://qwen.ai/blog?id=qwen3.5, 2025a. Accessed: 2026-02-26.

Qwen Team. Qwen3-4B. https://huggingface.co/Qwen/Qwen3-4B, 2025b. Hugging Face model card, best practices documentation, accessed February, 2026.

Akshit Sinha, Arvindh Arun, Shashwat Goel, Steffen Staab, and Jonas Geiping. The illusion of diminishing returns: Measuring long horizon execution in LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026.

Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md Rizwan Parvez, and Graham Neubig. Learning to filter context for retrieval-augmented generation. *arXiv preprint arXiv:2311.08377*, 2023.

Yuan-An Xiao, Pengfei Gao, Chao Peng, and Yingfei Xiong. Improving the efficiency of LLM agent systems through trajectory reduction. *arXiv preprint arXiv:2509.23586*, 2025.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: Improving retrieval-augmented LMs with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*, 2024.

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

Chenxiao Yang, Nathan Srebro, David McAllester, and Zhiyuan Li. PENCIL: Long thoughts with short memory. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025b.

Munazza Zaib, Wei Emma Zhang, Quan Z Sheng, Subhash Sagar, Adnan Mahmood, and Yang Zhang. Learning to select the relevant history turns in conversational question answering. In *International Conference on Web Information Systems Engineering*, pp. 334–348. Springer, 2023.

Alex L. Zhang, Tim Kraska, and Omar Khattab. Recursive language models. *arXiv preprint arXiv:2512.24601*, 2025. URL https://arxiv.org/abs/2512.24601.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wild-Chat: 1m ChatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bl8u7ZRlbM.

A Appendix

A.1 Data

We analyze two real-world multi-turn conversation datasets and one synthetic multi-turn conversation dataset.

WildChat. The `allenai/WildChat-4.8M` dataset (Zhao et al., 2024) consists of around 4.8 million real-world, multi-turn conversations sourced from ChatGPT interactions.

ShareLM. The `shachardon/ShareLM` dataset (Don-Yehiya et al., 2025) is a unified collection of human–model interaction data aggregated from multiple conversation sources; see the unified contributions section of: `https://huggingface.co/datasets/shachardon/ShareLM`.

Lost-in-Conversation. The `microsoft/lost_in_conversation` (Laban et al., 2026) dataset is a synthetic dataset designed to study the way LLMs respond to incrementally specified instructions. It consists of prompts from single-turn benchmarks (HumanEval, LiveCodeBench, Spider, Berkeley Function Calling Leaderboard, GSM8K, ToTTo, and Summary of a Haystack) into multi-turn dialogues via a sharding procedure. Each sharded instruction is composed of multiple shards that incrementally reveal the original query.

Data Filtering. For the real-world datasets, we load the first subset of the data: 5,000 conversations from WildChat and 50,000 from ShareLM. For WildChat, we filter for conversations originally conducted with GPT-4 models, the most recent models represented in the dataset. For ShareLM, we filter for conversations conducted with frontier models, GPTs, Gemini, and Claude, as identified by the conversation URL metadata (`chatgpt.com`, `gemini.google.com`, and `claude.ai`, respectively). For both datasets, we filter for conversations in English (when language metadata is available),[14] conversations with 5–10 rounds (where a round corresponds to one user–assistant exchange), and conversations spanning diverse topic categories, identified by the presence of any one of the following keywords:

- **Creative writing:** *essay*, *poem*, *novel*, *protagonist*, *antagonist*, *villain*, *hero*, *plot*, *monologue*, *fairy tale*, *science fiction*, *mystery*, *thriller*, *dystopian*, *historical fiction*, *metaphor*, *imagery*, *lyric*, *satire*
- **Science & medical:** *biology*, *organism*, *species*, *virus*, *bacteria*, *ecology*, *disease*, *diagnosis*, *treatment*, *symptom*, *pathology*, *clinical*, *patient*, *chemistry*, *physics*, *molecule*, *cell*, *protein*, *DNA*, *neurology*, *pharmacology*, *epidemiology*, *scientific evidence*, *hypothesis*, *experiment*, *laboratory*, *microscope*, *anatomy*, *physiology*, *medical*, *medicine*
- **Business & finance:** *business*, *finance*, *trading*, *stock*, *economics*, *marketing*, *investment*, *revenue*, *profit*, *pricing*, *market research*, *entrepreneur*, *supply chain*, *financial*, *accounting*, *budget*, *startup*, *venture capital*, *portfolio*, *interest rate*, *inflation*, *GDP*, *fiscal*, *monetary*, *business plan*, *business model*, *strategy*
- **Education:** *homework*, *assignment*, *exam*, *test preparation*, *study*, *tutor*, *explain to me*, *course*, *lesson*, *tutorial*, *academic*, *student*, *quiz*, *study guide*, *curriculum*, *teach me*, *learning*, *professor*, *lecture*, *semester*, *grade*, *classroom*, *education*, *school*, *university*
- **Math:** *equation*, *algebra*, *calculus*, *derivative*, *matrix*, *probability*, *statistics*, *geometry*, *trigonometry*, *polynomial*, *function*, *calculate*, *theorem*, *proof*, *lemma*, *corollary*, *formula*, *differential*, *linear algebra*, *graph theory*, *optimization*, *mathematics*, *mathematical*
- **Coding:** *code*, *python*, *javascript*, *java*, *c++*, *programming*, *class*, *method*, *algorithm*, *debug*, *error*, *compile*, *variable*, *array*, *loop*, *recursion*, *data structure*, *API*, *library*, *framework*, *script*, *implementation*, *runtime*, *bug*, *software*, *developer*, *coding*, *program*

Because filtering is keyword-based, any conversation containing one of the specified keywords is retained. Finally, we filter out conversations identified as toxic by the `detoxify` ML-based toxicity classifier (with a threshold set to 0.7).

[14]Upon manual inspection, the final dataset still contains a small subset of conversations in other languages.

Due to compute constraints (e.g., generating model responses for every round of every conversation across four models), we randomly sample 200 conversations from WildChat and 150 from ShareLM, yielding a total of 350 real-world conversations for our analysis.

For Lost-in-Conversation, we use the 103 math word problems from GSM8K.

A.2 Models

We evaluate three open-source, reasoning models spanning a range of sizes and backbones, **Qwen3-4B**, **DeepSeek-R1-Distill-Llama-8B**, **GPT-OSS-20B**, and one state-of-the-art model, **GPT-5.2**. Each model is served with its recommended sampling parameters:

- **DeepSeek-R1-Distill-Llama-8B**: temperature 0.6, top-p 0.95, max generation length 32,768 tokens, context window 32,768 tokens (DeepSeek-AI, 2025). This model is served via `vLLM`.
- **Qwen3-4B**: temperature 0.6, top-p 0.95, top-k 20, thinking mode enabled, max generation length 1,024 tokens, context window 32,768 tokens (Qwen Team, 2025b). This model is served via `vLLM`.
- **GPT-OSS-20B**: temperature 1.0, top-p 0.95; max generation length 2,048 tokens, context window 32,768 tokens (Agarwal et al., 2025). This model is served via HuggingFace Transformers.
- **GPT-5.2**: We use the default API parameters (OpenAI, 2025b), with reasoning effort set to `medium`. This model is served via the OpenAI Responses API.

A.3 Context Configurations

For each conversation, we generate responses under standard full-context prompting and three alternative context configurations:

Full Context (FC). The model sees all prior user and assistant turns.

Assistant-Omitted (AO). The model sees all prior user turns while assistant responses are replaced with a short placeholder in the conversation history. Additionally, the model receives a system prompt:

> *"In this conversation, previous assistant responses are shown as '[Response provided]' to save memory. These are placeholders indicating the turn was already answered and has passed. Focus only on answering the user's most recent message."*

After each turn, the assistant's full response is replaced with the placeholder `[Response provided]` to maintain the alternating user/assistant turn structure.

Last-Turn-Only (LT). The model receives only the immediately preceding user–assistant exchange plus the current user prompt. All earlier turns are discarded.

Summarized (SUM). The model sees all prior user turns, while assistant responses are replaced with one-sentence summaries. After each turn, the summary is generated by prompting the *same* model being evaluated with:

> *"Summarize the following assistant response in exactly one sentence: {response}*
> *One-sentence summary:"*

This summary is what gets appended to the conversation history in place of the original response.

Delete-Everything (DE). The model receives only the current user prompt with no prior conversation history. All previous user and assistant turns are discarded.

Figure 7 shows the `conversation_history` message list passed to each model's chat template for the four configurations. The example depicts a 3-turn conversation at the point of generating a response to turn 3. For the three open-weight models (DeepSeek-R1, Qwen3, and GPT-OSS), this message list is converted to a prompt string using HuggingFace Transformers' `tokenizer.apply_chat_template(conversation_history, tokenize=False, add_generation_prompt=True)`. For GPT-5, the message list is passed directly to OpenAI's Responses API.

Full Context (FC)

```
[
 {"role": "user",
  "content": "<user turn 1>"},
 {"role": "assistant",
  "content": "<asst response 1>"},
 {"role": "user",
  "content": "<user turn 2>"},
 {"role": "assistant",
  "content": "<asst response 2>"},
 {"role": "user",
  "content": "<user turn 3>"}
]
```

Assistant-Omitted (AO)

```
[
 {"role": "system",
  "content": "In this conversation,
   previous assistant responses are
   shown as '[Response provided]'
   to save memory. ..."},
 {"role": "user",
  "content": "<user turn 1>"},
 {"role": "assistant",
  "content": "[Response provided]"},
 {"role": "user",
  "content": "<user turn 2>"},
 {"role": "assistant",
  "content": "[Response provided]"},
 {"role": "user",
  "content": "<user turn 3>"}
]
```

Last-Turn-Only (LT)

```
[
 {"role": "user",
  "content": "<user turn 2>"},
 {"role": "assistant",
  "content": "<asst response 2>"},
 {"role": "user",
  "content": "<user turn 3>"}
]
```

Delete-Everything (DE)

```
[
 {"role": "user",
  "content": "<user turn 3>"}
]
```

Summarized (SUM)

```
[
 {"role": "user",
  "content": "<user turn 1>"},
 {"role": "assistant",
  "content": "<1-sentence summary
   of asst response 1>"},
 {"role": "user",
  "content": "<user turn 2>"},
 {"role": "assistant",
  "content": "<1-sentence summary
   of asst response 2>"},
 {"role": "user",
  "content": "<user turn 3>"}
]
```

Figure 7: Chat template message lists for each context configuration, shown for a 3-turn conversation generating a response to turn 3.

A.4 LLM-Judge System Prompts

To evaluate responses, we use GPT-5 as an LLM-judge. For every round in a conversation, the judge receives both the FC and alternative-context responses and picks a winner for each dimension (Quality and On-Topic). We randomize response ordering (which response is labeled "A" vs. "B") for each comparison to reduce position bias.

Below is the judge prompt used for the main text evaluation (Fig. 2), where the original dataset's assistant responses (i.e., the assistant responses from the source dataset, not from the model being evaluated) are used as context.

> **System Prompt:**
> You are an expert evaluator of AI assistant responses. You will evaluate two responses to round `{round_num}` of `{total_rounds}` in a multi-turn conversation.
>
> **CONVERSATION CONTEXT (User Prompts and Assistant Responses):**
> `{context_prompt}`
>
> **CURRENT ROUND'S USER PROMPT:**
> `{current_round_prompt}`
>
> **RESPONSE A:**
> `{first_resp}`
>
> **RESPONSE B:**
> `{second_resp}`
>
> **EVALUATION CRITERIA:**
> You must compare the two responses on TWO separate dimensions and pick a winner for each:
>
> 1. **Overall Quality**: How well does the response address the user's current prompt?
>   - Completeness in addressing the user's request
>   - Accuracy of information provided
>   - Clarity and coherence of the response
>   - Helpfulness and usefulness to the user
> 2. **On-Topic**: Does the response stay focused on the conversation?
>   - Stays on-topic and relevant to the conversation thread
>   - Avoids topic drift, tangents, or unrelated content
>   - Avoids repetitive phrases or unnecessary fillers
>
> **TASK:**
> For EACH dimension, directly compare the two responses and pick a winner. If the responses are roughly equal on a dimension, declare a tie. Provide your evaluation in the following JSON format:
>
> ```
> {
>   "quality_winner": "A" or "B" or "tie",
>   "quality_justification": "<explanation comparing both
>     responses on quality>",
>   "ontopic_winner": "A" or "B" or "tie",
>   "ontopic_justification": "<explanation comparing both
>     responses on on-topic relevance>",
>   "confidence": <float from 0.0 to 1.0 indicating
>     how confident you are in your judgments>
> }
> ```
>
> Be objective and thorough in your evaluation. Respond ONLY with valid JSON.

Below, we show a variant of the judge prompt where the assistant responses in context *come from the model currently generating the response*.

System Prompt:
You are an expert evaluator of AI assistant responses. You will evaluate two responses to round `{round_num}` of `{total_rounds}` in a multi-turn conversation.

CONVERSATION CONTEXT FOR RESPONSE A (All Past User and Assistant Turns):
`{context_for_a}`

RESPONSE A:
`{first_resp}`

CONVERSATION CONTEXT FOR RESPONSE B (All Past User and Assistant Turns):
`{context_for_b}`

RESPONSE B:
`{second_resp}`

EVALUATION CRITERIA:
You must compare the two responses on TWO separate dimensions and pick a winner for each:

1. **Overall Quality**: How well does the response address the user's current prompt?
   - Completeness in addressing the user's request
   - Accuracy of information provided
   - Clarity and coherence of the response
   - Helpfulness and usefulness to the user
2. **On-Topic**: Does the response stay focused on the conversation?
   - Stays on-topic and relevant to the conversation thread
   - Avoids topic drift, tangents, or unrelated content
   - Avoids repetitive phrases or unnecessary fillers

TASK:
For EACH dimension, directly compare the two responses and pick a winner. If the responses are roughly equal on a dimension, declare a tie. Provide your evaluation in the following JSON format:

```
{
  "quality_winner": "A" or "B" or "tie",
  "quality_justification": "<explanation comparing both
    responses on quality>",
  "ontopic_winner": "A" or "B" or "tie",
  "ontopic_justification": "<explanation comparing both
    responses on on-topic relevance>",
  "confidence": <float from 0.0 to 1.0 indicating
    how confident you are in your judgments>
}
```

Be objective and thorough in your evaluation. Respond ONLY with valid JSON.

Finally, we show a variant of the judge prompt where the LLM-judge is given only the prior user-side turns as context for evaluation.

System Prompt:
You are an expert evaluator of AI assistant responses. You will evaluate two responses to round `{round_num}` of `{total_rounds}` in a multi-turn conversation.

CONVERSATION CONTEXT (All User Prompts Up To This Round):
{context_prompt}

RESPONSE A:
{first_resp}

RESPONSE B:
{second_resp}

EVALUATION CRITERIA:
You must compare the two responses on TWO separate dimensions and pick a winner for each:

1. **Overall Quality**: How well does the response address the user's current prompt?
   - Completeness in addressing the user's request
   - Accuracy of information provided
   - Clarity and coherence of the response
   - Helpfulness and usefulness to the user
2. **On-Topic**: Does the response stay focused on the conversation?
   - Stays on-topic and relevant to the conversation thread
   - Avoids topic drift, tangents, or unrelated content
   - Avoids repetitive phrases or unnecessary fillers

TASK:
For EACH dimension, directly compare the two responses and pick a winner. If the responses are roughly equal on a dimension, declare a tie. Provide your evaluation in the following JSON format:

```
{
  "quality_winner": "A" or "B" or "tie",
  "quality_justification": "<explanation comparing both
    responses on quality>",
  "ontopic_winner": "A" or "B" or "tie",
  "ontopic_justification": "<explanation comparing both
    responses on on-topic relevance>",
  "confidence": <float from 0.0 to 1.0 indicating
    how confident you are in your judgments>
}
```

Be objective and thorough in your evaluation. Respond ONLY with valid JSON.

A.5 LM-judge Alignment and Score Justifications

To validate the reliability of the LM-judge, one author manually scored a random sample of 20 judgments per model (80 total across four models) on both the quality and on-topic rubrics. For each judgment, the author assessed whether FC wins, the reduced context configuration wins, or the two tie. The human and LM-judge agreed on the quality outcome in 74 out of 80 cases (92.5%) and on the on-topic outcome in 75 out of 80 cases (93.75%).

In addition to manually annotating response pairs, one of the authors conducted a manual review of the LM judge's one-sentence score justifications, which were provided beside each score.

Below, we highlight some of the common mistakes that the models tend to make in multi-turn conversations. Quotes are drawn directly from the GPT-5 judge's justifications and shown in italics.

DeepSeek-R1-Distill-Llama-8B. Why DeepSeek falls short in multi-turn conversations:

- Math/coding errors:
  - *"It contains arithmetic errors (claims 348TB instead of ~148TB)"*
  - *"[the response] contains sign/concept errors"*
  - *"The code is flawed in key ways..."*
  - *"The response stays focused on the user's issue but misdiagnoses the root cause and provides flawed code."*
- Factual inaccuracies:
  - *"The response contains major inaccuracies..."*
  - *"The response is inconsistent and error-prone..."*
  - *"The solution contains major physics errors."*
- Stuck on a past turn:
  - *"largely repeats earlier plans without addressing the user's stated priorities (redundancy, space efficiency, replacement cost)."*
- Fails to address the question:
  - *"The response does not answer the specific question about what '2 stripes' means or clarify ZFS terminology (stripe vs. vdev vs. mirror). It meanders into a general restatement of the overall design..."*
  - *"The response barely addresses the actual question about hot spare configuration."*
  - *"The response stays high-level and repeats a plan instead of executing concrete steps,..."*
- Fails to account for earlier requirements:
  - *"The revision fails to honor the user's earlier requirement to reserve the 2x18TB drives for their own backup pool and incorrectly claims the user wants them in the main pool."*
- Fails to comply with user specifications:
  - *"The response violates the required JSON-only..."*
  - *"The response attempts to modify the code but fails to meet key requirements"*

Qwen3-4B. Why Qwen falls short in multi-turn conversations:

- Math/coding errors:
  - *"[response] contains an incorrect intermediate result (0.4096) and then trails off without completing the corrected calculation."*
  - *"The response is rambling and indecisive, contains arithmetic and capacity-summing errors..."*
  - *"The response is muddled, contains arithmetic confusion"*
- Response rambles:
  - *"[response] is incoherent and highly repetitive,..."*
  - *"The response is rambling, speculative, and never reaches a conclusion..."*
  - *"[response] rambles through possibilities, mentions speculative or unverified sources..."*
  - *"[response is] overly long for a simple question."*
- Factual inaccuracies:
  - *"it appears to invent a citation (uncertain title/authors)..."*
  - *"[response] contains multiple inaccuracies about ZFS topology."*
  - *"The response incorrectly claims that 'Jupiter's Icy Moons Explorer' is not an official mission..."*
  - *"Factually incorrect: misidentifies the author..."*
  - *"[response] asserts a specific model version (GPT-3.5) without basis and likely inaccurately."*

  - *"The response contains multiple major inaccuracies..."*
  - *"[response] includes inaccuracies (e.g., implying an Excel/Sheets built-in RSI function..)"*
- Response is incomplete:
  - *"It rambles, repeats points, and ends mid-sentence..."*
  - *"it is incomplete and stops mid-derivation"*
  - *"solution is incomplete and stops before computing submerged depth"*
- Fails to address the final question:
  - *"It also gets bogged down in irrelevant disk/bay counting and does not provide a concise, definitive answer to the user's question."*
  - *"The answer is factually incorrect and does not address the user's question about '17776' by Jon Bois."*
  - *"The response is a long, repetitive internal monologue that does not provide concrete fixes, code, or actionable debugging steps."*
- Fails to address some part of a task:
  - *"The response fails to address the user's specific point..."*
  - *"The response violates the strict JSON-only requirement..."*

GPT-OSS-20B. Response quality looks much better than the smaller open models.

- Minor math/coding errors:
  - *"The response clearly explains the mortgage calculation, gives the correct monthly payment (˜$2,923.57), and provides useful breakdowns. However, it contains a small but concrete arithmetic error: total paid should be $877,071.00 (not $876,971.00)..."*
  - *"Calculations are accurate to reasonable rounding, though minor rounding differences could slightly change the final cents."*
  - *"contains multiple inaccuracies about ticker formats and instrument types..."*
- Provides a concise, complete response:
  - *"Accurate and complete answer with a concise, clear explanation matching the logic of the riddle."*
  - *"Accurate, concise, and well-reasoned."*
  - *"Highly comprehensive and actionable roadmap tailored to a working professional transitioning to AI engineering."*
  - *"The response is comprehensive, structured, and tailored to the user's goal and context..."*
  - *"[response is] coherent, concise, and helpful for the user's goal."*
  - *"The response directly provides references and explains why they match the user's requested content, which is helpful and clear."*
- Responses can still be rambling and incomplete:
  - *"It is incomplete, meandering, and never delivers a concrete topology..."*
  - *"It is a stream of brainstorming and speculation about potential sources without producing any actual, verifiable references or annotations..."*
  - *"The response is incomplete: it lists only two fully annotated sources and starts a third before cutting off."*
- Factually accurate at times:
  - *"The response correctly identifies that the import is wrong and provides a clear, concise fix with installation steps and a working GUI example."*
  - *"Accurate, concise, and well-reasoned. Correctly identifies the Louvre and Mona Lisa,..."*
- Factually inaccurate at other times:

  - *"contains several factual and calculation errors. It misinterprets the Lyft benefit (treats 5% as 1x and values $500 Lyft spend at only 500 points instead of ˜2,500 points)."*
  - *"key bibliographic details are inaccurate or likely fabricated."*
  - *"The response provides an elaborate profile with highly specific details that are almost certainly fabricated and unsourced."*

GPT-5.2. The model is able to provide high-quality responses, regardless of whether it see previous assistant responses.

- Clear, high-quality responses:
  - *"Clear, structured answer with useful caveats and concrete examples of larger/newer models."*
  - *"Clear, concise, and accurate. It addresses both possible interpretations: direct LEGO-style competitor (MEGA/Mega Bloks/Mega Construx) and broader toy rival (Playmobil)."*
- Code is effective:
  - *"Directly addresses coloring the values by using \textcolor with xcolor, provides correct syntax and a concise example."*
- Factually accurate responses:
  - *"Thorough, accurate continuation that expands on order-sensitive vs order-insensitive deduping, nested/unhashable contents via JSON, use of keep, and handling mixed/None values."*
- Full and direct at addressing of the user's request:
  - *"Provides a single, complete, and runnable code block that consolidates the snake game."*
  - *"The response directly addresses the persistent error, explains the likely cause (_NoValueType)..."*

A.6 Response Quality Can Be Maintained Under Much Less Context

For each configuration–model pair, we compare the win rate of each reduced-context setting (Summarized (SUM), Last Turn Only (LT), Assistant Omitted (AO), and Delete Everything (DE)) against Full Context (FC).

Below, we mark a cell $\geq$ if the reduced-context configuration performs significantly better than or is statistically indistinguishable from FC, and $\sim$ if the win-rate gap is small, meaning the bounds of the two confidence intervals are separated by no more than 10 percentage points, a margin we treat as negligible under the LLM-judge. Tables 2 and 3 report this for the original-response-in-context experiment (see corresponding bar plots in Fig. 2).

Under this criterion, SUM and LT are each comparable to FC for 4 out of 4 models on the On-Topic dimension and 3 out of 4 on Quality. AO is competitive for 2 out of 4 models on the On-Topic dimension and 1 out of 4 on Quality, and DE is never competitive with FC. Overall, these results indicate that while the summary of recent responses is often important, full responses and those further upstream are often dispensable.

Table 2: On-Topic: comparison of reduced-context configurations against Full Context (FC), per model. $\geq$ indicates that reduced context performs significantly better than or is statistically indistinguishable from FC (95% CIs overlap); $\sim$ indicates that the win-rate gap is within 10pp.

Config	Qwen3-4B	DeepSeek-R1	GPT-OSS-20B	GPT-5.2
Sum	$\geq$	$\geq$	$\geq$	$\geq$
LT	$\geq$	$\geq$	$\sim$	$\sim$
AO		$\sim$		$\sim$
DE				

Table 3: Quality: comparison of reduced-context configurations against Full Context (FC), per model. $\geq$ indicates that the reduced context performs significantly better than or is statistically indistinguishable from FC (95% CIs overlap); $\sim$ indicates that the win-rate gap is within 10pp.

Config	Qwen3-4B	DeepSeek-R1	GPT-OSS-20B	GPT-5.2
Sum	$\geq$	$\geq$		$\geq$
LT	$\geq$	$\geq$	$\sim$	
AO				$\sim$
DE				

A.7 Assistant-Side History Is Less Beneficial for New Asks Compared to Follow-Ups

As expected, prompts that introduce new, self-contained requests within an ongoing conversation fare better without assistant-side history than prompts that follow up on a prior round (see Fig. 8). More surprisingly, in real-world multi-turn conversations, such *new ask* prompts constitute a substantial fraction (36.4%) of user turns.

From manual inspection of a random sample of fifty chats, we find that conversations in our dataset can be loosely categorized into the following three types: (i) sequences of loosely-related standalone prompts, (ii) conversations consisting of a single, main prompt followed by related queries (e.g., analyzing a reading passage), and (iii) conversations centered on a single evolving intent (e.g., debugging or iterative refinement on a writing passage).

Moving from the conversation level to the prompt level, we find it helpful to categorize prompts according to their degree of dependence on prior assistant responses:

New Ask: non-initial user prompts that introduce a new, self-contained request within an ongoing conversation. Despite appearing mid-conversation, these prompts can be addressed using only the current user prompt, without dependence on prior conversations rounds (e.g., “Describe an unmade Christopher Nolan film,” “How do you call a function inside a function in Python?”).

Follow-up with Feedback: user prompts that provide concrete, actionable feedback on a prior assistant response (e.g., “Can the opening be a bit more inviting?” “Use Python instead of Java for the code example”).

Follow-up without Feedback: user prompts that reference a prior conversation round (may be a user turn or an assistant response) without any concrete feedback (e.g., “Reflect on your response,” “And does George like it too?”).

To classify prompts into categories at scale, we use GPT-5 as an automated annotator. We provide GPT-5 with the conversation history up to the current round along with the current user prompt, and ask the judge to classify the prompt into one of the three categories described above. See Section A.20 for the full classification prompt. In our dataset, new-ask prompts account for 36.4% of user turns, follow-up with feedback for 30.5%, and follow-up without feedback for 33.1%.

Using the prompt annotations, we examine how AO-context performance varies across prompt categories, focusing on the two models, Qwen3-4B and GPT-5.2, for which removing assistant-side history results in an overall decrease in response quality. In Fig. 8, we plot the pairwise win rate between FC and AO-context, averaged across the quality and on-topic dimensions. We find that assistant-side history is most beneficial for follow-up prompts, while performance remains comparable across the two configurations for new-ask prompts.

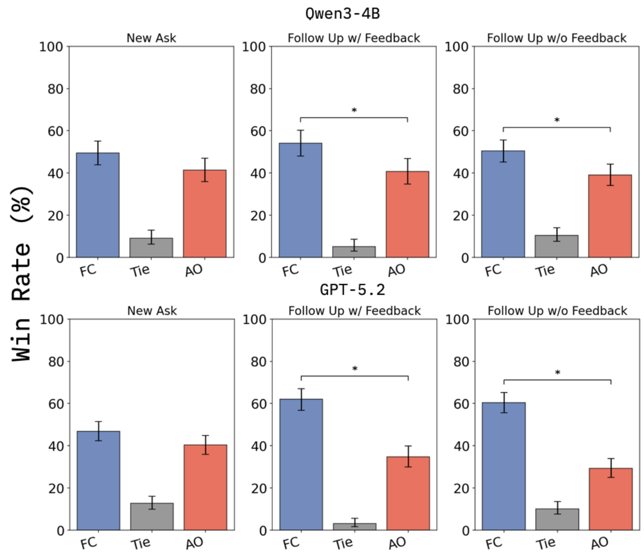

Figure 8: Pairwise win rates by prompt category (new ask, follow up with feedback, follow up without feedback) for Qwen3-4B (top) and GPT-5.2 (bottom), comparing Full-Context (FC) and Assistant-Omitted (AO) responses. Stars indicate statistically significant differences. Error bars indicate binomial proportion 95% confidence intervals.

A.8 Many Follow-Ups Remain Answerable Without Assistant History

We find that user-turn-only prompting often remains viable for follow-up prompts.

Upon manually inspecting 50 follow-up prompts that perform better under AO-context, we find that many provide sufficiently concrete instruction to be addressed *from scratch*. The current user prompt together with a prior user prompt, commonly the initial prompt or the immediately preceding one, often provides the needed information.

Motivated to categorize the follow-up prompts according to their degree of dependence on prior assistant responses, we identify the following three categories, and display examples of each in Fig. 3: (i) prompts that reference a past assistant turn but present concrete, actionable feedback, (ii) prompts that reference a past assistant turn without concrete feedback, and (iii) prompts that reference a past user turn.

In the first two cases (Fig. 3, left and middle), the follow-up prompt is either self-contained with concrete feedback or references only prior *user* turns. Thus, a model can plausibly respond from scratch without seeing the assistant-side history. In the third case (Fig. 3, right), prompts point to specific assistant outputs without providing enough detail, making prior assistant history necessary. Note that the third case requires storing only the *relevant* assistant turn.

The prevalence of follow-ups that provide concrete, self-contained feedback or rely solely on user-side context helps explain why AO-context still achieves win rates of roughly 40% for Qwen3-4B and 30% for GPT-5.2 across both follow-up categories. For more examples of conversations containing follow-up prompts, see Section A.16.

A.9 Summarized Response History Reduces Off-Target and Carried-Over Content

Manually examining the cases where summarized history was preferred over full context history by the LLM-judge, we find that a benefit of replacing prior assistant turn with summaries is that it leads to shorter, more focused downstream assistant responses. Keeping around verbose, half-baked thoughts from prior responses leads downstream responses to also meander, rehashing earlier content rather than addressing the current user prompt. Below are two such example of this behavior. Since the LLM-generated responses are long, we instead outline some key points from each response using Opus 4.7 and provide the judge's justification.

Example 1: RMSProp

[Turn 1]
USER:

> rmsprop (variance estimate in Adam) calculates a term, what is it

[Turn 2]
USER:

> Why is dividing by v helping? this means that updates to weights that were large updates before are going to be smaller now, right?

[Turn 3] (current turn)
USER:

> So if the network is far from the correct solution in a certain direction (e.g. training data slightly changed), this will prevent it from getting there fast? and would encourage training the weights that refer to small changes to be improved before others?

[Turn 3]
ASST: (FC history, 11,153 chars)

> The response delivers an eight-point list of general benefits of dividing by v: gradient normalization, variance and standard deviation, stabilization of optimization, adaptive learning rate, efficient optimization, comparison with other optimizers, preventing overshooting the solution, and handling different regions of the loss surface. This is essentially an answer to Turn 2 ("why does dividing by the variance help?"); only points 7 and 8 address Turn 3's actual question about the network being "far from the correct solution," and the response never directly confirms or denies the user's two hypotheses (that this slows convergence after a data shift, and that it prioritizes small-change weights first).

[Turn 3]
ASST: (Summarized history 3,754 chars)

> The response gives a four-point list built around the user's question on Turn 3 — stabilizing the learning rate, preventing overshooting the minimum, encouraging fine-tuning of smaller gradients first, and a closing note on training stability and efficiency. Each point maps onto what the user asked: points 2 and 4 address whether dividing by v keeps the network from overshooting when it is far from the solution, and point 3 directly answers whether small-change weights are refined before larger ones.

Judge's quality justification

> The summarized response directly addresses the user's specific questions and provides a clear, concise, and largely accurate explanation. The FC response is overly long, meandering, and contains confusion, making it less clear and helpful.

Takeaway. With the full history present, the response to Turn 2's question filled the context so that the Turn 3 answer drew on points from Turn 2's response. In contrast, the summarized history only kept the fact that Turn 2's question had already been answered in context, so that the response took on Turn 3's question more directly.

Example 2: Text Segmentation Function

[Turn 1]
USER:

> Implement: `def preprocess_text_segments(df: pd.DataFrame, max_words: int) -> pd.DataFrame:` — divide the text into segments of approx. `max_words` words each, don't split sentences, add a `segment_id` column formatted <testimony_id>_<segment_number>.

[Turn 3]
USER:

> The model is loaded above the loop *(a minor correction on the Turn 1 code)*

[Turn 4] (current prompt)
USER:

> implement this: `def segment_hebrew_spacy(text, nlp): doc = nlp(text); return [sent.text for sent in doc.sents] / def test_segment_hebrew_spacy(text, nlp):` — just for testing, instead of returning the sentences, split randomly eve...

[Turn 4]
ASST: (FC history, 21,174 chars)

> Instead of implementing the requested `test_segment_hebrew_spacy(text, nlp)` function — a different signature entirely, operating on a raw text string with no dataframe involved — the response reuses the Turn 1 `preprocess_text_segments` scaffolding almost wholesale: it keeps `testimony_id`, iterates over the dataframe, and emits segments as {`'testimony_id': ..., 'segment_id': ..., 'segment_text': ...`} records. It never produces a function matching the requested `test_segment_hebrew_spacy(text, nlp)` signature.

[Turn 4]
ASST: (Summarized history, 5,065 chars)

> The response works directly from the two function signatures given in the current prompt, reuses `segment_hebrew_spacy` to get sentences, then accumulates sentences into segments capped at 600 words, tracking a running word count and starting a new segment once the cap is reached — matching the requested `test_segment_hebrew_spacy` signature and behavior.

Judge's quality justification

> The summarized response implements the requested test function, segmenting roughly every 600 words ... and uses spaCy sentences to avoid splitting sentences. It provides runnable code ... The FC response, however, does not implement the requested function at all, instead returning an unrelated `preprocess_text_segments` implementation, making it far less responsive to the prompt.

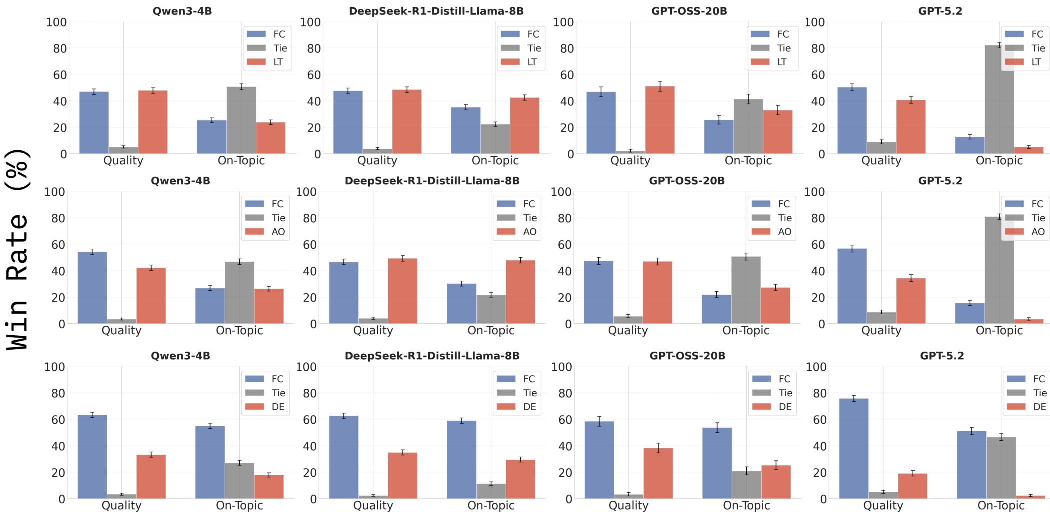

Figure 9: Pairwise win rates between Full Context (FC) and three reduced-context configurations across all four models (Qwen3-4B, DeepSeek-R1-Distill-Llama-8B, GPT-OSS-20B, and GPT-5.2), evaluated on the Quality and On-Topic dimensions. **Top row:** FC vs. Last Turn Only (LT), which retains only the most recent user–assistant round. **Middle row:** FC vs. Assistant Omitted (AO), which retains all user turns but replaces assistant responses with a placeholder. **Bottom row:** FC vs. Delete Everything (DE), which provides only the current user prompt. Error bars indicate binomial proportion 95% confidence intervals.

Takeaway. The full 36k-character Turn 1 response gave FC a large piece of code, and by Turn 4, an unrelated function request, FC reused that scaffold instead of reading the new function signature. The one-sentence summary kept only the *idea* of Turn 1 without the concrete code, leaving the model free to build a new implementation matching the actual requests of Turn 4.

A.10 Current Model In Context Evaluation

We run an experiment where the assistant responses shown in context belong to the same model that is generating the responses. Compare these results to that in Figure 2.

A.11 Sensitivity to Judge Context

Because our study investigates whether accumulated assistant responses can distract the responding model, one concern may be whether the judge itself may be subject to similar forms of distraction. To assess the sensitivity of our findings to the judge's context, we re-evaluate all responses under a which receives only the prior *user*-side turns.

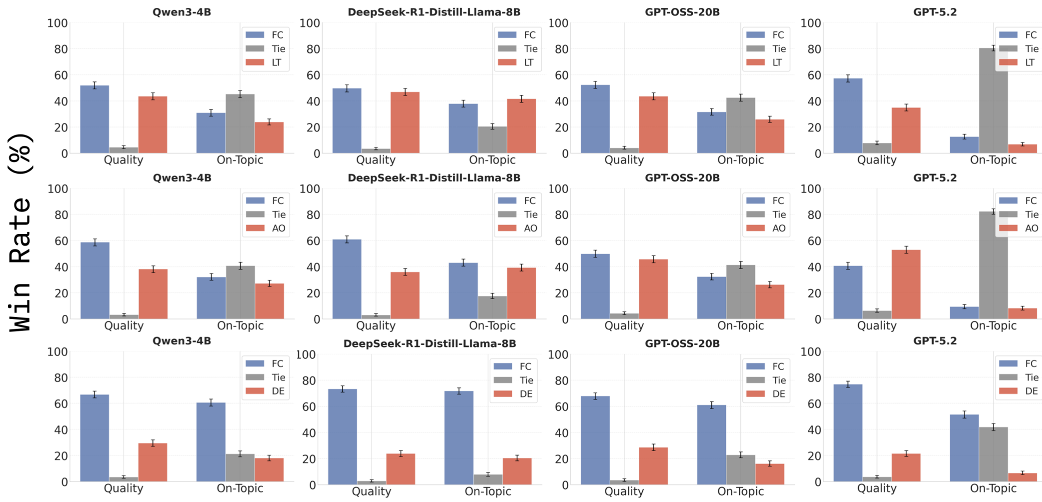

Figure 10: Pairwise win rates between Full Context (FC) and three reduced-context configurations across all four models (Qwen3-4B, DeepSeek-R1-Distill-Llama-8B, GPT-OSS-20B, and GPT-5.2), evaluated on the Quality and On-Topic dimensions. **Top row:** FC vs. Last Turn Only (LT), which retains only the most recent user–assistant round. **Middle row:** FC vs. Assistant Omitted (AO), which retains all user turns but replaces assistant responses with a placeholder. **Bottom row:** FC vs. Delete Everything (DE), which provides only the current user prompt. Error bars indicate binomial proportion 95% confidence intervals.

Fig. 10 presents pairwise win rates under the judge that receives only prior user-side turns, mirroring the main-text Figure 2. Notably, in the GPT-5.2 results, AO context is

A.12 Final-Answer-Only Evaluation

In our main experiments, the judge receives both the thinking trace and the final response when evaluating each model's output. To assess whether this choice affects our conclusions, we repeat the pairwise evaluation with the judge receiving only the final answers (i.e., with thinking traces removed).

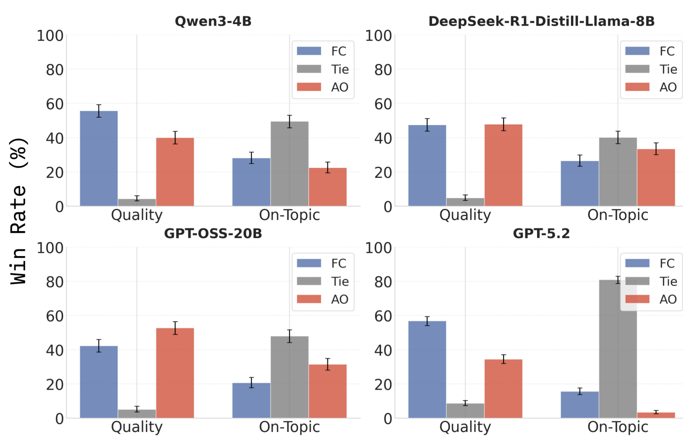

Figure 11: Pairwise win rates between Full-Context (FC) and Assistant-Omitted (AO) responses when the judge evaluates only final answers. Compare with Fig. 9, in which the judge receives both thinking traces and final answers.

The results (Fig. 11) are broadly consistent with judgements made on both reasoning traces and final responses (see Fig. 9). For Qwen3-4B and GPT-5.2, FC continues to outperform

Model	Fraction of Turns $\geq$ FC	Tokens Saved
GPT-5.2 *(self-generated context)*	0.916	37.7%
Qwen3-4B *(self-generated context)*	0.915	49.2%
DeepSeek-R1 *(historical context)*	0.875	49.4%
GPT-OSS-20B *(historical context)*	0.927	43.5%

Table 4: Fraction of turns meeting or exceeding the full-context (FC) baseline, and the corresponding token savings, across four models. Parentheses labels indicate whether the conversation history was self-generated or drawn from historical logs.

AO on response quality, while on-topic winrates remain similar. For GPT-OSS-20B, AO now shows a slight advantage on quality, whereas the two were roughly ties in Fig. 2. For DeepSeek-R1-Distill-Llama-8B, FC now shows a slight advantage over AO on quality, whereas the two were roughly tied in the main evaluation. Overall, the judge's access to reasoning traces does not substantially change the conclusions in the main text (see Fig. 2).

A.13 Multi-turn GSM8K

To complement the LM-judge evaluations with an objective, ground-truth metric, we evaluate math accuracy on GSM8K problems that have been transformed into multi-turn conversations.

Dataset. We use 103 sharded GSM8K problems from the `microsoft/lost_in_conversation` dataset (Laban et al., 2026). Each problem is decomposed into multiple shards that incrementally reveal the original GSM8K question. Each model generates responses turn-by-turn under both FC and AO context configurations, as described in Section A.3.

Answer extraction. We extract the model's final answer from the last-shard response, the turn where all information has been revealed. For models that produce chain-of-thought reasoning in <think> blocks (DeepSeek-R1-Distill-Llama-8B, Qwen3-4B), these blocks are stripped before extraction. We finally extract the candidate answer by finding the last bold-marked text (`**...**`) in the response; if no bold markers are present, we fall back to the last number in the response. The extracted text is cleaned by removing LaTeX escapes and currency symbols, and the last numeric value is taken as the candidate. The candidate answer is compared against the ground-truth from the GSM8K test split. We use the `math-verify` library to parse both answers into symbolic expressions and check equivalence. As a final verification step, we asked Claude-Opus-4.6 to scan the last-shard responses and confirm that the extracted candidate answer matches the intended final answer in the model output.

A.14 Evaluating adaptive assistant response omission across multiple LLM backbones

In this section, apply our adaptive policy trained on GPT-5.2 outputs (see Section 5) to predict whether to omit assistant responses for the other model backbones in our study.

We find that the policy transfers nicely across different model backbones (see Table 4). This new finding suggests that the decision of whether to retain assistant-side history is driven by properties of the conversation history, more than by model-specific behaviors.

A note on the experimental setting for DeepSeek-R1 and GPT-OSS: the original logistic regression policy (trained on GPT-5.2 outputs) was fit in the setting where the model's own responses were fed back into context. However, for DeepSeek-R1 and GPT-OSS, the assistant-omitted-only (AO) baseline already outperforms the full-context-only (FC) baseline under this condition (Fig. 9), meaning there is no quality headroom for an adaptive policy to recover – uniformly omitting assistant responses is already optimal. For these two models, we therefore choose to evaluate the adaptive policy on the historical-assistant-response setting (Fig. 2), where FC outperforms AO and an adaptive trade-off becomes meaningful.

A.15 Regression Analysis

Features. For each turn in every conversation, we extract the following features:

- **Numeric**: `round_num` (current round number), `total_rounds` (total number of rounds in the conversation), `context_length_user` (context length of user turns in characters), `context_length_asst` (context length of assistant turns in characters).
- **Categorical** (one-hot encoded): `prompt_type` (New Ask, Follow up w/o Feedback, Follow up w/ Feedback).
- **Current prompt**: the current round's user prompt, embedded with OpenAI's `text-embedding-3-large` (3,072 dimensions).
- **Conversation history**: the prior conversation turns (user and assistant), embedded with `text-embedding-3-large` (3,072 dimensions). To stay within the model's 8,191-token context window, we keep only the first and most recent (user, assistant) pairs when the history exceeds four turns. The opening turn often captures the initial problem statement, while the most recent turns capture the latest user feedback and model responses.

We reduce the prompt and history embeddings to 20 principal components (PCs) each. We concatenate the PCs with the base features. Evaluating 10, 20, 40, and 80 components, we find that 20 provided the strongest predictive performance.

Predicting LM-Judge Preferences. We train an L1-regularized logistic regression using scikit-learn's `LogisticRegression` with balanced class weights. We use `RandomizedSearchCV` to tune the regularization strength via 5-fold cross-validation, evaluating each fold with the F1 scores. For the predictive model, we find that the relationship between the features and the LLM judge's preference between FC and AO context is relatively weak, with a 5-fold cross-validated F1 score of only 0.6106 ± 0.0119.

We find that the prompt and conversation-history embeddings account for nearly all of the top-ranked features, with `category_new_ask` being the only non-embedding feature in the top 20. Table 5 reports the top 20 coefficients with 95% confidence intervals from the L1 logistic regression. None of the individual features reached statistical significance at the 5% level, suggesting that the relationship between these features and the LM-judge outcome is relatively weak. Note that the "Omit on New Ask" heuristic result shown in **??** is based on a 20% held-out test set; with a larger sample, this heuristic may perform somewhat better, as suggested by the trends in Fig. 8.

Table 5: Logistic regression coefficients with 95% confidence intervals. This table displays the 20 features with the largest absolute coefficient magnitudes.

Feature	Coef.	SE	95% CI
PromptPC3	−0.2003	0.1200	[−0.4355, 0.0350]
HistoryPC17	−0.1606	0.1056	[−0.3677, 0.0464]
HistoryPC2	0.1412	0.1320	[−0.1176, 0.4000]
category_new_ask	−0.1308	0.1098	[−0.3460, 0.0843]
PromptPC24	0.0963	0.0941	[−0.0881, 0.2807]
HistoryPC26	0.0920	0.0880	[−0.0805, 0.2644]
PromptPC10	0.0915	0.1200	[−0.1438, 0.3268]
PromptPC15	−0.0799	0.1035	[−0.2829, 0.1230]
HistoryPC23	0.0641	0.0887	[−0.1098, 0.2380]
HistoryPC7	−0.0635	0.1230	[−0.3046, 0.1776]
PromptPC36	−0.0626	0.0910	[−0.2410, 0.1159]
HistoryPC30	−0.0613	0.0844	[−0.2268, 0.1042]
HistoryPC28	−0.0588	0.0929	[−0.2409, 0.1233]
PromptPC4	−0.0534	0.1249	[−0.2981, 0.1914]
PromptPC35	0.0508	0.0877	[−0.1212, 0.2227]
PromptPC39	0.0489	0.0830	[−0.1139, 0.2116]
HistoryPC1	−0.0486	0.2075	[−0.4553, 0.3581]
HistoryPC31	0.0463	0.0879	[−0.1260, 0.2186]
HistoryPC15	0.0431	0.1032	[−0.1591, 0.2453]
HistoryPC6	0.0343	0.1239	[−0.2086, 0.2771]

A.16 Example Conversation Categories

As discussed in Section A.7, we identified three recurring conversation categories through qualitative analysis of 50 randomly sampled conversations. Below, we present representative examples of each category to illustrate the kinds of multi-turn interactions that appear in our datasets.

A.16.1 Category 1: Single Large Prompt Followed by Queries About that Prompt

In these conversations, the user provides a large initial prompt, such as a reading passage, a policy description, or a document, and subsequent turns ask questions about that prompt.

Example 1: Policy Program

[Turn 2]
USER:

> In 2009, President Obama's administration instituted a new program called "Cash for Clunkers," which was intended to boost aggregate demand in the short term by increasing the incentives for private consumption of vehicles after the economy had slumped. In addition to increasing consumption, this program was meant to allow citizens to use the money they earned selling an older "gas-guzzling" vehicle for purchasing a newer, fuel-efficient vehicle.
>
> Use the link below to read and analyze why the program was not as successful as people had hoped it would be. Can you think of any ways the program could have been changed to be more effective?

[Turn 3]
USER:

> https://www.nber.org/system/files/working_papers/w16351/w16351.pdf

[Turn 4]
USER:

> can you summarize the paper?

Example 2: Analyze a Reading Passage

[Turn 1]
USER:

> What's your take on the following text? [insert text]

[Turn 2]
USER:

> Do you think it is a good text?

[Turn 3]
USER:

> What influences or inspirations do you notice in the text? What's the most similar thing to it?

[Turn 4]
USER:

> What interpretations or meanings do you think this text has?

[Turn 5]
USER:

> What do you think about the numerous allusions to the idea of reincarnation in the text? Do you think there may be another interpretation based on this idea?

[Turn 6]
USER:

> How deeply is this text inspired by Brazilian culture? Do you think there may be references to hidden songs or works in his prose?

The user provides a literary text in Turn 1 and then asks a series of analytical questions about it across Turns 2–6. Each question probes a different aspect of the text but all depend on the passage provided in Turn 1.

A.16.2 Category 2: Loosely Related Standalone Queries

In these conversations, each turn is, for the most part, self-contained. The questions may share a broad topic but do not depend on one another or on prior assistant responses.

Example 3: Model Temperature

[Turn 1]
USER:

> Is it known why AI language models tend to generate repetitive text when using a low temperature? Rate from 1 to 10 your confidence that your answer is correct.

[Turn 2]
USER:

> Does temperature hyperparameter affect the probability calculations for the next token? Rate from 1 to 10 your confidence that your answer is correct.

[Turn 3]
USER:

> Does temperature affect the probability calculations for the next token that is output by the neural network? Rate from 1 to 10 your confidence that your answer is correct.

[Turn 4]
USER:

> Does the temperature setting affect the pre-softmax values? Rate from 1 to 10 your confidence that your answer is correct.

Example 4: Create a Film

[Turn 1]
USER:

> Describe an unmade Christopher Nolan film

[Turn 2]
USER:

> Outline an unmade Wes Anderson film

[Turn 3]
USER:

> Ken Rafinan

[Turn 4]
USER:

> Karla Manikad

A.16.3 Category 3: Cohesive Conversations with Iterative User Feedback

In these conversations, later turns provide explicit feedback, corrections, or refinements that build on prior assistant responses. The conversation follows a cohesive thread, and the user iteratively steers the model toward a goal.

Example 5: NAS Disk Topology Design

[Turn 1]
USER:

> Provide a design for a disk topology for a NAS built on TrueNAS Scale, as well as a dataset layout. The available disks are as follows:
>
> - 2x 18TB disks - 5x 14TB disks - 3x 12TB disk - 4x 8TB disks - 2x 120GB disks - 2x SLOW 8TB drives
>
> There are 17 drive bays available. The two smallest disks are to be used for a mirrored pool that servers as a boot device. The two slow drives are SMR disks that will be used in their own pool to provide a Time Machine target for some Macs. You are free to design a topology to optimize redundancy, space, and performance. The data being stored includes video files, music files, disk images, archived software, photos, and some text files. While much of the data could be recreated or downloaded, some of it is impossible to replace. You may leave bays available for a hot spare or to allow for future expansion. I prefer not to use RAIDZ, as mirrored arrays rebuild faster.
>
> If you need more information before creating your design, please provide me with a short questionnaire.

[Turn 2]
USER:

> 1. Let's not get this granular. 2. Far less than 0.5TB per year. 3. Some irreplaceable data may have more backup targets, but won't be backup up more frequently. Speed is generally not an issue as long as I can sustain 4k video streaming. 4. I intend to use primarily existing hardware but I could buy a small number of disks if needed. 5. The main goal is to prevent any data loss.

[Turn 4]
USER:

> Performance is the least important consideration for me. Redundancy, space efficiency, and finally replacement cost are my priorities. can you revise the plan? No need to mention the dataset layout, I got it.

Example 6: UDP Chatroom Development

[Turn 1]
USER:

> can you make a tax calculator only using methods or features specific to ruby?

[Turn 2]
USER:

> can you make a simple program that allows for a simple client-server udp python chatroom?

[Turn 3]
USER:

> can you add a feature to the program such that clients can privately message to each other using a json command "/msg"?

[Turn 4]
USER:

> does not seem to use json

[Turn 5]
USER:

> can you also add a feature so that multiple clients can create their own private group chat within the udp chat room?

[Turn 6]
USER:

> it doesn't seem to be working e.g. the clients are unable to join groups

[Turn 7]
USER:

> lastly, can you add a feature for clients to send and receive images and files, as well as adding emoji support for texts?

[Turn 8]
USER:

> can you verify your answer

[Turn 9]
USER:

> can you show the more optimal solution then?

Example 7: Webdriver Session Troubleshooting

[Turn 1]
USER:

I have a webdriver session object i saved in a file and then called but i get this error:

```
File "C:\Users\jason\Desktop\wikichat\omnigpt4\test1.py", line 48, in
<module>
message(driver)
File "C:\Users\jason\Desktop\wikichat\omnigpt4\test1.py", line 15, in
message
textbox = driver.find_element("xpath",
"//*[@id=""component-5""]/label/textarea")
^^^^^^^^^^^^^^^^^^^
AttributeError: 'str' object has no attribute 'find_element'
```

[Turn 2]
USER:

how can i transform the webdriver object from a string back to a webdriver object

[Turn 3]
USER:

but I have a webdriver instance open in a program and I want to control it from another python program

---

A.17 LLM Annotation Prompts

A.18 Context Pollution Annotator Prompt

To quantify the prevalence of context pollution, we prompt Claude Haiku 4.5 to examine each assistant turn (from round 2 onward) using conversations produced by full-context prompting. The annotator classifies each turn as pollution-present or pollution-absent, assigns a pollution category, identifies the source turns, and provides a justification.

System Prompt:
You are an expert annotator identifying context pollution in multi-turn AI conversations. Context pollution occurs when earlier assistant responses introduce errors, hallucinations, or stylistic artifacts that propagate into later turns.

FULL CONVERSATION HISTORY (up to and including the current round):
`{interleaved_context}`

CURRENT ROUND BEING EVALUATED: Round `{round_num}` of `{total_rounds}`

CATEGORIES OF CONTEXT POLLUTION:

1. `incorrect_code_carryover` — Code from an earlier turn is reused even though it is incorrect, or syntax from a previously used programming language appears after the user has requested a different language.
2. `formula_carryover` — A formula introduced in an earlier turn is reused because the model conditions on it, even though it is incorrect or applied in the wrong context.
3. `factual_hallucination_carryover` — A false or unsupported factual claim introduced earlier is repeated or treated as true in later responses.
4. `intent_override` — The assistant continues following the narrative, assumptions, or task structure from an earlier response instead of responding to the current user instruction.

5. `reasoning_trace_carryover` — Verbose reasoning patterns or stylistic artifacts from earlier turns are repeated or expanded in later responses.

TASK:
Examine the assistant's response in Round {`round_num`}. Determine whether it exhibits context pollution from any earlier assistant response. Context pollution means the assistant's response in this round is negatively influenced by content from its own earlier responses — for example, repeating an earlier error, carrying over incorrect code or formulas, hallucinating facts it introduced before, overriding the user's current intent with assumptions from prior turns, or exhibiting stylistic artifacts from earlier reasoning.

Important: Only flag pollution that is clearly harmful or incorrect. Normal continuity in a conversation (e.g., correctly referencing earlier work) is NOT pollution.

Return JSON:

```
{
  "pollution_present": "present" or "absent",
  "pollution_category": "<category_name>" or "none",
  "pollution_source_turns": [<list of integer turn
    numbers that introduced the pollution>] or [],
  "justification": "<explanation of why pollution
    is or is not present>",
  "confidence": <float from 0.0 to 1.0>
}
```

Respond ONLY with valid JSON.

A.19 Referenced Turn Annotator Prompt

To identify which prior turns each user prompt references, we prompt Claude Haiku 4.5 to classify each user turn (from round 2 onward) as a *follow-up* or *new ask*, and to identify the specific prior turns that the user references. Referenced turns are expressed as relative distances from the current turn, with the notation $N(\mathrm{u})$ or $N(\mathrm{a})$ indicating whether the user prompt or assistant response is referenced at distance N.

System Prompt:
Classify this user turn from a multi-turn conversation.

CONVERSATION CONTEXT (all prior turns):
{`conversation_context`}

CURRENT USER TURN TO CLASSIFY (Turn {current_turn_index}):
{`current_user_prompt`}

CLASSIFICATION RULES:
First, identify context-dependent elements in the current user turn:

- Pronouns with unclear referents (it, these, those, "the [noun introduced in previous turn]")
- Negations implying correction ("X is not right", "No", "Not X", "Don't", "Never")
- Short fragments that only make sense as responses (e.g., "Really?", "Not X?")
- Imperatives about previous behavior (e.g., "do not repeat...")

CRITICAL: If ANY context-dependent elements exist, the turn CANNOT be `new_ask`.

CATEGORIES:

1. **follow_up**: The user references, provides feedback about, or builds on any previous conversation turn. Examples: "Make it shorter", "X is not right", "Add more examples", "Reflect on your response", "The second one is wrong"
2. **new_ask**: A completely standalone prompt. Must be fully understandable without ANY prior context.

REFERENCED TURNS:
If the turn is a `follow_up`, identify which previous turns the current user turn references, responds to, or builds on. A "round" is a (user prompt, assistant response) pair. Count rounds back from the current turn: round 1 = the immediately preceding round, round 2 = the round before that, etc.

Within each round, specify whether the user prompt (u) or assistant response (a) or both are referenced. Use the notation $N(\mathrm{u})$ or $N(\mathrm{a})$ where N is the round distance.

For example, if the current user turn is in Round 4:

- `1(a)` = the user references the assistant response from Round 3
- `1(u)` = the user references their own prompt from Round 3
- `2(a)` = the user references the assistant response from Round 2

TASK:
Output ONLY this JSON (no other text):

```
{
  "category": "<follow_up or new_ask>",
  "referenced_turns":
    ["<round_distance>(<role>)", ...],
  "confidence": <1-10>
}
```

A.20 New Ask or Follow up: Annotator Prompt

To understand which prompts benefit from seeing assistant-side history, we find it helpful to categorize prompts according to their degree of dependence on prior assistant responses. To categorize prompts at scale, we use GPT-5 as an LM-judge to group prompts into one of the following three categories: *new_ask*, *follow-up with feedback*, or *follow-up without feedback*.

System Prompt:
Classify this user turn from a multi-turn conversation.

CONVERSATION CONTEXT (previous turns):
{`conversation_context`}

CURRENT USER TURN TO CLASSIFY:
{`current_round_prompt`}

CLASSIFICATION RULES:
First, identify context-dependent elements in the current turn:

- Pronouns with unclear referents (this, that, it, these, those, "the [noun]")
- Negations implying correction ("X is not right", "No", "Not X", "Don't", "Never")
- References to numbered/ordered items (the first, the second, etc.)
- Short fragments that only make sense as responses (e.g., "Really?", "Not X?")
- Imperatives about previous behavior (e.g., "do not repeat…")

CRITICAL: If ANY context-dependent elements exist, the turn CANNOT be `new_ask`.

CATEGORIES:

1. **new_ask**: A completely standalone prompt. Must be fully understandable without ANY prior conversation.
2. **feedback**: The user provides concrete, actionable feedback on a previous response. Examples: "Make it shorter", "X is not right, use Y instead", "Add more examples"
3. **no_feedback**: The user references a previous response without concrete feedback. Examples: "Reflect on your response", "The second one is wrong", "Not X?"

TASK:
Output ONLY this JSON (no other text):

```
{
  "context_dependent_elements": ["<list any found>"],
  "category": "<name>",
  "confidence": <1-10>
}
```

A.21 Numerical Score Judge Prompt

The judge receives the complete conversation history (past user prompts interleaved with assistant responses).

System Prompt:
You are an expert evaluator of AI assistant responses. You will evaluate a response to round {`round_num`} of {`total_rounds`} in a multi-turn conversation.

FULL CONVERSATION CONTEXT (All User Prompts And This Assistant's Previous Responses):
{`context_prompt`}

CURRENT ROUND'S USER PROMPT:
{`current_round_prompt`}

ASSISTANT'S RESPONSE:
{`response`}

EVALUATION CRITERIA:
You must evaluate this response on TWO separate dimensions:

1. **Overall Quality Score (1–10):** How well does the response address the user's current prompt?
   - Completeness in addressing the user's request
   - Accuracy of information provided
   - Clarity and coherence of the response
   - Helpfulness and usefulness to the user
2. **On-Topic Score (1–10):** Does the response stay focused on the conversation?
   - Stays on-topic and relevant to the conversation thread
   - Avoids topic drift, tangents, or unrelated content
   - Avoids repetitive phrases or unnecessary fillers

TASK:
Evaluate this response on BOTH dimensions (1–10 scale). Provide your evaluation in the following JSON format:

```
{
  "quality_score": <integer from 1 to 10>,
  "ontopic_score": <integer from 1 to 10>,
  "quality_justification": "<brief explanation>",
  "ontopic_justification": "<brief explanation>"
}
```

Scoring guide for BOTH dimensions: 9–10: Excellent; 7–8: Good; 5–6: Adequate; 3–4: Poor; 1–2: Very poor.
Be objective and thorough in your evaluation. Respond ONLY with valid JSON.

A.22 Examples of Context Pollution

In this section, we present surfaced examples of context pollution drawn from various models and spanning various domains (math, coding, machine learning, research, and creative writing). Each example shows ways in which previous assistant responses in full-context (FC) prompting may introduce errors, hallucinations, or stylistic artifacts that propagate into subsequent turns, while the assistant-omitted (AO) prompting avoids these issues.

Notably, we observe instances of context pollution even in GPT-5.2, indicating that this phenomenon is not confined to smaller models: frontier models are also susceptible to being misled by their own prior generations.

Instances of context pollution are highlighted with a red background, along with explanatory annotations that appear in red. In cases where the judge's justifications suggest potential context pollution, we provide the full conversation to Claude Opus 4.5 for further review to identify the specific locations in the response exhibiting context pollution. In all examples shown below, the annotations and commentary are initially provided by Claude Opus 4.5, and then manually reviewed and edited by a human.

A.22.1 Example 1: Code syntax leakage (UMAP to t-SNE)

Generator Model: GPT-5.2 **Domain:** Coding **Context Pollution:** Code parameter carry-over

In Turn 2, the user asks the model to write a UMAP dimensionality-reduction function. The model produces correct code with UMAP-specific parameters, including `metric="jaccard"` (appropriate for UMAP). Two turns later, the user says "Use TSNE instead." The FC response carries over two UMAP-specific elements into the t-SNE code, introducing bugs: (1) `metric="jaccard"`, which is incompatible with scikit-learn's t-SNE solver.

[Turn 2]
USER:
good, now create a new function to plot UMAP dimensionality reduction of the df

ASSISTANT (Full Context):

```
def plot_umap_labels(
df,
color=None,
n_neighbors=15,
min_dist=0.1,
metric="jaccard", # good default for binary label vectors
random_state=42,
...
):
```

[*Full UMAP implementation with correct parameters.*]

[Turn 4]
USER:
Use TSNE instead

ASSISTANT (Full Context):

```
def plot_tsne_labels(
df,
color=None,
perplexity=30,
learning_rate="auto",
n_iter=1000,
metric="jaccard",  # works well for binary vectors ← carried from UMAP
random_state=42,
...
):
tsne = TSNE(
n_components=2,
perplexity=perplexity,
learning_rate=learning_rate,
max_iter=n_iter,
metric=metric,
init="pca" if metric == "euclidean" else "random",
random_state=random_state
)
```

ASSISTANT (Assistant-Omitted Context):

```
def plot_tsne_by_binary_column(
df,
color_col="Evaluation",
perplexity=30,
learning_rate="auto",
n_iter=1000,
random_state=42,
...
):
tsne = TSNE(
n_components=2,
perplexity=perplexity,
learning_rate=learning_rate,
n_iter=n_iter,
init="pca",
random_state=random_state
)
```

Analysis. The FC response rewrites the UMAP function as t-SNE but carries over a UMAP-specific artifact. Specifically, `metric="jaccard"` is incompatible with scikit-learn's Barnes-Hut t-SNE solver. The AO response, without the UMAP code in context, writes clean t-SNE code with standard parameters (`init="pca"`, correct `n_iter` usage) and additionally includes logic to exclude the coloring column from the feature matrix.

Judge Quality Scores: FC = 3.0, AO = 8.0.

A.22.2 *Example 2: Inherited factual inaccuracy*

Generator Model: Qwen3 **Domain:** Writing **Context Pollution:** Inherited Factual Hallucination

In a multi-turn conversation about child narrators in post-*Huckleberry Finn* literature, the user asks the model to write a research paper (Turn 5) and then requests a polished abstract (Turn 6). In Turn 5, the FC model lists *The Bell Jar* (Plath) and *The Handmaid's Tale* (Atwood) as examples of novels featuring child narrators; neither novel is actually narrated by a child. When the user requests a new abstract in Turn 6, the model copies these incorrect examples into the methods section, carrying the error forward from its own previous generation.

[Turn 5]
USER:
write a research paper using polished language with literary jargon on novels featuring child narrators post huckleberry finn and their different uses in each novel.

ASSISTANT (Full Context):

> **Abstract**
> . . . Through a close reading of texts such as *The Catcher in the Rye* (1951) by J.D. Salinger, *Lord of the Flies* (1954) by William Golding, *The Bell Jar* (1963) by Sylvia Plath , *The Handmaid's Tale* (1985) by Margaret Atwood , and *The Nickel Boys* (2022) by Colson Whitehead, this paper argues that the child narrator in adult literature post-Huck Finn serves as both a mirror and a critique of the adult world. . .
> *The highlighted novels are not narrated by a child.*

[Turn 6]
USER:
Give a polished abstract using Research methodology and aims in academic language with literary jargon for the following [user provides essay text about child narrators, mentions neither *The Bell Jar* nor *The Handmaid's Tale*]. . .

ASSISTANT (Full Context):

> **Methodology**
> The study employs a close reading of canonical and contemporary texts, utilizing literary criticism, historical context, and narrative analysis to evaluate the function of child narrators. Key texts include *The Catcher in the Rye* (Salinger), *Lord of the Flies* (Golding), *The Bell Jar* (Plath) , *The Handmaid. . .*
> *Error carries over from Turn 5 response*

ASSISTANT (Assistant-Omitted Context):

> The AO response produces an abstract that discusses child narrators using appropriate authors (Dickens, Twain, and contemporary writers) without the incorrect examples.

Analysis. The error originates in Turn 5, where the model incorrectly categorizes *The Bell Jar* (narrated by an adult, Esther Greenwood, in college) and *The Handmaid's Tale* (narrated by an adult woman, Offred) as child-narrated novels. When the model sees its own Turn 5 output in context during Turn 6, it copies the same incorrect text list into the new abstract's Methodology section.

Judge Quality Scores: FC = 5.0, AO = 9.0.

A.22.3 Example 3: Misattributed citation

Generator Model: GPT-5.2 **Domain:** Policy **Context Pollution:** Hallucinated citation likely drawing from an earlier turn source

The user asks about a policy program, "Cash for Clunkers", and provides a link to an NBER paper that discusses the program. In Turn 1, the model cannot open the link and instead synthesizes an analysis drawing on multiple sources, likely including Busse, Knittel, and Zettelmeyer (NBER w18610). When the user later provides the actual paper link (Mian & Sufi, NBER w16351) and asks for a summary in Turn 4, the FC model misattributes the paper to "Busse, Knittel, and Zettelmeyer", authors from a *different* related NBER paper that the model likely drew upon in Turn 1.

[Turn 1]
USER:
In 2009, President Obama's administration instituted a new program called "Cash for Clunkers," which was intended to boost aggregate demand. . . Use the link below to read and analyze why the program was not as successful as people had hoped. . .

ASSISTANT (Full Context):

> I can't open the link you referenced from here, but the main criticisms economists raised about **Cash for Clunkers (CARS, 2009)** are fairly consistent across analyses. The program largely worked mechanically (it increased new-car sales during the program window), but it was often viewed as less successful than hoped. . .
>
> *Without the link, the model decides to synthesize from multiple sources*
>
> [*Detailed analysis follows, drawing on concepts from Busse/Knittel/Zettelmeyer (w18610), Hoekstra et al., and lifecycle emissions research.*]

[Turn 4]
USER:
can you summarize the paper?

[*User had provided the actual NBER link (w16351, Mian & Sufi) in Turn 3.*]

ASSISTANT (Full Context):

> **Paper:** *The Effects of "Cash for Clunkers" on Auto Sales and the Environment* (NBER Working Paper 16351; Busse, Knittel, and Zettelmeyer)
>
> *Incorrectly Attributed citation: w16351 is by Mian & Sufi, not Busse et al.*
>
> The authors evaluate the 2009 **Cash for Clunkers (CARS)** program's impact on: (1) new-vehicle sales, (2) vehicle prices/discounts, (3) fuel economy and emissions. . .

ASSISTANT (Assistant-Omitted Context):

> The NBER working paper (W16351) evaluates the 2009 **Car Allowance Rebate System (CARS)**—"Cash for Clunkers"—and asks whether it actually increased *total* auto sales or mainly shifted the timing of purchases. . .
>
> [*Correctly does not attribute specific authors, avoids hallucinated citation.*]

Analysis. Because the LLM was unable to open the link in Turn 1, the FC model synthesized its response from multiple related studies in its training data. The author list "Busse, Knittel, and Zettelmeyer" corresponds to a real but *different* NBER paper (w18610) on gasoline prices and automobile markets. When the user provides the actual paper link (w16351, by Mian & Sufi) in Turn 3 and asks for a summary in Turn 4, the FC model conflates the earlier synthesized sources with the new paper, producing a hallucinated citation. The AO response, without the contaminated Turn 1 output in context, does not misattribute the paper.

Judge Quality Scores: FC = 4.0, AO = 8.0.

A.22.4 *Example 4: Persistent style patterns override new instructions*

Generator Model: Qwen3 **Domain:** Coding **Context Pollution:** Prior response style overrides new user instruction

In Turn 1, the user asks how to call nested functions in Python, and the model produces a detailed tutorial with code examples. In Turn 2, the user issues a new request: "Reflect on your response." The FC model ignores the reflection instruction and instead produces *another full tutorial* on nested functions in the same style as Turn 1, rather than performing the requested meta-cognitive reflection. The model's own prior output anchors the model such that it does not shift to a different response type.

[Turn 1]
USER:
How can I call a function that is inside a function python

ASSISTANT (Full Context):

[*Produces a detailed tutorial covering: (1) calling inner functions within the outer function, (2) returning inner functions for external access, (3) scope and closures, (4) code examples with* `outer()/inner()` *patterns. Approximately 500 words with multiple code blocks.*]

[Turn 2]
USER:
Reflect on your response

ASSISTANT (Full Context):

To call a function that is defined inside another function in Python, you can either call it directly within the outer function or return it so it can be accessed from outside. Here's a step-by-step explanation with examples:

Produces another tutorial similar in style to the one in Turn 1 instead of reflecting

1. Call the Inner Function Inside the Outer Function

```
def outer():
def inner():
print("This is the inner function")
inner()
```

2. Access the Inner Function from Outside

[*Continues with the same tutorial format as Turn 1*]

ASSISTANT (Assistant-Omitted Context):

> When reflecting on the response, it's important to clarify that functions can indeed be defined and called within other functions in Python, a concept known as **nested functions**. Here's a breakdown of the key points:
>
> 1. **Definition**: A function can be defined inside another function... 2. **Calling**: The inner function can be called directly within the outer function... 3. **Scope**: The inner function can reference variables from the outer function, but it cannot modify variables in the outer function unless they are declared as `nonlocal` or `global`.
>
> [*Generates a reflection on what the prior response covered, explaining its strengths and limitations.*]

Analysis. The FC model's Turn 2 response completely ignores the user's instruction to "reflect" and instead regenerates the same tutorial content from Turn 1. The judge notes: "The response does not follow the user's instruction to reflect on the prior answer; instead, it provides a full tutorial and exposes internal chain-of-thought, which is inappropriate." The AO response, without the Turn 1 tutorial anchoring its generation, at least attempts to perform a reflection by summarizing key points and discussing scope considerations, though it too falls short of a fully meta-cognitive reflection.

Judge Quality Scores: FC = 3.0, AO = 7.0.

A.22.5 Example 5: Mathematical formula misapplied downstream

Generator Model: DeepSeek-R1 **Domain:** ML Concepts **Context Pollution:** A mathematical expression is carried over across turns, causing confusion.

In Turn 2, the model introduces a Boltzmann-style formula $p(y) = \exp(\theta \cdot s(y))/Z$ and uses θ as a "temperature" parameter. In this parameterization, however, θ actually functions as *inverse* temperature ($1/T$), so higher θ means *more* peaked distributions. When the user asks in Turn 3 to explain how temperature affects output diversity, the FC model reuses the formula but *reverses* the interpretation: it claims, "Lower Temperature ($\theta < 1$)" produces a "more spread-out probability distribution," which contradicts the standard understanding that lower temperature yields less diverse outputs.

[Turn 2] — ASSISTANT (Full Context):

> ...The probability of generating token y given context x can be modeled as:
>
> `p(y | x) = exp(θ · s(y, x)) / Z(θ, x)`
>
> where θ is the temperature parameter and $s(y, x)$ is the unnormalized log-score...

[Turn 3] — USER:

> How does the model's temperature setting impact the diversity and creativity of the generated outputs? On a scale of 1-10, how confident are you in this response?

[Turn 3] — ASSISTANT (Full Context):

> ...Using the formula from our earlier discussion, $p(y \mid x) = \exp(\theta \cdot s(y, x))/Z$:
>
> Lower Temperature ($\theta < 1$): The exponent $\theta \cdot s(y, x)$ is scaled down,

resulting in a more spread-out (uniform) probability distribution. ← *reversed: lower T should yield peaked, not spread-out, probability distributions.*

For example, if token scores are $[2.0, 1.0, 0.5]$ and $\theta = 0.5$:

$p(y_1) = \exp(0.5 \times 2.0)/Z = 0.45$

$p(y_2) = \exp(0.5 \times 1.0)/Z = 0.30$

$p(y_3) = \exp(0.5 \times 0.5)/Z = 0.22$

[Turn 3] — ASSISTANT (Assistant-Omitted Context):

> Temperature acts as a scaling factor applied to the model's output logits before the softmax function. . . **Low temperature** ($T < 1$): Amplifies differences between logits, making the distribution sharper and more peaked around the highest-scoring tokens. . . **High temperature** ($T > 1$): Flattens the distribution, giving lower-scoring tokens a higher relative probability. . .
>
> On a scale of 1–10, my confidence in this response is **10/10**.

Analysis. The Turn 2 formula uses θ in the *numerator* of the exponent ($\exp(\theta \cdot s)$), making it behave as inverse temperature: larger θ amplifies score differences, producing a more peaked distribution. When the FC model reuses this formula in Turn 3, it conflates θ with the standard temperature T (which appears in the *denominator*: $\exp(s/T)$) and reverses the directionality. The AO response, free from any earlier formula, explains temperature correctly.

Judge Quality Scores: FC = 3.0, AO = 6.0.

A.22.6 Example 6: Compounding failures in code

Generator Model: GPT-4 **Domain:** Coding **Context Pollution:** Cascading patch failures from accumulated broken code

In a 10-turn WildChat conversation, the user asks for a genetic algorithm. Turns 1–6 proceed successfully: the user iteratively requests modifications (output formatting, numpy removal, roulette wheel selection, elitism removal) and the model handles all correctly. In Turn 7, the user reports that the population count is inconsistent (5 or 7 instead of 10) and asks to add a `crossover_rate` parameter. The model's fix introduces a *new* population-size bug, triggering a cascade of patch-on-patch failures across Turns 8–10, each producing a different incorrect population count. The user ultimately gives up and asks the model to generate a summary prompt *so they can abandon the conversation and begin a new chat*.

[Turn 7]
USER:

> . . . And also. When I tried to run multiple time to see the random output. The individual are not fixed to 10. I not sure why. Can you check or explain to me. Output were differ from 5 individual to 7 individual. Instead of a full 10 individual which we start at individual 0. Please explain

ASSISTANT:

> [*Adds* `crossover_rate = 0.2` *to the config section and patches the evolution loop:*]
>
> ```
> for gen in range(num_generations):
> num_offspring = int(population_size * crossover_rate)
> ```
> ← *= 2 with rate 0.2*

```
parents = selection(population, population_size - num_offspring)
offspring = crossover(parents, num_offspring)
offspring = mutation(offspring, mutation_rate)
population = offspring ← population shrinks to 2
```

[Turn 8]
USER:
So I did what you told me. But now the final population is fixed to only 2 individual, which are individual 0 and individual 1. What happen?

ASSISTANT:

> [*Apologizes and swaps the parent/offspring formula:*]
>
> ```
> for gen in range(num_generations):
> num_parents = int(population_size * crossover_rate) ← = 2
> parents = selection(population, num_parents)
> offspring = crossover(parents, population_size - num_parents) ← 2 parents
> → 8 offspring
> offspring = mutation(offspring, mutation_rate)
> population = offspring ← population is now 8, not 10
> ```

[Turn 9]
USER:
Now it fixed to 8 individual. from individual 0 to 7. the config are the same

ASSISTANT:

> [*Attempts yet another patch, modifying the selection function and changing the formula again:*]
>
> ```
> for gen in range(num_generations):
> num_parents = int(population_size * (1 - crossover_rate)) ← = 8
> parents = selection(population, num_parents)
> offspring = crossover(parents, population_size - num_parents) ← 8 parents
> → 2 offspring
> offspring = mutation(offspring, mutation_rate)
> population = offspring ← back to 2 individuals
> ```

[Turn 10]
USER:

> Nope, now it revert back to 2 individual.
>
> Since you as a chatgpt don't really have a long term memory. Can you analysis our interaction or our conversation and generate maybe a prompt or a text or even passage including the code? So that when the next time I make a new conversation, I could just pasted the text/prompt/passage containing code and instruction. So that you can continue where we left off ...

Analysis. The root cause is that the `crossover()` function produces offspring *in pairs* (two children per crossover operation). When `crossover_rate = 0.2` and `population_size = 10`, the model needs to ensure that the number of offspring from crossover plus the number of survivors equals exactly 10. However, the model never examines the `crossover()` function itself or traces the arithmetic end-to-end. Instead, it repeatedly shuffles the same two variables (`num_parents` and `num_offspring`) between three equivalent-but-wrong formulas:

- Turn 7: `num_offspring = int(10 * 0.2) = 2` → population = 2
- Turn 8: `num_parents = int(10 * 0.2) = 2`, offspring from remaining 8 → population = 8
- Turn 9: `num_parents = int(10 * 0.8) = 8`, offspring from remaining 2 → population = 2

Each failed fix becomes part of the conversation context, and the model patches on top of its own previous patches. The accumulating broken code fragments prevent the model from stepping back to reason holistically about the system. A correct fix would involve ensuring that `crossover()` always produces exactly `population_size` individuals (e.g., by selecting `population_size / 2` parent pairs and producing 2 children each), but this requires examining the crossover function's implementation rather than adjusting the evolution loop formulas.

The model's own erroneous fixes accumulate in the conversation history, creating a self-reinforcing cycle where each patch is constrained by the frame established by previous patches. In the end, the user makes a decision to abandon the conversation and "start a new chat."

A.22.7 Example 7: Reasoning trace carryover

Generator Model: Qwen3-4B **Domain:** Coding **Context Pollution:** Reasoning trace from prior turn gets amplified in a subsequent turn

In Turn 1, the user asks about components for a recipe class. The model's reasoning trace produces a runaway list of "Also [include this]..." items when enumerating possible features. By Turn 3 ("What about user interaction?"), the FC model's reasoning trace has grown to 50+ "Also..." items, producing ∼800 words of largely redundant output, while the AO response—free from the prior pathological trace—produces a concise, structured response.

[Turn 1]
USER:
What are the components that a class to hold recipes should have? It should follow h-recipes, be able to share in SEO schemas and be a flexible tool for internet use. What else should it support?

ASSISTANT (Full Context) — Reasoning Trace:

`<think>`
...We should enumerate components: attributes like name, description, ingredients, steps, preparation time, cooking time, total time, servings, nutrition info, images, video, tags, source, author, rating, comments, categories...

Also ability to handle special fields: allergens, dietary categories (vegan, gluten-free).

Also support for image lazy loading, alt attributes.

Also support for caching, version control, serialization, import/export (CSV, JSON, XML).

Also support for unit conversions.

Also support for front-end frameworks.

Also mention "h-recipe" is microformat for recipes.

Also mention that support for SEO: JSON-LD markup for Google, rich snippets.

Also mention multi-step instructions, sub-steps. Should support sub-recipes.
Also consider integration with Alexa or Google Assistant voice cooking instructions.
Also support for generating share links: WhatsApp, Facebook, email.
Also support for "public" vs "private" recipes.
Also support for "alternateName".
30 "Also..." items total
</think>
The reasoning trace establishes a pathological "Also [add X]..." enumeration pattern, listing features without structure or prioritization.

[Turn 3]
USER:
What about user interaction (reviews, scores etc.)?

ASSISTANT (Full Context) — Reasoning Trace:

<think>
Need to elaborate about user interaction: reviews, scores, likes, ratings, comments. Should support methods for adding, editing, retrieving reviews...

Also ability to flag. Also include 'helpful' votes. Also handle moderation.
Also method to get aggregated rating. Also maybe track 'review_count'.
Also support for star rating system. Also support for user-specified rating: 1–5.
Also support for 'score' as numeric maybe 1–10.
Also support for 'tags' like 'vegan', 'gluten-free'.
Also support for 'user_interaction' flags: 'likes', 'dislikes', 'favorites'.
Also ability to export reviews to JSON.
Also support for comments thread (nested).
Also maybe support 'review_published_date'.
Also support for 'moderation status' like 'approved', 'pending', 'rejected'.
Also ability to fetch reviews by user, by rating threshold.
Also support for 'review_score' numeric and star mapping.
50+ "Also..." items total
</think>
Reasoning trace carries over the enumerative "Also..." pattern from Turn 1, producing 50+ items listed in this way.

[*Final answer is a bloated skeleton class with redundant attributes and methods, reflecting the unstructured enumeration from the reasoning trace.*]

ASSISTANT (Assistant-Omitted Context) — Reasoning Trace:

<think>
Okay, the user is asking about user interaction features like reviews and scores for a recipe class. They mentioned the class should follow h-recipes, be SEO-friendly, and flexible. So, I need to think about how to extend the class to handle user interactions.

First, I remember that the previous class had components like title, ingredients, instructions, etc. Now, adding reviews and scores would require new attributes. Maybe a list of reviews, each with a user, rating, comment, and date. Also, a score average would be useful for SEO and user experience.

I should include methods to add reviews, calculate the average score, and maybe retrieve them. But the user said not to fill in logic, just outline the structure. So, I'll add variables like self.reviews and self.average_score...

I need to make sure the class is extensible. Maybe include a method to get reviews, and a way to update the average score. But again, just the structure. So, variables for

reviews list and average score. Methods to add reviews and calculate the average. Also, perhaps a way to retrieve all reviews.
</think>

ASSISTANT (Assistant-Omitted Context) — Response:

To support user interaction (reviews, scores, etc.), the class should include:

Variables
`self.reviews`: A list of user reviews (e.g., `[{"user": "john_doe", "rating": 5, "comment": "Amazing recipe!", "date": "2023-10-05"}]`).
`self.average_score`: A float representing the average rating (e.g., `4.5`).
`self.total_reviews`: An integer tracking the number of reviews.

Methods
`add_review(user, rating, comment)`: Adds a new review to the list.
`calculate_average_score()`: Computes the average rating from reviews.
`get_reviews()`: Returns all reviews.
`update_average_score()`: Recalculates the average score after adding/removing reviews.

SEO/Schema Support
Include structured data for reviews (e.g., `Review` schema) in metadata. Allow embedding reviews in HTML for search engines to index.

Flexibility
Support multiple user identifiers (e.g., usernames, IDs). Allow optional filtering of reviews by date, rating, or user.

This structure ensures the class can handle user interactions while remaining compatible with SEO standards and scalable for internet use.

Analysis. The FC model's reasoning trace in Turn 1 establishes a pathological "Also [add X]" enumeration pattern. When this output is included in context for subsequent turns, the model's reasoning trace reproduces and amplifies the pattern: Turn 1 has ∼30 "Also…" items, Turn 3 has 50+. The resulting FC response is ∼800 words of largely redundant feature suggestions, while the AO response, free from the 'Also [add X]" enumeration pattern, produces a concise, structured response.

Judge Quality Scores: FC = 3.0, AO = 8.0.

A.22.8 Example 8: Hallucinated code carryover

Generator Model: GPT-5.2 **Domain:** Coding **Context Pollution:** Hallucinated import paths persist across turns

In a multi-turn conversation about building an Android chart using the Vico library, the model introduces hallucinated import paths in Turn 3, importing from fabricated `compose.m3.*` package paths (e.g., `compose.m3.chart.bar.barChart`, `compose.m3.axis.rememberBottomAxis`). In Turn 4, the user explicitly corrects the API, providing the exact Vico 1.6.5 syntax with correct function names and dependencies. The FC model acknowledges the correction ("You're right") and updates the function names (e.g., `barChart()` → `columnChart()`) but *retains the hallucinated* `compose.m3.*` *import paths* from its own Turn 3 response. The AO model, without the prior hallucinated code in context, produces the correct import paths.

[Turn 3]
ASSISTANT (Full Context):

[*Produces Vico chart code with hallucinated import paths:*]

```
import com.patrykandpatrick.vico.compose.m3.axis.rememberBottomAxis
import com.patrykandpatrick.vico.compose.m3.axis.rememberStartAxis
import com.patrykandpatrick.vico.compose.m3.chart.Chart
import com.patrykandpatrick.vico.compose.m3.chart.bar.barChart
import com.patrykandpatrick.vico.core.entry.ChartEntryModelProducer
```

The `compose.m3.*` paths do not exist in the Vico library; the correct paths use `compose.chart.*` and `compose.axis.*`.

[Turn 4]
USER:

That's actually not right. Here is the proper way to use Vico 1.6.5:

```
Chart(
chart = columnChart(),
chartModelProducer = chartEntryModelProducer1,
startAxis = startAxis(),
bottomAxis = bottomAxis(),
)
```

ASSISTANT (Full Context):

You're right—with Vico 1.6.5…

```
import com.patrykandpatrick.vico.compose.m3.axis.bottomAxis
import com.patrykandpatrick.vico.compose.m3.axis.startAxis
import com.patrykandpatrick.vico.compose.m3.chart.Chart
import com.patrykandpatrick.vico.compose.m3.chart.column.columnChart
```

Function names updated (`barChart` → `columnChart`, `rememberBottomAxis` → `bottomAxis`), but the hallucinated `compose.m3.*` paths from Turn 3 persist.

ASSISTANT (Assistant-Omitted Context):

```
import com.patrykandpatrick.vico.compose.chart.Chart
import com.patrykandpatrick.vico.compose.chart.column.columnChart
import com.patrykandpatrick.vico.compose.axis.horizontal.bottomAxis
import com.patrykandpatrick.vico.compose.axis.vertical.startAxis
import com.patrykandpatrick.vico.core.entry.ChartEntryModelProducer
import com.patrykandpatrick.vico.core.entry.entriesOf
```

[*Correct import paths:* `compose.chart.*`, `compose.axis.horizontal.*`, `compose.axis.vertical.*`.]

Analysis. In Turn 4, the hallucinated *import paths* `compose.m3.*` are carried directly from its own Turn 3 response. The hallucinated paths persist into Turn 5 as well. The AO model, without the Turn 3 hallucination in context, produces the correct Vico import structure.

A.22.9 Example 9: Invented assumption carryover

Generator Model: Qwen3-4B **Domain:** Math (GSM8K) **Context Pollution:** Hallucinated assumption carried across turns leads to an incorrect answer.

The problem asks how much money John makes selling planks cut from a tree. When the user gives only the height of the tree in Turn 2, the model invents a 3D cylinder model of the tree with a made-up 10-inch diameter. Each subsequent turn references Turn 1's hallucinated diameter length.

[Turn 2]
USER:
John is starting with an 80-foot tree.

ASSISTANT (Full Context):

Assuming the tree is a **cylinder** with a **diameter of 10 inches** ($\approx$ 0.833 feet) and height of 80 feet:

$$\text{Volume} = \pi \times \left(\frac{d}{2}\right)^2 \times h \approx \text{44.2 cubic feet}$$

Both the cylinder and the 10-inch diameter are assumptions that the model creates.

[Turn 3]
USER:
he can use 80% of the tree to make logs.

ASSISTANT (Full Context):

Usable Volume $= 44.2 \times 0.8 = 35.4$ cubic feet.
$\leftarrow$ References the made-up 44.2 from Turn 2.

[Turn 4]
USER:
he cuts the usable portion into 4-foot logs.

ASSISTANT (Full Context):

Volume per 4-foot log: $\pi \times (0.4165)^2 \times 4 \approx 2.17$ cubic feet.

Number of logs: $35.4/2.17 \approx 16.3$ logs.
$\leftarrow$ Uses the 35.4 from Turn 3.

[Turn 5]
USER:
he turns each 4-foot log into 5 planks.

ASSISTANT (Full Context):

Total planks: $16.3 \times 5 = 81.5$ planks.
$\leftarrow$ Uses the 16.3 from Turn 4.

[Turn 6]
USER:
each plank sells for $1.20.

ASSISTANT (Full Context):

Total earnings:

$$81.5 \times 1.20 = \$97.80$$

John makes $97.80 from selling the planks.

Response is incorrect due to following an assumption made all the way back in Turn 2: $44.2 \rightarrow 35.4 \rightarrow 16.3 \rightarrow 81.5 \rightarrow \$97.80$ *instead of $96.*

ASSISTANT (Assistant-Omitted Context):

Breakdown:
1. Usable tree length: 80% of 80 feet = **64** feet.
2. Number of 4-foot logs: $64 \div 4 =$ **16** logs.
3. Total planks: $16 \times 5 =$ **80** planks.
4. Total revenue: $80 \times \$1.20 =$ **$96**.

Analysis. The model hallucinates assumptions in Turn 2 that ultimately lead to an incorrect answer in Turn 6. Once committed to the hallucinated assumptions, every subsequent turn references the previous turn's answer. **Gold Answer:** $96. **FC Answer:** $97.80 (incorrect). **AO Answer:** $96 (correct).

A.22.10 Example 10: Intermediate answer misapplied

Generator Model: GPT-5.2 **Domain:** Math (GSM8K) **Context Pollution:** Intermediate answer overwrites a cumulative calculation.

This problem requires keeping track of the total amount of tea consumed by a group of students. Turn 3 introduces 4 students drinking 1.5 quarts each, and Turn 4 introduces 16 students drinking 2 quarts each, with 10 quarts remaining. The correct total consumption is $4 \times 1.5 + 16 \times 2 = 6 + 32 = 38$ quarts, giving a starting amount of $38 + 10 = 48$ quarts $= 12$ gallons. The model computes a correct intermediate answer of 4 gallons in Turn 3 (using only the first group), but in Turn 4 it treats the new information as a *replacement* rather than an *addition*, computing $16 \times 2 + 10 = 42$ quarts $= 10.5$ gallons, discarding the 6 quarts consumed by the first group.

[Turn 3]
USER:
four students consumed 1.5 quarts of tea each

ASSISTANT (Full Context):

Four students drank 1.5 quarts each, so the amount consumed was:

$$4 \times 1.5 = 6 \text{ quarts}$$

Since 10 quarts were remaining after the party, the amount at the start was:

$$10 + 6 = 16 \text{ quarts}$$

Convert quarts to gallons (4 quarts = 1 gallon):

$$16 \div 4 = 4 \text{ gallons}$$

There were 4 gallons of tea at the start of the party.

Correct given the information so far, but this intermediate answer confuses the model in Turn 4.

[Turn 4]
USER:
16 students each drank 2 quarts of tea

ASSISTANT (Full Context):

If **16 students each drank 2 quarts**, then the amount consumed was:

$$16 \times 2 = 32 \text{ quarts}$$

← Only counts the new group; drops the 6 quarts from the 4 students in Turn 3

With **10 quarts remaining** after the party, the starting amount was:

$$32 + 10 = 42 \text{ quarts}$$

Convert to gallons (4 quarts = 1 gallon):

$$42 \div 4 = 10.5 \text{ gallons}$$

There were 10.5 gallons of tea at the start of the party.

Incorrect. The model interprets Turn 4 as replacing rather than adding to Turn 3.

ASSISTANT (Assistant-Omitted Context):

Total tea consumed $= 4 \times 1.5 + 16 \times 2 = 6 + 32 = 38$ quarts.
Starting amount = consumed + remaining $= 38 + 10 = 48$ quarts.
Convert to gallons: $48 \div 4 = 12$ gallons.

There were 12 gallons of tea at the start of the party.

Analysis. The FC model's Turn 3 response provides an intermediate answer of 4 gallons based on partial information. When Turn 4 introduces additional information, the model restarts the calculation using *only* the new group ($16 \times 2 = 32$ quarts) rather than accumulating both groups ($6 + 32 = 38$ quarts).

Gold Answer: 12 gallons. **FC Answer:** 10.5 gallons (incorrect). **AO Answer:** 12 gallons (correct).

A.22.11 Example 11: Reasoning error causes confusion downstream

Generator Model: Qwen3-4B **Domain:** Math (GSM8K) **Context Pollution:** Prior flawed reasoning poisons subsequent turns

The problem describes Jerry's two pools that both leak water at 4 gallons per minute. In Turn 4, the model incorrectly reasons that "if both pools lose water at the same rate, the ratio remains constant". However, these equivalent leak rates would preserve the *difference*, not the *ratio*. In Turn 5, the FC model inherits this belief and confuses itself "if they both lose water at the same rate, the ratio between them should stay the same. . . the ratio changed from 2:1 to 4:1. . . that doesn't make sense." This then leads to a new hallucination, where the model sets up the "4 minutes ago" condition as $B - 16 = 2(S - 16)$ (subtracting leaked water from current values, as if pools had *less* water in the past) instead of the correct $B + 16 = 2(S + 16)$, obtaining $S = -8$ and declaring the problem "impossible."

[Turn 4]
USER:
4 minutes ago, the big pool had twice as much water as the small pool

ASSISTANT (Full Context):

Let me denote the current amount of water in the small pool as S and the big pool as B.

If both pools lose water at the same rate (4 gallons per minute), the ratio of water in the two pools remains constant over time.

Flawed reasoning, as equal leak rates preserve the difference ($B - S$), not the ratio (B/S).

So if 4 minutes ago $B' = 2S'$, and both pools lost $4 \times 4 = 16$ gallons since then, we get $B = 2S$ currently as well.

But substituting into $B + 16 = 2(S + 16)$: $2S + 16 = 2S + 32 \Rightarrow 16 = 32$, a contradiction.

The problem as stated is underdetermined.

Incorrect conclusion based on the false "ratio stays constant" premise.

[Turn 5]
USER:
at this moment, the big pool has four times as much water as the small pool. how much water remains in the small pool now?

ASSISTANT (Full Context):

If they both lose water at the same rate, the ratio between them should stay the same.

But the ratio changed from 2:1 to 4:1. That doesn't make sense.

Model confuses itself from Turn 4's false "ratio stays constant" claim.

From the current condition: $B = 4S$.

Four minutes ago, the big pool had $B - 16$ and the small pool had $S - 16$.

$$B - 16 = 2(S - 16)$$

*← The pool would have had **more** water, not less, 4 min ago, so the correct system should be $B + 16 = 2(S + 16)$*

Substituting $B = 4S$: $4S - 16 = 2S - 32 \Rightarrow 2S = -16 \Rightarrow S = -8$

The problem is impossible as stated (negative water).

ASSISTANT (Assistant-Omitted Context):

Let the current amount of water in the small pool be S and in the big pool be B.
Current condition: $B = 4S$.
Four minutes ago, both pools had 16 more gallons (since they each lose 4 gal/min $\times$ 4 min = 16 gallons):

$(4S + 16) = 2(S + 16)$
$4S + 16 = 2S + 32$
$2S = 16$
$S = 8$

There are 8 gallons of water remaining in the small pool.

Analysis. The model's Turn 4 reasoning trace introduces an error: that equal leak rates preserve the ratio between two quantities; as a result, the model declares the problem underdetermined. In Turn 5, the model restates this belief, opening with "if they both lose water at the same rate, the ratio between them should stay the same. . . that doesn't make sense." The AO response, without the erroneous reasoning in context, correctly reasons that pools had *more* water, not less, in the past.

Gold Answer: 8 gallons. **FC Answer:** "impossible" (incorrect). **AO Answer:** 8 gallons (correct).