ISOLATED CONSISTENCY: IMPROVE MULTI-TURN REASONING VIA TURN-LEVEL ISOLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-turn conversation lets users specify a task gradually, but language models often fail even after all requirements have been provided. One difficulty is that the model starts answering before it has the full task. These early responses remain in the conversation and can shape later answers, even when new requirements invalidate their assumptions. Restating the complete task clarifies what the model should solve, but does not tell it which earlier responses to trust. These responses may offer useful intermediate reasoning while also carrying mistaken assumptions, so neither keeping nor discarding them all is an obvious choice. We observe that responses containing errors often make up only a minority of assistant turns. This suggests that isolating earlier responses can keep these errors out of most candidate contexts, rather than exposing every attempt to them. Instead of training a selector to decide what to retain, we introduce a simple but effective method, Isolated Consistency (IC), that is training-free, compatible with both white-box and black-box models, revisits the task with each earlier response in isolation, and selects the final answer through agreement across the resulting candidates. Across seven models, IC yields absolute accuracy gains over standard multi-turn prompting in mathematical reasoning (***+13.64%***), code generation (***+14.43%***), text-to-SQL database querying (***+27.50%***), and structured action generation (***+47.49%***). With the same candidate budget and voting rule, IC improves macro-average accuracy by 3.19% over the strongest multi-turn baseline. Remarkably, the benefit persists even on the frontier model *GPT-5.5*, where IC raises the four-task average by 25.87% over standard multi-turn prompting and by 3.54% over the strongest multi-turn baseline.

1 INTRODUCTION

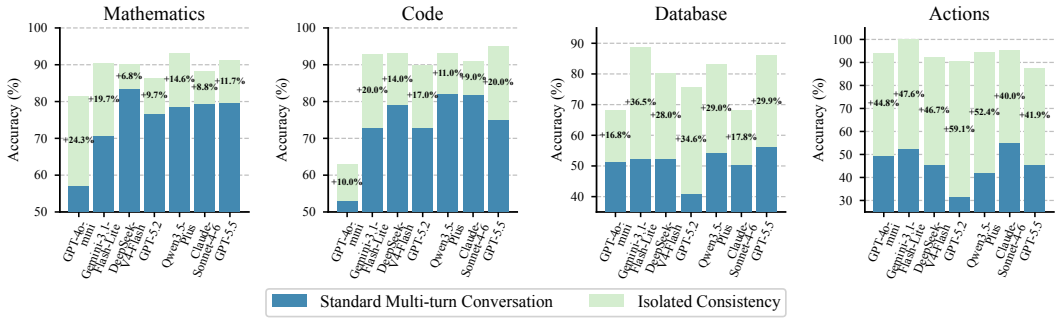


Figure 1: Isolated Consistency significantly improves performance in multi-turn conversations.

Long multi-turn conversations are becoming a common interface for working with large language models, as reflected in large-scale conversation logs and studies of real-world use (Zheng et al., 2024; Handa et al., 2025). The rapid adoption of chatbots can make multi-turn interaction appear to be a largely solved capability. Yet a body of evidence shows that models still struggle to maintain, update, and revise information across turns, so conversational context remains a challenge even for

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

advanced systems (Laban et al., 2026; Huang et al., 2026; Li et al., 2025; Sinha et al., 2026). Context management determines not only how much text the model must process, but also which parts of the exchange continue to influence its next answer. Longer histories increase attention cost and can make relevant information harder to use reliably (Dao et al., 2022; Eyuboglu et al., 2026; Liu et al., 2024; Hong et al., 2025; Lee et al., 2026). Deployed and research systems summarize, compact, mask, or omit older material (Anthropic, 2025a;b; Cursor, 2026; Lopopolo, 2026; Xiao et al., 2025; Lindenbauer et al., 2025; Liu et al., 2026b). These operations do more than control length and cost: they determine what the model remembers. That decision is especially consequential when users specify tasks gradually. Multi-turn interaction is useful because users can begin with an incomplete request, inspect the answer, and add constraints. Studies find that underspecified requests are common and that non-experts often reveal requirements gradually rather than formulate complete prompts upfront (Herlihy et al., 2024; Chang et al., 2025; Zamfirescu-Pereira et al., 2023; Knoth et al., 2024; Zipf, 2016). The resulting history grows along two different tracks: user messages progressively define what should be solved, while assistant messages record attempts made before that definition was complete. When the final requirement arrives, the user side may now specify the complete task, but the assistant side can still carry assumptions formed under its earlier versions. Lost in Conversation shows that this mismatch reduces accuracy even after all necessary information is present (Laban et al., 2026). Recovery therefore requires the model not only to incorporate the final constraints, but also to revise reasoning already embedded in the dialogue.

Standard multi-turn prompting preserves conversational continuity by conditioning each new response on the entire preceding transcript, but this also keeps every earlier assistant response in context. Earlier context-compression methods largely treated the full transcript as the target to preserve (Jiang et al., 2023; Li, 2023), whereas work on conversational question answering has explicitly selected only the history turns relevant to the current request (Choi et al., 2018; Reddy et al., 2019; Qu et al., 2019; Do et al., 2022; Zaib et al., 2023). We study recovery within the standard full-history protocol, where the user continues the same conversation after completing the task specification. A natural strategy is to collect the requirements into a recap and ask the model to solve the completed task again. The recap makes the user's final intent explicit and recovers much of the accuracy lost to incremental disclosure (Laban et al., 2026). Yet it is appended to the conversation rather than replacing what came before. The model now conditions on two records at once: a current specification of the task and earlier answers based on less complete specifications. Keeping those answers implicitly assumes that the full assistant history is harmless, or useful, when the model tries again. Prior evidence of over-reliance on self-generated content, reversals under conversational challenge, and propagation of early errors makes this assumption uncertain (Huang et al., 2026; Laban et al., 2023; Jiang et al., 2025; Li et al., 2025). Earlier responses may preserve useful intermediate reasoning, but the same responses may also carry assumptions that no longer apply. Recovery must therefore use this history without allowing every old mistake to shape every new attempt.

Whether such selective use is possible depends on how widely the mistakes that matter are distributed across the conversation. If nearly every assistant response contributes to the final failure, little reliable history remains to preserve. If only a small subset does, the other responses may still provide useful context. We label each intermediate response by whether it is incorrect and whether its error contributes to the final failure. Averaged over the four task categories, 23.11–37.07% of turns are labeled incorrect across the seven evaluated models, but only **10.35–21.75%** are labeled as contributing to the final failure (Figure 2). Most assistant turns are therefore not identified as causes of the eventual failure. When failures arise from a minority of turns, we naturally consider generating several candidates from different assistant turns and using agreement to outvote the few candidates exposed to harmful responses. Self-consistency (Wang et al., 2023) provides a familiar way to turn repeated sampling into a final answer: it samples multiple reasoning paths and selects the most common

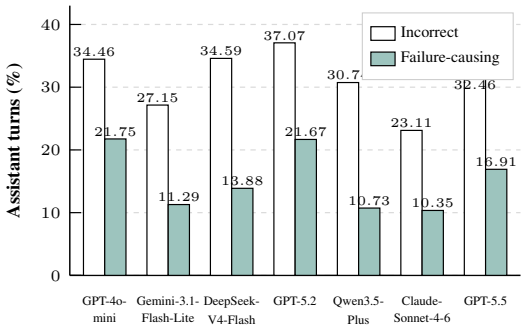


Figure 2: Turn-level error rates across Math, Code, Database, and Actions. White marks errors; green marks errors contributing to final failure.

output. After a recap, however, all of its candidates still see the same assistant responses. Sampling varies how the model continues from that context, but not the earlier reasoning on which every continuation is conditioned. A harmful response can therefore steer several candidates toward the same mistake, leaving voting with little disagreement to exploit. To use the turn-level concentration observed above, the candidates must also differ in the assistant history they see. If each attempt sees a different assistant turn, the few harmful responses need not influence every candidate, while the remaining candidates can reconsider the task without seeing them. Agreement can then favor answers supported by the unaffected candidates. The same support pattern can also prioritize isolated views for inspection: a candidate backed by few other views is more likely to be unreliable.

We therefore take an embarrassingly simple route and introduce **Isolated Consistency (IC)**, which uses agreement across isolated assistant turns to recover the final answer. Rather than deciding which earlier responses are trustworthy, IC creates one candidate context for each assistant turn. Every context preserves all user messages and the same clean recap, but exposes the model to only one assistant response while masking the rest. The model solves the same completed task several times, each time reconsidering it alongside a different piece of its earlier reasoning. IC then groups equivalent outputs and returns the answer with the most support. In this way, recovery no longer depends on a single decision about which earlier response to keep; it depends on whether an answer remains stable across different views of the assistant history. This perspective leads to two contributions: ① We introduce Isolated Consistency, a training-free method that votes across isolated assistant turns while keeping the completed task fixed. It supports white-box and black-box models. Candidate disagreement also helps identify earlier assistant turns that warrant inspection, without requiring additional model calls. ② We evaluate IC on seven models from different model families: GPT-4o-mini, Gemini-3.1-Flash-Lite, DeepSeek-V4-Flash, GPT-5.2, Qwen3.5-Plus, Claude-Sonnet-4-6, and GPT-5.5. The evaluation covers four analytical tasks. Relative to standard multi-turn prompting, IC improves mathematical reasoning (***+13.64%***), code generation (***+14.43%***), text-to-SQL database querying (***+27.50%***), and structured action generation (***+47.49%***). Remarkably, these gains remain substantial even on the frontier model *GPT-5.5*, where IC improves by 25.87% over standard multi-turn prompting and by 3.54% over the strongest multi-turn baseline.

2 METHOD

Multi-turn conversation is valuable precisely because users do not need to fully specify a task before the interaction begins. They can start with an initial goal, inspect the model's response, and use what they learn to clarify or extend the request. Each turn thereby updates a shared conversational state rather than beginning a new interaction (Lewis, 1979). For each new turn to build on this process, standard prompting passes the entire transcript back to the model whenever it answers (Huang et al., 2026). This is a natural way to preserve continuity: the model can see every request, clarification, and response that led to the current state. Yet it also ties conversational continuity to full-history conditioning. The model's own earlier answers remain in the context alongside the user's requirements, even when those answers were produced for an incomplete version of the task. Learned memory managers address this problem by training a model to decide what to retain (Liu et al., 2026b); IC instead asks whether structured variation of the visible history can support recovery without additional training.

Following the sharded-conversation setting of Laban et al. (2026), let Q denote a complete task with target answer $y^\star$. Its requirements are divided into shards $q_1, \ldots, q_K$ and revealed through user messages $U_1, \ldots, U_K$; the model returns A_i after each U_i. Given system prompt S, the completed transcript is

$$H_K = [S, U_1, A_1, U_2, A_2, \ldots, U_K, A_K]. \tag{1}$$

By turn K, the user messages $U_{1:K}$ jointly specify Q. Yet each A_i records an attempt made from only the requirements available at turn i and the model outputs that preceded it. Consequently, the completed transcript contains two records with different status: a current specification of the task and provisional attempts to solve its earlier versions. Stating Q again therefore resolves what the model should solve, but not which earlier reasoning it should inherit. A new answer still conditions on $A_{1:K}$, so repeated generation changes the continuation while leaving the potentially harmful assistant history untouched. We examine the intermediate assistant turns more closely and observe a useful asymmetry: responses judged to contribute to the final failure are concentrated in a minority of turns (Figure 2). That concentration suggests a natural remedy. If only a few attempts are influenced

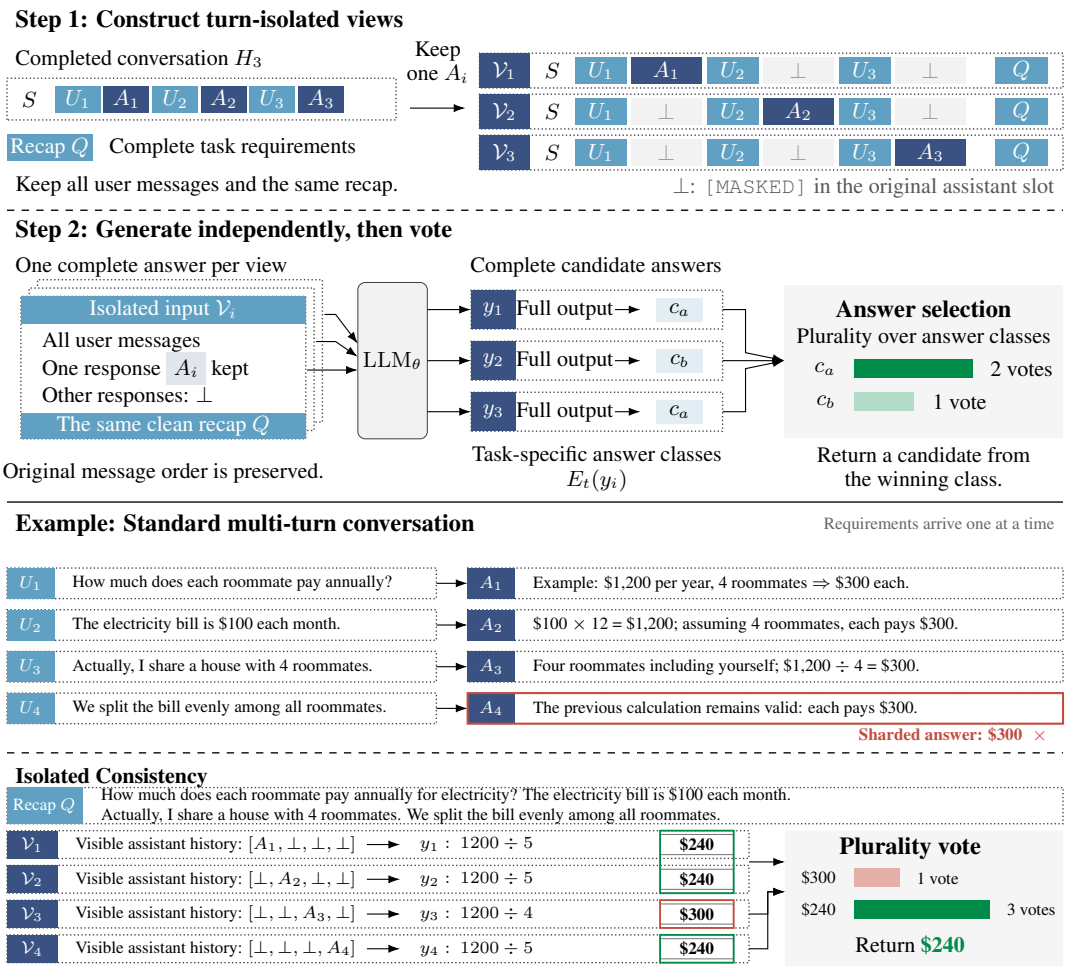


Figure 3: Isolated Consistency retains one assistant response per view and selects the plurality answer. In this real GPT-4o-mini example, three of four isolated views correct the standard conversation's $300 answer to $240.

by harmful turns, repeated sampling and agreement can allow the unaffected candidates to determine the final answer. Self-consistency follows this general principle by sampling several reasoning paths and selecting the most common output (Wang et al., 2023). When it is applied directly to the completed conversation, however, every candidate is sampled from the same assistant history:

$$y_i^{\mathrm{SC}} \sim p_\theta(\cdot \mid S, U_{1:K}, A_{1:K}, Q), \qquad i = 1, \ldots, K. \tag{2}$$

The sampled outputs y_i^{SC} may differ, but the conditioning history $A_{1:K}$ does not. A harmful turn can still steer several candidates toward the same mistake, leaving voting with little disagreement to exploit. To make agreement useful here, candidates must differ not only in their sampled continuations, but also in the assistant history they receive. We therefore isolate the assistant turns before generation. The resulting method, Isolated Consistency (IC), constructs one candidate context per assistant turn, retains a different response in each context, and aggregates the answers through agreement.

Isolated Consistency. Concretely, IC turns this idea into K views of the same completed task. Every view contains the system prompt S, all user messages $U_{1:K}$, and the complete task Q, so the model is always asked to solve the same problem. What changes from one view to the next is the visible assistant history. Starting from transcript H_K, view i keeps A_i and replaces every other assistant response with the marker $\perp_A$. We implement this marker as [MASKED] in the original

assistant-message slot, preserving the original message order:

$$\mathcal{V}_i = [S, U_1, \widetilde{A}_1^{(i)}, \ldots, U_K, \widetilde{A}_K^{(i)}, Q], \qquad \widetilde{A}_j^{(i)} = \begin{cases} A_j, & j = i, \\ \perp_A, & j \neq i. \end{cases} \tag{3}$$

The same model then answers once from each fixed view. As Figure 3 illustrates, every $\mathcal{V}_i$ keeps the same user messages $U_{1:K}$ and the same completed task Q, so all candidates solve exactly the same task. The only difference is which earlier assistant response A_i remains visible. This separation allows useful reasoning in A_i to help candidate i without also passing a mistake in A_i to every other candidate. Errors repeated across several responses may still affect several views, but an error confined to one response can influence only the candidate generated from that view. Once the candidates have been generated, IC looks for the answer that reappears across the largest number of views. Equivalent answers may be phrased differently, so E_t maps each output to a common answer class before the votes are counted. The generation and voting steps can therefore be written as

$$y_i \sim p_\theta(\cdot \mid \mathcal{V}_i) \quad (i = 1, \ldots, K), \qquad \widehat{c} = \arg\max_c \sum_{i=1}^{K} \mathbf{1}[E_t(y_i) = c], \tag{4}$$

In summary, Isolated Consistency returns a representative candidate from the winning class $\widehat{c}$ and breaks ties with a fixed rule. More importantly, isolation turns the assistant history itself into a controlled source of candidate diversity. Full-history self-consistency asks the same context for different continuations, whereas IC asks different pieces of earlier reasoning to support the same completed task. Because IC and full-history self-consistency use the same candidate budget and plurality rule, they differ only in the assistant history shown to each candidate. This controlled comparison directly tests whether sampling from different assistant turns improves recovery over repeatedly sampling from the same full history. We evaluate this effect across seven models and four analytical tasks in the next section.

3 Experiments

We evaluate IC through a sequence of comparisons that separates its main effect from the conditions under which that effect is most useful. Section 3.1 defines the shared tasks, models, baselines, and evaluation protocol. Section 3.2 then tests whether changing the visible assistant history improves recovery beyond restating the completed task and repeatedly sampling from the same transcript. Section 3.3 examines how this advantage changes with the number of candidate views and conversation length, and whether agreement among isolated views provides a useful diagnostic signal.

3.1 Setup

Benchmarks and evaluation metrics. We use four analytical tasks from Lost in Conversation (Laban et al., 2026): **Math**, grade-school arithmetic derived from GSM8K (Cobbe et al., 2021); **Code**, Python function synthesis from HumanEval and LiveCodeBench-style sources (Chen et al., 2021; Jain et al., 2025); **Database**, text-to-SQL over Spider schemas (Yu et al., 2018); and **Actions**, structured API calls based on the Berkeley Function Calling Leaderboard (Patil et al., 2025). The source benchmark rewrites a complete instruction into conversational, decontextualized shards whose joint content preserves the original task (Choi et al., 2021; Laban et al., 2026). Each task therefore requires a final answer that satisfies all shards, including requirements disclosed after the model has already responded. We follow the benchmark evaluators: normalized exact match for Math, execution for Code and SQL (Zhong et al., 2020), and semantic equivalence to the required calls for Actions. We macro-average accuracy over tasks and then models, reporting scores in percent and changes in points. Further details are provided in Appendix C.

Base models. We evaluate GPT-4o-mini (OpenAI, 2024), Gemini-3.1-Flash-Lite (Google, 2026), DeepSeek-V4-Flash (DeepSeek-AI, 2026), GPT-5.2 (OpenAI, 2025), Qwen3.5-Plus (Qwen Team, 2026), Claude-Sonnet-4-6 (Anthropic, 2026), and GPT-5.5 (OpenAI, 2026). The same model produces both the original assistant responses and the recovery candidates, with recovery temperature 1.0. Appendices C and D give the implementation details and prompts.

Compared methods. SHARDED is the unrecovered final answer from the original multi-turn trajectory. Recap (RC) appends the completed task Q to the full transcript and generates one new answer,

Table 1: Accuracy (%) of multi-turn recovery methods across seven models and four analytical tasks. Parentheses show changes from Sharded; Restart is the gray idealized reference. Blue and green mark the best and second-best recovery results.

Model	Method	Accuracy (%)				
		Math	Code	Database	Actions	Average
GPT-4o-mini	Sharded	57.28	53.00	51.40	49.52	52.80
	Recap	68.93 (+11.65)	58.00 (+5.00)	59.81 (+8.41)	87.62 (+38.10)	68.59 (+15.79)
	SC	71.84 (+14.56)	59.00 (+6.00)	59.81 (+8.41)	88.57 (+39.05)	69.81 (+17.01)
	AO	82.52 (+25.24)	63.00 (+10.00)	71.96 (+20.56)	79.05 (+29.53)	74.13 (+21.33)
	SUM	78.64 (+21.36)	59.00 (+6.00)	52.34 (+0.94)	71.43 (+21.91)	65.35 (+12.55)
	MT-OSC	63.11 (+5.83)	56.00 (+3.00)	47.66 (-3.74)	47.62 (-1.90)	53.60 (+0.80)
	IC	**81.55 (+24.27)**	**63.00 (+10.00)**	**68.22 (+16.82)**	**94.29 (+44.77)**	**76.77 (+23.97)**
	Restart	88.35	67.00	89.72	94.29	84.84
Gemini-3.1-Flash-Lite	Sharded	70.59	73.00	52.34	52.38	62.08
	Recap	87.25 (+16.66)	83.00 (+10.00)	83.18 (+30.84)	97.14 (+44.76)	87.64 (+25.56)
	SC	86.41 (+15.82)	85.00 (+12.00)	83.18 (+30.84)	97.14 (+44.76)	87.93 (+25.85)
	AO	89.32 (+18.73)	83.00 (+10.00)	79.44 (+27.10)	69.52 (+17.14)	80.32 (+18.24)
	SUM	77.67 (+7.08)	79.00 (+6.00)	70.09 (+17.75)	80.95 (+28.57)	76.93 (+14.85)
	MT-OSC	75.73 (+5.14)	73.00 (+0.00)	48.60 (-3.74)	52.38 (+0.00)	62.43 (+0.35)
	IC	**90.29 (+19.70)**	**93.00 (+20.00)**	**88.79 (+36.45)**	**100.00 (+47.62)**	**93.02 (+30.94)**
	Restart	95.15	90.00	93.46	100.00	94.65
DeepSeek-V4-Flash	Sharded	83.50	79.00	52.34	45.71	65.14
	Recap	88.35 (+4.85)	85.00 (+6.00)	72.90 (+20.56)	87.62 (+41.91)	83.47 (+18.33)
	SC	87.38 (+3.88)	95.00 (+16.00)	76.64 (+24.30)	90.48 (+44.77)	87.38 (+22.24)
	AO	79.61 (-3.89)	75.00 (-4.00)	55.14 (+2.80)	46.67 (+0.96)	64.11 (-1.03)
	SUM	77.67 (-5.83)	94.00 (+15.00)	49.53 (-2.81)	57.14 (+11.43)	69.59 (+4.45)
	MT-OSC	77.67 (-5.83)	85.00 (+6.00)	51.40 (-0.94)	49.52 (+3.81)	65.90 (+0.76)
	IC	**90.29 (+6.79)**	**93.00 (+14.00)**	**80.37 (+28.03)**	**92.38 (+46.67)**	**89.01 (+23.87)**
	Restart	85.44	88.00	89.72	94.29	89.36
GPT-5.2	Sharded	76.70	73.00	41.12	31.43	55.56
	Recap	85.44 (+8.74)	86.00 (+13.00)	60.75 (+19.63)	86.67 (+55.24)	79.72 (+24.16)
	SC	89.32 (+12.62)	86.00 (+13.00)	68.22 (+27.10)	89.52 (+58.09)	83.27 (+27.71)
	AO	81.55 (+4.85)	88.00 (+15.00)	55.14 (+14.02)	66.60 (+35.17)	72.82 (+17.26)
	SUM	75.73 (-0.97)	87.00 (+14.00)	44.86 (+3.74)	69.52 (+38.09)	69.28 (+13.72)
	MT-OSC	73.79 (-2.91)	84.00 (+11.00)	43.93 (+2.81)	57.14 (+25.71)	64.72 (+9.16)
	IC	**86.41 (+9.71)**	**90.00 (+17.00)**	**75.70 (+34.58)**	**90.48 (+59.05)**	**85.65 (+30.09)**
	Restart	86.41	88.00	79.44	90.48	86.08
Qwen3.5-Plus	Sharded	78.64	82.00	54.21	41.90	64.19
	Recap	85.44 (+6.80)	89.00 (+7.00)	79.44 (+25.23)	91.43 (+49.53)	86.33 (+22.14)
	SC	91.26 (+12.62)	91.00 (+9.00)	81.31 (+27.10)	94.29 (+52.39)	89.47 (+25.28)
	AO	91.26 (+12.62)	87.00 (+5.00)	69.16 (+14.95)	69.52 (+27.62)	79.24 (+15.05)
	SUM	81.55 (+2.91)	86.00 (+4.00)	70.09 (+15.88)	68.57 (+26.67)	76.55 (+12.36)
	MT-OSC	80.58 (+1.94)	81.00 (-1.00)	51.40 (-2.81)	49.52 (+7.62)	65.63 (+1.44)
	IC	**93.20 (+14.56)**	**93.00 (+11.00)**	**83.18 (+28.97)**	**94.29 (+52.39)**	**90.92 (+26.73)**
	Restart	97.09	96.00	96.26	95.24	96.15
Claude-Sonnet-4-6	Sharded	79.41	82.00	50.47	55.24	66.78
	Recap	90.29 (+10.88)	89.00 (+7.00)	59.81 (+9.34)	96.19 (+40.95)	83.82 (+17.04)
	SC	86.27 (+6.86)	90.00 (+8.00)	67.29 (+16.82)	94.29 (+39.05)	84.46 (+17.68)
	AO	73.79 (-5.62)	81.00 (-1.00)	53.27 (+2.80)	47.62 (-7.62)	63.92 (-2.86)
	SUM	87.38 (+7.97)	82.00 (+0.00)	47.66 (-2.81)	53.33 (-1.91)	67.59 (+0.81)
	MT-OSC	80.58 (+1.17)	79.00 (-3.00)	43.93 (-6.54)	46.67 (-8.57)	62.55 (-4.23)
	IC	**88.24 (+8.83)**	**91.00 (+9.00)**	**68.22 (+17.75)**	**95.24 (+40.00)**	**85.68 (+18.90)**
	Restart	93.20	94.00	87.85	99.05	93.53
GPT-5.5	Sharded	79.61	75.00	56.07	45.71	64.10
	Recap	89.32 (+9.71)	85.00 (+10.00)	71.96 (+15.89)	86.67 (+40.96)	83.24 (+19.14)
	SC	88.35 (+8.74)	95.00 (+20.00)	76.64 (+20.57)	85.71 (+40.00)	86.43 (+22.33)
	AO	76.70 (-2.91)	90.00 (+15.00)	55.14 (-0.93)	63.81 (+18.10)	71.41 (+7.31)
	SUM	77.67 (-1.94)	86.00 (+11.00)	42.99 (-13.08)	64.76 (+19.05)	67.86 (+3.76)
	MT-OSC	78.64 (-0.97)	84.00 (+9.00)	51.40 (-4.67)	51.43 (+5.72)	66.37 (+2.27)
	IC	**91.26 (+11.65)**	**95.00 (+20.00)**	**85.98 (+29.91)**	**87.62 (+41.91)**	**89.97 (+25.87)**
	Restart	95.15	89.00	86.92	87.62	89.67

while Self-Consistency (SC) (Wang et al., 2023) draws K candidates from the same recap prompt and applies the answer-grouping and plurality rule used by IC. IC uses the same candidate count and voting rule, but generates one candidate from each isolated assistant-turn view. We also compare three training-free, text-only history-management methods. Assistant-Omitted (AO) replaces each assistant response with `[Response provided]`; Summarized (SUM) (Huang et al., 2026) replaces it with a one-sentence summary generated by the evaluated model; and MT-OSC (Singh et al., 2026) uses a prompted condenser and rule-based decider to compact the dialogue. Restart provides an idealized reference by joining all instruction shards into one user message and solving

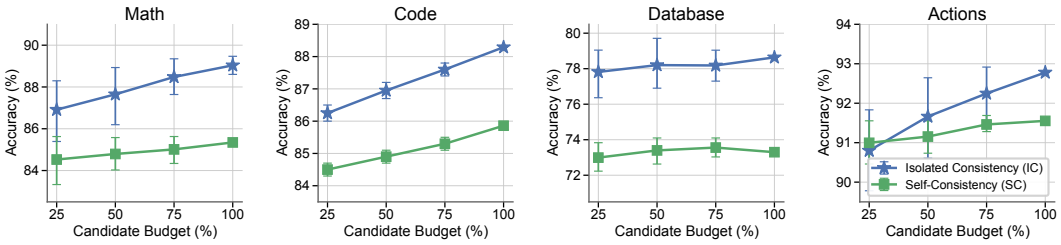


Figure 4: Matched-budget comparison between IC and SC on Math, Code, Database, and Actions, macro-averaged over seven models. Each point uses 25%, 50%, 75%, or 100% of the available candidate pool. Error bars show the observed variation across matched candidate subsets.

them without the original conversation history. In practice, this would require users to reconstruct and resubmit the completed task, or require the system to infer when an evolving request has become complete and initiate a fresh interaction at the appropriate moment.

Comparison protocol. To isolate the effect of varying the assistant history, we compare IC with SC under the same candidate budget and aggregation rule. SC samples all K candidates from a shared full-history recap context, whereas IC generates one candidate from each isolated assistant turn. RC shows how much recovery comes from making the completed task explicit before repeated sampling is introduced. The reset pairs separate the instruction to reconsider an earlier answer from the choice of visible history, while Restart serves as an idealized single-turn reference. Differences between displayed aggregate scores are computed from the reported two-decimal values.

3.2 Main Results

Does isolating assistant turns improve recovery beyond repeated sampling from the same history? Table 1 presents the complete results. IC reaches 87.29% average accuracy, compared with 84.10% for SC. This 3.19% gain is obtained with the same number of candidates and the same voting rule. The comparison therefore isolates candidate context: SC repeatedly samples from one full-history recap, whereas IC retains a different assistant response for each candidate.

Which stages of recovery account for the improvement? Restating the task already raises accuracy from 61.52% for Sharded to 81.83% for RC. Repeated sampling and voting raise the score further to 84.10%, and varying the retained assistant turn increases it again to 87.29%. These successive comparisons separate three sources of recovery: making the final requirements explicit, generating more than one attempt, and changing the history seen by those attempts. By contrast, AO, SUM, and MT-OSC reach 72.28%, 70.45%, and 63.03%, respectively. Their smaller gains show that omitting or compressing the accumulated history does not substitute for first recovering the complete task in this sharded setting.

Does the improvement hold across models? Relative to SC, IC improves the four-task average by 1.22% to 6.96% across all seven evaluated models. The gains include 6.96% for GPT-4o-mini, 5.09% for Gemini-3.1-Flash-Lite, and 3.54% for GPT-5.5. At the finer model–task level, IC improves 24 of 28 comparisons, ties two, and is lower in two; it also exceeds RC in 26 of 28 comparisons. Thus, the average improvement reflects a broad cross-model pattern rather than a result driven by one model, while the individual comparisons identify the remaining exceptions.

Does turn isolation contribute equally across tasks? Averaged across models, IC improves over SC by 5.34% on Database, 2.92% on Math, 2.43% on Code, and 2.04% on Actions. Actions illustrates why these stages should be separated: RC alone raises accuracy from 45.98% to 90.48%, after which IC reaches 93.47%. Most of the Actions recovery therefore comes from restating the completed requirements, whereas Database gains more from varying the assistant history.

Taken together, these results show that multi-turn recovery depends not only on restating the completed task or sampling more answers, but also on controlling which earlier model responses shape each new attempt. We next examine when this advantage emerges and whether agreement across isolated histories can also reveal where the conversation went wrong.

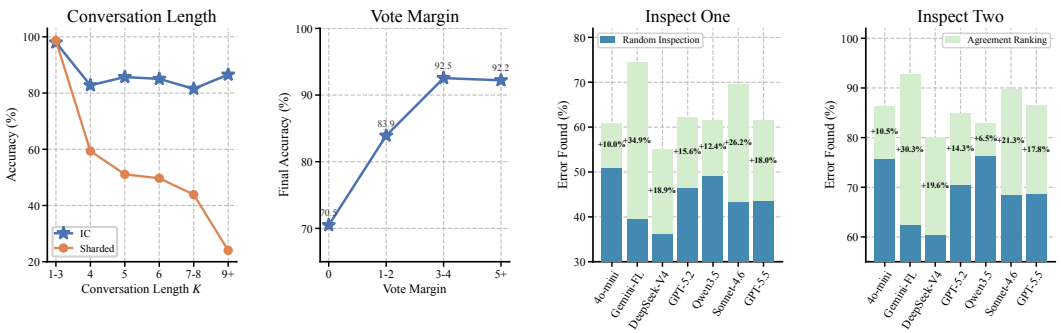


Figure 5: Recovery and diagnostic behavior of IC. From left to right: IC and Sharded accuracy across conversation-length bins; final IC accuracy grouped by vote margin; the probability that the lowest-support candidate is incorrect; and the probability that at least one of the two lowest-support candidates is incorrect. The last two panels compare agreement-based inspection with matched random inspection on the conversations containing both correct and incorrect candidates.

3.3 Further Analysis

How many isolated views are needed for IC to improve recovery? The main comparison uses one candidate per assistant turn, but a smaller sample may suffice. For conversations shared by IC and SC, we match their candidate pools, randomly retain 25%, 50%, 75%, or 100%, and vote using the original grouping rule. We draw 200 subsamples for Math, Database, and Actions, while Code uses separately recorded matched-budget ranges based on executable-test pass vectors. At 25% of the matched pool, IC already reaches 85.44% four-task macro-average accuracy, compared with 83.25% for SC. The gap grows from 2.19 to 3.17 points as the budget increases to 100%, while IC itself improves to 87.19% on this matched subset. Figure 4 shows the clearest separation on Database and Math, with a smaller but consistent advantage on Code. On Actions, the two methods are similar at the smallest budget, but IC improves steadily as additional isolated views are included. Thus, IC does not need every available view to produce a gain, although additional views make the agreement signal more effective.

When is recovery most valuable? Prior work reports degradation over longer interaction horizons and propagation of early errors through subsequent steps (Sinha et al., 2026; Li et al., 2025). We therefore group the original trajectories by the number of assistant turns and compare their unrecovered Sharded answers with the final IC vote. Figure 5 (left) shows little need for recovery in trajectories with at most three assistant turns, where Sharded already reaches 98.64% accuracy. From four turns onward, however, Sharded falls from 59.39% to 24.04% in the longest bin, while IC remains between 81.50% and 86.54%. The corresponding recovery gap grows from 23.37 points at four turns to 62.50 points at nine or more turns. Across these mixed task and model populations, the widening gap identifies longer conversations as the setting in which recovery provides the greatest practical benefit, maintaining reliable answers as standard multi-turn performance declines.

Can the vote margin indicate when IC's answer is reliable? Figure 5 (center-left) groups conversations by the difference between the two largest answer-class counts. When the vote is tied, the selected answer is correct in 70.48% of conversations. Accuracy rises to 83.91% for margins of one or two votes and to 92.54% for margins of three or four, after which it remains similar at 92.21%. The margin therefore provides a practical confidence signal, separating uncertain tied votes from strongly supported answers whose accuracy exceeds 92% once the margin reaches three.

Can agreement help identify which isolated views are likely to be wrong? We test this question on the conversations that contain at least one correct and one incorrect candidate, since all-correct and all-incorrect sets cannot reveal whether support separates the two. Figure 5 (center-right and right) shows that the lowest-support candidate is incorrect in 63.32% of these conversations, compared with 44.47% under matched random ranking. Inspecting the two lowest-support candidates finds an incorrect answer in 86.18% of conversations, compared with 69.20% for random selection. The corresponding pairwise ranking AUC is 62.64%, and every model shows a positive one-candidate gain over random. Agreement therefore turns voting into a diagnostic tool: rank-

ing candidates by support directs developers to the isolated history views most likely to contain an incorrect answer, helping them inspect the source of a failure more quickly.

4 Related Work

Multi-turn evaluation. Recent benchmarks extend dialogue evaluation to longer interactions and task-specific settings such as tool use, fairness, mathematics, and entangled instructions (Zheng et al., 2023; Bai et al., 2024; Kwan et al., 2024; Duan et al., 2024; Deshpande et al., 2025; Wang et al., 2024; Fan et al., 2025; Liang et al., 2024; Han et al., 2025); other work trains multi-turn instruction following directly (Sun et al., 2024). Studies of underspecification examine missing details, ambiguity, contextual evaluation, and clarification (Herlihy et al., 2024; Wildenburg et al., 2024; Malaviya et al., 2025; Shaikh et al., 2025; Peng et al., 2023; Chang et al., 2025), while prompt rewriting and intent alignment improve the user-side request (Sarkar et al., 2025; Kim et al., 2024; Terry et al., 2024). Lost in Conversation instead fixes the task and evaluator while revealing requirements incrementally (Laban et al., 2026). This controlled setting separates failure caused by incremental interaction from failure caused by an ambiguous final task. IC studies the next question: once every requirement is available, how should recovery use assistant responses produced before the complete task was known?

History management. Conversational question answering selects the prior turns most relevant to the current question (Qu et al., 2019; Do et al., 2022; Zaib et al., 2023). Prompt-compression and memory systems shorten transcripts, edit trajectories, select retained content, or retrieve external state (Jiang et al., 2023; Li, 2023; Khattab et al., 2021; Pan et al., 2024; Yang et al., 2025; Anthropic, 2025a; Liu et al., 2026b; Xiao et al., 2025; Packer et al., 2023; Zhang et al., 2026). Closer inference-time methods omit or summarize assistant responses (Huang et al., 2026), compress dialogue windows (Singh et al., 2026), or reconstruct the current instruction (Khalid et al., 2025; Liu et al., 2026a); training-based alternatives modify the policy or memory (Chen et al., 2026; Luo et al., 2026). Whether selected, compressed, or reconstructed, these approaches produce one history representation before generating the answer. That choice is difficult when earlier responses contain both useful deductions and obsolete assumptions. IC keeps the model fixed, avoids choosing one history in advance, and instead votes across several isolated views.

Answer aggregation. Self-consistency samples several reasoning paths from one prompt and selects the most common answer (Wang et al., 2023), but every candidate remains conditioned on the same history. Integrative Decoding introduces context diversity by combining token predictions across inputs built from sampled responses (Cheng et al., 2025), which requires access to model prediction scores during decoding. IC likewise uses agreement across contexts, but varies the retained assistant turn and votes over complete text outputs. It therefore remains compatible with text-only black-box models and avoids exposing users to several inconsistent generations (Lee et al., 2024). Unlike self-refinement and reflection (Madaan et al., 2023; Shinn et al., 2023), IC does not first ask the model to identify and revise the faulty response; agreement both selects the final answer and prioritizes earlier turns for inspection.

5 Conclusion

In a multi-turn conversation, a model often begins solving the task before the user has finished specifying it. When the final requirement arrives, the transcript therefore contains both the completed request and earlier answers built on incomplete information. Restating the task and sampling more answers helps, but leaves a source of failure untouched when every attempt still sees the same potentially misleading responses. Isolated Consistency addresses this problem by keeping the user's requirements fixed, letting each new attempt see only one earlier assistant response, and voting across the resulting answers. Across seven models and four analytical tasks, IC outperforms the strongest multi-turn baseline on every model; the same vote also helps developers prioritize unreliable views for inspection. These results suggest a practical change in how multi-turn systems should use history: preserve the user's accumulated requirements, but treat the assistant's earlier responses as alternatives to be tested rather than facts that every later answer must inherit. IC thus preserves conversational continuity while allowing the model to reconsider its own earlier answers.

AI USE STATEMENT

We used generative AI for coding assistance, and improving the clarity of author-written text. In our experiments, GPT-4o-mini served as the user simulator and the system agent for response classification and answer extraction, while the seven evaluated models generated the task responses. The authors reviewed all AI-assisted work and take full responsibility for the paper.

ETHICS STATEMENT

This work uses controlled benchmark conversations and public task datasets; it does not use private user conversations or collect personal data. The method generates multiple model responses and therefore increases inference cost. Its benchmark gains should not be interpreted as guarantees in consequential applications, where outputs require independent verification.

REPRODUCIBILITY STATEMENT

Our implementation extends the public Lost in Conversation framework. Appendix C documents the experimental pipeline, generation settings, answer grouping, and evaluation, while Appendix D provides the complete prompts. Algorithm 1 specifies the full IC procedure. The code is provided in the supplementary materials.

REFERENCES

Anthropic. Context editing and tool result clearing. Claude Developer Documentation, 2025a. URL https://platform.claude.com/docs/en/build-with-claude/context-editing.

Anthropic. Effective context engineering for AI agents. Anthropic Engineering Blog, 2025b. URL https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents.

Anthropic. Introducing Claude Sonnet 4.6. Anthropic News, 2026. URL https://www.anthropic.com/news/claude-sonnet-4-6.

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, and Wanli Ouyang. MT-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7421–7454. Association for Computational Linguistics, August 2024. doi: 10.18653/v1/2024.acl-long.401. URL https://aclanthology.org/2024.acl-long.401/.

Tuhin Chakrabarty, Philippe Laban, Divyansh Agarwal, Smaranda Muresan, and Chien-Sheng Wu. Art or artifice? large language models and the false promise of creativity. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24. Association for Computing Machinery, 2024. ISBN 9798400703300. doi: 10.1145/3613904.3642731. URL https://doi.org/10.1145/3613904.3642731.

Tuhin Chakrabarty, Philippe Laban, and Chien-Sheng Wu. AI-slop to AI-polish? aligning language models through edit-based writing rewards and test-time computation. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=jeDYcjuZIV.

Serina Chang, Ashton Anderson, and Jake M. Hofman. ChatBench: From static benchmarks to human-AI evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26009–26038. Association for Computational Linguistics, July 2025. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1262. URL https://aclanthology.org/2025.acl-long.1262/.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian,

Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Tianlang Chen, Shirley Wu, and Jure Leskovec. Found in conversation: Llms teach themselves to close the multi-turn gap. *arXiv preprint arXiv:2605.24432*, 2026.

Yi Cheng, Xiao Liang, Yeyun Gong, Wen Xiao, Song Wang, Yuji Zhang, Wenjun Hou, Kaishuai Xu, Wenge Liu, Wenjie Li, Jian Jiao, Qi Chen, Peng Cheng, and Wayne Xiong. Integrative decoding: Improve factuality via implicit self-consistency. In *International Conference on Learning Representations*, 2025.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2174–2184. Association for Computational Linguistics, October-November 2018. doi: 10.18653/v1/D18-1241. URL https://aclanthology.org/D18-1241/.

Eunsol Choi, Jennimaria Palomaki, Matthew Lamm, Tom Kwiatkowski, Dipanjan Das, and Michael Collins. Decontextualization: Making sentences stand-alone. *Transactions of the Association for Computational Linguistics*, 9:447–461, 2021. doi: 10.1162/tacl_a_00377. URL https://aclanthology.org/2021.tacl-1.27/.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Cursor. Dynamic context discovery. Cursor Blog, 2026. URL https://cursor.com/blog/dynamic-context-discovery.

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-v4: Towards highly efficient million-token context intelligence, 2026.

Kaustubh Deshpande, Ved Sirdeshmukh, Johannes Baptist Mols, Lifeng Jin, Ed-Yeremai Hernandez-Cardona, Dean Lee, Jeremy Kritz, Willow E. Primack, Summer Yue, and Chen Xing. MultiChallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18632–18702. Association for Computational Linguistics, July 2025. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.958. URL https://aclanthology.org/2025.findings-acl.958/.

Xuan Long Do, Bowei Zou, Liangming Pan, Nancy F. Chen, Shafiq Joty, and Ai Ti 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.

Haodong Duan, Jueqi Wei, Chonghua Wang, Hongwei Liu, Yixiao Fang, Songyang Zhang, Dahua Lin, and Kai Chen. BotChat: Evaluating LLMs' capabilities of having multi-turn dialogues. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 3184–3200. Association for Computational Linguistics, June 2024. doi: 10.18653/v1/2024.findings-naacl.201. URL https://aclanthology.org/2024.findings-naacl.201/.

Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Atri Rudra, James Y Zou, Azalia Mirhoseini, and Christopher Re. Cartridges: Lightweight and general-purpose long context representations via self-study. In *International Conference on Learning Representations*, volume 2026, pp. 42642–42687, 2026.

Zhiting Fan, Ruizhe Chen, Tianxiang Hu, and Zuozhu Liu. FairMT-bench: Benchmarking fairness for multi-turn dialogue in conversational LLMs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=RSGoXnS9GH.

Google. Gemini 3.1 Flash-Lite. Google AI for Developers, 2026. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.1-flash-lite.

Chi Han, Xin Liu, Haodong Wang, Shiyang Li, Jingfeng Yang, Haoming Jiang, Zhengyang Wang, Qingyu Yin, Liang Qiu, Changlong Yu, Yifan Gao, Zheng Li, Bing Yin, Jingbo Shang, and Heng Ji. Can language models follow multiple turns of entangled instructions? In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 25445–25460. Association for Computational Linguistics, November 2025. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.1387. URL https://aclanthology.org/2025.findings-emnlp.1387/.

Kunal Handa, Alex Tamkin, Miles McCain, Saffron Huang, Esin Durmus, Sarah Heck, Jared Mueller, Jerry Hong, Stuart Ritchie, Tim Belonax, Kevin K. Troy, Dario Amodei, Jared Kaplan, Jack Clark, and Deep Ganguli. Which economic tasks are performed with ai? evidence from millions of claude conversations. *ArXiv*, abs/2503.04761, 2025. URL https://api.semanticscholar.org/CorpusID:276885088.

Christine Herlihy, Jennifer Neville, Tobias Schnabel, and Adith Swaminathan. On overcoming miscalibrated conversational priors in LLM-based chatbots. In *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024. URL https://openreview.net/forum?id=iXQIgIsmIr.

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

Jenny Y. Huang, Leshem Choshen, Wei Sun, Omar Khattab, Ramón Fernandez Astudillo, Mehul Damani, Tamara Broderick, and Jacob Andreas. Do LLMs benefit from their own words? In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=l6T6MrLgWp.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Dongwei Jiang, Jingyu Zhang, Orion Weller, Nathaniel Weir, Benjamin Van Durme, and Daniel Khashabi. SELF-[IN]CORRECT: llms struggle with discriminating self-generated responses. In *Thirty-Ninth AAAI Conference on Artificial Intelligence, AAAI 2025*, pp. 24266–24275. AAAI Press, 2025. doi: 10.1609/AAAI.V39I23.34603. URL https://doi.org/10.1609/aaai.v39i23.34603.

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. 13358–13376, 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 (UncertaiNLP 2025)*, 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.

Yoonsu Kim, Kihoon Son, Seoyoung Kim, and Juho Kim. Beyond prompts: Learning from human communication for enhanced AI intent alignment. *CoRR*, abs/2405.05678, 2024. doi: 10.48550/ARXIV.2405.05678. URL https://doi.org/10.48550/arXiv.2405.05678.

Nils Knoth, Antonia Tolzin, Andreas Janson, and Jan Marco Leimeister. AI literacy and its implications for prompt engineering strategies. *Comput. Educ. Artif. Intell.*, 6:100225, 2024. doi: 10.1016/J.CAEAI.2024.100225. URL https://doi.org/10.1016/j.caeai.2024.100225.

Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. MT-eval: A multi-turn capabilities evaluation benchmark for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20153–20177. Association for Computational Linguistics, November 2024. doi: 10.18653/v1/2024.emnlp-main.1124. URL https://aclanthology.org/2024.emnlp-main.1124/.

Philippe Laban, Lidiya Murakhovs'ka, Caiming Xiong, and Chien-Sheng Wu. Are you sure? challenging llms leads to performance drops in the flipflop experiment. *ArXiv*, abs/2311.08596, 2023. URL https://api.semanticscholar.org/CorpusID:265213308.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. Llms get lost in multi-turn conversation. In *International Conference on Learning Representations*, volume 2026, pp. 54738–54778, 2026.

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.

Yoonjoo Lee, Kihoon Son, Tae Soo Kim, Jisu Kim, John Joon Young Chung, Eytan Adar, and Juho Kim. One vs. many: Comprehending accurate information from multiple erroneous and inconsistent ai generations. *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, 2024. URL https://api.semanticscholar.org/CorpusID:269635304.

David Lewis. Scorekeeping in a language game. *J. Philos. Log.*, 8(1):339–359, 1979. doi: 10.1007/BF00258436. URL https://doi.org/10.1007/BF00258436.

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*, 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.

Zhenwen Liang, Dian Yu, Wenhao Yu, Wenlin Yao, Zhihan Zhang, Xiangliang Zhang, and Dong Yu. Mathchat: Benchmarking mathematical reasoning and instruction following in multi-turn interactions. *arXiv preprint arXiv:2405.19444*, 2024.

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. *arXiv preprint arXiv:2508.21433*, 2025.

Geng Liu, Fei Zhu, Rong Feng, Changyi Ma, Shiqi Wang, and Gaofeng Meng. Intent mismatch causes llms to get lost in multi-turn conversation. *arXiv preprint arXiv:2602.07338*, 2026a.

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*, 2026b.

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

Shu Tong Luo, Wenqin Liu, Rui Liu, Mingming Gong, and Jiaxian Guo. Multi-turn reasoning when context arrives in pieces: Scalable sharding and memory-augmented rl. *arXiv preprint arXiv:2606.12941*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S37hOerQLB.

Chaitanya Malaviya, Joseph Chee Chang, Dan Roth, Mohit Iyyer, Mark Yatskar, and Kyle Lo. Contextualized evaluations: Judging language model responses to underspecified queries. *arXiv preprint arXiv:2411.07237*, 2025. URL https://arxiv.org/abs/2411.07237.

OpenAI. GPT-4o mini: Advancing cost-efficient intelligence. OpenAI Blog, 2024. URL https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

OpenAI. Introducing GPT-5.2. OpenAI Blog, 2025. URL https://openai.com/index/introducing-gpt-5-2/.

OpenAI. Introducing GPT-5.5. OpenAI Blog, 2026. URL https://openai.com/index/introducing-gpt-5-5/.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560, 2023. doi: 10.48550/ARXIV.2310.08560. URL https://doi.org/10.48550/arXiv.2310.08560.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky 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. Association for Computational Linguistics, August 2024. doi: 10.18653/v1/2024.findings-acl.57. URL https://aclanthology.org/2024.findings-acl.57/.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): from tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Hao Peng, Xiaozhi Wang, Jian-Hui Chen, Weikai Li, Yunjia Qi, Zimu Wang, Zhili Wu, Kaisheng Zeng, Bin Xu, Lei Hou, and Juan-Zi Li. When does in-context learning fall short and why? a study on specification-heavy tasks. *ArXiv*, abs/2311.08993, 2023. URL https://api.semanticscholar.org/CorpusID:265212914.

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. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019. doi: 10.1162/tacl_a_00266. URL https://aclanthology.org/Q19-1016/.

Rupak Sarkar, Bahareh Sarrafzadeh, Nirupama Chandrasekaran, Nagu Rangan, Philip Resnik, Longqi Yang, and Sujay Kumar Jauhar. Conversational user-ai intervention: A study on prompt rewriting for improved llm response generation. *ArXiv*, 2025. URL https://arxiv.org/abs/2503.16789.

Omar Shaikh, Hussein Mozannar, Gagan Bansal, Adam Fourney, and Eric Horvitz. Navigating rifts in human-LLM grounding: Study and benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20832–20847. Association for Computational Linguistics, July 2025. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1016. URL https://aclanthology.org/2025.acl-long.1016/.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2023.

Jyotika Singh, Fang Tu, Miguel Ballesteros, Weiyi Sun, Sandip Ghoshal, Michelle Yuan, Yassine Benajiba, Sujith Ravi, and Dan Roth. Mt-osc: Path for llms that get lost in multi-turn conversation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 27137–27160, 2026.

Akshit Sinha, Arvindh Arun, Shashwat Goel, Steffen Staab, and Jonas Geiping. The illusion of diminishing returns: Measuring long horizon execution in llms. In *International Conference on Learning Representations*, volume 2026, pp. 35184–35211, 2026.

Yuchong Sun, Che Liu, Kun Zhou, Jinwen Huang, Ruihua Song, Xin Zhao, Fuzheng Zhang, Di Zhang, and Kun Gai. Parrot: Enhancing multi-turn instruction following for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9729–9750. Association for Computational Linguistics, August 2024. doi: 10.18653/v1/2024.acl-long.525. URL https://aclanthology.org/2024.acl-long.525/.

Michael Terry, Chinmay Kulkarni, Martin Wattenberg, Lucas Dixon, and Meredith Ringel Morris. Interactive ai alignment: Specification, process, and evaluation alignment. *ArXiv*, 2024. URL https://arxiv.org/abs/2311.00710.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. MINT: Evaluating LLMs in multi-turn interaction with tools and language feedback. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=jp3gWrMuIZ.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Frank Wildenburg, Michael Hanna, and Sandro Pezzelle. Do pre-trained language models detect and understand semantic underspecification? ask the DUST! In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9598–9613. Association for Computational Linguistics, August 2024. doi: 10.18653/v1/2024.findings-acl.572. URL https://aclanthology.org/2024.findings-acl.572/.

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.

Chenxiao Yang, Nathan Srebro, David McAllester, and Zhiyuan Li. PENCIL: long thoughts with short memory. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver*, volume 267 of *Proceedings of Machine Learning Research*, 2025.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 3911–3921, 2018.

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.

J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. Why johnny can't prompt: How non-ai experts try (and fail) to design llm prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23. Association for Computing Machinery, 2023. ISBN 9781450394215. doi: 10.1145/3544548.3581388. URL https://doi.org/10.1145/3544548.3581388.

Alex L. Zhang, Tim Kraska, and Omar Khattab. Recursive language models. *CoRR*, 2026. URL https://arxiv.org/abs/2512.24601.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=uccHPGDlao.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. LMSYS-chat-1m: A large-scale real-world LLM conversation dataset. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=BOfDKxfwt0.

Ruiqi Zhong, Tao Yu, and Dan Klein. Semantic evaluation for text-to-SQL with distilled test suites. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 396–411. Association for Computational Linguistics, November 2020. doi: 10.18653/v1/2020.emnlp-main.29. URL https://aclanthology.org/2020.emnlp-main.29/.

George Kingsley Zipf. *Human behavior and the principle of least effort: An introduction to human ecology*. Ravenio books, 2016.

The appendix first reports the additional reset experiment and representative case studies. It then records the implementation and prompts used in the study, and closes with a discussion of the evidence and its scope.

A Additional Results: Reset Instruction

Table 2: Task-level accuracy for standard and reset variants. Parentheses on RC-R and IC-R show changes from RC and IC, respectively; Restart is the gray idealized reference.

Model	Method	Accuracy (%)				
		Math	Code	Database	Actions	Average
GPT-4o-mini	Sharded	57.28	53.00	51.40	49.52	52.80
	RC	68.93	58.00	59.81	87.62	68.59
	RC-R	75.73 (+6.80)	64.00 (+6.00)	64.49 (+4.68)	85.71 (-1.91)	72.48 (+3.89)
	IC	81.55	63.00	68.22	94.29	76.77
	IC-R	**85.44 (+3.89)**	**69.00 (+6.00)**	**71.96 (+3.74)**	**94.29 (0.00)**	**80.17 (+3.40)**
	Restart	88.35	67.00	89.72	94.29	84.84
Gemini-3.1-Flash-Lite	Sharded	70.59	73.00	52.34	52.38	62.08
	RC	87.25	83.00	83.18	97.14	87.64
	RC-R	87.25 (0.00)	89.00 (+6.00)	85.05 (+1.87)	98.10 (+0.96)	89.85 (+2.21)
	IC	90.29	93.00	88.79	100.00	93.02
	IC-R	**92.23 (+1.94)**	**95.00 (+2.00)**	**88.79 (0.00)**	**100.00 (0.00)**	**94.01 (+0.99)**
	Restart	95.15	90.00	93.46	100.00	94.65
DeepSeek-V4-Flash	Sharded	83.50	79.00	52.34	45.71	65.14
	RC	88.35	85.00	72.90	87.62	83.47
	RC-R	85.44 (-2.91)	88.00 (+3.00)	79.44 (+6.54)	93.33 (+5.71)	86.55 (+3.08)
	IC	90.29	93.00	80.37	92.38	89.01
	IC-R	**90.29 (0.00)**	**94.00 (+1.00)**	**84.11 (+3.74)**	**92.38 (0.00)**	**90.20 (+1.19)**
	Restart	85.44	88.00	89.72	94.29	89.36
GPT-5.2	Sharded	76.70	73.00	41.12	31.43	55.56
	RC	85.44	86.00	60.75	86.67	79.72
	RC-R	84.47 (-0.97)	89.00 (+3.00)	60.75 (0.00)	89.52 (+2.85)	80.94 (+1.22)
	IC	86.41	90.00	75.70	90.48	85.65
	IC-R	**91.26 (+4.85)**	**92.00 (+2.00)**	**73.83 (-1.87)**	**89.52 (-0.96)**	**86.65 (+1.00)**
	Restart	86.41	88.00	79.44	90.48	86.08
Qwen3.5-Plus	Sharded	78.64	82.00	54.21	41.90	64.19
	RC	85.44	89.00	79.44	91.43	86.33
	RC-R	93.20 (+7.76)	94.00 (+5.00)	82.24 (+2.80)	94.29 (+2.86)	90.93 (+4.60)
	IC	93.20	93.00	83.18	94.29	90.92
	IC-R	**94.17 (+0.97)**	**95.00 (+2.00)**	**93.46 (+10.28)**	**95.24 (+0.95)**	**94.47 (+3.55)**
	Restart	97.09	96.00	96.26	95.24	96.15
Claude-Sonnet-4-6	Sharded	79.41	82.00	50.47	55.24	66.78
	RC	90.29	89.00	59.81	96.19	83.82
	RC-R	87.25 (-3.04)	89.00 (0.00)	61.68 (+1.87)	94.29 (-1.90)	83.06 (-0.76)
	IC	88.24	91.00	68.22	95.24	85.68
	IC-R	**90.29 (+2.05)**	**90.00 (-1.00)**	**65.42 (-2.80)**	**93.33 (-1.91)**	**84.76 (-0.92)**
	Restart	93.20	94.00	87.85	99.05	93.53
GPT-5.5	Sharded	79.61	75.00	56.07	45.71	64.10
	RC	89.32	85.00	71.96	86.67	83.24
	RC-R	91.26 (+1.94)	87.00 (+2.00)	75.70 (+3.74)	86.67 (0.00)	85.16 (+1.92)
	IC	91.26	95.00	85.98	87.62	89.97
	IC-R	**90.29 (-0.97)**	**96.00 (+1.00)**	**77.57 (-8.41)**	**87.62 (0.00)**	**87.87 (-2.10)**
	Restart	95.15	89.00	86.92	87.62	89.67

Turn isolation changes which earlier response remains visible, while a reset instruction changes how the model is asked to use that response. We evaluate the two interventions separately by adding the same soft-reset instruction to RC and IC.

Table 2 reports the full per-task scores for Sharded, RC, RC-R, IC, IC-R, and Restart. Changes for RC-R and IC-R are measured against RC and IC, respectively, so the table shows both the overall comparison and the effect of adding the reset instruction.

Soft reset improves average recovery, but its effect depends on the model. The instruction raises RC from 81.83 to 84.14 and IC from 87.29 to 88.30. IC-R has the highest overall average among the evaluated history-conditioned methods and exceeds RC-R by 4.16 points. It also outperforms RC-R in 24 model–task comparisons, ties in one, and is lower in three.

Reset complements turn isolation by controlling how the model uses the response that remains visible. It improves RC for six of the seven models and IC for five. For Claude-Sonnet-4-6, reset changes RC by -0.76 points and IC by -0.92 points; for GPT-5.5, it improves RC by 1.92 points while changing IC by -2.10 points. Across tasks, IC-R improves Math, Code, and Database, with

a small reduction on Actions. Together, these results separate two decisions in recovery: IC determines which earlier response the model sees, while reset changes how the model continues from that response.

B Qualitative Case Studies

This appendix directly reproduces three GPT-4o-mini examples from the saved experiment records. We retain the original wording of every system, user, and assistant message, the completed-task recap, and every generated IC and full-history candidate.

B.1 Database: Removing Columns Inherited from Earlier Answers

The following transcript is reproduced verbatim from the saved run.

Standard multi-turn conversation

```
===== SYSTEM =====
You are helping a user write SQL queries to a database. If something is
↪  not clear, you can ask the user to clarify what they need. The
↪  schema for the database being accessed is the following:

CREATE TABLE stadium (
    Stadium_ID INT   PRIMARY KEY,
    Location TEXT,
    Name TEXT,
    Capacity INT,
    Highest INT,
    Lowest INT,
    Average INT,
    UNIQUE (Stadium_ID)
);

CREATE TABLE singer (
    Singer_ID INT   PRIMARY KEY,
    Name TEXT,
    Country TEXT,
    Song_Name TEXT,
    Song_release_year TEXT,
    Age INT,
    Is_male bool,
    UNIQUE (Singer_ID)
);

CREATE TABLE concert (
    concert_ID INT   PRIMARY KEY,
    concert_Name TEXT,
    Theme TEXT,
    Stadium_ID TEXT,
    Year TEXT,
    UNIQUE (concert_ID),
    FOREIGN KEY (Stadium_ID) REFERENCES stadium(Stadium_ID)
);

CREATE TABLE singer_in_concert (
    concert_ID INT   PRIMARY KEY,
    Singer_ID TEXT   PRIMARY KEY,
    PRIMARY KEY (concert_ID, Singer_ID),
    UNIQUE (concert_ID, Singer_ID),
    FOREIGN KEY (Singer_ID) REFERENCES singer(Singer_ID),
    FOREIGN KEY (concert_ID) REFERENCES concert(concert_ID)
);

===== USER 1 =====
```

````
datasheet of singers

===== ASSISTANT 1 =====
It seems you're looking for the data or structure regarding the singers
↪  in the database. To retrieve the datasheet of singers, you can use
↪  the following SQL query:

```sql
SELECT * FROM singer;
```

This query will return all the records and fields in the `singer`
↪  table, including `Singer_ID`, `Name`, `Country`, `Song_Name`,
↪  `Song_release_year`, `Age`, and `Is_male`.

If you need specific details or a subset of the data, please clarify!

===== USER 2 =====
the list should be sorted by age

===== ASSISTANT 2 =====
To retrieve the datasheet of singers sorted by age, you can use the
↪  following SQL query:

```sql
SELECT * FROM singer
ORDER BY Age;
```

This query will return all records from the `singer` table sorted in
↪  ascending order by the `Age` column. If you would like the results
↪  in descending order (from oldest to youngest), you can modify the
↪  query like this:

```sql
SELECT * FROM singer
ORDER BY Age DESC;
```

Let me know if you need further assistance!

===== USER 3 =====
I want the list ordered from oldest to youngest.

===== ASSISTANT 3 =====
To retrieve the list of singers ordered from oldest to youngest, you
↪  can use the following SQL query:

```sql
SELECT * FROM singer
ORDER BY Age DESC;
```

This query will return all records from the `singer` table sorted in
↪  descending order by the `Age` column, thus listing the oldest
↪  singers first. If you have any more requests or need further
↪  assistance, feel free to ask!

===== USER 4 =====
I also want to see the countries of the singers in the list.

===== ASSISTANT 4 =====
````

````
The query I provided earlier already includes the `Country` column, as
↪  it selects all columns from the `singer` table. To ensure that the
↪  list includes all relevant information, here's the same query
↪  explicitly listing the fields you want, including the country of
↪  each singer:

```sql
SELECT Singer_ID, Name, Country, Song_Name, Song_release_year, Age,
↪  Is_male
FROM singer
ORDER BY Age DESC;
```

This query will return the `Singer_ID`, `Name`, `Country`, `Song_Name`,
↪  `Song_release_year`, `Age`, and `Is_male` of each singer, ordered
↪  from oldest to youngest. Let me know if you need any other
↪  modifications or additional information!

===== USER 5 =====
also include the names of the singers in the details.

===== ASSISTANT 5 =====
The previous query already includes the names of the singers as it
↪  selects the `Name` column. Here's a reiteration of the SQL query
↪  with all relevant details, including singer names:

```sql
SELECT Singer_ID, Name, Country, Song_Name, Song_release_year, Age,
↪  Is_male
FROM singer
ORDER BY Age DESC;
```

This query will display the following columns for each singer:

- `Singer_ID`
- `Name`
- `Country`
- `Song_Name`
- `Song_release_year`
- `Age`
- `Is_male`

If you need anything else or further modifications, please let me know!

===== USER 6 =====
also want to see the ages of the singers in the list.

===== ASSISTANT 6 =====
The query I provided earlier already includes the ages of the singers,
↪  as it selects the `Age` column. However, to emphasize that the ages
↪  will be visible in the results, here's the SQL query again:

```sql
SELECT Singer_ID, Name, Country, Song_Name, Song_release_year, Age,
↪  Is_male
FROM singer
ORDER BY Age DESC;
```

This query will return:

- `Singer_ID`
- `Name`
- `Country`
- `Song_Name`
````

```
- `Song_release_year`
- `Age` (which shows the ages of the singers)
- `Is_male`

This will list all the required details, including the names and ages
↪  of the singers ordered from oldest to youngest. If you have any
↪  other requests or need further clarification, feel free to ask!
```

COMPLETED-TASK RECAP SUPPLIED TO RECOVERY

```
Just to recapitulate, the entire task is:

You must generate the valid SQL query that answers the given user query
↪  in natural language.

The schema of the database you are responding to is as follows:
CREATE TABLE stadium (
    Stadium_ID INT   PRIMARY KEY,
    Location TEXT,
    Name TEXT,
    Capacity INT,
    Highest INT,
    Lowest INT,
    Average INT,
    UNIQUE (Stadium_ID)
);

CREATE TABLE singer (
    Singer_ID INT   PRIMARY KEY,
    Name TEXT,
    Country TEXT,
    Song_Name TEXT,
    Song_release_year TEXT,
    Age INT,
    Is_male bool,
    UNIQUE (Singer_ID)
);

CREATE TABLE concert (
    concert_ID INT   PRIMARY KEY,
    concert_Name TEXT,
    Theme TEXT,
    Stadium_ID TEXT,
    Year TEXT,
    UNIQUE (concert_ID),
    FOREIGN KEY (Stadium_ID) REFERENCES stadium(Stadium_ID)
);

CREATE TABLE singer_in_concert (
    concert_ID INT   PRIMARY KEY,
    Singer_ID TEXT   PRIMARY KEY,
    PRIMARY KEY (concert_ID, Singer_ID),
    UNIQUE (concert_ID, Singer_ID),
    FOREIGN KEY (Singer_ID) REFERENCES singer(Singer_ID),
    FOREIGN KEY (concert_ID) REFERENCES concert(concert_ID)
);

The user query is:
Consider all the following:
- datasheet of singers
- ensure the list is ordered by their age
- start ordering the singers with the oldest first
- include the singers' countries in the details
- include the names of the singers as part of the details
- include the ages of the singers in the list
```

```
Rules:
- [Single Answer] Produce a single valid SQL query based on the
↪  database schema and the user query.
- [Only SQL] Do not include any other text or comments.
- [Validity] The SQL query must be valid and executable.
- [Single Line] Produce your entire response in a single line.

Now please complete the task correctly taking all the information into
↪  account.
```

ISOLATED CONSISTENCY CANDIDATES

Every view uses the system message, user messages, and recap printed above. The indicated assistant turn remains visible in its original slot; every other assistant response is replaced by `[MASKED]`.

IC view 1: retain Assistant 1. Recorded evaluator outcome: Correct (✓).

```
```sql
SELECT Name, Country, Age FROM singer ORDER BY Age DESC;
```
```

IC view 2: retain Assistant 2. Recorded evaluator outcome: Correct (✓).

```
```sql
SELECT Name, Country, Age FROM singer ORDER BY Age DESC;
```
```

IC view 3: retain Assistant 3. Recorded evaluator outcome: Correct (✓).

```
```sql
SELECT Name, Country, Age FROM singer ORDER BY Age DESC;
```
```

IC view 4: retain Assistant 4. Recorded evaluator outcome: Incorrect (×).

```
```sql
SELECT Singer_ID, Name, Country, Song_Name, Song_release_year, Age,
↪  Is_male FROM singer ORDER BY Age DESC;
```
```

IC view 5: retain Assistant 5. Recorded evaluator outcome: Incorrect (×).

```
```sql
SELECT Singer_ID, Name, Country, Song_Name, Song_release_year, Age,
↪  Is_male FROM singer ORDER BY Age DESC;
```
```

IC view 6: retain Assistant 6. Recorded evaluator outcome: Incorrect (×).

```
```sql
SELECT Singer_ID, Name, Country, Age FROM singer ORDER BY Age DESC;
```
```

FULL-HISTORY VOTING CANDIDATES

Each candidate receives the complete standard transcript and the same recap. The outputs below are reproduced in vote-index order.

Full-history candidate 1. Recorded evaluator outcome: Incorrect (×).

```
SELECT Singer_ID, Name, Country, Song_Name, Song_release_year, Age,
↪  Is_male FROM singer ORDER BY Age DESC;
```

Full-history candidate 2. Recorded evaluator outcome: Incorrect (×).

```
SELECT Singer_ID, Name, Country, Song_Name, Song_release_year, Age,
↪  Is_male FROM singer ORDER BY Age DESC;
```

Full-history candidate 3. Recorded evaluator outcome: Incorrect (×).

````
```sql
SELECT Singer_ID, Name, Country, Song_Name, Song_release_year, Age,
↪  Is_male FROM singer ORDER BY Age DESC;
```
````

Full-history candidate 4. Recorded evaluator outcome: Incorrect (×).

```
SELECT Singer_ID, Name, Country, Song_Name, Song_release_year, Age,
↪  Is_male FROM singer ORDER BY Age DESC;
```

Full-history candidate 5. Recorded evaluator outcome: Incorrect (×).

````
```sql
SELECT Singer_ID, Name, Country, Song_Name, Song_release_year, Age,
↪  Is_male FROM singer ORDER BY Age DESC;
```
````

Full-history candidate 6. Recorded evaluator outcome: Incorrect (×).

````
```sql
SELECT Singer_ID, Name, Country, Song_Name, Song_release_year, Age,
↪  Is_male FROM singer ORDER BY Age DESC;
```
````

RECORDED VOTE OUTCOME

IC selects the three-column query with three votes. Full-history voting selects the seven-column query with six votes.

B.2 ACTIONS: CORRECTING A PROPAGATED TOOL PARAMETER

The following transcript is reproduced verbatim from the saved run.

STANDARD MULTI-TURN CONVERSATION

```
===== SYSTEM =====
You are an expert in composing functions. You are given a question and
↪  a set of possible functions. Based on the question, you will need
↪  to make one or more function/tool calls to achieve the purpose.
If none of the functions can be used, point it out. If the given
↪  question lacks the parameters required by the function, also point
↪  it out.
You should only return the function calls in your response.

If you decide to invoke any of the function(s), you MUST put it in the
↪  format of [func_name1(params_name1=params_value1,
↪  params_name2=params_value2...), func_name2(params)]
You SHOULD NOT include any other text in the response.

Here is a list of functions in JSON format that you can invoke.
```

```
[{'name': 'calculate_present_value', 'description': 'Calculate the
↪  present value of a future cash flows stream.', 'parameters':
↪  {'type': 'dict', 'properties': {'payment_per_year': {'type':
↪  'integer', 'description': 'The payment received per year.'},
↪  'interest_rate': {'type': 'float', 'description': 'The interest
↪  rate applied per period.'}, 'years': {'type': 'integer',
↪  'description': 'The total number of years.'}}, 'required':
↪  ['payment_per_year', 'interest_rate', 'years']}}]

===== USER 1 =====
Calculate the Present Value of an investment

===== ASSISTANT 1 =====
The question lacks the parameters required by the function, such as
↪  payment_per_year, interest_rate, and years.

===== USER 2 =====
The investment pays $1000 each year.

===== ASSISTANT 2 =====
The question still lacks the parameters required by the function,
↪  specifically the interest_rate and years.

===== USER 3 =====
The interest rate is 5%

===== ASSISTANT 3 =====
The question still lacks the parameters required by the function,
↪  specifically the years.

===== USER 4 =====
Calculate it for a duration of 10 years

===== ASSISTANT 4 =====
[calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=10)]

===== USER 5 =====
Also want to calculate it for 20 years.

===== ASSISTANT 5 =====
[calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=10), calculate_present_value(payment_per_year=1000,
↪  interest_rate=5.0, years=20)]

===== USER 6 =====
And finally, I want to calculate it for 30 years.

===== ASSISTANT 6 =====
[calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=10), calculate_present_value(payment_per_year=1000,
↪  interest_rate=5.0, years=20),
↪  calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=30)]

===== USER 7 =====
Please return all these scenarios in one dashboard.

===== ASSISTANT 7 =====
The question requests a format or function that isn't provided in the
↪  available functions. Thus, I cannot fulfill this request.
```

COMPLETED-TASK RECAP SUPPLIED TO RECOVERY

```
Just to recapitulate, the entire task is:
```

```
- Calculate the Present Value of an investment
- The investment pays $1000 per year
- The interest rate is 5%
- Calculate it for a duration of 10 years
- Also, calculate it for 20 years
- Finally, calculate it for 30 years
- return all these scenarios in one dashboard


Now please complete the task correctly taking all the information into
↪  account.
```

ISOLATED CONSISTENCY CANDIDATES

Every view uses the system message, user messages, and recap printed above. The indicated assistant turn remains visible in its original slot; every other assistant response is replaced by `[MASKED]`.

IC view 1: retain Assistant 1. Recorded evaluator outcome: Correct (✓).

```
[calculate_present_value(payment_per_year=1000, interest_rate=0.05,
↪  years=10), calculate_present_value(payment_per_year=1000,
↪  interest_rate=0.05, years=20),
↪  calculate_present_value(payment_per_year=1000, interest_rate=0.05,
↪  years=30)]
```

IC view 2: retain Assistant 2. Recorded evaluator outcome: Correct (✓).

```
[calculate_present_value(payment_per_year=1000, interest_rate=0.05,
↪  years=10), calculate_present_value(payment_per_year=1000,
↪  interest_rate=0.05, years=20),
↪  calculate_present_value(payment_per_year=1000, interest_rate=0.05,
↪  years=30)]
```

IC view 3: retain Assistant 3. Recorded evaluator outcome: Correct (✓).

```
[calculate_present_value(payment_per_year=1000, interest_rate=0.05,
↪  years=10), calculate_present_value(payment_per_year=1000,
↪  interest_rate=0.05, years=20),
↪  calculate_present_value(payment_per_year=1000, interest_rate=0.05,
↪  years=30)]
```

IC view 4: retain Assistant 4. Recorded evaluator outcome: Incorrect (×).

```
[calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=10), calculate_present_value(payment_per_year=1000,
↪  interest_rate=5.0, years=20),
↪  calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=30)]
```

IC view 5: retain Assistant 5. Recorded evaluator outcome: Correct (✓).

```
[calculate_present_value(payment_per_year=1000, interest_rate=0.05,
↪  years=10), calculate_present_value(payment_per_year=1000,
↪  interest_rate=0.05, years=20),
↪  calculate_present_value(payment_per_year=1000, interest_rate=0.05,
↪  years=30)]
```

IC view 6: retain Assistant 6. Recorded evaluator outcome: Incorrect (×).

```
[calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=10), calculate_present_value(payment_per_year=1000,
↪  interest_rate=5.0, years=20),
↪  calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=30)]
```

IC view 7: retain Assistant 7. Recorded evaluator outcome: Correct (✓).

```
[calculate_present_value(payment_per_year=1000, interest_rate=0.05,
↪  years=10), calculate_present_value(payment_per_year=1000,
↪  interest_rate=0.05, years=20),
↪  calculate_present_value(payment_per_year=1000, interest_rate=0.05,
↪  years=30)]
```

FULL-HISTORY VOTING CANDIDATES

Each candidate receives the complete standard transcript and the same recap. The outputs below are reproduced in vote-index order.

Full-history candidate 1. Recorded evaluator outcome: Incorrect (×).

```
[calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=10), calculate_present_value(payment_per_year=1000,
↪  interest_rate=5.0, years=20),
↪  calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=30)]
```

Full-history candidate 2. Recorded evaluator outcome: Incorrect (×).

```
[calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=10), calculate_present_value(payment_per_year=1000,
↪  interest_rate=5.0, years=20),
↪  calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=30)]
```

Full-history candidate 3. Recorded evaluator outcome: Incorrect (×).

```
[calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=10), calculate_present_value(payment_per_year=1000,
↪  interest_rate=5.0, years=20),
↪  calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=30)]
```

Full-history candidate 4. Recorded evaluator outcome: Incorrect (×).

```
[calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=10), calculate_present_value(payment_per_year=1000,
↪  interest_rate=5.0, years=20),
↪  calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=30)]
```

Full-history candidate 5. Recorded evaluator outcome: Incorrect (×).

```
[calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=10), calculate_present_value(payment_per_year=1000,
↪  interest_rate=5.0, years=20),
↪  calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=30)]
```

Full-history candidate 6. Recorded evaluator outcome: Incorrect (×).

```
[calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=10), calculate_present_value(payment_per_year=1000,
↪  interest_rate=5.0, years=20),
↪  calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=30)]
```

Full-history candidate 7. Recorded evaluator outcome: Incorrect (×).

```
[calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=10), calculate_present_value(payment_per_year=1000,
↪  interest_rate=5.0, years=20),
↪  calculate_present_value(payment_per_year=1000, interest_rate=5.0,
↪  years=30)]
```

RECORDED VOTE OUTCOME

IC selects the calls with `interest_rate=0.05` by five votes to two. Full-history voting selects `interest_rate=5.0` with seven votes.

B.3 MATH: CORRECTING AN INHERITED DISPLACEMENT ERROR

The following transcript is reproduced verbatim from the saved run.

STANDARD MULTI-TURN CONVERSATION

```
===== SYSTEM =====
As an expert problem solver solve step by step the following
↪  mathematical questions.

===== USER 1 =====
how far is Elvis from his house?

===== ASSISTANT 1 =====
To answer the question "How far is Elvis from his house?", I need more
↪  specific details or context.

1. **Identify "Elvis":** Are we referring to a specific Elvis (e.g.,
↪  Elvis Presley, a character named Elvis, or someone you know)?

2. **Define the locations:** What are the two points we are comparing?
↪  The location of Elvis and the location of his house.

3. **Method of measurement:** How is the distance being measured (e.g.,
↪  straight line, walking distance, driving distance)?

If you provide more information, I can definitely help you work through
↪  it step by step!

===== USER 2 =====
Elvis starts his trip from his house.

===== ASSISTANT 2 =====
To determine how far Elvis is from his house on his trip, we need to
↪  establish the specifics of the trip. Here is a step-by-step
↪  framework that we can use to solve this:

1. **Define the Starting Point:**
   - Elvis starts from his house, which is considered the initial point
   ↪  (Point A).

2. **Define the Destination:**
   - We need to know where Elvis is going (Point B). This could be a
   ↪  specific address, landmark, or distance.

3. **Traveling Method:**
   - Determine how Elvis is traveling (e.g., walking, driving, etc.),
   ↪  as this can affect the total distance if there are specific
   ↪  routes to consider.

4. **Calculate the Distance:**
   - If the destination is known, we can either:
```

```
     - Measure the straight-line distance using coordinates (if
     ↪  available).
     - Use a mapping service or GPS to find the distance based on the
     ↪  route taken.
     - If the distance covered while traveling is given (e.g.,
     ↪  miles/kilometers), that can be used directly.

5. **Final Assessment:**
   - The distance Elvis is from his house will be equal to the distance
   ↪  traveled to his destination.

For a conclusive answer, please provide the following details:
- The destination Elvis is traveling to.
- The travel method or distance if already known.

With this information, we can accurately determine how far Elvis is
↪  from his house.

===== USER 3 =====
Elvis drives west for 5 hours.

===== ASSISTANT 3 =====
To calculate how far Elvis is from his house after driving west for 5
↪  hours, we need to consider the speed at which he is traveling.
↪  Here's a step-by-step approach:

1. **Determine Elvis's Speed:**
   - Let's assume a typical driving speed. For example, if Elvis drives
   ↪  at an average speed of 60 miles per hour (mph), we can calculate
   ↪  the distance based on that. If you have a different speed in
   ↪  mind, please let me know.

2. **Calculate the Distance Traveled:**
   - The formula to determine distance is:
     \[
     \text{Distance} = \text{Speed} \times \text{Time}
     \]
   - If we assume he is driving at 60 mph:
     \[
     \text{Distance} = 60 \, \text{miles/hour} \times 5 \, \text{hours}
     ↪  = 300 \, \text{miles}
     \]
   - Therefore, if he drives at that rate, he would be **300 miles**
   ↪  from his house after 5 hours.

3. **Final Result:**
   - After driving west for 5 hours at an average speed of 60 mph,
   ↪  Elvis is **300 miles** away from his house.

If you have a different speed in mind or if there are additional
↪  factors (like traffic, rest stops, etc.), please let me know so we
↪  can adjust the calculations accordingly!

===== USER 4 =====
After that, he turns around to change direction.

===== ASSISTANT 4 =====
Once Elvis has driven for 5 hours and covered a distance of 300 miles,
↪  if he turns around to change direction and heads back towards his
↪  house, we can clarify what will happen next.

### Step-by-Step Calculation:

1. **Initial Distance from House:**
   - After driving 300 miles west, he is 300 miles away from his house.
```

```
2. **Turn Around:**
   - By turning around, he begins to drive back towards his house in
   ↪  the opposite direction (east).

3. **Determine the Time/Distance for Return:**
   - If he continues to drive at the same average speed of 60 mph when
   ↪  returning, we can calculate the time it will take him to return
   ↪  home:
   \[
   \text{Time to Return} = \frac{\text{Distance}}{\text{Speed}} =
   ↪  \frac{300 \text{ miles}}{60 \text{ miles/hour}} = 5 \text{
   ↪  hours}
   \]

4. **Return Journey:**
   - It will take him another 5 hours to return to his house, driving
   ↪  the same distance of 300 miles back east.

5. **Final Distance from House After Return:**
   - Once he completes his return journey (after 5 hours of driving
   ↪  back), he will arrive at his house, making him **0 miles** away
   ↪  from his house.

### Summary:
- After driving for 5 hours west (300 miles away from his house) and
↪  then turning around, it will take him another 5 hours to return
↪  home. Thus, when he arrives back, he will be at **his house**, or
↪  **0 miles** away from it.

If there are any other details or further questions about Elvis's
↪  journey, feel free to ask!

===== USER 5 =====
He also drives east for 8 hours after turning around.

===== ASSISTANT 5 =====
Got it! Let's analyze the situation step by step based on Elvis driving
↪  east for 8 hours after turning around.

### Starting Point:
1. **Initial Distance from House:**
   - After driving west for 5 hours, Elvis is **300 miles** away from
   ↪  his house.

2. **Turning Around:**
   - He turns around and starts driving back east towards his house. As
   ↪  previously calculated, it will take him **5 hours** to return
   ↪  home, covering another **300 miles** back.

3. **Returning Home:**
   - By the end of those 5 hours of driving east, he will arrive back
   ↪  at his house, which is **0 miles** away from his previous
   ↪  position.

### Driving East for 8 More Hours:
4. **Continuing East:**
   - After reaching his house, he then drives **east for another 8
   ↪  hours**. Assuming he continues at the same speed of **60 mph**:
   \[
   \text{Distance} = \text{Speed} \times \text{Time} = 60 \text{
   ↪  miles/hour} \times 8 \text{ hours} = 480 \text{ miles}
   \]

### Final Distance from House:
5. **Final Position:**
```

```
   - After driving **480 miles** east from his home, Elvis is now **480
   ↪  miles** away from his house.

### Summary:
- Elvis drove back to his house (0 miles away) and then drove east for
↪  8 hours at 60 mph, resulting in a new distance of **480 miles**
↪  east from his house.

If you have any further questions or need additional calculations, feel
↪  free to ask!

===== USER 6 =====
He was going pretty slow overall, average speed was just 18 mph during
↪  the whole trip.

===== ASSISTANT 6 =====
Thank you for providing the average speed of 18 mph for the entire
↪  trip. Let's reevaluate Elvis's trip using this information.

### Step-by-Step Breakdown:

1. **Initial Trip West:**
   - **Time:** 5 hours
   - **Distance traveled west:**
     \[
     \text{Distance} = \text{Speed} \times \text{Time} = 18 \text{
     ↪  miles/hour} \times 5 \text{ hours} = 90 \text{ miles}
     \]

2. **Elapsed Distance:**
   - After driving west for 5 hours, Elvis is **90 miles** away from
   ↪  his house.

3. **Turning Around:**
   - Elvis turns around and drives back east towards his house. Since
   ↪  he has to cover the same distance to return, he drives 90 miles
   ↪  back east.

4. **Returning Home:**
   - To calculate the time taken to return home at the same average
   ↪  speed of 18 mph:
   \[
   \text{Time to Return} = \frac{90 \text{ miles}}{18 \text{
   ↪  miles/hour}} = 5 \text{ hours}
   \]

5. **Distance After Returning:**
   - After 5 hours of driving back east, he reaches his house, which is
   ↪  **0 miles** away from his initial position.

### Driving East for 8 More Hours:
6. **Continuing East:**
   - After reaching his house, Elvis drives east for another 8 hours at
   ↪  18 mph:
   \[
   \text{Distance} = \text{Speed} \times \text{Time} = 18 \text{
   ↪  miles/hour} \times 8 \text{ hours} = 144 \text{ miles}
   \]

### Final Distance from House:
7. **Final Position:**
   - After driving **144 miles** east from his house, Elvis is now
   ↪  **144 miles** away from his house.

### Summary:
```

```
- Elvis was initially 90 miles west of his house after the first 5-hour
↪  drive. He then returned home (0 miles away) and subsequently drove
↪  east for 8 hours at an average speed of 18 mph, resulting in a
↪  final distance of **144 miles** east of his house.

If you have any additional questions or need further calculations, feel
↪  free to ask!
```

COMPLETED-TASK RECAP SUPPLIED TO RECOVERY

```
Just to recapitulate, the entire task is:

Q: - how far is Elvis from his house?
- Elvis begins his journey from his house.
- he drives west for 5 hours.
- after that, he turns around to change direction.
- then, he drives east for 8 hours.
- his average speed was 18 mph during the entire trip.

A:

Now please complete the task correctly taking all the information into
↪  account.
```

ISOLATED CONSISTENCY CANDIDATES

Every view uses the system message, user messages, and recap printed above. The indicated assistant turn remains visible in its original slot; every other assistant response is replaced by `[MASKED]`.

IC view 1: retain Assistant 1. Recorded evaluator outcome: Correct (✓).

```
To determine how far Elvis is from his house, we will calculate the
↪  total distance he traveled in both directions and then find his
↪  final position relative to his starting point (his house).

### Step-by-step Calculation

1. **Calculate the distance traveled west:**
   - Elvis drives west for **5 hours**.
   - Average speed = **18 mph**.
   - Distance = Speed × Time = 18 mph × 5 hours = **90 miles**.

2. **Calculate the distance traveled east:**
   - Elvis drives east for **8 hours**.
   - Average speed = **18 mph**.
   - Distance = Speed × Time = 18 mph × 8 hours = **144 miles**.

3. **Determine his position relative to his starting point:**
   - After the westward trip, Elvis is **90 miles west** of his house.
   - After turning around and driving east for 144 miles, we can
   ↪  calculate his final position.
   - Starting from 90 miles west, he drives 144 miles east:
     \[
     144 \text{ miles (east)} - 90 \text{ miles (west)} = 54 \text{
     ↪  miles (east)}.
     \]

### Conclusion
Elvis is **54 miles east** of his house after completing his trip.
```

IC view 2: retain Assistant 2. Recorded evaluator outcome: Correct (✓).

```
To determine how far Elvis is from his house after his entire trip, we
↪  will follow these steps:
```

```
1. **Calculate the distance traveled west:**
   - Distance = Speed × Time
   - For the westward trip:
     - Speed = 18 mph
     - Time = 5 hours
   - Distance west = 18 mph × 5 hours = 90 miles

2. **Calculate the distance traveled east:**
   - For the eastward trip:
     - Speed = 18 mph
     - Time = 8 hours
   - Distance east = 18 mph × 8 hours = 144 miles

3. **Determine the net distance from the house:**
   - Elvis first drives 90 miles west and then drives 144 miles back
   ↪  east.
   - The net distance from his house is calculated as:
   - Net distance = Distance traveled east - Distance traveled west
   - Net distance = 144 miles - 90 miles = 54 miles

**Conclusion:** Elvis is 54 miles east of his house after completing
↪  his trip.
```

IC view 3: retain Assistant 3. Recorded evaluator outcome: Correct (✓).

```
Sure! Let's break down the trip step by step, taking into account all
↪  the details provided.

1. **Trip Details:**
   - **Initial position:** Elvis starts at his house (point A).
   - **First leg:** Drives west for 5 hours.
   - **Second leg:** Turns around and drives east for 8 hours.
   - **Average speed for the entire trip:** 18 mph.

2. **Calculate the total distance traveled:**
   - We need to calculate how far Elvis traveled during each leg of the
   ↪  trip.

   **First Leg (Driving West):**
   - Speed: 18 mph
   - Time: 5 hours
   - Distance traveled west:
     \[
     \text{Distance}_\text{west} = \text{Speed} \times \text{Time} = 18
     ↪  \, \text{mph} \times 5 \, \text{hours} = 90 \, \text{miles}
     \]

   **Second Leg (Driving East):**
   - Speed: 18 mph
   - Time: 8 hours
   - Distance traveled east:
     \[
     \text{Distance}_\text{east} = \text{Speed} \times \text{Time} = 18
     ↪  \, \text{mph} \times 8 \, \text{hours} = 144 \, \text{miles}
     \]

3. **Determine Elvis's final position relative to his house:**
   - After driving west, Elvis is 90 miles away from his house.
   - He then turns around and drives east for 144 miles.

4. **Net Displacement:**
   - After driving east, we need to determine his total displacement
   ↪  from the house by subtracting the distance traveled west from
   ↪  the distance traveled east:
     \[
```

```
     \text{Net Displacement} = \text{Distance}_\text{east} -
     ↪  \text{Distance}_\text{west} = 144 \, \text{miles} - 90 \,
     ↪  \text{miles} = 54 \, \text{miles}
     \]

Thus, Elvis ends up **54 miles east** from his house after completing
↪  the entire trip.
```

IC view 4: retain Assistant 4. Recorded evaluator outcome: Correct (✓).

```
Let's break down Elvis's journey step by step to determine how far he
↪  is from his house:

### Step 1: Calculate Distance Driven West

1. **Average Speed West:**
   - Let's assume that his average speed while he drives west was **18
   ↪  mph** (to simplify, we will use the overall average speed for
   ↪  the entire trip, as the specific speeds for each leg are not
   ↪  given).

2. **Time Driving West:**
   - He drives west for **5 hours**.

3. **Distance Driven West:**
   \[
   \text{Distance}_{\text{west}} = \text{Speed} \times \text{Time} = 18
   ↪  \, \text{mph} \times 5 \, \text{hours} = 90 \, \text{miles}
   \]

### Step 2: Turn Around

After driving **90 miles west**, he turns around.

### Step 3: Calculate Distance Driven East

1. **Time Driving East:**
   - He then drives east for **8 hours**.

2. **Distance Driven East:**
   \[
   \text{Distance}_{\text{east}} = \text{Speed} \times \text{Time} = 18
   ↪  \, \text{mph} \times 8 \, \text{hours} = 144 \, \text{miles}
   \]

### Step 4: Determine his Position Relative to Home

- After driving **90 miles west**, he is **90 miles** away from his
↪  house.
- After turning around and driving **144 miles east**, we can find his
↪  position relative to his house:

### Net Distance Calculation
- He initially drove **90 miles west** and then drove **144 miles
↪  east**:
\[
\text{Net Distance from House} = \text{Distance East} - \text{Distance
↪  West} = 144 \, \text{miles} - 90 \, \text{miles} = 54 \,
↪  \text{miles}
\]

### Final Answer

Thus, **Elvis is 54 miles east from his house.**
```

IC view 5: retain Assistant 5. Recorded evaluator outcome: Correct (✓).

```
Let's solve the problem step by step, taking into account all the
↪  information provided, including the average speed.

### Step 1: Calculate Total Distance Traveled

1. **Total Trip Time**:
   - Elvis drives west for 5 hours and then east for 8 hours:
   \[
   \text{Total time} = 5 \text{ hours} + 8 \text{ hours} = 13 \text{
   ↪  hours}
   \]

2. **Average Speed**:
   - The average speed during the entire trip was 18 mph.

3. **Total Distance**:
   - We can calculate the total distance traveled using the formula:
   \[
   \text{Total distance} = \text{Average speed} \times \text{Total
   ↪  time}
   \]
   \[
   \text{Total distance} = 18 \text{ mph} \times 13 \text{ hours} = 234
   ↪  \text{ miles}
   \]

### Step 2: Calculate Distances in Each Direction

4. **Distance Traveled West**:
   - Elvis drives west initially for 5 hours. Let's denote his speed
   ↪  going west as \( v_w \). We assume that his speed was the same
   ↪  while driving west and east until he starts driving back.
   - Let's calculate the distance traveled west:
   \[
   \text{Distance west} = v_w \times 5
   \]

5. **Distance Traveled East**:
   - After turning around, he drives east for 8 hours. Let's denote his
   ↪  speed going back as \( v_e \).
   - The distance traveled east can be expressed as:
   \[
   \text{Distance east} = v_e \times 8
   \]

### Step 3: Set Up the Equation for Total Distance

6. Based on the distances and knowing the direction he traveled:
   - The total distance equals the distance traveled west minus the
   ↪  distance traveled east (accounting for the return trip):
   \[
   v_w \times 5 + v_e \times 8 = 234
   \]

### Step 4: Determine Speeds

7.  To find the distances going each direction, let's assume that he
↪  drove at the same average speed in both directions for
↪  simplification:
   - Since his average speed for the total trip is 18 mph, the overall
   ↪  distance calculation can be structured:
   \[
   \text{Distance west} + \text{Distance east} = \text{Total distance}
   \]
```

```
   - Hence, if we let \( v \) = 18 mph, we replace:
   - In total:
   \[
   18 \text{ mph} \times 5 + 18 \text{ mph} \times 8 = 234
   \]

### Step 5: Final Position Calculation

8. We can summarize the distances:
   - He traveled west for a total of \( 90 \) miles (5 hours at 18
   ↪  mph).
   - He traveled east for a total of \( 144 \) miles (8 hours at 18
   ↪  mph).

9. **Net Position**:
  - After turning around and driving back to his house, he drove a
  ↪  total net distance of:
   \[
   \text{Net distance from house} = \text{Distance east} -
   ↪  \text{Distance back} = (90 + 144) - 90 = 54 \text{ miles}
   \]
   * Thus, after completing this trip, Elvis ended up **54 miles east**
   ↪  of his starting point at home.

### Conclusion:
**The final answer is**: Elvis is now **54 miles away** from his house.
```

IC view 6: retain Assistant 6. Recorded evaluator outcome: Incorrect (×).

```
Let's carefully work through the problem step-by-step to determine how
↪  far Elvis is from his house, given all the information provided.

### 1. Initial Trip West:
- **Direction:** West
- **Time:** 5 hours
- **Average Speed:** 18 mph

#### Distance Traveled West
\[
\text{Distance} = \text{Speed} \times \text{Time} = 18 \text{
↪  miles/hour} \times 5 \text{ hours} = 90 \text{ miles}
\]
So, after driving west for 5 hours, Elvis is **90 miles west** of his
↪  house.

### 2. Turning Around:
- After driving west, Elvis turns around and heads back towards the
↪  east.

### 3. Returning Home:
- The distance to return to his house is the same, **90 miles**.

#### Time Taken to Return Home
\[
\text{Time to Return} = \frac{90 \text{ miles}}{18 \text{ miles/hour}}
↪  = 5 \text{ hours}
\]
After driving back for 5 hours, he arrives back at his house, which
↪  means he is now **0 miles from his house**.

### 4. Driving East for 8 More Hours:
- Now Elvis drives east for another 8 hours at the same average speed
↪  of 18 mph.

#### Distance Traveled East
\[
```

```
\text{Distance} = \text{Speed} \times \text{Time} = 18 \text{
↪  miles/hour} \times 8 \text{ hours} = 144 \text{ miles}
\]
After driving back east for 8 hours, he is **144 miles away from his
↪  house**.

### Final Conclusion:
Elvis is **144 miles from his house** in the east direction after the
↪  entire trip.

Thank you for your patience, and if there are more questions or further
↪  clarifications needed, please let me know!
```

FULL-HISTORY VOTING CANDIDATES

Each candidate receives the complete standard transcript and the same recap. The outputs below are reproduced in vote-index order.

Full-history candidate 1. Recorded evaluator outcome: Correct (✓).

```
Sure! Let's walk through the task step by step, using all the
↪  information provided, and summarize it correctly to determine how
↪  far Elvis is from his house.

### Task Breakdown:
1. **Starting Point:**
   - Elvis begins his journey at his house.

2. **Driving West:**
   - **Time:** 5 hours
   - **Average speed:** 18 mph
   - **Distance traveled west:**
     \[
     \text{Distance} = \text{Speed} \times \text{Time} = 18 \text{
     ↪  miles/hour} \times 5 \text{ hours} = 90 \text{ miles}
     \]
   - Therefore, after driving west for 5 hours, Elvis is **90 miles
   ↪  away from his house**.

3. **Turning Around:**
   - After driving west, Elvis turns around and starts to drive back
   ↪  towards his house.

4. **Driving East:**
   - Now, consider the portion of his trip when he drives east:
   - **Time spent driving east:** 8 hours
   - **Distance traveled east:**
     \[
     \text{Distance} = \text{Speed} \times \text{Time} = 18 \text{
     ↪  miles/hour} \times 8 \text{ hours} = 144 \text{ miles}
     \]

5. **Net Distance from House After Driving East:**
   - From his earlier position (90 miles west of his house), he drives
   ↪  back towards (east from that position):
   - At this point, after driving the additional 144 miles east, we
   ↪  calculate his new position relative to his house.

6. **Calculating Final Position:**
   - After returning 90 miles back to his house, he has:
     - **Back at house:** 0 miles (from the 90 miles he was
     ↪  originally).
   - Then he drives 144 miles **beyond** his house:
     - New distance from the house:
     \[
```

```
    \text{Final position} = 144 \text{ miles} - 90 \text{ miles} = 54
    ↪  \text{ miles}
    \]

   So, Elvis is now **54 miles** east of his house.

### Summary of Final Answer:
**Elvis is 54 miles away from his house.**

If you need further clarification or have additional questions, feel
↪  free to ask!
```

Full-history candidate 2. Recorded evaluator outcome: Incorrect (×).

```
Certainly! Let's summarize the entire task step-by-step and provide the
↪  final answer regarding how far Elvis is from his house, while
↪  considering all the given information accurately.

---

### Given Information:
- **Starting Point:** Elvis starts his journey from his house.
- **Trip West:** He drives west for **5 hours**.
- **Direction Change:** He turns around and drives east.
- **Trip East:** He drives east for **8 hours**.
- **Average Speed:** His average speed during the entire trip is **18
↪  mph**.

---

### Step-by-Step Calculation:

1. **Calculate Distance Driven West:**
   - Distance traveled west for 5 hours at 18 mph:
   \[
   \text{Distance}_{\text{west}} = \text{Speed} \times \text{Time} = 18
   ↪  \text{ miles/hour} \times 5 \text{ hours} = 90 \text{ miles}
   \]

2. **Distance from House After Driving West:**
   - After driving west, Elvis is **90 miles away** from his house.

3. **Turning Around:**
   - He now turns around and drives back towards his house. He has to
   ↪  cover the same distance of **90 miles** to return home.

4. **Calculate Time to Return Home:**
   - It would take him the same distance time to return at 18 mph:
   \[
   \text{Time}_{\text{return}} = \frac{90 \text{ miles}}{18 \text{
   ↪  miles/hour}} = 5 \text{ hours}
   \]

5. **Distance After Returning Home:**
   - After 5 hours of driving back east, he arrives back at his house,
   ↪  making him **0 miles away** from his house at this point.

6. **Driving East for 8 More Hours:**
   - After reaching home, he then drives east for an additional 8 hours
   ↪  at 18 mph:
   \[
   \text{Distance}_{\text{east}} = \text{Speed} \times \text{Time} = 18
   ↪  \text{ miles/hour} \times 8 \text{ hours} = 144 \text{ miles}
   \]

7. **Final Distance from House:**
```

```
   - After driving **144 miles** east from home, Elvis is now **144
   ↪  miles away** from his house.

---

### Conclusion:
**Final Answer: Elvis is 144 miles away from his house.**

If you have any further questions or need additional clarification,
↪  feel free to ask!
```

Full-history candidate 3. Recorded evaluator outcome: Incorrect (×).

```
Let's carefully analyze the entire journey of Elvis step-by-step and
↪  summarize the final result concisely.

### Given Information:
- Elvis starts his trip from his house.
- He drives west for 5 hours.
- After that, he turns around and drives east for 8 hours.
- His average speed during the entire trip is **18 mph**.

### Step 1: Driving West
- **Westward journey time:** 5 hours
- **Distance traveled west:**
  \[
  \text{Distance} = \text{Speed} \times \text{Time} = 18 \text{
  ↪  miles/hour} \times 5 \text{ hours} = 90 \text{ miles}
  \]

### Step 2: Returning to His House
- After driving west for 5 hours and reaching 90 miles away from his
↪  house, he turns around and drives back toward his house.

- **Distance back to his house:** 90 miles
- **Time to return home:**
  \[
  \text{Time to Return} = \frac{\text{Distance}}{\text{Speed}} =
  ↪  \frac{90 \text{ miles}}{18 \text{ miles/hour}} = 5 \text{ hours}
  \]

  After this time, he reaches his house (0 miles away).

### Step 3: Driving East
- After returning home, Elvis drives east for **8 hours.**
- **Distance traveled east:**
  \[
  \text{Distance} = \text{Speed} \times \text{Time} = 18 \text{
  ↪  miles/hour} \times 8 \text{ hours} = 144 \text{ miles}
  \]

### Final Distance from His House
- After driving **144 miles east** from his house, Elvis is now:

\[
\text{Distance from his house} = 144 \text{ miles}
\]

### Final Answer
**Elvis is 144 miles away from his house.**

If you have any further questions or require more assistance, feel free
↪  to ask!
```

Full-history candidate 4. Recorded evaluator outcome: Incorrect (×).

```
Sure! Let's summarize the entire journey of Elvis step-by-step, taking
↪  all the provided information into account and ultimately answering
↪  the question about how far Elvis is from his house at the end of
↪  his trip.

### Breakdown of the Journey

1. **Initial Position:**
   - Elvis starts his journey from his house (Position A = 0 miles).

2. **Driving West:**
   - **Duration:** 5 hours
   - **Average Speed:** 18 mph
   - **Distance Traveled West:**
     \[
     \text{Distance} = \text{Speed} \times \text{Time} = 18 \text{
     ↪  miles/hour} \times 5 \text{ hours} = 90 \text{ miles}
     \]
   - **Elvis is now:** 90 miles west of his house.

3. **Turning Around:**
   - He then turns around to go back east towards his house.

4. **Time to Return Home:**
   - Since he is now 90 miles away from his house, at the same speed of
   ↪  18 mph, the time taken to return home:
   \[
   \text{Time to Return} = \frac{90 \text{ miles}}{18 \text{
   ↪  miles/hour}} = 5 \text{ hours}
   \]
   - After 5 hours of driving back east, Elvis reaches his house
   ↪  (Position B = 0 miles).

5. **Driving East for 8 More Hours:**
   - After reaching his house, Elvis continues to drive east for 8
   ↪  hours at 18 mph.
   - **Distance Traveled East:**
     \[
     \text{Distance} = \text{Speed} \times \text{Time} = 18 \text{
     ↪  miles/hour} \times 8 \text{ hours} = 144 \text{ miles}
     \]
   - **Now he is:** 144 miles east of his house.

### Final Result
After completing his entire trip, Elvis is **144 miles away** from his
↪  house in the east direction.

### Summary Answer
**Elvis is 144 miles away from his house at the end of his trip.**
```

Full-history candidate 5. Recorded evaluator outcome: Incorrect (×).

```
Let's summarize and calculate the entire distance Elvis is from his
↪  house step-by-step, taking into account all the information
↪  provided:

### Breakdown of the Journey:

1. **Starting Point:**
   - Elvis starts at his house (let's consider this as 0 miles from his
   ↪  house).

2. **Driving West:**
   - **Time driving west:** 5 hours
   - **Average speed:** 18 mph
   - **Distance traveled west:**
```

```
     \[
     \text{Distance} = \text{Speed} \times \text{Time} = 18 \text{ mph}
     ↪  \times 5 \text{ hours} = 90 \text{ miles}
     \]
   - After driving west for 5 hours, Elvis is now **90 miles** away
   ↪  from his house.

3. **Turning Around:**
   - After reaching 90 miles west of his house, he turns around and
   ↪  starts driving east to return home.

4. **Returning Home:**
   - To return to his house, he needs to cover the same distance of 90
   ↪  miles.
   - **Time to return home:**
     \[
     \text{Time to Return} = \frac{90 \text{ miles}}{18 \text{ mph}} =
     ↪  5 \text{ hours}
     \]
   - After driving back for 5 hours, he reaches his house again, making
   ↪  him **0 miles** away from his house.

5. **Driving East for 8 More Hours:**
   - After reaching his house, he then drives east for 8 hours.
   - **Distance traveled east:**
     \[
     \text{Distance} = 18 \text{ mph} \times 8 \text{ hours} = 144
     ↪  \text{ miles}
     \]

### Final Distance from House:

6. **Final Position:**
   - At the end of his entire journey (after driving 144 miles east
   ↪  from his house), Elvis is now **144 miles away from his house**.

### Conclusion:

**A:** After completing his journey:
- Elvis is **144 miles** away from his house.

If you have any further questions or need additional assistance, feel
↪  free to ask!
```

Full-history candidate 6. Recorded evaluator outcome: Incorrect (×).

```
Let's go through the entire journey step-by-step, taking into account
↪  all the given information to determine how far Elvis is from his
↪  house at the end of the trip.

### Step-by-Step Breakdown:

1. **Starting Point:**
   - Elvis starts his journey from **his house**.

2. **Driving West:**
   - **Duration:** 5 hours
   - **Average Speed:** 18 mph
   - **Distance traveled west:**
     \[
     \text{Distance} = \text{Speed} \times \text{Time} = 18 \text{
     ↪  miles/hour} \times 5 \text{ hours} = 90 \text{ miles}
     \]
   - After this leg of the journey, Elvis is **90 miles** west of his
   ↪  house.
```

```
3. **Turning Around:**
   - Elvis turns around and drives back east towards his house. He will
   ↪  cover the same distance of **90 miles** to return home.

4. **Driving Back Home:**
   - To calculate the time taken to return to his house at 18 mph:
   \[
   \text{Time to return} = \frac{90 \text{ miles}}{18 \text{
   ↪  miles/hour}} = 5 \text{ hours}
   \]
   - After driving back for **5 hours**, Elvis reaches his house, which
   ↪  means he is **0 miles** away from it at this point.

5. **Driving East for 8 More Hours:**
   - After returning home, Elvis then drives east for an additional **8
   ↪  hours** at **18 mph**.
   - **Distance traveled east:**
   \[
   \text{Distance} = \text{Speed} \times \text{Time} = 18 \text{
   ↪  miles/hour} \times 8 \text{ hours} = 144 \text{ miles}
   \]

### Final Distance from House:
6. **Final Position:**
   - After driving **144 miles** east from his house, Elvis is now
   ↪  **144 miles** away from his house.

### Conclusion:
Based on the entire journey:
- After driving west for 5 hours, returning home, and then driving east
↪  for 8 hours, Elvis is **144 miles east** from his house.

Thus, the final answer to your question is:
**Elvis is 144 miles away from his house.**
```

RECORDED VOTE OUTCOME

IC returns the correct answer of 54 miles with five votes to one. Full-history voting returns the incorrect answer of 144 miles with five votes to one.

C IMPLEMENTATION AND EXPERIMENTAL DETAILS

This appendix summarizes the experimental pipeline, generation settings, and task-specific aggregation rules. Appendix D provides the complete prompts.

C.1 INFERENCE PROCEDURE

Algorithm 1 Isolated Consistency

Input: Transcript $H_K = [S, U_1, A_1, \ldots, U_K, A_K]$; complete task Q
Input: Model p_θ; task-specific answer-grouping rule E_t
Output: Final answer $\widehat{y}$

for $i = 1, \ldots, K$ **do** ▷ Views can run in parallel
  $\mathcal{V}_i \leftarrow \textsc{Copy}(H_K)$
  Replace every assistant response in $\mathcal{V}_i$ with `[MASKED]`
  Restore A_i in its original message slot
  Append Q and sample $y_i \sim p_\theta(\cdot \mid \mathcal{V}_i)$
  $c_i \leftarrow E_t(y_i)$
end for
$\widehat{c} \leftarrow \arg\max_c \sum_{i=1}^{K} \mathbf{1}[c_i = c]$
return a fixed representative answer from class $\widehat{c}$

C.2 Conversation and recovery construction

We perform one run for each example, model, and condition. A task-specific system message begins each conversation. The user simulator then reveals at most one new instruction shard per turn, and the evaluated model produces the assistant response. After each response, a system agent classifies the response and extracts a task answer when present. The original sharded run ends when the answer is correct or all shards have been revealed. GPT-4o-mini serves as both the user simulator and the system agent, while evaluator metadata remains outside the assistant's context.

All recovery methods receive the same completed task Q, constructed deterministically from the stored shards in their original order. Math and Actions place each shard on a separate bullet line. Code preserves its formatting preamble and starter code, while Database includes the schema and all query requirements. Restart presents Q in a fresh conversation. RC appends Q to the original transcript, and RC-R adds the reset instruction to the same recap.

IC constructs one view for each instruction shard. A view retains the first assistant response produced after that shard was revealed and replaces responses linked to the other shards with `[MASKED]`. System and user messages remain unchanged and in their original order. IC-R uses the same views with the reset recap. The evaluated model generates one complete answer from every view, after which the candidates are grouped and voted on.

C.3 Generation settings

Assistant generation, user simulation, and recap generation use temperature 1.0. Response classification and model-based answer extraction use temperature 0.0 with structured output. Generated responses are limited to 1,000 tokens for all seven evaluated models, and sampling does not use a fixed seed. We use five parallel workers for sharded generation and recap recovery; this changes scheduling but not the candidate set.

C.4 Answer grouping and evaluation

Math and Database use GPT-4o-mini to extract a verbatim answer span from the generated response. Code extracts the final Python implementation from Markdown code blocks, and Actions passes the complete response to its structured-call evaluator. The default voting rule strips surrounding whitespace for Math and Actions; for Database, it also removes SQL fences and collapses whitespace. Ties are broken lexicographically with a fixed rule.

Code uses executable verification. Each candidate program is executed to obtain a Boolean pass vector, and candidates vote as complete pass vectors rather than through independent per-test majorities. The program associated with the most frequent vector is returned.

Final correctness follows the benchmark evaluator for each task. Math applies the benchmark's numeric normalization before exact comparison. Database uses the Spider execution evaluator with value plugging enabled and distinctness disabled. Actions parses the predicted function calls and compares them with the required calls. Code is correct only when the selected program passes every test. Statistics require at least one usable candidate for a conversation, and final-evaluation exceptions count as incorrect predictions.

D Prompt Templates

The following blocks reproduce the prompts used in our experiments. Spelling, capitalization, and punctuation are unchanged; visual line wrapping is only for typesetting. Double-bracket fields are replaced with task or conversation content before each request. We include the recovery prompts, task inputs, user simulation, response classification, and answer extraction.

D.1 Recap and reset messages

The recap's `[[SAMPLE_TEXT]]` field contains the completed task text, filled by the task-specific concatenation function. Each populated block is appended as the final user message after the preserved or masked conversation. RC and IC share the ordinary recap; RC-R and IC-R share the

reset recap. The models produce an ordinary assistant answer after this message, with no request to compare candidates, inspect other views, or report agreement with an earlier response.

Ordinary recap.

```
Just to recapitulate, the entire task is:

[[SAMPLE_TEXT]]

Now please complete the task correctly taking all the information into
account.
```

Soft-reset recap.

```
Forget the messages above. The entire task is:

[[SAMPLE_TEXT]]

Now please complete the task correctly.
```

D.2 Task system messages and complete-task inputs

The system message is sent with the system role. The complete-task input is sent directly as the user message for Restart or inserted inside the recap for recovery. For Restart and recap, the question or instruction field contains the shard-based text described in Appendix C. Code experiments use the LiveCodeBench system message with both Code task inputs.

Math system message.

```
As an expert problem solver solve step by step the following
mathematical questions.
```

Math task input.

```
Q: [[QUESTION]]
A:
```

Code system message.

```
You are an expert Python programmer. You will be given a question
(problem specification) and will generate a correct Python program that
matches the specification and passes all tests.

Format:
- [Standalone] Make sure that your answer consists of only one Python
function at the top level. Do not wrap with a class or split into
multiple functions.
```

LiveCodeBench task input.

```
### Question:
[[QUESTION]]

### Format: [[FORMATTING_PREAMBLE]]
[[FORMATTING]]

### Answer: (use the provided format with backticks)
```

HumanEval task input.

```
You must generate a valid Python function that satisfies the following
instruction:

Instruction:
```

```
[[INSTRUCTION]]

Rules:
- [Format] You must only return the Python function, matching the
signature exactly.
- [Validity] The Python function must be valid and executable. Do not
include any other text or comments. (```python```)
```

Database system message.

```
You are helping a user write SQL queries to a database. If something is
not clear, you can ask the user to clarify what they need. The schema
for the database being accessed is the following:

[[SCHEMA]]
```

Database task input.

```
You must generate the valid SQL query that answers the given user query
in natural language.

The schema of the database you are responding to is as follows:
[[DATABASE_SCHEMA]]

The user query is:
[[USER_QUERY]]

Rules:
- [Single Answer] Produce a single valid SQL query based on the
database schema and the user query.
- [Only SQL] Do not include any other text or comments.
- [Validity] The SQL query must be valid and executable.
- [Single Line] Produce your entire response in a single line.
```

Actions system message.

```
You are an expert in composing functions. You are given a question and
a set of possible functions. Based on the question, you will need to
make one or more function/tool calls to achieve the purpose.
If none of the functions can be used, point it out. If the given
question lacks the parameters required by the function, also point it
out.
You should only return the function calls in your response.

If you decide to invoke any of the function(s), you MUST put it in the
format of [func_name1(params_name1=params_value1,
params_name2=params_value2...), func_name2(params)]
You SHOULD NOT include any other text in the response.

Here is a list of functions in JSON format that you can invoke.

[[FUNCTIONS]]
```

Actions task input.

```
[[QUESTION]]
```

Task-field substitutions. For Math and Actions, `[[QUESTION]]` is the bullet-form shard list. For Code, `[[QUESTION]]` or `[[INSTRUCTION]]` is the initial shard followed by the remaining bullet-form shards. The LiveCodeBench formatting fields use the following fixed preamble and a Python code block containing the sample's starter code, or the literal comment `# YOUR CODE HERE` when it is empty. For Database, `[[SCHEMA]]` and `[[DATABASE_SCHEMA]]` contain the database schema, while `[[USER_QUERY]]` contains the shard list. For Actions, `[[FUNCTIONS]]` contains the function definitions supplied with the task.

```
You will use the following starter code to write the solution to the
problem and enclose your code within delimiters.
```

D.3 User simulation and response classification

For user simulation, `[[CONVERSATION_SO_FAR]]` contains the user and assistant messages with role tags and blank lines between messages; the system message is omitted. The revealed and unrevealed shard fields are JSON lists. For classification, the conversation field contains only the latest assistant response. Both prompts request JSON output.

User simulator.

```
You are simulating a user of an interactive LLM system (like ChatGPT).
The user is inherently lazy, and answers in short form, providing only
minimal information to the system. You should not be proactive.

Here's the conversation so far:
[[CONVERSATION_SO_FAR]]

Here are the shards that have already been revealed:
[[SHARDS_REVEALED]]

Here are all the shards that have not been revealed yet:
[[SHARDS_NOT_REVEALED]]

You must generate a response to the conversation so far. Here are the
rules:
- [Providing a Shard] You can reveal the content of a shard to the
system in your response if it will help the system move closer to
answering the problem. You should select the shard to reveal that is
most "basic" and is the current most relevant shard.
- [One Shard at a Time] You should only reveal at most one shard at a
time.
- [Reveal Entire Shard] If you reveal a shard, you must make sure to
include *all the information in the shard*. For example, if the shard
is "your symptoms are that you have a headache in the mornings", your
response can't just be ``yeah I have headaches'', you must say ``yup
mostly headaches in the mornings``.
- [Irrelevant Clarifications] If the system asks you a question
irrelevant to the shards, asks you a generic question (``Can you give
me a hint?``), you should respond with an answer that does not provide
a shard. (``I don't know``, ``Is that really important?``, etc.) You
should not reveal any information beyond what is available in the
shards.
- [No Repeated Shards] You should not reveal the same shard more than
once. Carefully review the shards revealed already, and only reveal a
shard if its `shard_id` is not on the list.
- [Rephrase Shards] If you reveal a shard, you should rephrase it in a
conversational way. Do not copy the shard verbatim.
- [Do Not Ask Questions] Your response should always be declarative
sentences, and not questions.
- [Brevity of Response] You should favor being succint. Your answer can
also have typos, improper grammar, capitalization, etc. You are
simulating a real person talking to an AI, who is in a hurry.
- [Format] Your response should be formatted as a JSON object with the
following keys:
    - `response`: The response to the conversation so far.
    - `shard_id`: The shard you are revealing to the system. The
    shard_id can be an integer, or -1 if you did not reveal any shards.
For example:
{"response": "I don't know", "shard_id": -1}
or:
{"response": "yeah I want it to [...]", "shard_id": 1}
```

Response-type classifier.

```
You are reviewing a multi-turn conversation between a user and an
assistant, and are given the last turn of the conversation.

Here is the full specification of the problem the system is attempting
to solve:
[[INITIAL_SHARD]]

Specification:
[[SHARDS]]

You must classify the response of the assistant according to the
response type:
- `answer_attempt`: The response contains a complete answer attempt to
the user's question (not templated or hypothetical), that can be
extracted verbatim. See the task-specific answer description for more
details.
- `clarification`: The response  is short (less than 100 words) and
contains a single question addressed to the user that directly inquires
about an aspect of the user's query. A clarification turn cannot be
long (see `discussion`), cannot contain a vague question (see
`discussion`) and cannot contain multiple questions (see
`interrogation`).
- `interrogation`: The response contains multiple questions addressed
to the user, sometimes organized in a list or bullet-points.
- `discussion`: The response discusses the question in detail, without
providing a final answer, asking a specific clarification question, or
a refusal to answer. The response may or may not contain a vague
question (e.g., \What else can I help you with?").
- `hedge`: The response contains multiple answer candidates based on
hypotheticals (ifs) or branching (case 1, case 2) with corresponding
descriptions.
- `refuse`: The response contains an explicit or implicit refusal to
answer the user's question without a follow-up question or a request.
- `missing`: The response is empty/blank.

You must output your answer in the following JSON format:
{"response_type": "refuse|missing|answer_attempt|hedge|clarification|␣
interrogation|discussion"}

Rules:
- The assistant giving a hint at how an answer could look like is not a
final answer. You should only select `answer_attempt` if the
conversation could end at this stage with the user having an entirely
final answer to the problem they've formulated.
- [Task Specific Answer] [[ANSWER_DESCRIPTION]]

Conversation's last turn:
[[CONVERSATION_SO_FAR]]
```

D.4 Answer extraction and task-specific descriptions

Math uses the direct extraction template, while Database uses the prefix–suffix template. In both cases, `[[ASSISTANT_RESPONSE]]` contains the latest assistant response with its role tag. Code and Actions are parsed directly rather than using these extraction templates. The descriptions below populate `[[ANSWER_DESCRIPTION]]` during classification and extraction.

Direct answer extraction.

```
You are reviewing a response from an AI assistant to a user query.
Your goal is to extract verbatim what the answer is. The answer to the
user query is expected to be: [[ANSWER_DESCRIPTION]]

Rules:
```

```
- [Exact Answer Only] only extract the exact answer, and nothing else
(including ``` for code blocks, or intro/outro text).
- [Verbatim Only] Only extract verbatim text, do not modify the text in
any way. If there's a typo, an error, you must absoltutely include it,
and not correct it in any way. The program will check that the string
you extract is exactly present as a substringz in the passage provided.
- [String output] the <answer_str> must be a string, not a number and
not a dictionary.

You must output your answer in the following JSON format:
{"answer": "<answer_str>"}

Response:
[[ASSISTANT_RESPONSE]]
```

Prefix–suffix answer extraction.

```
You are reviewing a multi-turn conversation between a user and an
assistant, and are given the last turn of the conversation.
In the final response from the assistant, a final answer has been
provided. Your goal is to extract verbatim what the answer is:
- If the answer is short (less than 10 words), then you should copy
verbatim what the answer is in the `answer` field.
- If the answer is long, then you should produce the answer with an
ellipses, to indicate the exact start and end of the answer (e.g, ```def
funny_function(n): [...]  return funny_output```). You should include
*at least* 4 words or one full line for the start (before the ellipses)
and *at least* 4 words or one full line for the end (after the
ellipses), such that the answer can be identified exactly.

Rules:
- [Exact Answer Only] only extract the exact answer, and nothing else
(including ``` for code blocks, or intro/outro text).
- [Verbatim Only] Only extract verbatim text, do not modify the text in
any way. If there's a typo, an error, you must absoltutely include it,
and not correct it in any way.
- [Task Specific Answer] [[ANSWER_DESCRIPTION]]
- [String output] the <answer_str> must be a string, not a number and
not a dictionary.

You must output your answer in the following JSON format:
{"answer": "<answer_str>"}

Conversation's last turn:
[[ASSISTANT_RESPONSE]]
```

Math answer description.

```
The answer should be a single number (it could be decimal, or negative,
or a fraction, etc.).
```

Code answer description.

```
A final answer must be a valid Python function that is defined (def
...) and returns something. The function should handle multiple test
cases according to the problem specification. If the answer is just a
natural language explanation, it is not a valid answer. Special symbols
like newlines and quotes must be escaped.
```

Database answer description.

```
If the response contains a complete SQL query (and not just partial or
templated SQL used as an example), then it is an answer attempt. You
must only extract the SQL query, nothing before or after, as it will be
executed as is.
```

Actions answer description.

```
The answer should be a series of valid function calls in the format of
[func_name1(params_name1=params_value1, params_name2=params_value2...),
func_name2(params)]. An answer may contain multiple function calls.
```

E LIMITATIONS

Our evaluation uses clean recaps and known recovery points in simulated, English-language conversations. This controlled setting does not capture uncertain task boundaries, imperfect recaps, or open-ended tasks such as creative writing, whose outputs require different notions of quality (Chakrabarty et al., 2024; 2025). At the same time, fixing the completed task and recovery point allows the experiments to isolate the effect of assistant history across seven models and four distinct analytical tasks. We report aggregate scores rather than paired uncertainty estimates, while the IC–SC improvement appears on every evaluated model and in 24 of 28 model–task comparisons.

IC generates K candidates and therefore costs more than a single recap call. This budget is matched exactly by SC, however, and all K views can be processed in parallel. The comparison consequently attributes the accuracy gain to the candidate contexts rather than to additional samples or a different voting rule. Reducing the number of required views is a natural extension, and the budget analysis already shows that IC retains an advantage when only 25% of the candidate pool is used.

Our implementation also uses deliberately simple components: a literal `[MASKED]` marker, one retained response per view, and plurality voting over task-specific answer groups. These choices can omit reasoning spread across several turns or separate equivalent surface forms. Yet this simple, training-free procedure improves all seven models without access to their internal states, leaving richer multi-turn views and semantic grouping as opportunities for further gains.