SeDT: Sentence-Transformer Decision-Transformer Conditioning for Multi-Turn Conversation Reliability

Anonymous ACL submission

Abstract

Large language models (LLMs) achieve impressive performance when a task is fully specified in a single turn, yet the same models lose up to 39% of that performance when the identical task is revealed incrementally across multiple turns, a phenomenon documented at scale as *Lost in Conversation*. Crucially, this collapse is almost entirely a *reliability* failure; the best case, the aptitude only falls 16%, while the unreliability more than doubles (+112%). We argue that the root cause is structural, a flat conversation history assigns equal implicit weight to every prior turn, giving the model no signal to distinguish a critical constraint from incidental dialog. We present **SeDT Se**ntence-transformer **D**ecision-**T**ransformer, a training-free inference-time method that resolves this by importing return-to-go conditioning from offline reinforcement learning. SeDT annotates each conversation shard with a cumulative relevance score derived from three complementary semantic, lexical, and positional signals and presents the full annotated history to the model at the final turn, without weight changes, without training data, and without discarding context. Evaluated on the Lost-in-Conversation benchmark in three LLMs and three generation tasks, SeDT outperforms the sharded baseline in all nine model-task combinations, with gains up to $+37.7\%$ in mean performance $\bar{P}$ and simultaneous reductions in unreliability in seven of the nine combinations. In short, telling the model which past turns matter is sufficient to substantially recover the performance lost in conversation.

1 Introduction

Ask a language model to write a function and it will succeed (Rachapudi et al., 2026c,b,a). Give it the same problem one constraint at a time, the function name first, the input format next, the edge cases last, and it will quietly fall apart. Laban et al. (2025) document this failure at scale, 15 state-of-the-art (SoTA) LLMs evaluated on six tasks over 200,000 simulated conversations suffer an average performance drop of 39% when the same instruction is revealed across turns rather than all at once. This is not a contrived setting. Analysis of large scale real world LLM conversations confirms that multi-turn, underspecified interaction is the norm rather than the exception (Zheng et al., 2023a), and that users, particularly novice ones, rarely specify all requirements upfront (Herlihy et al., 2024).

The cause is structural. A multi-turn conversation is concatenated into a flat context window in which every prior turn carries equal implicit weight. The model receives no signal about which turns specified critical constraints and which were conversational scaffolding. Transformer attention compounds this; models systematically neglect middle-context information even in single-turn settings (Liu et al., 2024), and multi-turn conversations stack this bias across turns the model cannot distinguish from noise. Four concrete failure modes follow directly, premature commitment to an answer before all constraints are revealed, over-reliance on incorrect intermediate responses, loss of middle-turn constraints in favor of the first and last turns, and verbose drift that introduces false assumptions (Laban et al., 2025).

The offline reinforcement learning community faced a structurally identical problem. An agent learning from a flat replay buffer has no signal about which transitions were valuable. The Decision Transformer (Chen et al., 2021a) resolved this by annotating each trajectory step with its return-to-go (RTG), telling the agent what mattered rather than leaving it to infer importance from a flat buffer. The same insight applies to multi-turn LLM inference, yet existing solutions fall short; finetuning requires curated multi-turn training data and weight modification making it impractical at inference time, and users reveal constraints incrementally not by choice but by necessity (Herlihy

et al., 2024). We import this parallel directly into inference-time prompt construction. A multi-turn conversation is a trajectory; each *shard*, an atomic piece of information revealed in a single turn, is a step, and the semantic relevance of that shard to the final output goal is the reward. Just as the Decision Transformer tells the agent which steps mattered, SeDT tells the model which turns matter.

To our knowledge, no prior work identifies equal implicit turn weighting in flat conversation contexts as the structural driver of multi-turn reliability collapse; prior work attributes the failure to intent misalignment (Liu et al., 2026) or model unreliability (Laban et al., 2025) rather than to the context representation itself.

We present **SeDT** (**Se**ntence-transformer **D**ecision-**T**ransformer conditioning), a training-free inference-time method that resolves flat-context weighting by annotating each prior shard with a cumulative relevance score and presenting the full RTG-annotated history to the model at the final turn changing no weights, requiring no training data, and discarding no context. On the Lost-in-Conversation benchmark (Laban et al., 2025), SeDT consistently outperforms the sharded baseline across all three evaluated LLMs and tasks, with gains up to $+37.7\%$ in mean performance $\bar{P}$ and simultaneous reductions in unreliability, confirming that the lost-in-conversation problem is at least partially one of context weighting, addressable at inference time without training. SeDT requires no task-specific data, no model modification, and at most one additional LLM call.

Contributions. Our main contributions are as follows:

- **Problem identification** Flat-context turn weighting identified as the structural root cause of multi-turn reliability collapse, with a formal parallel to Decision Transformer RTG conditioning.
- **SeDT** A training-free inference-time method that requires zero model modification and zero training data.
- **Three-signal relevance** A semantic, lexical and positional relevance formulation that directly counteracts the four documented failure modes of multi-turn LLMs.
- **RTG-grounded self-correction** A two-guard correction mechanism that introduces zero hurt cases while providing a conservative verification pathway.

2 Background

2.1 Lost in Conversation

Laban et al. (2025) document a systematic performance gap between single-turn, fully-specified interaction and multi-turn, underspecified interaction at scale. Their sharding framework decomposes a fully-specified instruction into atomic information shards revealed one per turn, enabling controlled comparison while holding task content constant. Running n independent simulations per example at temperature $T{=}1.0$ yields three metrics: average performance $\bar{P}$ (mean score), aptitude A_{90} (90th percentile, capturing best-case capability) and unreliability U (the 90th$-$10th percentile gap, where lower is better). The central finding is that the lost-in-conversation problem manifests itself primarily as an unreliability explosion ($+112\%$) rather than an aptitude collapse (-16%). Therefore, a reliable solution must reduce U along with improving $\bar{P}$.

2.2 Related Work

Multi-turn evaluation A growing body of work evaluates LLMs in multi-turn settings through *episodic* benchmark settings where each conversation turn introduces a self-contained subtask that can be evaluated independently, without requiring the model to fuse information accumulated across turns (Zheng et al., 2023b; Bai et al., 2024; Kwan et al., 2024; Wang et al., 2023). Although these benchmarks capture important capabilities such as refinement (iteratively improving a response based on user feedback) and tool use (calling external APIs or executing code across turns), they do not require the model to fuse underspecified information accumulated across turns and, as a result, systematically overestimate multi-turn performance (Laban et al., 2025).

Attention bias and long-context failures Transformer models exhibit a well-documented U-shaped attention bias, disproportionately attending to tokens at the beginning and end of long contexts while neglecting middle content (Liu et al., 2024). This phenomenon extends naturally to multi-turn conversation, where middle turns carrying critical constraints receive insufficient attention (Laban et al., 2025). Static self-attention has been identified as the root cause of score dilution in long-context settings, motivating test-time approaches that address this at the query level (Bansal et al., 2025).

Context management and recapitulation Beyond evaluation, a separate line of work addresses the multi-turn problem by modifying how conversation history is presented to the model. One family of approaches uses recapitulation, a strategy in which prior user turns are restated verbatim in the current context to ensure that the model has access to all previous constraints. Standard recapitulation appends all prior shards as a restatement at the final turn, while snowball recapitulation grows this restatement cumulatively at every turn (Laban et al., 2025). A subtler alternative is starting a new conversation for each sub-task, which empirically improves performance (Laban et al., 2025) by resetting the flat context, but this discards conversational history entirely and is unavailable when constraints arrive incrementally from a real user. Finetuning resolves flat weighting by updating model weights on curated multi-turn data, but requires task-specific corpora, retraining for every new model or domain, and is unavailable at inference time.

Intent alignment and instruction following When faced with incomplete or ambiguous questions, LLMs exhibit systematic response patterns. Some models hedge, producing vague or noncommittal answers that avoid committing to an incomplete specification. Others issue clarification requests, asking the user to provide missing information before proceeding. A third pattern is the premature direct response, where the model assumes the most likely interpretation and responds immediately, often incorrectly (Herlihy et al., 2024). All three patterns reflect the same underlying problem; the model cannot determine user intent from an incomplete context. Improving multi-turn instruction following has been pursued through demonstration-guided training (Sun et al., 2024) and through architectural frameworks that decouple intent inference from task execution by reconstructing a single-turn instruction from conversation history (Liu et al., 2026).

Return-to-go conditioning Sequence modeling approaches to offline reinforcement learning have demonstrated that conditioning on desired outcomes, rather than learning value functions, produces robust and controllable agent behavior (Chen et al., 2021a). Dense sentence representations optimized for semantic similarity have since enabled goal-conditioned reasoning in language settings (Reimers and Gurevych, 2019; Song et al., 2020), providing the computational foundation for translating outcome-conditioned sequence modeling into prompt-level context management. SeDT builds on both lines of work.

3 Method: SeDT

3.1 The Formal Parallel

The structural analogy between the Decision Transformer (Chen et al., 2021a) and SeDT is precise. In the Decision Transformer, a trajectory is a sequence of interactions between an agent and its environment recorded as a series of steps. At each step t, the agent observes a state s_t describing the current situation, takes an action a_t, and receives a signal $\hat{R}_t$ called the return-to-go, which represents the total reward the agent can still accumulate from that step onward to the end of the trajectory. By conditioning on these return-to-go values, the Decision Transformer teaches the agent to associate high-value steps with high future reward, producing the full trajectory $[(\hat{R}_1, s_1, a_1), \ldots, (\hat{R}_T, s_T, a_T)]$ as input to generate the next action. In SeDT, we draw a direct parallel to multi-turn conversation. The conversation history plays the role of the trajectory, each shard revealed in a single turn plays the role of a step, and the semantic relevance of that shard to the final output goal plays the role of reward. Just as the Decision Transformer annotates each step with how much future reward it is worth, SeDT annotates each shard with how much goal-relevant information is still to come from that turn onward, which we also call the return-to-go of that shard. The conversation history thus becomes $[(\hat{R}_1, \text{shard}_1), \ldots, (\hat{R}_T, \text{shard}_T)]$, conditioning the final answer on return-to-go values:

$$\begin{aligned} \text{DT:} \quad & [\hat{R}_1, s_1, a_1]\ [\hat{R}_2, s_2, a_2]\ \cdots \rightarrow a_T \\ \text{SeDT:} \quad & [\hat{R}_1, \text{shard}_1]\ [\hat{R}_2, \text{shard}_2]\ \cdots \rightarrow \hat{y}_T, \end{aligned}$$

where $\hat{R}_t = \sum_{t'=t}^{T-1} \text{rel}(t')$ is the cumulative relevance from turn t to the final turn T, and T is the total number of shards in the conversation. In the Decision Transformer, a high $\hat{R}_t$ tells the agent that much future reward is still achievable from this step. In SeDT, a high $\hat{R}_t$ tells the model that much goal-relevant information is still to come from this turn, and the model should carefully attend to the constraints that follow it. The parallel is not merely metaphorical; it directly motivates every design decision that follows. Figure 1 shows the overview of SeDT.

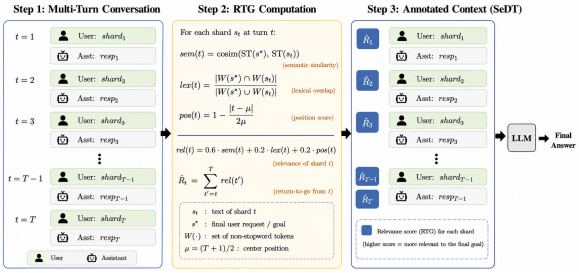

Figure 1: Overview of the SeDT pipeline.

3.2 Goal-Based Anchor and Shard Embedding

The anchor s^* is the expected output goal, which the model must produce at the final turn. We use task-typed anchors: *"Calculate and give the final numerical answer"* (math), *"Return all required function calls with correct parameters"* (actions), and *"Write a complete Python function that solves the problem"* (code). For each shard s_t with intermediate response r_t, we embed $\mathrm{ST}(s_t \| r_t)$ if $|r_t| \leq 30$ characters and $\mathrm{ST}(s_t)$ otherwise, preventing the verbose intermediate output from distorting the relevance estimate.

3.3 Three-Signal Relevance Scoring

A shard may be semantically relevant to the goal without sharing surface keywords and vice versa. To capture complementary dimensions of relevance, we combine three signals:

$$\mathrm{rel}(t) = \alpha \cdot \mathrm{sem}(t) + \beta \cdot \mathrm{lex}(t) + \gamma \cdot \mathrm{pos}(t), \quad (1)$$

with weights α=0.6, β=0.2, γ=0.2, chosen to reflect the relative discriminative power of each signal as empirically validated through the analysis of the signal contribution in Section 5. Semantic similarity to the output goal carries the most weight because it captures goal relevance that surface overlap misses, while lexical and positional signals provide complementary corrections at lower weights.

Signal 1 (Semantic) Deep alignment between the shard and the goal anchor, computed as $\mathrm{sem}(t) = \mathrm{cosim}(\mathrm{embed}(s^*), \mathrm{embed}_t)$, where $\mathrm{embed}(\cdot)$ denotes dense vector representations produced by the all-mpnet-base-v2 sentence transformer (Reimers and Gurevych, 2019) and cosim denotes cosine similarity between two vectors.

Signal 2 (Lexical) Surface keyword overlap between the goal anchor and the shard, computed as $\mathrm{lex}(t) = |W(s^*) \cap W(s_t)| / |W(s^*) \cup W(s_t)|$, where $W(s)$ denotes the set of unique words in the string s, $\cap$ is the intersection of sets and $\cup$ is the union of sets. This is the Jaccard similarity between the word sets of the anchor and the shard, capturing surface-level overlap that semantic embeddings may miss when domain-specific keywords appear verbatim across both.

Signal 3 (Positional) An inverted-U boost that counteracts both the suffix-sum bias of the RTG computation and the U-shaped positional attention of transformers (Liu et al., 2024; Vaswani et al., 2017), calculated as $\mathrm{pos}(t) = 1 - |t - \mu|/(2\mu)$, where $t \in \{0, \ldots, T-1\}$ is the zero-indexed position of the current shard, T is the total number of shards in the conversation, and $\mu = (T-1)/2$ is the midpoint of the shard indices. The intuition is straightforward; the formula assigns the highest score of 1.0 to the middle turn and decreases symmetrically towards both first and last turns. This directly counteracts the transformer's natural tendency to over-attend to the beginning and end of the context while neglecting the middle, ensuring that constraints revealed in the middle of the conversation receive a relevance boost proportional to how far they are from the endpoints. This signal directly addresses the loss-of-middle-turns failure mode documented in Laban et al. (2025). A *minimum guarantee* for the last shard ensures that the final shard, which by the construction of RTG accumulates the lowest cumulative score, receives an RTG of at least the mean RTG across all shards: $\hat{R}_{T-1} \leftarrow \max(\hat{R}_{T-1}, \frac{1}{T}\sum_t \hat{R}_t)$.

Table 1 illustrates the full RTG annotation on a real GSM8K example. Shards above the mean relevance are marked **CONFIRMED**; below-average shards are marked **UNCERTAIN**. In particular, shard t=4 has the lowest raw relevance (rel=0.12), yet the last-shard guarantee increases its RTG from 0.12 to 0.61, preventing the model from discarding the final constraint.

3.4 RTG-Conditioned Prompt and Confidence Labels

In the final turn, SeDT presents the complete history annotated with RTG and augments the system prompt with confidence labels: shards with $\mathrm{rel}(t) \geq \bar{\rho}$ are marked **CONFIRMED**; below-average shards are marked **UNCERTAIN**. This gives the model both a quantitative relevance signal and a categorical one, reinforcing the constraints to prioritize and directly targeting all four failure modes of Laban et al. (2025). **Premature commitment** is addressed by RTG reorienting attention toward the full goal-weighted history rather than

t	Shard	sem	lex	pos	rel	RTG	Label
0	What % of students in hip-hop dance?	.08	.00	.50	.15	1.03	UNCERTAIN
1	There are 20 students in the dance class	.13	.07	.75	.24	0.88	CONFIRMED
2	20% of the students chose contemporary	.11	.08	1.00	.28	0.64	CONFIRMED
3	25% of the remaining opted for jazz	.13	.07	.75	.24	0.36	CONFIRMED
4	The rest are in the hip-hop dance group	−.01	.13	.50	.12	$0.61^{\dagger}$	UNCERTAIN

Table 1: RTG annotation for a real GSM8K example with anchor *"Calculate and give the final numerical answer"*. †Raised by the last-shard minimum guarantee from raw 0.12.

the earliest shard. **Overreliance on incorrect intermediates** is reduced by UNCERTAIN labeling, which reduces the influence of low-relevance prior responses. **Loss of middle turns** is countered by Signal 3, which explicitly increases the relevance of middle turns. **Verbose drift** is prevented by excluding intermediate responses longer than 30 characters from embedding, preventing noisy outputs from inflating relevance scores.

3.5 RTG-Guided Self-Correction

When the final-turn RTG signal indicates a weak conversation, SeDT triggers a verification step. Naïve triggers such as $\hat{R}_{\text{last}} \leq \bar{R}$ are unsuitable because RTG is a suffix sum, making this condition trivially always true. Instead, we define a *weakness score*:

$$W = (-z_{\text{last}}) \times u_r, \tag{2}$$

where $z_{\text{last}} = (\text{rel}(T-1) - \bar{\rho})/\sigma_\rho$ is the normalized relevance of the final shard and u_r is the fraction of UNCERTAIN shards. The adaptive threshold $\tau = 0.5 \times (1 - \bar{\rho})$ is reduced when the global conversation is weak globally, allowing for a more aggressive correction. Self-correction occurs when $W > \tau$.

The verifier's response replaces the initial answer only via *RTG-grounded acceptance*: the replacement is accepted if and only if $a_1 \neq a_v$ and $\text{cov}(v, \mathcal{C}) > \text{cov}(r_1, \mathcal{C})$, where $\mathcal{C}$ is the CONFIRMED shard set and coverage is defined as

$$\text{cov}(y, \mathcal{C}) = \frac{1}{|\mathcal{C}|} \sum_{s \in \mathcal{C}} \mathbf{1}[\exists\, w \in W_{>3}(s) : w \in W(y)], \tag{3}$$

the fraction of confirmed shards for which at least one content word (longer than three characters) appears in the response y. This guard prevents a stochastic verifier from overwriting a correct answer simply by disagreeing.

4 Experimental Setup

Dataset and tasks. We evaluate on the `microsoft/lost_in_conversation` HuggingFace dataset (Laban et al., 2025), using all available sharded examples: actions (105 instructions from the Berkeley Function Calling Leaderboard), math / GSM8K (Cobbe et al., 2021) (103 instructions), and code (Chen et al., 2021b; Jain et al., 2024)(100 instructions; 45 HumanEval + 55 LiveCodeBench medium). These three tasks represent the binary-correctness subset of the benchmark, where the evaluation is unambiguous: functional accuracy for code, exact match for API calls, and numerical match for math.

Models. We evaluate three LLMs spanning three families: GPT-4o-mini (Achiam et al., 2023) (OpenAI), Gemini 2.5 Flash (Team et al., 2023) (Google) and Llama 3.3-70B (Grattafiori et al., 2024) (Meta), covering proprietary and open-weight models at different scales.

Evaluation protocol. Following Laban et al. (2025), we perform n=5 independent simulations per example at temperature T=1.0, with `max_tokens = 1000`. The Metrics $\bar{P}$, A, and U are computed per sample from the n run scores and averaged across the corpus. We terminate conversations early on a correct answer, preventing later incorrect turns from overwriting a correct intermediate; this is particularly consequential for code tasks, where partial specifications can yield valid partial solutions.

Sharded baseline. The Sharded baseline presents all prior shards as a plain multi-turn conversation with no relevance annotation. All other settings for LLM, the system prompt, and the scorer are identical to SeDT. Task-specific scorers follow the base paper exactly: flexible regex for math, AST-based evaluation for API calls, and test-case execution for code.

Algorithm 1 SeDT: RTG-Conditioned Multi-Turn Inference

Require: Shards $\{s_1, \ldots, s_T\}$, intermediate responses $\{r_1, \ldots, r_{T-1}\}$, goal anchor s^*, LLM $\mathcal{M}$, weights α=0.6, β=0.2, γ=0.2
Ensure: Final answer $\hat{y}_T$

for $t = 0$ **to** $T-1$ **do**
  $\text{text}_t \leftarrow s_t \| r_t$ **if** $|r_t| \leq 30$ **else** s_t
  $\mathbf{e}_t \leftarrow \text{ST}(\text{text}_t)$
end for
$\mathbf{e}^* \leftarrow \text{ST}(s^*)$
for $t = 0$ **to** $T-1$ **do**
  $\text{sem}(t) \leftarrow \text{cosim}(\mathbf{e}^*, \mathbf{e}_t)$
  $\text{lex}(t) \leftarrow |W(s^*) \cap W(s_t)| \,/\, |W(s^*) \cup W(s_t)|$
  $\mu \leftarrow (T-1)/2$
  $\text{pos}(t) \leftarrow 1 - |t - \mu| \,/\, (2\mu)$
  $\text{rel}(t) \leftarrow \alpha \cdot \text{sem}(t) + \beta \cdot \text{lex}(t) + \gamma \cdot \text{pos}(t)$
end for
for $t = T-1$ **downto** 0 **do**
  $\hat{R}_t \leftarrow \sum_{t'=t}^{T-1} \text{rel}(t')$
end for
$\hat{R}_{T-1} \leftarrow \max(\hat{R}_{T-1},\ \frac{1}{T}\sum_t \hat{R}_t)$
$\bar{\rho} \leftarrow \frac{1}{T}\sum_t \text{rel}(t)$
for $t = 0$ **to** $T-1$ **do**
  $\ell_t \leftarrow$ **CONFIRMED** **if** $\text{rel}(t) \geq \bar{\rho}$ **else** **UNCERTAIN**
end for
Build annotated history $\mathcal{H} \leftarrow \{[\hat{R}_t,\ \ell_t,\ s_t,\ r_t]\}_{t=0}^{T-1}$
$\hat{y}_1 \leftarrow \mathcal{M}(\mathcal{H},\ s_T)$
$\sigma_\rho \leftarrow \text{std}(\{\text{rel}(t)\})$, $u_r \leftarrow |\{t : \ell_t =$ **UNCERTAIN**$\}|/T$
$z_{T-1} \leftarrow (\text{rel}(T-1) - \bar{\rho}) \,/\, \sigma_\rho$
$W \leftarrow (-z_{T-1}) \times u_r$
$\tau \leftarrow 0.5 \times (1 - \bar{\rho})$
if $W > \tau$ **then**
  $\hat{y}_v \leftarrow \mathcal{M}(\mathcal{H},\ s_T,\ \text{verify})$
  $\mathcal{C} \leftarrow \{s_t : \ell_t =$ **CONFIRMED**$\}$
  if $\hat{y}_v \neq \hat{y}_1$ **and** $\text{cov}(\hat{y}_v, \mathcal{C}) > \text{cov}(\hat{y}_1, \mathcal{C})$ **then**
    $\hat{y}_1 \leftarrow \hat{y}_v$
  end if
end if
return $\hat{y}_1$

5 Results

5.1 Main Results

Table 2 and Figure 2 present the mean performance for all nine combinations of model tasks. SeDT consistently outperforms the flat-history sharded baseline on every model and every task.

In the actions task (105 examples), the gains range from +11.2 to +37.7. Gemini 2.5 Flash (Team et al., 2023) records the largest single gain (+37.7) and the highest average gain across all three tasks (+18.0 points), reaching 78.1, more than halfway back to its single-turn ceiling of 88.4. GPT-4o-mini (Achiam et al., 2023) and Llama 3.3-70B (Grattafiori et al., 2024) also improve substantially, by +26.3 and +11.2, respectively.

In math (103 examples, GSM8K (Cobbe et al., 2021)), all three models improve consistently: GPT-4o-mini (Achiam et al., 2023) gains +11.5, Gemini 2.5 Flash (Team et al., 2023) gains +10.7, and Llama 3.3-70B (Grattafiori et al., 2024) gains +8.9. In code (100 examples), the gains hold for all models: +8.4, +5.6, and +10.2 for GPT-4o-mini (Achiam et al., 2023), Gemini 2.5 Flash (Team et al., 2023) and Llama 3.3-70B (Grattafiori et al., 2024), respectively. The Code gains are smaller in absolute terms, reflecting the inherent difficulty of LiveCodeBench (Jain et al., 2024) medium problems, but SeDT improves consistently even in this challenging regime.

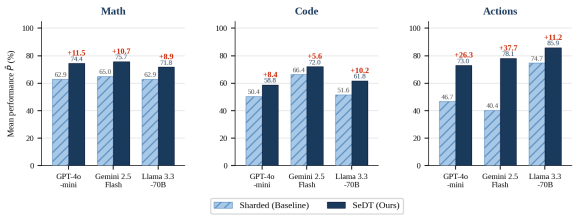

Figure 2: Mean performance $\bar{P}$ (%) for Sharded (hatched) and SeDT (solid) across three tasks and three model families. SeDT consistently outperforms the sharded baseline on all nine model-task combinations.

5.2 Reliability Analysis

Table 3 presents aptitude A and unreliability U in all nine model-task combinations. SeDT improves A on every combination, confirming that gains reflect genuine improvement in best-case capability rather than noise reduction, and reduces U in seven of nine, with the strongest reductions in math (Llama 3.3-70B (Grattafiori et al., 2024), $\Delta U = -7.6$) and actions (Gemini 2.5 Flash (Team et al., 2023), $\Delta U = -9.6$). The two exceptions Actions / Llama 3.3-70B (Grattafiori et al., 2024) ($\Delta U = +0.2$) and Code / GPT-4o-mini (Achiam et al., 2023) ($\Delta U = +2.6$) both occur where the Sharded U is already the lowest within its task, leaving limited room for further reduction — confirming that SeDT tightens score distributions precisely where the reliability improvement the lost-in-conversation literature calls for (Laban et al., 2025) is most needed.

5.3 Signal Contribution Analysis

To understand which of the three relevance signals drives performance, Table 4 reports $\bar{P}$, A, and U on a representative subset of 25 GSM8K (Cobbe et al., 2021) examples with GPT-4o-mini (Achiam et al., 2023), ablating each signal in turn.

Each single-signal variant underperforms SeDT

Model	Method	Actions	Math	Code	Average
GPT-4o-mini (Achiam et al., 2023)	Full†	94.1	88.1	75.9	86.0
	Sharded	46.7	62.9	50.4	53.3
	SeDT	**73.0**	**74.4**	**58.8**	**68.7**
	Gain	*+26.3*	*+11.5*	*+8.4*	*+15.4*
Gemini 2.5 Flash (Team et al., 2023)	Full†	88.4	90.6	97.0	92.0
	Sharded	40.4	65.0	66.4	57.3
	SeDT	**78.1**	**75.7**	**72.0**	**75.3**
	Gain	*+37.7*	*+10.7*	*+5.6*	*+18.0*
Llama 3.3-70B (Grattafiori et al., 2024)	Full†	95.0	91.7	72.0	86.2
	Sharded	74.7	62.9	51.6	63.1
	SeDT	**85.9**	**71.8**	**61.8**	**73.2**
	Gain	*+11.2*	*+8.9*	*+10.2*	*+10.1*

Table 2: Mean performance $\bar{P}$ (%) across three LLMs and three tasks. Full† is the single-turn upper bound from Laban et al. (2025); Sharded is the flat-history multi-turn baseline; SeDT is our method. Gain is the absolute improvement over Sharded.

		Sharded		**SeDT** ours		**Delta**	
Task	**Model**	A	U	A	U	ΔA	ΔU
Actions	GPT-4o-mini (Achiam et al., 2023)	54.3	14.5	**79.0**	**13.5**	+24.7	−1.0
	Gemini 2.5 Flash (Team et al., 2023)	56.4	30.7	**87.4**	**21.1**	+31.0	−9.6
	Llama 3.3-70B (Grattafiori et al., 2024)	80.6	11.6	**91.0**	11.8	+10.4	+0.2
Math	GPT-4o-mini (Achiam et al., 2023)	77.7	29.3	**88.9**	**28.9**	+11.2	−0.4
	Gemini 2.5 Flash (Team et al., 2023)	80.6	32.4	**85.8**	**23.5**	+5.2	−8.9
	Llama 3.3-70B (Grattafiori et al., 2024)	84.3	44.3	**87.6**	**36.7**	+3.3	−7.6
Code	GPT-4o-mini (Achiam et al., 2023)	63.2	24.8	**72.6**	27.4	+9.4	+2.6
	Gemini 2.5 Flash (Team et al., 2023)	80.2	27.6	**83.6**	**24.8**	+3.4	−2.8
	Llama 3.3-70B (Grattafiori et al., 2024)	69.6	36.2	**76.8**	**31.6**	+7.2	−4.6

Table 3: Aptitude A (90th percentile) and unreliability U (90th−10th percentile gap, lower is better), computed per-example and averaged across the corpus following Laban et al. (2025).

by 8.0–11.2 points in $\bar{P}$, confirming that no individual signal is sufficient. The semantic signal alone ($\bar{P}$=56.8) performs worse despite receiving the highest weight in the full model, because cosine similarity without positional correction is vulnerable to suffix-sum bias. The position signal alone ($\bar{P}$=60.0) provides the strongest single-signal baseline, consistent with middle-turn neglect being a primary failure mode. Combining all three signals ($\bar{P}$=68.0) outperforms the best uniform weighting ($\bar{P}$=63.2) by 4.8 points, confirming that α=0.6, β=0.2, γ=0.2 reflects a meaningful prior on the importance of the signal rather than the choice of incidental hyperparameter.

To isolate whether the gains are due to the RTG prompt format or the calculated relevance signal, we include a Random RTG condition that replaces the computed scores with uniform random values in $[0, 1]$ while keeping the prompt structure identical. Random RTG ($\bar{P}$=60.8) outperforms the sharded baseline, confirming that the instruction-tuned models respond to numerical prefixes from pretraining exposure (Ouyang et al., 2022; Wei et al., 2022). SeDT ($\bar{P}$=68.0) further outperforms Random RTG by 7.2 points in $\bar{P}$ and by 13.6 points in U (33.6 $\rightarrow$ 20.0), demonstrating that structured relevance adds substantial signal beyond the format alone.

5.4 Self-Correction Mechanism Analysis

We evaluated the RTG-guided self-correction mechanism on 15 randomly drawn GSM8K (Cobbe et al., 2021) examples (75 runs total, `random.seed(42)`) with GPT-4o-mini, recording Guard 1 trigger and Guard 2 acceptance decisions at each run. The verifier triggers on 28.0% of runs (21/75), indicating that $W > \tau$ fires selectively in genuinely uncertain conversations; in most triggered cases the verifier confirms the original answer without replacement, and when it disagrees Guard 2 blocks replacement unless the verifier demonstrates strictly higher confirmed-shard coverage. Across all 75 runs,

Configuration	α	β	γ	$\bar{P}$	A	U
Semantic only	1.00	0.00	0.00	56.8	70.4	29.6
Lexical only	0.00	1.00	0.00	59.2	72.8	27.2
Position only	0.00	0.00	1.00	60.0	73.6	27.2
Uniform weights	0.33	0.33	0.34	63.2	75.2	24.0
Random RTG	0.60	0.20	0.20	60.8	76.8	33.6
SeDT(ours)	0.60	0.20	0.20	**68.0**	**80.0**	**20.0**

Table 4: Signal contribution analysis on 25 GSM8K (Cobbe et al., 2021) examples with GPT-4o-mini (Achiam et al., 2023).

zero hurt cases are recorded, confirming that the mechanism introduces no errors while providing a conservative verification pathway.

5.5 Why RTG Conditioning Works Without Training

A natural question is why a model responds meaningfully to return-to-go prefixes it was not explicitly trained on. Instruction-tuned LLMs are extensively exposed to relevance-annotated text during pretraining; code comments, document outlines, meeting agendas, and editorial markup all of which condition content on explicit importance signals, suggesting RTG-style numerical prefixes fall within their learned distribution (Ouyang et al., 2022; Wei et al., 2022). Additionally, the RTG-annotated history functions as an implicit in-context demonstration, the pattern $[\hat{R}_t, \ell_t, s_t]$ repeated through turns teaches the model that higher-scored shards carry more constraint weight, without any gradient signal (Brown et al., 2020). Consistent with this, the two models with the highest gains, Gemini 2.5 Flash ($+18.0$) and GPT-4o-mini ($+15.4$), are the most capable instruction-tuned models in our evaluation, while Llama 3.3-70B records the lowest gain ($+10.1$).

5.6 Comparison with Prior Approaches

Recapitulation strategies Laban et al. (2025) evaluate two recapitulation baselines on GPT-4o-mini (Achiam et al., 2023); Recap, which includes a complete restatement of all prior shards as a final turn, and Snowball, which cumulatively restates shards at every turn. Both improve over the sharded baseline, yet fall short of the single-turn ceiling, with Recap achieving an average $\bar{P}$ of 66.5 and Snowball 61.8 across four tasks. The per-task breakdowns for these baselines are not publicly available, which excludes a direct per-task comparison. SeDT achieves 68.7 on GPT-4o-mini across our three evaluated tasks, comparing favorably to both reported averages, while requiring no modification of the conversation structure and no assumption about which turn is final.

Intent reconstruction Liu et al. (2026) propose a Mediator-Assistant framework that decouples intent inference from task execution, reconstructing ambiguous conversation history into a fully-specified single-turn instruction before passing it to the assistant. The Mediator requires task-specific historical interaction data and invokes 2–3 additional LLM calls per inference. On GPT-4o-mini (Achiam et al., 2023), it achieves $\bar{P}$ of 85.7, 77.7 and 66.9 on Actions, Math, and Code respectively (Liu et al., 2026). SeDT achieves 73.0, 74.4, and 58.8, remaining competitive on Math while requiring zero task-specific data and at most one additional LLM call. The two approaches are complementary; the Mediator addresses intent alignment through experience-driven reconstruction, while SeDT addresses context weighting through return-to-go conditioning.

6 Conclusion

We identified flat-context turn weighting as a structural root cause of multi-turn reliability collapse and proposed **SeDT**, a training-free inference-time method importing return-to-go conditioning from offline reinforcement learning. The key insight is that conversation shards map to trajectory steps and semantic relevance serves as reward. SeDT annotates each shard with a cumulative relevance score from three complementary signals and presents the annotated history at the final turn, changing no weights and requiring no training data. Across three LLMs and three tasks, SeDT outperforms the sharded baseline in all nine combinations, improving mean performance while reducing unreliability in seven. The self-correction mechanism introduces zero hurt cases.

Limitations

Several design decisions and resource constraints scope the current evaluation. We use n=5 runs per example rather than n=10 as in Laban et al. (2025), following Liu et al. (2026) who also adopt n=5 and demonstrate stable estimates at this sample size; the U metric may marginally underestimate the true variance. Due to computational constraints, we evaluated three of the six tasks in Laban et al. (2025), as the remaining tasks (Database, Data-to-Text, Summary) require SQL execution infrastructure or long-context pipelines that exceed our resource budget. Goal anchors are task-typed natural language strings that describe the expected output, analogous to system prompt design and can be specified with minimal effort for any new task, though automatic anchor generation remains future work. The signal weights α=0.6, β=0.2, γ=0.2 were validated on 25 GSM8K examples with GPT-4o-mini (Achiam et al., 2023); however, consistent gains across all nine combinations of model-task with these fixed weights provide indirect evidence of robustness. A systematic weight sweep across all tasks and model families, complete six-task evaluation, and automatic anchor generation are left to future work.

E1 - AI Assistant Disclosure

An AI writing assistant was used for paper-prose drafting, paragraph restructuring, and figure / table layout suggestions. The assistant was *not* used to generate or fabricate experimental numbers, dataset records, annotator labels, or finding interpretations. All empirical results, dataset construction scripts, model selection decisions, are author-generated and have been independently verified against the locked release. The specific assistant and version will be named in the camera-ready; anonymising the vendor here preserves double-blind review.

Ethics Statement

This work uses publicly available datasets and API-based language models. No human subjects were involved and no new data were collected. The proposed method is inference time only and introduces no additional risks beyond those of the underlying models.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, and 1 others. 2024. 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)*, pages 7421–7454.

Rachit Bansal, Aston Zhang, Rishabh Tiwari, Lovish Madaan, Sai Surya Duvvuri, Devvrit Khatri, David Brandfonbrener, David Alvarez-Melis, Prajjwal Bhargava, Mihir Sanjay Kale, and 1 others. 2025. Let's (not) just put things in context: Test-time training for long-context llms. *arXiv preprint arXiv:2512.13898*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021a. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021b. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Christine Herlihy, Jennifer Neville, Tobias Schnabel, and Adith Swaminathan. 2024. On overcoming miscalibrated conversational priors in llm-based chatbots. *arXiv preprint arXiv:2406.01633*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Live-

codebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. 2024. 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*, pages 20153–20177.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025. Llms get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*.

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

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

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Jagadeesh Rachapudi, Pranav Singh, Ritali Vatsi, Praful Hambarde, and Amit Shukla. 2026a. Repair: Interactive machine unlearning through prompt-aware model repair. *arXiv preprint arXiv:2604.12820*.

Jagadeesh Rachapudi, Ritali Vatsi, Praful Hambarde, and Amit Shukla. 2026b. Bid-lora: A parameter-efficient framework for continual learning and unlearning. *arXiv preprint arXiv:2604.12686*.

Jagadeesh Rachapudi, Ritali Vatsi, Pranav Singh, Praful Hambarde, and Amit Shukla. 2026c. Backflush: Knowledge-free backdoor detection and elimination with watermark preservation in large language models. *arXiv preprint arXiv:2605.12529*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 3982–3992.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33:16857–16867.

Yuchong Sun, Che Liu, Kun Zhou, Jinwen Huang, Ruihua Song, Wayne Xin Zhao, Fuzheng Zhang, Di Zhang, and Kun Gai. 2024. 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)*, pages 9729–9750.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 2023. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. *arXiv preprint arXiv:2309.10691*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P Xing, and 1 others. 2023a. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *arXiv preprint arXiv:2309.11998*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.