

ERGO: Entropy-guided Resetting for Generation Optimization in Multi-turn Language Models

Haziq Mohammad Khalid[‡]

Athikash Jeyaganthan

Timothy Do

Yicheng Fu

Sean O'Brien

Vasu Sharma

Kevin Zhu

Algoverse AI Research

haziqkhalid04@gmail.com , psyaj9@nottingham.ac.uk , timothy.k.do@sjsu.edu

Abstract

Large Language Models (LLMs) suffer significant performance degradation in multi-turn conversations when information is presented incrementally. Given that multi-turn conversations characterize everyday interactions with LLMs, this degradation poses a severe challenge to real world usability. We hypothesize that abrupt increases in model uncertainty signal misalignment in multi-turn LLM interactions, and we exploit this insight to dynamically realign conversational context. We introduce ERGO (Entropy-guided **R**esetting for **G**eneration **O**ptimization), which continuously quantifies internal uncertainty via Shannon entropy over next token distributions and triggers adaptive prompt consolidation when a sharp spike in entropy is detected. By treating uncertainty as a first class signal rather than a nuisance to eliminate, ERGO embraces variability in language and modeling, representing and responding to uncertainty. In multi-turn tasks with incrementally revealed instructions, ERGO yields a 56.6% average performance gain over standard baselines, increases aptitude (peak performance capability) by 24.7%, and decreases unreliability (variability in performance) by 35.3%, demonstrating that uncertainty aware interventions can improve both accuracy and reliability in conversational AI.

1 Introduction

Large Language Models (LLMs) have become the primary interface for conversational AI systems, enabling users to interact through multi-turn exchanges. However, recent research has documented a critical limitation: LLMs often get 'lost' in conversation and experience substantial performance degradation in multi-turn conversations compared to single-turn interactions (Laban et al., 2025; Gupta et al., 2024). This degradation manifests as

reduced accuracy, lower confidence, and a 112% increase in unreliability, posing significant challenges for real-world deployment (Laban et al., 2025).

While prior work has measured this degradation, existing mitigation strategies remain limited. Approaches based on task classification, retrieval, or context compression lack generality or require fine-tuning (Wu et al., 2023).

We hypothesize that spikes in model uncertainty signal moments of conversational drift and by explicitly representing this uncertainty and monitoring its fluctuations, we can detect when an LLM begins getting 'lost' in conversation. We introduce ERGO (Entropy-guided **R**esetting for **G**eneration **O**ptimization), the first practical intervention framework that dynamically monitors internal uncertainty signals and resets context when needed. ERGO computes Shannon entropy over next-token probability distributions (Malinin and Gales, 2018; Xiao and Wang, 2022) as an internal behavioral signal to detect spikes in uncertainty that indicate breakdown in comprehension. When such spikes occur, ERGO triggers entropy-guided prompt reconstruction, mitigating accumulated ambiguity and restoring coherence. Unlike static prompt engineering, ERGO's reconstruction is dynamically triggered by entropy thresholds and systematically preserves only those contextual elements that sustain both internal coherence and external task performance, discarding accumulated noise. A visual representation of this can be seen in Figure 1.

Empirical results demonstrate that targeted interventions based on uncertainty peaks not only recover task accuracy but also improve consistency across turns. In extensive simulations with incrementally revealed instructions, ERGO improves average performance by 56.6% compared to standard multi-turn baselines, increases aptitude levels by 24.7% (best-case performance capability), and

[‡]Lead Author

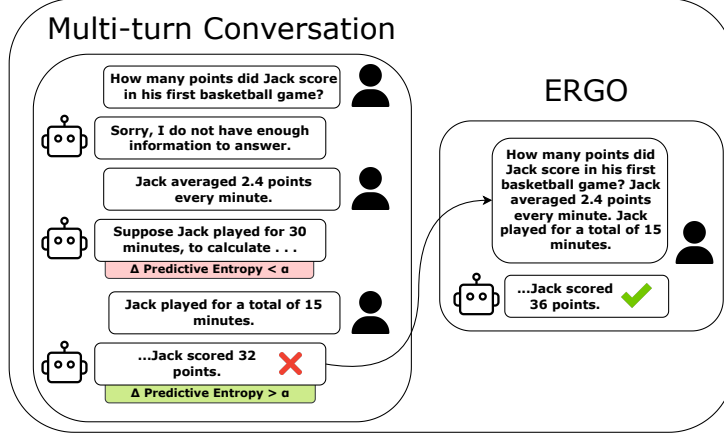


Figure 1: Illustrative comparison of a standard multi-turn conversational AI and the ERGO system

reduces the increased unreliability (variability in response consistency) observed in multi-turn settings by 35.3%. Furthermore, ERGO outperforms existing alternative strategies, and triggers resets with greater precision and timing compared to alternate baselines illustrating the potential of uncertainty-aware methods for robust conversational AI. To verify our findings and reproduce the results, please refer to the code repository found at the following link: <https://github.com/haziq-exe/ERGO>

2 Background and Related Works

Recent work has documented significant performance degradation in multi-turn LLM conversations. Laban et al. (2025) demonstrated that model performance dropped by 39% on average in multi-turn settings across six domains. Gupta et al. (2024) formalized task-switch sensitivity using probability ratios, showing how conversation history compounds model confusion. While Laban et al. (2025) managed to mitigate average performance losses by 15-20%, their approaches faced substantial verbosity and practicality constraints (Sec 5.4). Agent-based frameworks (Wu et al., 2023) explore system-level solutions but do not target fundamental model limitations during generation.

2.1 Entropy Based Uncertainty Estimation

Entropy-based uncertainty estimation provides the theoretical basis for our method, grounding ERGO’s use of internal model signals. Prior work has used predictive entropy to quantify model confidence in classification and generation tasks (Malinin and Gales, 2018; Xiao and Wang, 2022), implicitly linking internal uncertainty to exter-

nal behavior. More recent approaches extend this to semantic-level uncertainty using semantic-aware entropy measures (Kuhn et al., 2023) or trainable proxies derived from hidden representations (Kossen et al., 2024). While these methods improve semantic fidelity, they often rely on sampling or auxiliary models. In contrast, we use token-level entropy, computed directly from the model’s next-token distribution, as a low-cost proxy for real-time monitoring. Unlike prior work that applies entropy primarily for evaluation or filtering, we use it as a temporal signal to detect context degradation and trigger prompt restructuring.

2.2 Inference-Time Interventions

Inference-time control methods intervene on frozen models by manipulating internal activations, modifying output logits, or reranking candidate outputs. For example, Li et al. (2024) introduced activation-level interventions to elicit truthful answers without fine-tuning, shifting hidden states toward truthful completions. Similarly, Turner et al. (2024) developed activation engineering techniques that steer the behavior of the model by editing intermediate representations during decoding. These methods act directly on the output path of the model and often rely on internal signal manipulation.

In contrast, our approach introduces a policy layer outside of the model that monitors uncertainty and intervenes by restructuring the user’s input. We do not modify the internal computation or sampling process of the model.

2.3 Backtracking and Prompt Restructuring

Several recent approaches have explored controlled backtracking during generation. Cundy and Er-

mon (2024) augmented the decoding space with a ‘backspace’ action to revert low-probability generations, while Zhang et al. (2024) uses a special [RESET] token to discard unsafe prefixes. Other strategies such as Self-Refine (Madaan et al., 2023) allowed iterative refinement by prompting the model to critique and revise its own output. These methods operate on generated content and typically require multi-step decoding or auxiliary supervision.

Our intervention departs from this paradigm by focusing on upstream correction. Instead of rewriting the model’s response, we update the user’s prompt to recover task coherence, using rising entropy as the intervention trigger. This shifts the optimization target from output correction to input re-specification, which is more lightweight and avoids cumulative reasoning errors. To our knowledge, this is the first method that uses entropy-based signals to restructure user input mid-conversation, rather than adjusting the model’s internal behavior or downstream output.

3 Entropy-Guided Context Resetting

3.1 Rise in Average Token Level Entropy

At each turn of the conversation, the average token-level entropy is calculated by measuring the uncertainty of the model’s token probability distribution when generating each token in its output.

Suppose the model produces a sequence of tokens t_1, t_2, \dots, t_n at a given turn. For each token t_i , the model assigns a probability distribution P_i over the vocabulary V , where $P_i(v)$ is the probability assigned to token $v \in V$ at position i .

The entropy at position i is computed as:

$$H_i = - \sum_{v \in V} P_i(v) \log P_i(v)$$

The average token-level entropy \bar{H} for the turn (covering n generated tokens) is then:

$$\bar{H} = \frac{1}{n} \sum_{i=1}^n H_i$$

This metric quantifies the model’s overall uncertainty when generating the turn. Higher \bar{H} indicates greater uncertainty and a more diffuse token distribution, while the lower \bar{H} indicates more confident and peaked predictions (Malinin and Gales, 2018; Xiao and Wang, 2022).

For each subsequent turn t in the conversation, the change in average token-level entropy is calculated to monitor fluctuations in model uncertainty.

Let $\bar{H}^{(t)}$ denote the average token-level entropy at turn t , as defined previously.

The change in predictive entropy between consecutive turns is defined as:

$$\Delta \bar{H}^{(t)} = \bar{H}^{(t)} - \bar{H}^{(t-1)}$$

A positive $\Delta \bar{H}^{(t)}$ indicates that the uncertainty of the model has risen relative to the previous turn.

3.2 Threshold-Based Trigger for Context Reset

A predefined, calibrated entropy change threshold τ is established. When the change in predictive entropy satisfies the following condition:

$$\Delta \bar{H}^{(t)} > \tau$$

The system deems that the uncertainty of the model is rising beyond an acceptable margin. This is interpreted as a signal that the evolving conversation context may be inducing compounding uncertainty or drift. A detailed analysis of the threshold selection process is provided in Appendix A, while an analysis of ERGO’s sensitivity to entropy thresholds is provided in Appendix B.

3.3 Context Reset Protocol

Upon detection of $\Delta \bar{H}^{(t)} > \tau$, an automated context reset protocol is initiated. This protocol proceeds in the following steps:

I. Prompt Rewriting:

The user’s inputs up to turn t are provided to the model. The model is asked to rewrite these inputs into a single-turn, optimized prompt that preserves relevant task information while reducing ambiguity and redundancy.

II. Isolated Generation (Simulate New Chat):

The rewritten prompt is passed into a new instance of the model, simulating a stateless chat environment with no memory of prior turns. The model then generates a response R_{opt} to this rewritten prompt.

III. Branch Continuation:

A new dialogue branch is created that begins from the rewritten prompt and response. This maintains continuity from the optimized state rather than the potentially degraded original context.

FULL	SHARDED	ERGO
Solve the following problem using all the information given to you:	How old was Bill when he first saw Comet Halley?	How old was Bill when he first saw Comet Halley?
- How old was Bill when he first saw Comet Halley?	Comet Halley takes 75 years to complete one orbit around the sun	Comet Halley takes 75 years to complete one orbit around the sun
- Comet Halley takes 75 years to complete one orbit around the sun	Bill's dad was 30 years old when he saw the comet	Bill's dad was 30 years old when he saw the comet
- Bill's dad was 30 years old when he saw the comet	Bill saw Comet Halley for the second time when he was three times as old as his dad was when his dad saw it	Bill saw Comet Halley for the second time when he was three times as old as his dad was when his dad saw it
- Bill saw Comet Halley for the second time when he was three times as old as his dad was when his dad saw it	MODEL ANSWER: 75 ✗	RESET OCCURS AFTER ROW
MODEL ANSWER: 15 ✓		Bill's dad was 30 years old when he saw Comet Halley. The comet takes 75 years to complete one orbit around the sun. Bill saw Comet Halley for a second time when he was three times as old as his dad was when his dad saw it. How old was Bill when he first saw Comet Halley?
		MODEL ANSWER: 15 ✓

Figure 2: Example Llama3.1-8B run on a GSM8K question with FULL, SHARDED and ERGO settings. Each row represents a separate prompt given to the model while each table represents a context window.

4 Experimentation Background

4.1 Simulation Scale & Parameters

Our simulation follows the protocol of Laban et al. (2025) with the only change being the implementation of ERGO. We evaluate a suite of five leading instruction-tuned LLMs: **Phi-4** (Abdin et al., 2024), **LLaMA 3.1-8B Instruct** (Grattafiori et al., 2024), **GPT-4o** (Hurst et al., 2024), **GPT-4.1** (OpenAI, 2025), and **GPT-4o-mini** (OpenAI, 2024). All models are used in their publicly released variants without additional fine-tuning.

Generation settings are standardized across models with temperature set to 1.0. For entropy calculations, we note an important methodological constraint: OpenAI models provide access to only the top-20 logprobs through their API. This limitation affects the precision of entropy estimates, particularly for tasks with shorter responses such as *Actions* and *Data-to-text* (Sec 4.2), where the restricted probability space may not capture the full uncertainty of the model’s predictions.

We conduct 3 independent simulation runs for each dataset using 100-question samples, with the exception of the Data-to-text dataset (Sec 4.2), for which evaluations were performed on a 50-question subset over 3 runs. All other experimental settings and baseline figures are adopted directly from Laban et al. (2025).

We compare three settings:

FULL: Simulates a single-turn, fully-specified conversation using the sharded instruction. The shards are combined into a single bullet-point list (one shard per line), prefaced by a direc-

tive to complete the task using all listed points. This setting serves as an upper bound for performance, providing a target for evaluating how closely multi-turn intervention methods can approximate single-turn optimality.

SHARDED: Sequential shard presentation as in the original (Laban et al., 2025) LLMs-lost-in-conversation experiment.

ERGO: Our entropy-guided reset mechanism applied upon exceeding the entropy threshold.

Figure 2 provides an example of a run on each setting. This evaluation isolates the effect of ERGO relative to both single-pass and original multi-turn baselines.

4.2 Tasks

We evaluated models on five representative generation tasks, each framed as a multi-turn interaction over sharded instructions and augmented them with our entropy-guided context resetting method (Section 3). For each task, we used 220-325 constructed prompts from the datasets created by Laban et al. (2025). We simulate a multi-turn conversation, feeding the model one shard at a time. At each assistant turn, we compute the average token-level entropy and track its change $\Delta \bar{H}^{(t)}$. Whenever $\Delta \bar{H}^{(t)}$ exceeds the calibrated threshold τ , we invoke our reset protocol - prompt rewriting, isolated regeneration, branch continuation - before continuing.

Below we briefly summarize what the assistant must do in each task:

CODE: Convert natural-language problem description into a correct Python function. Outputs

are validated by executing against the reference test suite (Chen et al., 2021; Jain et al., 2024).

DATABASE: Given a database schema and a user request, generate an SQL query that returns the requested data. Correctness is checked by running the query on the Spider-derived database (Yu et al., 2018).

ACTIONS: Given API schemas plus high-level user instruction, emit valid code-style API calls that fulfill the intent. This is verified against the Berkeley Function Calling Leaderboard definitions (Yan et al., 2024).

DATA-TO-TEXT: Take a structured data table and metadata and write a single caption that highlights its key insight. Adapted from ToTTo and evaluated using BLEU (scaled 0-100) (Parikh et al., 2020; Papineni et al., 2002).

MATH: Solve an elementary math story problem by carrying out each arithmetic step and returning the numeric result. Simulates day-to-day problems LLMs may be tasked with by users. GSM8K problems were used and scored by exact match (Cobbe et al., 2021).

4.3 Metric Selection

We assess LLM performance in multi-turn tasks by repeating simulations for each instruction and collecting success scores from multiple runs, following Laban et al. (2025). Each score, ranging from 0 to 100, reflects task success.

4.4 Per-Run Scoring

I. Binary-Correctness Tasks (Code, Database, API, Math): A correct response at any turn yields a score of 100, and the run ends. Otherwise, the score is 0.

II. Refinement Task (Data-to-Text): The final output is evaluated using BLEU, rescaled to 0-100.

4.5 Aggregate Metrics

From the scores collected across the 3 runs, we compute three metrics:

- **Average Performance (\bar{P}):** Average performance per instruction for a given task.

- **Aptitude (A^{90}):** 90th-percentile score, measures a model’s peak capability, indicating its potential to deliver high-quality results in critical multi-turn tasks. Averaged across all tasks.

- **Unreliability (U_{10}^{90}):** Difference between 90th and 10th percentiles, quantifies response variability, where lower values reflect greater consistency, essential for user trust and system reliability in long-horizon interactions. Averaged across all tasks.

Formulae and more information on metrics is available in Appendix D.

5 Results & Discussion

5.1 Aptitude and Unreliability Improvements

Figure 3 shows that ERGO demonstrates exceptional gains in aptitude, often exceeding single-turn performance levels, while substantially reducing unreliability compared to multi-turn baselines, two metrics introduced by (Laban et al., 2025) to capture model consistency across conversations.

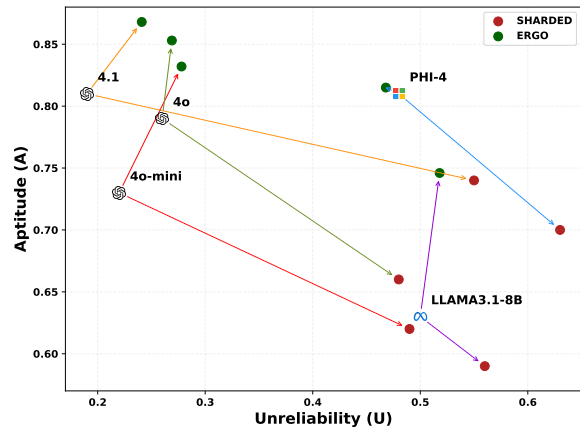


Figure 3: Effect of SHARDED and ERGO on Aptitude and Unreliability. Icons represent models FULL performance. Green dots represent performance with ERGO while red dots represent SHARDED performance

These results indicate that our intervention not only fully recovers the aptitude lost in the transition from single-turn to multi-turn settings and achieves aptitude levels exceeding single-turn baselines, but also makes behavior significantly more

Model	Code			Database			Actions			Data-to-Text			Math		
	📄	📁	🔄	📄	📁	🔄	📄	📁	🔄	📄	📁	🔄	📄	📁	🔄
🦋 Llama3.1-8b	21.2	21.7	52.0 [↑]	47.7	25.9	64.3 [↑]	83.0	45.5	60.0 [↑]	15.7	13.3	12.3 ⁺	62.6	37.4	65.7 [↑]
🌀 4o-mini	66.7	50.3	66.7 [↑]	90.7	40.2	93.3 [↑]	92.2	52.4	92.0 [↑]	31.2	19.8	22.0 [↑]	88.0	58.7	85.0 [↑]
🌈 Phi-4	48.4	39.1	55.0 [↑]	79.6	33.1	62.0 [↑]	76.0	34.1	65.7 [↑]	28.6	23.2	28.0 [↑]	90.4	52.5	85.3 [↑]
🌀 4.1	88.7	72.6	81.7 [↑]	86.5	46.0	96.0 [↑]	98.5	62.9	84.7 [↑]	54.4	28.6	31.0 [↑]	89.7	70.7	91.7 [↑]
🌀 4o	82.9	61.3	76.3 [↑]	91.7	42.3	95.7 [↑]	97.1	65.0	82.0 [↑]	32.2	20.5	27.0 [↑]	91.9	67.9	89.3 [↑]

Table 1: Average Performance \bar{P} comparison across three settings: 📄 **FULL** (single-turn), 📁 **SHARDED** (multi-turn baseline), and 🔄 **ERGO** (multi-turn with entropy-guided resetting). Arrow represents change in performance for 🔄 relative to 📁, with arrow size representing magnitude of change.

stable compared to multi-turn settings across repeated trials. When comparing to standard sharded conversations, the average aptitude across models rose by 24.7%, achieving performance levels that surpass single-turn baselines, while unreliability declined by 35.3% compared to multi-turn settings.

5.2 Average Performance Gains

In addition to aptitude and unreliability improvements, Table 1 shows that ERGO delivers substantial performance improvements across all models compared to baseline multi-turn setups. By detecting moments of confusion and restarting interactions, models avoid becoming "lost" in conversational flow. Nearly every dataset and model combination shows increased average success rates, with performance improving by 56.6% on average and several model-task combinations achieving over 100% gains compared to original multi-turn baselines.

Models frequently exceeded single-turn baseline performance in both average performance and aptitude as our method only corrects derailment when calculated confusion rises significantly. This preserves the model’s ability to iteratively reason and refine responses across shards while preventing the compounding errors typical in prolonged multi-turn contexts. This approach effectively merges both paradigms’ strengths: single-turn stability and clarity when needed, and iterative compositional reasoning when the model remains on track.

Moreover, performance on the 📄 Data-to-Text task improves over the multi-turn baseline, though less substantially than in other datasets. This is partly due to model-specific constraints. **LLaMA**

3.1-8B struggles to rewrite large, structured prompts effectively (e.g., full tables), limiting the benefit of consolidation. **GPT models** face difficulties in triggering resets, as entropy estimates are less reliable, only top-20 log-probabilities are available, and outputs are typically short, reducing entropy sensitivity. **Phi-4** performs best, nearing single-turn levels, likely because it supports accurate entropy tracking and handles prompt rewriting more effectively. These results indicate model-dependent limitations in applying our method to high-input-structure tasks.

5.3 Evaluating Entropy-Guided Resets vs. Random Resets and Fixed Resets

We compared entropy-based context resets against random and fixed-interval baselines using Llama3.1-8B across three tasks: 📄 *Database*, 📁 *Actions*, and 📄 *Math*. In these ablations, we retained all experimental settings from the main condition, with the only change being that each metric was tested on 50 question samples instead of 100. The random baseline used uniformly random triggers with unconstrained reset frequency. The fixed baseline triggered resets every five shards (quintet reset), matching the average reset frequency of Llama3.1-8B observed in our ERGO system. For more information on computation and average reset frequency across models, please refer to Appendix C.

The results, visualized in Figure 4, demonstrate a clear advantage for ERGO over baseline approaches. Entropy-guided resets consistently outperformed both random and fixed reset strategies while demonstrating adaptive scaling behavior. In

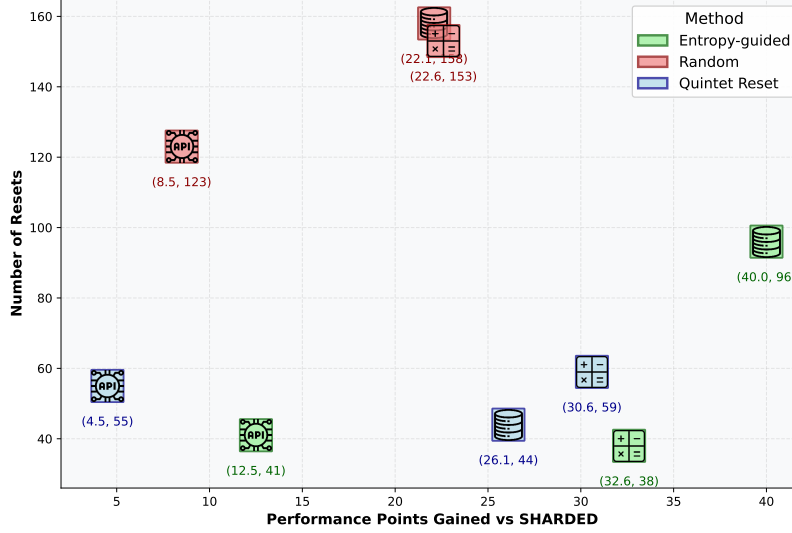


Figure 4: Comparison of performance point gains (percentage-point increase in accuracy relative to SHARDED) and number of resets across entropy-guided, random, and quintet reset methods on Database, Actions, and Math tasks. Icons represent their respective task with their color determining method used.

the Database task, ERGO achieved a performance gain of 0.400 using 96 resets, compared to the quintet baseline’s 0.261 gain with only 44 resets. This demonstrates the system’s ability to increase intervention frequency when encountering greater model uncertainty. Conversely, in the Actions task, ERGO required only 41 resets, fewer than both baselines, while still achieving superior performance (0.125 gain versus 0.045 and 0.085 for random and fixed approaches, respectively). This adaptive behavior indicates that entropy guided resets effectively allocate computational resources by intervening only when necessary, scaling both up and down based on task complexity and model confusion levels.

The primary risk posed by resets is semantic drift. Poorly timed or excessive context rewriting can lose critical details through increased abstraction, compromising semantic faithfulness to the original input (Dreyer et al., 2023). This degradation in semantic faithfulness can offset or even negate the benefits of resetting. Furthermore, resets incur computational overhead; each reset involves having two additional forward passes through the model. Together, these considerations underscore why the frequency and timing of resets must be carefully controlled. Not only to avoid wasted com-

putation, but, more critically, to prevent semantic degradation.

5.4 Comparison to Existing Intervention Strategies






To contextualize the effectiveness of **ERGO**, we compare its performance against two alternative strategies introduced by Laban et al. (2025): **SNOWBALL** and **RECAP**. Both methods attempt to mitigate information loss in multi-turn settings by explicitly reintroducing previously seen content.

SNOWBALL: Reiterates all prior shards at each new turn, effectively growing the prompt cumulatively. This ensures full task visibility at each step but leads to increasing context length and potential repetition issues.

RECAP: Reiterates all prior shards only at the final turn. While more efficient, the authors note that this strategy is impractical in real-world deployments, since the system would not know *prior* when the final user input will occur.

Our method significantly outperforms both **SNOWBALL** and **RECAP** across both model variants. For GPT-4o-mini, ERGO nearly closes the gap between SHARDED and FULL baselines entirely, while for GPT-4o, ERGO performs well above competing approaches and within 3.2 points

Model	FULL	SHARDED	SNOWBALL	RECAP	ERGO
GPT-4o-mini	73.8	44.3	54.0	57.7	71.8
GPT-4o	79.2	51.4	57.4	66.3	75.6

Table 2: Comparison of average performance across  Code,  Database,  Actions,  Data-to-Text and  Math tasks.

of the full information upper bound, as shown in Table 2. ERGO’s advantages over static repetition-based strategies are twofold: it prevents input bloating at each iteration unlike SNOWBALL, and operates without requiring prior knowledge of the final input unlike RECAP.

5.5 Evaluating Length Bias in Entropy-Based Reset Triggers

One potential concern regarding ERGO’s entropy-based reset mechanism is whether it inadvertently functions as a proxy for response length. Specifically, since entropy is calculated over token probability distributions, it is plausible that longer outputs, which involve more tokens and potentially more diffuse distributions, may naturally exhibit higher entropy. If true, this would raise the possibility that ERGO’s resets are effectively triggered by length increases rather than genuine uncertainty spikes, undermining the validity of entropy as an internal behavioral signal.

We analyze response behavior from the Phi-4 model across all tasks and questions used in the main evaluation suite. For each turn t in a given multi-turn conversation, we compute two quantities relative to the previous turn: the change in average token-level entropy, $\Delta\bar{H}(t)$, and the change in response length, $\Delta L(t)$, measured in tokens.

We evaluate the relationship between these using two standard correlation metrics: Spearman’s rank correlation coefficient (ρ), which captures monotonic associations without assuming linearity (Spearman, 1904), and Pearson’s correlation coefficient (r), which quantifies the strength of linear correlation (Pearson, 1895). The results for the Phi-4 model are summarized in Table 3.

The Spearman result indicates no meaningful monotonic relationship between changes in entropy and length. The Pearson coefficient, while statistically significant due to the large sample size, has

negligible magnitude and a negative sign, indicating no positive linear correlation.

These findings demonstrate that entropy fluctuations are not systematically associated with output length changes in the Phi-4 model. This supports the claim that ERGO’s reset mechanism is not driven by verbosity or token count, but rather by internal signals of model uncertainty. Entropy-based resets therefore retain validity as an independent control signal rather than acting as a surrogate for response length.

	Coefficient	p-value
Spearman’s ρ	−0.0143	0.4525
Pearson’s r	−0.0796	2.7×10^{-5}

Table 3: Correlation between changes in entropy and response length for the Phi-4 model.

6 Conclusion

Our results show that ERGO effectively mitigates multi-turn LLM performance degradation by using Shannon entropy to detect model confusion and trigger prompt restructuring. Despite its simplicity, Shannon entropy serves as a reliable signal for targeted context consolidation, minimizing unnecessary resets. ERGO consistently outperformed existing methods, achieving 56.6% performance gains over standard baselines, improving aptitude by 24.7%, and reducing unreliability by 36.3%. Correlation analysis confirmed that entropy-based resets reflect genuine model uncertainty rather than response length. As a practical, model-agnostic framework, ERGO enhances conversational coherence in real-world deployments, with future work focused on advanced context consolidation strategies such as multi-stage summarization and adaptive techniques for long-form conversations.

Limitations

While ERGO achieves significant improvements in multi-turn performance via entropy-guided resets, certain avenues for future work remain.

Context Simplification: ERGO’s resets currently consolidate only *user* inputs, omitting *assistant* responses. This design enables lightweight, stateless resets but limits fidelity in open-ended dialogues where assistant turns introduce key entities or reasoning steps. Without full dialogue trace consolidation, resets may discard critical context.

Threshold Adaptation: ERGO uses model-specific entropy thresholds calibrated on GSM8K, that are then fixed across datasets. While this methodology has shown to have inherit sensitivity and adapt to model capabilities (Appendix A & B). More dynamic or task-aware thresholding could improve precision.

These limitations represent natural progressions for ERGO toward broader, more general-purpose deployment. They do not challenge the core mechanism but point to extensions that scale the system into richer, more adaptable dialogue settings.

References

- M. Abdin, J. Aneja, H. Behl, S. Bubeck, R. Eldan, S. Gunasekar, M. Harrison, R. J. Hewett, M. Javaheripi, P. Kauffmann, and 1 others. 2024. [Phi-4 Technical Report](#). In *arXiv preprint*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Girish Hernandez, Chelsea Edwards, Yuri Burda, Nicholas Joseph, and 1 others. 2021. 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, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Chris Cundy and Stefano Ermon. 2024. [Sequence-match: Imitation learning for autoregressive sequence modelling with backtracking](#). *Preprint*, arXiv:2306.05426.
- Markus Dreyer, Mengwen Liu, Feng Nan, Sandeep Atluri, and Sujith Ravi. 2023. [Evaluating the tradeoff between abstractiveness and factuality in abstractive summarization](#). In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2089–2105, Dubrovnik, Croatia. Association for Computational Linguistics.
- A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, and 1 others. 2024. [The LLaMA 3 Herd of Models](#). In *arXiv preprint*.
- Akash Gupta, Ivaxi Sheth, Vyas Raina, Mark Gales, and Mario Fritz. 2024. [LLM Task Interference: An Initial Study on the Impact of Task-Switch in Conversational History](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford, and 1 others. 2024. [GPT-4o System Card](#). In *arXiv preprint*.
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.
- Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. 2024. [Semantic entropy probes: Robust and cheap hallucination detection in llms](#). *arXiv preprint arXiv:2406.15927*.
- Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*.
- Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025. [LLMs Get Lost In Multi-Turn Conversation](#). In *Proceedings of the 2025 Conference on Language Modeling (COLM)*.
- Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024. [Inference-time intervention: Eliciting truthful answers from a language model](#). *Preprint*, arXiv:2306.03341.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. [Self-refine: Iterative refinement with self-feedback](#). *Preprint*, arXiv:2303.17651.
- Andrey Malinin and Mark Gales. 2018. Predictive uncertainty estimation via prior networks. *Advances in neural information processing systems*, 31.

OpenAI. 2024. [OpenAI o3 and o4-mini System Card](#).

OpenAI. 2025. [Introducing gpt-4.1 in the api](#).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. [Bleu: a Method for Automatic Evaluation of Machine Translation](#). In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Ankur P Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. [ToTTo: A controlled table-to-text generation dataset](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186.

Karl Pearson. 1895. [Note on regression and inheritance in the case of two parents](#). *Proceedings of the Royal Society of London*, 58:240–242.

Charles Spearman. 1904. [The proof and measurement of association between two things](#). *The American Journal of Psychology*, 15(1):72–101.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. 2024. [Steering language models with activation engineering](#). *Preprint*, arXiv:2308.10248.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2023. [AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation](#). In *Proceedings of the 2023 Conference on Language Modeling (COLM)*.

Yuxia Xiao and William Yang Wang. 2022. Uncertainty quantification with pre-trained language models: A large-scale empirical analysis. *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7608–7621.

Fanjia Yan, Huanzhi Mao, Charlie Cheng-Jie Ji, Tianjun Zhang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Berkeley function-calling leaderboard. <https://gorilla.cs.berkeley.edu/leaderboard.html>.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, and 1 others. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of EMNLP*, pages 811–820.

Yiming Zhang, Jianfeng Chi, Hailey Nguyen, Kartikeya Upasani, Daniel M. Bikel, Jason Weston, and Eric Michael Smith. 2024. [Backtracking improves generation safety](#). *Preprint*, arXiv:2409.14586.

A Threshold Selection Procedure






Model Name	Version	τ	Percentile	Provider
 Phi-4 A	N/A	0.1	90th	HuggingFace
 Llama3.1-8b	N/A	0.03	65th	HuggingFace
 GPT-4.1	gpt-4.1-2025-04-14	0.2	90th	OpenAI API
 GPT-4o-mini	gpt-4o-mini-2024-07-18	0.2	85th	OpenAI API
 GPT-4o	gpt-4o-2024-08-06	0.3	90th	OpenAI API

Table 4: Model versions, thresholds, and calibration percentiles used in our experiments. (Versions included where applicable.)

To determine appropriate entropy thresholds (τ) for triggering context resets, we conducted a calibration procedure specific to each model. The goal was to identify a rise in entropy that reliably signals when a model is ‘lost’ in the conversation, that is, when its internal uncertainty increases sharply, suggesting that it is struggling to integrate or reason over the accumulated context.

For each model, we selected a held-out subset of approximately ~ 80 shard-level examples from the GSM8K dataset. These examples were drawn from outside the final evaluation set to avoid contamination, with GSM8K being chosen due to its hybrid structure, requiring both reasoning and natural language generation. We then ran each model in a standard multi-turn setting over these shards and computed the change in average token-level predictive entropy at each turn.

From the resulting distribution of entropy rises, we selected a threshold based on a percentile aligned with the model’s baseline aptitude on GSM8K. For instance, since GPT-4.1 achieves a baseline aptitude of $\sim 90\%$ on GSM8K in single-turn settings, we selected the 90th percentile of the entropy rise distribution as its reset threshold. The underlying rationale was to calibrate the threshold so that only the most atypical (high-entropy)

turns, those statistically associated with likely failure, would trigger an intervention. Details of the models used, including their version identifiers, selected entropy thresholds, and corresponding calibration percentiles, are summarized in Table 4.

Once determined, this threshold was fixed across all datasets for a given model. We made this decision intentionally, as our goal was to evaluate the feasibility of a general-purpose, model-specific threshold rather than tuning thresholds for each dataset individually. This “one-size-fits-all” approach allows for a more robust and realistic assessment of whether entropy-based context resets can generalize across tasks without requiring per-task adjustment.

Interestingly, while both GPT-4.1 and Phi-4 shared the same 90th percentile threshold, Phi-4 triggered significantly more resets during evaluation. This was due to Phi-4’s strong performance on GSM8K but much weaker performance on the broader set of tasks. This divergence illustrates that the system remains sensitive to task-specific confusion, with the number of resets scaling appropriately even under a fixed, model-specific threshold, highlighting the adaptive behavior of the method across domains. More information on number of resets incurred available in Appendix C.

B Sensitivity to Entropy Threshold (τ)

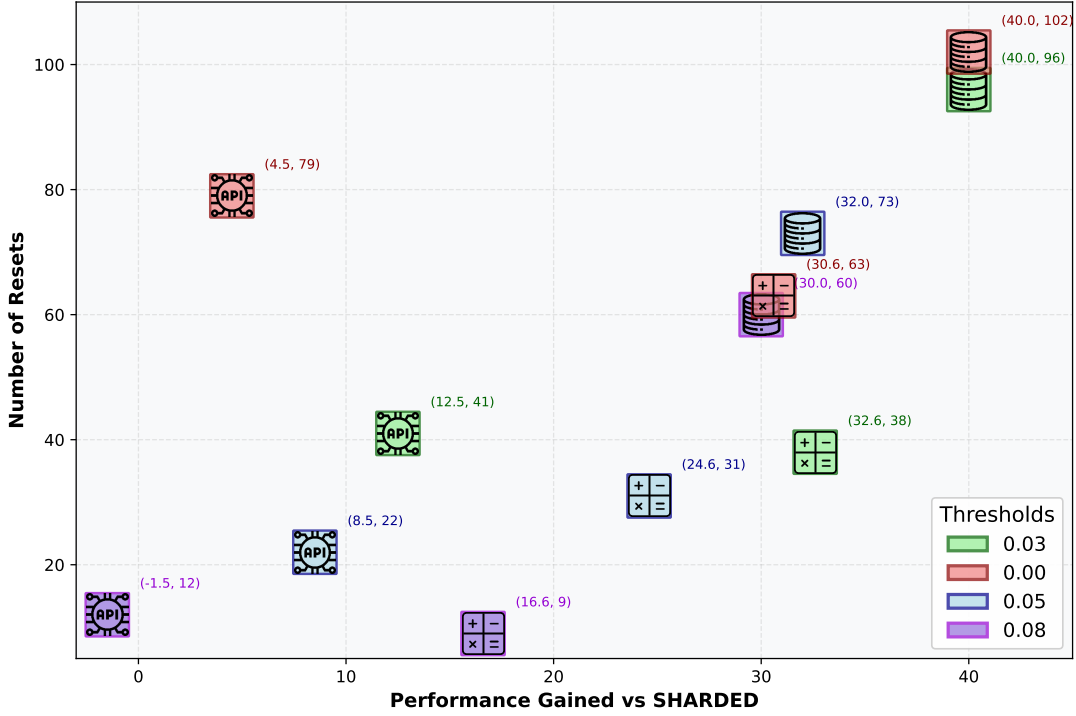


Figure 5: Comparison of maximum performance points gains (highest increase in accuracy when compared to ϕ) and number of resets between different thresholds across Database, Actions, and Math tasks.

To evaluate the sensitivity of our method to the entropy threshold parameter τ , we conducted an ablation study using the same controlled setup described in Section 5.3 with the Llama3.1-8B model on the *Database*, *Actions*, and *Math* tasks. The only variable changed in this study was the value of τ , the threshold used to trigger entropy-guided resets. We tested four settings: $\tau \in \{0.00, 0.03, 0.05, 0.08\}$, where 0.03 corresponds to the threshold selected for the main experiments.

The results, visualized in Figure 5 showed a clear performance peak at $\tau = 0.03$, which consistently achieved the highest gains across all tasks. This setting struck a balance between reactivity and restraint, triggering resets selectively at moments of genuine confusion without introducing excessive rewrites that risk semantic drift. In contrast, the lowest threshold $\tau = 0.00$ resulted in the highest

number of resets and either matched or underperformed the 0.03 setting, suggesting that overly aggressive resetting is not beneficial and may lead to instability due to frequent context rewrites.

At the other extreme, the highest threshold $\tau = 0.08$ yielded the fewest resets and consistently underperformed, likely due to failing to intervene even when the model was demonstrably confused. The intermediate value $\tau = 0.05$ behaved as expected, yielding results that were approximately midpoint between 0.03 and 0.08 in both performance and reset count.

Taken together, these findings support the robustness of our selected threshold and highlight the importance of calibrating reset triggers to maintain a balance between informativeness and intervention overhead.

C Computational Cost and Reset Overhead Analysis

Model	Average Performance	\sim Shards per Reset	Threshold Percentile
GPT-4o	75.6	51	92nd
GPT-4.1	77.2	38	90th
GPT-4o-mini	71.8	29	85th
Phi-4	59.2	7	90th
Llama3.1-8B	50.9	5	63rd

Table 5: Average Performance with ERGO along with the number of shards before reset occurs for each model and its threshold percentile, measured as an average across all datasets.

A key consideration in deploying entropy-guided context resets is the computational overhead they introduce. In our system, two sources of computational cost must be considered: (1) the cost of computing predictive entropy at each turn, and (2) the cost incurred when a context reset is triggered.

Entropy Computation Cost: While more advanced measures of model uncertainty such as semantic entropy require sampling multiple outputs over the same input (Kuhn et al., 2023), our method uses token-level Shannon entropy, which is extracted directly from the next-token probability distribution during generation. This choice imposes negligible additional cost beyond standard decoding and was selected for its practicality and compatibility with real-time systems.

Reset Overhead: Each reset introduces two additional forward passes through the model: one to rewrite the accumulated user context into a consolidated prompt, and a second to respond to that prompt. This introduces latency and compute proportional to the number of resets triggered per run. Table 5 showcases the average performance of models with ERGO along with the approximate number of shards per reset and the selected threshold percentile for each model. Averaged across all datasets, one question equates to ~ 6 shards.

These results reflect the adaptive nature of the system: more capable models (e.g., GPT-4.1, GPT-

4o) experience fewer high-entropy turns and thus require fewer resets, minimizing overhead. Conversely, less capable models like Phi-4 trigger resets more frequently, aligning with their observed confusion.

Prompt Length Reduction: An additional consequence of context resets is that they tend to truncate the context window, potentially removing stale or redundant information. Across all runs, the average token length of model prompts for questions where resets occurred was 260 tokens, compared to 309 tokens in questions where no resets were triggered. While this reduction does not eliminate the cost of the reset itself, it may partially offset it by reducing input size in subsequent turns.

Retrieval-Augmented Consolidation (Future Work): More advanced consolidation techniques, such as retrieval-augmented synthesis, could further improve the quality of resets but would introduce additional retrieval and ranking costs. We leave the exploration of such hybrid architectures to future work.

Taken together, these results indicate that while entropy-guided resets do introduce compute overhead via additional forward passes, the system remains adaptive. Reset frequency scales with model confusion, and thresholds derived from a single reasoning heavy dataset generalize effectively across diverse tasks.

D Metrics

D.1 Metric Selection

LLMs employ a stochastic decoding process, yielding different outputs even under fixed prompts and sampling parameters. We leverage this by repeating our multi-turn simulation on each sharded instruction and observing the resulting success scores. Let

$$S = \{S_i\}_{i=1}^N$$

be the set of scores from N independent runs on a single instruction, where each $S_i \in [0, 100]$ measures task success at the end of that simulation.

D.1.1 Per-run scoring:

I. Binary-correctness tasks (Code, Database, API, Math): At each turn, we evaluate the model’s response; if it produces a correct solution at any turn, we immediately assign $S_i = 100$ and terminate that run. If no turn yields a correct answer, $S_i = 0$.

II. Refinement task (Data-to-Text): We compute the native metric (BLEU for data-to-text; joint coverage/attribution score for summarization) on the final generated output and rescale it to $[0, 100]$.

D.1.2 Aggregate metrics

From the per-run scores S , we define three summary statistics, following the methodology from [Laban et al. \(2025\)](#):

$$\bar{P} = \frac{1}{N} \sum_{i=1}^N S_i \quad (1)$$

$$A^{90} = \text{percentile}_{90}(S) \quad (2)$$

$$U_{10}^{90} = \text{percentile}_{90}(S) - \text{percentile}_{10}(S) \quad (3)$$

- \bar{P} (Average Performance): An unbiased estimate of the model’s mean score on an instruction.

- A^{90} (Aptitude): Estimates the 90th-percentile performance, reflecting what one can achieve in the top decile of runs.

- U^{90} (Unreliability): Measures the gap between the 90th and 10th percentiles, capturing the degree of stochastic variability in outputs.

Aptitude and Unreliability are computed per instruction and then averaged over the full set of tasks. Binary-correctness accuracy is mapped onto the 0–100 scale to ensure every task’s score aligns.