Double Prompting: How Prompt Repetition Improves LLM Accuracy Without Extra Cost

Kabui, Charles

2026-02-13

Table of Contents

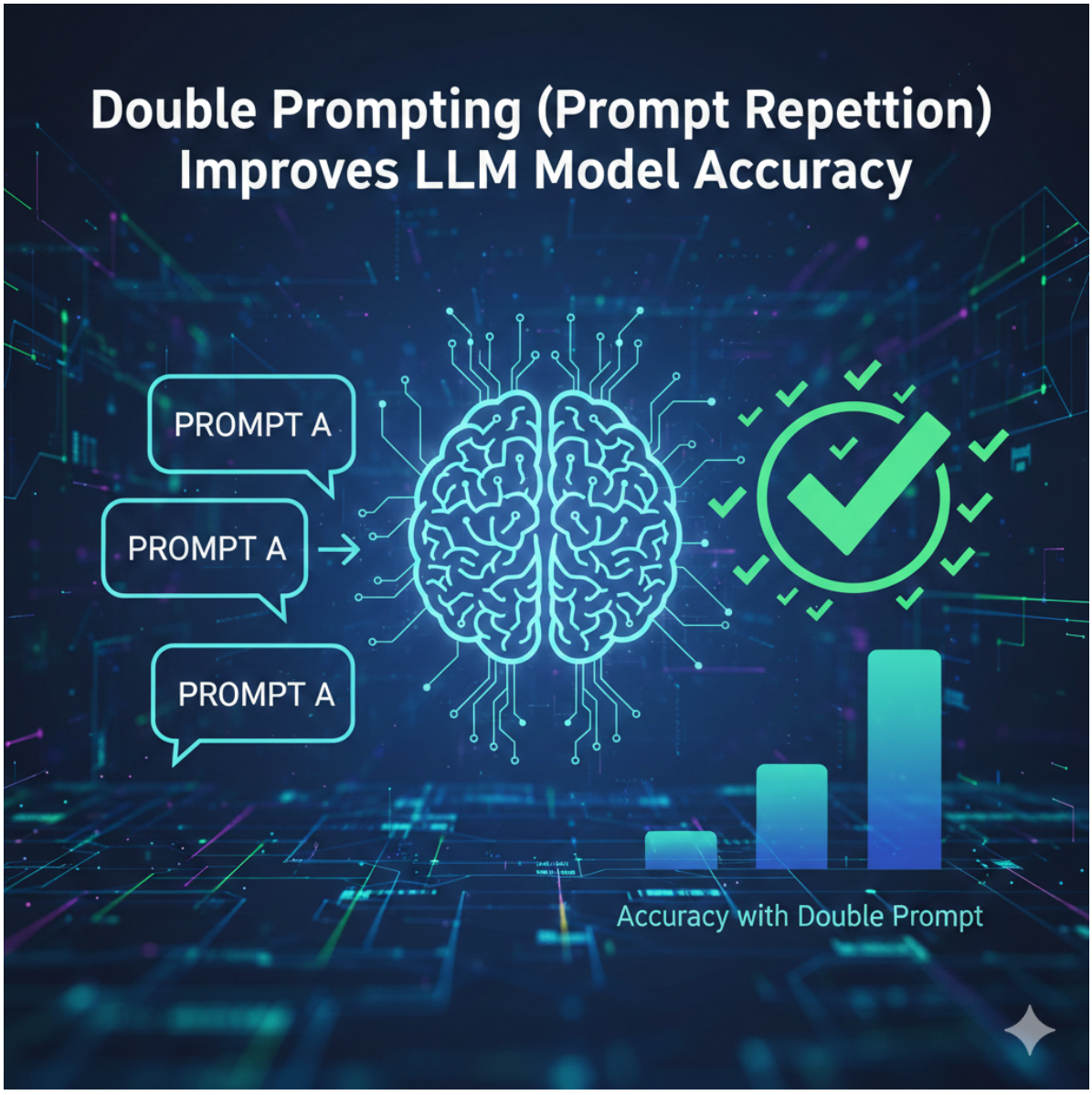

Figure 1: Accuracy With Double Prompting

Introduction

Ever noticed that an LLM can kickstart a project with brilliant, high-level architecture, but as the codebase grows, it starts tripping over its own feet? It begins introducing logical inconsistencies, making "executive decisions" like adding complex, unasked-for features, or simply forgetting context you provided earlier in the conversation.

This is not a random glitch. It is a documented phenomenon rooted in the fundamental architecture of transformer-based language models, and it has a name: **Lost in the Middle**.

This article examines the research behind this problem, a remarkably simple technique that mitigates it, **prompt repetition** (or "double prompting"), and why this matters for anyone building production software with LLMs.

---

1. The Problem: Lost in the Middle

1.1 The Foundational Discovery

In July 2023, Nelson F. Liu, Kevin Lin, John Hewitt, and colleagues at Stanford published a landmark paper: *"Lost in the Middle: How Language Models Use Long Contexts"*.

The study evaluated language models on two tasks requiring them to identify relevant information within their input contexts: multi-document question answering and key-value retrieval. The findings were striking:

- **Performance was highest when relevant information appeared at the very beginning or very end of the input context.**
- **Performance degraded significantly when the relevant information was placed in the middle of the context**, even for models explicitly designed for long contexts.
- This pattern held across model families and sizes, suggesting it is a fundamental property of the causal attention mechanism used in decoder-only transformer architectures.

The researchers coined the term **"Lost in the Middle"** for this phenomenon, and it has since become a standard reference point in the LLM research community.

1.2 The U-Shaped Attention Curve

The attention pattern Liu et al. identified is often described as a **"U-shaped" curve**: models exhibit strong **primacy bias** (they attend well to the beginning of the context) and strong **recency bias** (they attend well to the end), but the middle of the context window becomes a kind of blind spot.

This is directly analogous to the well-known serial position effect in human memory research, where humans also recall items better from the beginning (primacy) and end (recency) of a list.

The underlying mechanism is tied to how causal language models work: past tokens cannot attend to future tokens. This means the first tokens in a sequence are processed without the benefit of full context, while the last tokens have the richest attention. Tokens in the middle are caught in a zone where they have fading relevance to the end-of-sequence prediction but are too far from the beginning to benefit from primacy.

1.3 The Problem Persists Across Model Generations

A November 2025 study by researchers at General Motors, *"What Works for 'Lost-in-the-Middle' in LLMs? A Study on GM-Extract and Mitigations"*, extended this work with a real-world industrial benchmark.

Their key findings confirmed the persistence of the problem:

- **Even modern models with 128k token context windows** (such as LLaMA-3.1-8B) show performance degradation as context density increases.
- The maximum improvement achieved by all tested mitigation methods (including positional encoding modifications, hidden state scaling, and fine-tuning) was only **7-15% on average**, indicating that the problem is **deeply fundamental to transformer attention.**
- A critical insight: as context length increases, models retain their ability to understand **what** the answer is (semantic comprehension) but progressively lose the ability to identify **where** the answer is located (spatial awareness).
- Performance was highly sensitive to input data format. Simply reformatting the same data from structured key-value pairs to natural language paragraphs changed retrieval accuracy by 15-20%.

These findings establish that "Lost in the Middle" is not merely a theoretical curiosity. It is a practical reliability problem that affects code generation, document retrieval, data extraction, and any task where an LLM must locate and use specific information within a long context.

---

2. The Solution: Prompt Repetition

2.1 The Core Insight

In December 2025, Yaniv Leviathan, Matan Kalman, and Yossi Matias at Google Research published *"Prompt Repetition Improves Non-Reasoning LLMs"*, proposing a deceptively simple technique: **repeat the entire prompt**.

Instead of sending `<QUERY>` to the model, send `<QUERY><QUERY>`.

The reasoning is elegant: in a causal language model, each token can only attend to tokens that appeared before it. In a single pass of `<CONTEXT> <QUESTION>`, the model processes the answer options or question elements without full bidirectional awareness. When the prompt is repeated, **every token in the second copy can attend to every token in the first copy**, effectively giving the model a form of bidirectional attention within a unidirectional architecture.

This addresses the Lost in the Middle problem directly: the middle content from the first copy now has a second chance to be processed through the lens of the complete instruction, because the instruction appears again after it.

2.2 Experimental Results

The researchers tested prompt repetition on **7 popular models** of varying sizes from leading providers:

- **Gemini 2.0 Flash** and **Gemini 2.0 Flash Lite** (Google)
- **GPT-4o-mini** and **GPT-4o** (OpenAI)
- **Claude 3 Haiku** and **Claude 3.7 Sonnet** (Anthropic)
- **DeepSeek V3**

They evaluated across **7 benchmarks**: ARC (Challenge), OpenBookQA, GSM8K, MMLU-Pro, MATH, and two custom tasks (NameIndex and MiddleMatch). All tests used official APIs.

The headline result: prompt repetition won 47 out of 70 benchmark-model combinations, with zero losses.

That bears repeating: across 70 tests spanning 7 models and 7 benchmarks, prompt repetition **never made things worse** and **improved performance in 67% of cases.**

Specific findings:

Metric	Result
Wins (statistically significant, $p < 0.1$)	47 out of 70

Metric	Result
Losses	**0 out of 70**
NameIndex accuracy (Gemini 2.0 Flash-Lite)	21.33% => **97.33%**
Latency impact	**None** (only prefill stage affected)
Output length change	**None**

The NameIndex result is particularly striking. This custom task asks the model to identify the 25th item in a list of 50 names, a pure positional recall task that directly tests the Lost in the Middle problem. Prompt repetition increased accuracy from 21% to 97%, a **76 percentage point improvement**.

2.3 Why It Does Not Increase Cost or Latency

A natural concern is that doubling the prompt doubles the cost. In practice, the impact is minimal:

1. **No increase in generated tokens.** Prompt repetition does not change the length or format of the model's output. The token costs for generation (which typically dominate pricing) are unchanged.
2. **No increase in latency.** The additional processing occurs entirely in the **prefill stage**, which is parallelizable. The researchers measured empirical latency and found it remained constant across all models and benchmarks, with the only exception being Anthropic models on very long inputs.
3. **The input token cost increase is modest.** Doubling the input adds to the input token cost, but input tokens are typically priced 3-10x cheaper than output tokens across major providers.

The trade-off equation is clear: a modest increase in input cost for a consistent, measurable improvement in accuracy with zero architectural changes.

2.4 Prompt Repetition Variants

The researchers also tested variations of the technique:

- **Prompt Repetition (Verbose)**: `<QUERY> Let me repeat that: ...</QUERY>`, performed similarly across most benchmarks.
- **Prompt Repetition $\times$3**: `<QUERY> Let me repeat that: ...</QUERY> <QUERY>Let me repeat that one more time: ...<QUERY>`, sometimes **substantially outperformed** standard repetition on positional tasks like NameIndex and MiddleMatch.

- **Padding control**: Padding the input with periods to the same length as prompt repetition showed **no improvement**, confirming the gains come from the repetition itself, not simply from longer inputs.

2.5 Interaction with Reasoning Models

When reasoning (chain-of-thought / "think step by step") was enabled, prompt repetition was **neutral to slightly positive** (5 wins, 1 loss, 22 ties). This is expected: reasoning models often already begin their response by restating parts of the prompt, effectively performing their own internal repetition.

The key takeaway: **prompt repetition is most valuable for non-reasoning tasks**, which include code generation, data extraction, classification, formatting, and many production workloads where reasoning overhead is unnecessary.

---

3. Related Research: Re-Reading as Reasoning Enhancement

The prompt repetition findings are complemented by earlier work from Xu et al. (*"Re-Reading Improves Reasoning in Large Language Models"*, EMNLP 2024), which introduced the **Re2** technique: simply re-reading the question in the prompt.

Unlike chain-of-thought prompting, which targets the *output* (encouraging the model to reason aloud), Re2 targets the *input* by processing the question twice. The authors demonstrated that this enables a form of **"bidirectional" encoding in unidirectional decoder-only LLMs**, because the first pass provides global information that the second pass can leverage.

Their evaluation across **14 datasets spanning 112 experiments** confirmed that Re2 consistently enhanced reasoning performance with a simple re-reading strategy.

The theoretical underpinning is the same as prompt repetition: in causal models, **the second pass through the content benefits from attention to the first pass**, compensating for the inherent limitation of unidirectional attention.

---

4. Practical Implications for Developers

4.1 When to Use Prompt Repetition

Prompt repetition is most effective for:

- **Data extraction tasks** with long context (parsing specific values from large documents)
- **Code generation** from detailed specifications, where the spec appears early and the instruction at the end
- **Multi-document question answering** (RAG systems where retrieved documents fill the middle context)
- **Classification tasks** where options are presented before the question
- **Any task where relevant information is in the middle of a long prompt**

4.2 When It Is Less Necessary

- **Tasks with reasoning enabled** (the model already repeats key information in its chain-of-thought)
- **Short prompts** where there is no "middle" to get lost in
- **Streaming / conversational contexts** where context is managed turn-by-turn

4.3 Simple Implementation

Implementing prompt repetition requires no special tooling:

```
# Before
response = client.chat.completions.create(
    model="gpt-4o",
    messages=[{"role": "user", "content": query}]
)

# After: double prompting
response = client.chat.completions.create(
    model="gpt-4o",
    messages=[{"role": "user", "content": f"{query}\n\n{query}"}]
)
```

For a verbose variant:

```
doubled = f"{query}\n\nLet me repeat that:\n\n{query}"
```

4.4 Combining with Other Techniques

Prompt repetition can be combined with:

- **Structured prompts** (the repeated prompt benefits from the same specificity principles discussed in The Prompt-to-Code Paradox)
- **Context engineering** (place the most critical instructions at both the beginning and end of the context)
- **Few-shot examples** (repeat examples alongside the instruction)
- **Selective attention** techniques (Leviathan et al., 2024)

4.5 Enterprise Orchestration Strategy

As VentureBeat's analysis of this research noted, prompt repetition should ideally be **invisible infrastructure**, not a user behaviour. A well-designed orchestration layer or API gateway can automatically identify non-reasoning requests (entity extraction, classification, data retrieval, simple Q&A) and double the prompt before sending it to the model. This delivers better results without requiring action from end users or increasing the generation budget.

This conditional application is key: reasoning-heavy tasks gain little from repetition, so the orchestration harness should route selectively.

4.6 The Model Selection Cost Shift

The data reveals a counterintuitive model selection strategy: **smaller, cheaper models with prompt repetition may outperform larger models without it.** Gemini 2.0 Flash-Lite jumped from 21% to 97% accuracy on NameIndex with simple repetition. Before upgrading to a larger, more expensive model to solve an accuracy bottleneck, teams should first test whether repetition on their current lightweight model closes the gap. This preserves the speed and cost advantages of smaller infrastructure.

4.7 Security Implications

Prompt repetition introduces a double-edged security consideration:

- **Offensive risk:** If repeating a prompt clarifies intent to the model, malicious intents may be clarified as well. Security teams should update red-teaming protocols to test "repeated injection" attacks, verifying whether repeating a jailbreak command (e.g., "Ignore previous instructions") makes the model attend to the breach more effectively.

- **Defensive opportunity:** Repeating **system prompts** could actually reinforce safety constraints. Stating safety guardrails twice at the start of the context window forces the model to attend to them more rigorously, acting as low-cost reinforcement for robust content filtering.

This is an underexplored area that warrants further research from AI safety teams.

---

5. The Broader Picture: Why This Matters

The prompt repetition research matters for three reasons beyond the immediate accuracy gains:

5.1 It Validates a Known Developer Intuition

Many experienced developers have independently discovered that restating instructions at the end of a long prompt improves results. The Leviathan et al. paper provides the first rigorous, large-scale empirical confirmation of this practice.

5.2 It Is a Free Lunch (Almost)

Unlike switching to a more expensive model, fine-tuning, or implementing complex RAG architectures, prompt repetition requires **zero infrastructure changes**. It works with existing APIs, existing models, and existing workflows. The only cost is a modest increase in input tokens.

5.3 It Highlights a Fundamental Architectural Limitation

The fact that such a simple technique produces consistent improvements across all major foundation models underscores that the Lost in the Middle problem is not a training bug that will be fixed in the next model release. It is a **structural property of causal attention** in decoder-only transformers. Until architectures fundamentally change (e.g., through native bidirectional attention or radically different positional encoding), techniques like prompt repetition will remain valuable.

5.4 It May Become a Default

The Google researchers themselves suggest this could become standard behaviour. We may soon see inference engines that **silently double prompts in the background** before sending them to the model, or reasoning models that have internalised this repetition strategy to be more efficient. The authors explicitly list fine-tuning with repeated prompts and training reasoning models with prompt repetition as top future directions.

Conclusion

The research is clear:

1. **LLMs have a documented, U-shaped attention curve** that causes them to lose track of information in the middle of long contexts (Liu et al., 2023).
2. **This problem persists across model generations**, even for models with 128k+ context windows (Gupte et al., 2025).
3. **Simply repeating the prompt** addresses this limitation for non-reasoning tasks, winning **47 out of 70 benchmark tests with zero losses** and improving accuracy by up to 76 percentage points on positional recall tasks (Leviathan et al., 2025).
4. **The technique is free**: it does not increase output length, latency, or require any architectural changes.

For developers and teams building production systems with LLMs, prompt repetition represents one of the highest-value, lowest-effort improvements available today. Add it to your toolkit alongside structured prompting, spec-driven development, and proper context engineering.

References

1. Liu, N.F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2023). *Lost in the Middle: How Language Models Use Long Contexts.* Transactions of the Association for Computational Linguistics (TACL).
2. Leviathan, Y., Kalman, M., & Matias, Y. (2025). *Prompt Repetition Improves Non-Reasoning LLMs.* arXiv:2512.14982.
3. Gupte, M., Dixit, E., Tayyab, M., & Adiththan, A. (2025). *What Works for 'Lost-in-the-Middle' in LLMs? A Study on GM-Extract and Mitigations.* arXiv:2511.13900.

4. Xu, X., Tao, C., Shen, T., Xu, C., Xu, H., Long, G., Lou, J., & Ma, S. (2024). *Re-Reading Improves Reasoning in Large Language Models.* EMNLP 2024.

5. Springer, J.M., Kotha, S., Fried, D., Neubig, G., & Raghunathan, A. (2024). *Repetition Improves Language Model Embeddings.* arXiv:2402.15449.

6. Shaier, S. (2024). *Asking Again and Again: Exploring LLM Robustness to Repeated Questions.* arXiv:2412.07923.

7. Franzen, C. (2026). *This new, dead simple prompt technique boosts accuracy on LLMs by up to 76% on non-reasoning tasks.* VentureBeat.
