What is Context Rot? How to fix it?

This article will help you understand why the effectiveness of LLMs gradually decreases throughout a conversation and what you can do to prevent that.

Have you ever noticed that the longer you talk to an AI, the less effective it becomes?

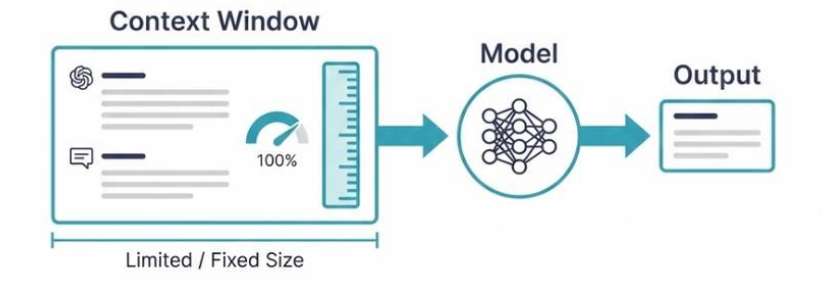
Why is this happening? Recent research is shedding light on why this occurs. It's called Context Rot—contextual degradation. The more input data you provide to a large language model , the worse its performance becomes. It may ignore some input data or overprocess others. How this performance degradation occurs is complex.

In large-scale language models, input quality affects output quality. There are several ways we can influence input quality:

1. Clearly state what you want. (Often referred to as the "prompt writing technique").
2. It provides the necessary context so that the larger language model can do what we want. (Often referred to as "context creation techniques").

If you're new to command-line interfaces like Claude Code or the Codex, or using Claude Cowork to create your own AI workflows, this article will help you understand why LLM performance degrades throughout a conversation and what you can do to prevent it.

What is a context window?



The context window is the amount of information an AI model can process at one time. In other words, the context window is a metaphor we use to refer to what takes up input into a large language model. You can think of it as the model's short-term memory. It's everything the model can process at the same time.

The context window has a fixed size, not an infinite one. The amount of input you can enter at the same time is limited by the size of the context window. And this size varies depending on the model.

Model	Context window
Claude Opus 4.5	200K tokens
Claude Sonnet 4.5	200K tokens (1 million at usage level 4)
GPT-5.2	400K tokens
Gemini 2.5 Pro	1M token

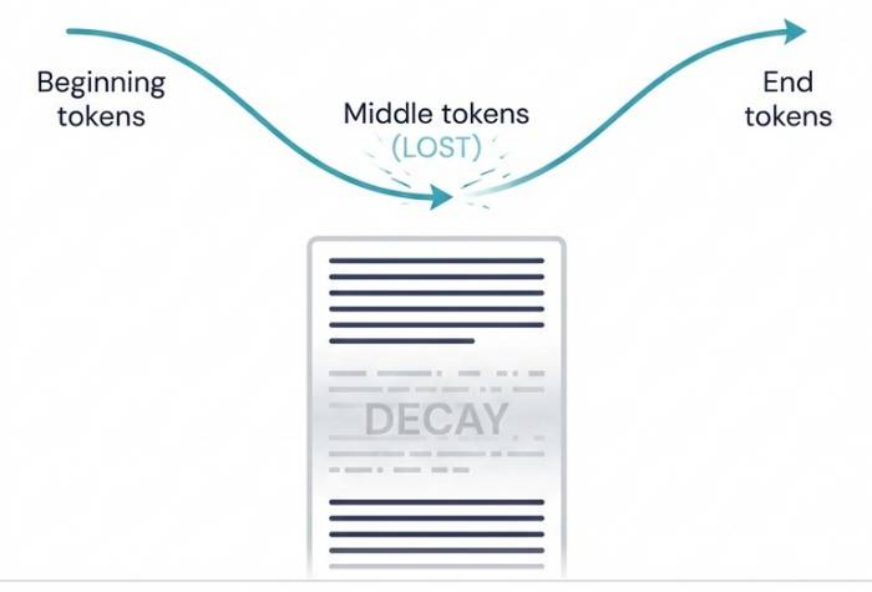
The input data for the model doesn't just include what you type in. Each application adds a system prompt placed before the user's message. Depending on the application context, the system prompt might include a description of the tool (e.g., from the MCP server), available skills, plugins, etc.

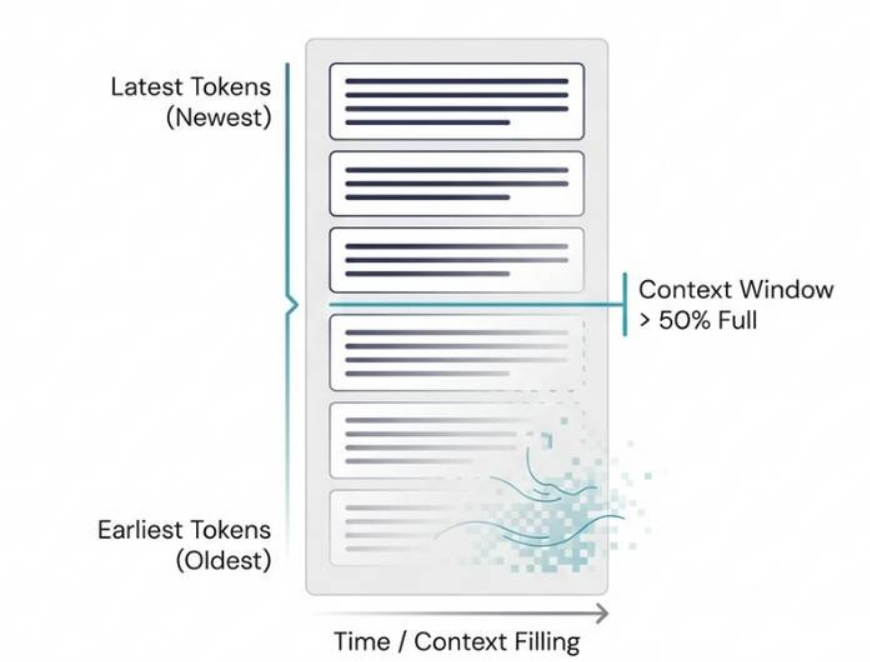
In a back-and-forth conversation, the entire conversation history is typically included in the context window at each turn. This is the only way the model can keep track of the conversation. While many models have large context windows, in practice, these windows can quickly become full. And we'll see that filling the context window can cause problems.

Why are context windows of fixed size?

Several factors limit the size of the context window, including the amount of computation required to process the input, the amount of memory needed to store the values ??during processing, and the size of the input used during training.

What is the phenomenon of Context Degradation?





The degree of context degradation varies depending on how full the context window is.

In November 2023, Liu et al. published a paper titled "Lost in the Middle," presenting the first evidence of contextual degradation. The authors found that as the context window fills up, models begin to prioritize tokens at the beginning and end of the input. Tokens in the middle "disappear."

In 2025, several other researchers published papers exploring this pattern, and the term "context rot" was coined. Here's a quick summary of what the studies have found so far:

1. The original study by Liu et al. (2023) used simple "find the needle in the haystack" tasks to test contextual degradation. In the "find the needle in the haystack" problem, the model was asked whether a sentence ("the needle") was in context ("the haystack"). The study showed that as the context window filled up, the models performed worse on these simple tasks.
2. Paulsen (2025) has shown that context is degraded across many different types of tasks (not just "find a needle in a haystack" tasks) and often with far fewer tokens for more complex tasks.
3. Veseli et al (2025) found that the U-shaped pattern (where LLM prioritizes tokens at the beginning and end) found by Liu et al (2023) only exists when the context is less than 50%. When the context is more than 50% full, Veseli et al (2025) found a different pattern: Context degrades with distance from the end, where LLM prioritizes more recent tokens, then those in the middle, compared to the initial tokens.
4. Researchers hypothesized that the models were having trouble retrieving information, meaning the model couldn't find the "needle in the haystack." But Du et al (2025) – through some clever experiments – showed that it wasn't a retrieving problem. It was simply a function of input length. In one of the more interesting experiments, they replaced all tokens that weren't "needle" in the input with whitespace. Their thinking was that if this were a retrieving problem, then the "needle" would now be obvious – but they still found similar evidence of contextual degradation in these modified tasks.

So what does all this mean?

Even with a large context window size, we don't want to fill it completely. Performance will decrease when it's full and degrade in various ways.

1. When the context window is filled below 50%, the model will lose the tokens in the middle.
2. When the context window is filled above 50%, the model will lose its first tokens.

It is very difficult to manage context when using LLMs on a web browser.

Web browser interfaces (such as ChatGPT/Claude) do not allow users to view the status, size, or degradation of the context window. Users have no control over this. Web interfaces of popular models do not provide any information about how full the context window is.

First of all, we must acknowledge a major limitation when using ChatGPT, Claude, or Gemini on a web browser. These applications don't let us know how full the context window is. You may have been using ChatGPT since its launch without even knowing what a context window is. That's a problem.

In fact, these tools give the impression that the context window is infinite, as they allow you to chat forever. But now you know they aren't. Context degrades over time. The longer the conversation, the worse the model's performance.

This also means you have very little control over what appears in the context window (beyond what you type), how and when to collapse or clear it, or even knowing when you need to do so. It means you can't influence the model's performance. The only tool for managing "context errors" in a web browser is to start a new conversation.

Starting a new conversation will clear the context window. And you should do this regularly. If you're working with a language learning model (LLM) in a web browser, start a new conversation whenever:

1. You start a new topic.
2. The model performed an operation incorrectly, and you want it to try again.

3. The conversation started getting long (e.g., more than 15 messages). Request a conversation summary and use that summary to start a new conversation.

These tips will help you keep the context window compact.

But we can do much more when we can see what's in the context window, how full it is, and know exactly when to minimize it. This is one of the things people love about Claude Code.