**AI Explained**

# Memory, Attention and Limits (Why AI ‘Forgets’)

By Dr Peter Fields

**THE 2-MINUTE SUMMARY**

The conversation you can scroll through and the information supplied to answer your next message are two different things. A product can carry forward earlier turns, retrieve selected material, summarise old exchanges or leave some of them out.[2,3]

The **context window** is the amount of information the model can work with in a single request, measured in tokens. A token is a small unit of text, such as a word, part of a word or a punctuation mark. Material can fit within the window and still be hard for the model to use. In one series of tests, moving the same evidence into the middle of a long input made it harder for models to use.[4,5]

Saved preferences and past-chat references are separate product features. For a task that matters, put the current instructions and the relevant evidence where the assistant can use them, then check what it actually relied on.[3,8]

## The vanishing instruction

Suppose you start a working session by asking for British spelling, sources and continuous prose. Twenty turns later, after two drafts and a long report, the assistant switches to American spelling and misses a qualification on page eighteen. Both the instruction and the qualification are still visible when you scroll back. How much of that material was available to the model, and how well did it use what it received? Those are separate questions.

## What fits in the workspace?

A tokenizer divides text into tokens. The number of tokens in a page varies with the text and the tokenizer. A page of code and a page of English prose can take up different amounts of the same context window.[1]

A chat application assembles the material for the next answer. A short conversation may fit in full. In a long session the application may compress earlier turns into a summary, an example of what developers call **compaction**, or retrieve selected passages. Uploading a report can give the application access to the document without putting every page into every request. A summary can preserve the decision to use the second draft while losing the reason it superseded the first.[2,3]

**Geek Box · What shares the context window?**

Depending on the system, the request can include product or application instructions, earlier turns, a description of available tools, an excerpt from a document, or the output of a search. Some models also allocate space to reasoning before producing the visible answer. A context limit governs what can fit in a request. A message, rate or usage limit governs what the service allows you to send or run. The rules can differ across products and plans.[2,3,10]

## Why a bigger window does not ensure a better answer

Nelson Liu and colleagues varied the position of relevant information in tasks including multi-document question answering. Models often used evidence more successfully near the beginning or end of the input than in the middle. Later work, including RULER, tested combinations of length and task complexity and found large differences between advertised capacity and the length at which some models sustained performance on those tasks. These findings are specific to the models and tasks tested.[4,5]

OpenAI's GPT-4.1 guide recommends putting key instructions on both sides of a long input. Anthropic's Claude guidance recommends putting long documents first and the question at the end. If placement matters, try both arrangements with your own material and check which gives better results.[11]

Length is only part of it. Chroma's technical report found that the content of the additional material also mattered. In a working thread, a superseded draft can be a particularly plausible distraction because it uses the names, facts and language of the current one. Label versions explicitly and identify the controlling document.[6]

**Geek Box · Why attention gets expensive**

Imagine comparing each word in a transcript with every earlier word. Double the length and you make roughly four times as many comparisons. A text-generating Transformer using full attention over earlier tokens faces the same growth in attention calculations. End-to-end time and cost depend on the architecture and implementation. FlashAttention, for example, computes exact attention while reducing transfers between levels of GPU memory.[7]

## What comes back in a new chat?

Cross-chat memory is a feature of the application. ChatGPT distinguishes saved memories from information drawn from past chats. Claude also documents profile preferences and past-chat search, with its own settings. An assistant can appear to remember how you like a briefing formatted because that preference is made available again. Separately, retrieval can supply passages from an uploaded document or connected collection. The visible behaviour alone tells you little about how a particular memory was stored, selected or supplied.[2,8,9]

## How to use it better

- **Give a long project a short handover.** Record the current objective, decisions and open issues. Keep the governing version with the new task. In the opening example, repeat the British-spelling instruction when you request the next draft. That makes it explicit for that turn, whatever happened to the original instruction.
- **Ask for the passage before the conclusion.** For a long report, start by asking for the clause, surrounding qualification and page or section. Check both against the original.
- **Review saved preferences in the product settings.** If something surprising appears in a new chat, find out which memory or history settings apply to that product.

## Sources and notes


1. Sennrich, Haddow and Birch (2016), Neural Machine Translation of Rare Words with Subword Units, ACL, pp 1715–1725. An early subword method. Source
2. OpenAI, Conversation State and Responses Overview, API documentation. Examples of how an application carries forward state and compacts context. Source
3. Anthropic (2025), Effective Context Engineering for AI Agents, 29 September. Context contents, compaction and retrieval. Source
4. Liu et al. (2024), Lost in the Middle: How Language Models Use Long Contexts, TACL 12, pp 157–173. Source
5. Hsieh et al. (2024), RULER: What's the Real Context Size of Your Long-Context Language Models?, COLM. Source
6. Hong, Troynikov and Huber (2025), Context Rot: How Increasing Input Tokens Impacts LLM Performance, Chroma technical report. A provider-independent study of the tested models. Source
7. Vaswani et al. (2017), Attention Is All You Need, NeurIPS, §3.2 and Table 1; Dao et al. (2022), FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness, NeurIPS. Source 1 · Source 2
8. OpenAI, Memory in ChatGPT, and Anthropic, Understanding Claude's Personalization Features. Product documentation. Source 1 · Source 2
9. Lewis et al. (2020), Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, NeurIPS, pp 9459–9474. Early research on supplying retrieved material during generation. Source
10. OpenAI, Reasoning Models, API documentation. An example of accounting for reasoning tokens. Source
11. OpenAI, GPT-4.1 Prompting Guide, §2, Prompt Organization, and Anthropic, Claude Prompting Best Practices, Long Context Prompting. Model-specific developer guidance. Source 1 · Source 2