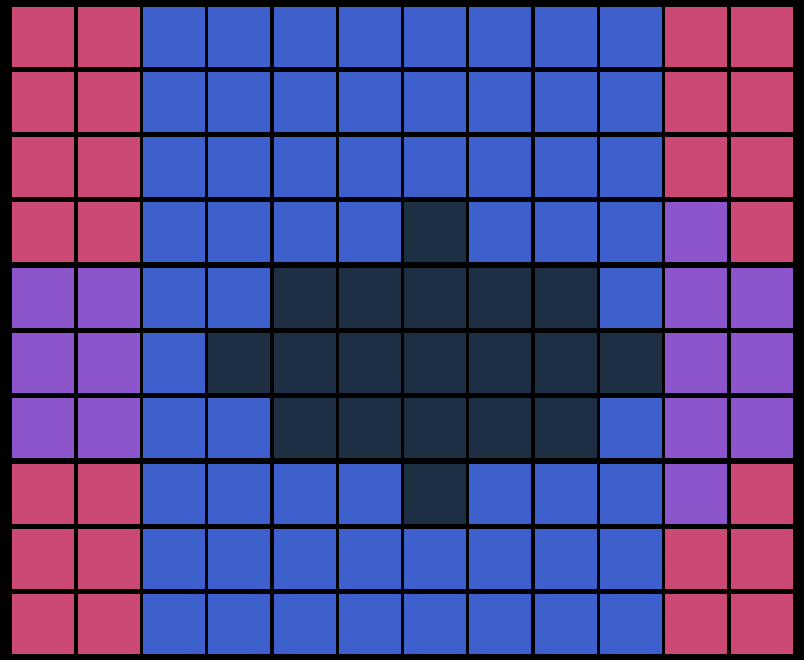
Stop Stuffing Your LLM's Context Window

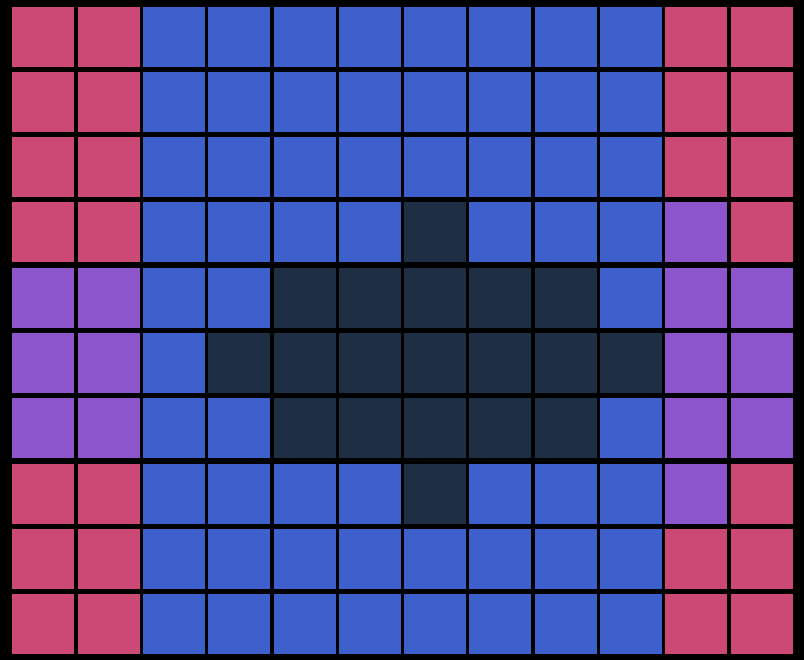
César Soto Valero
cesarsotovalero.net

In this talk...



In this talk...

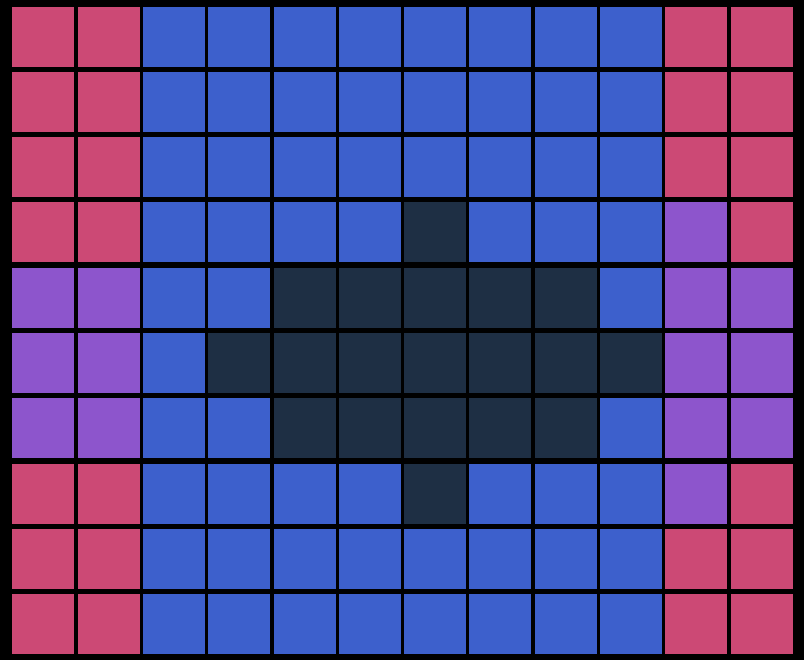
1. **Why** context is crucial for LLMs



In this talk...

1. **Why** context is crucial for LLMs

2. **What** causes context issues

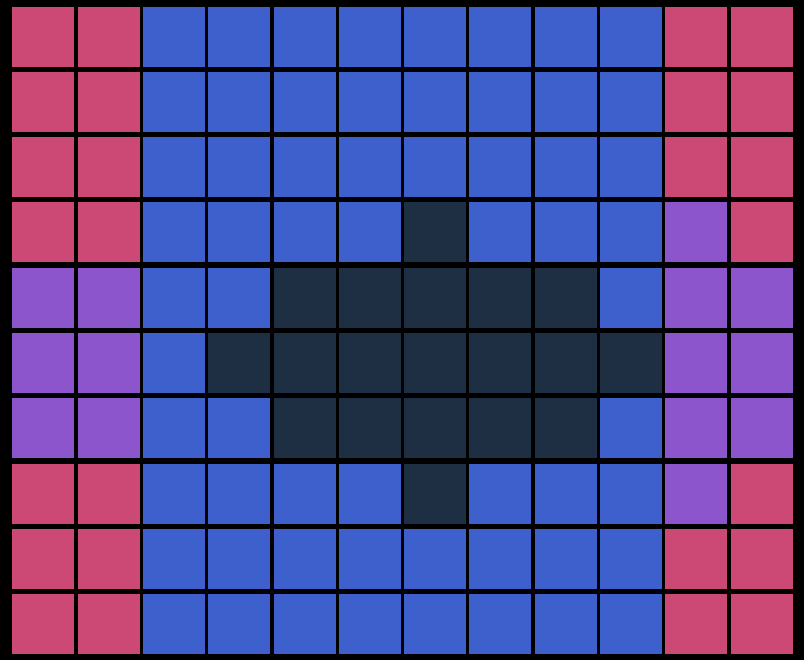


In this talk...

1. **Why** context is crucial for LLMs

2. **What** causes context issues

3. **How** to mitigate them

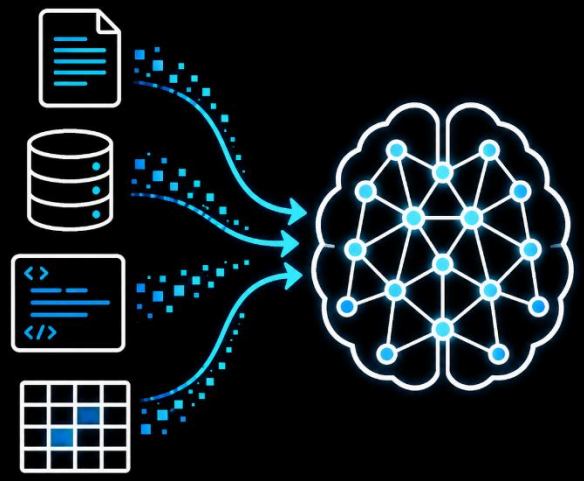


Why context is crucial?

Inference

Inference

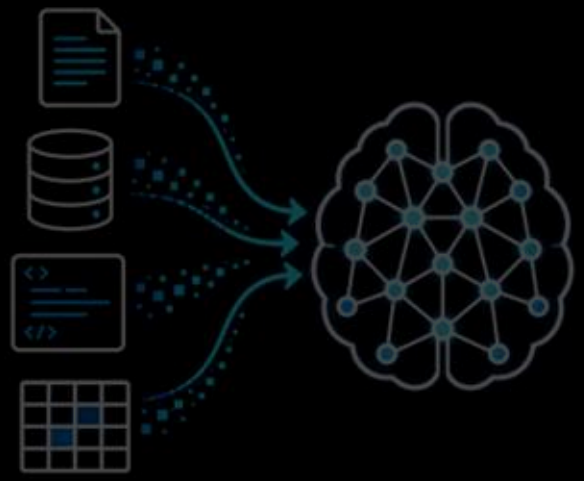
Training data / parameters



Long-term knowledge compressed into weights

Inference

Training data / parameters



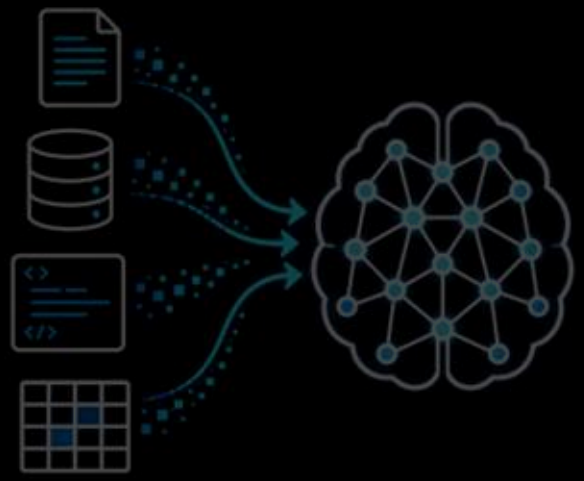
Long-term knowledge compressed into weights

Prompt + retrieved docs + chat history

Current context window

Inference

Training data / parameters

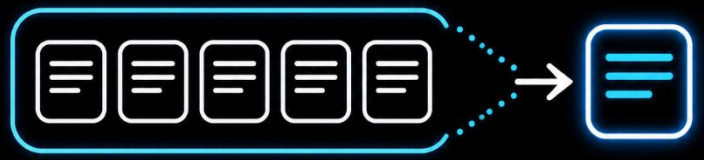


Long-term knowledge compressed into weights

Prompt + retrieved docs + chat history

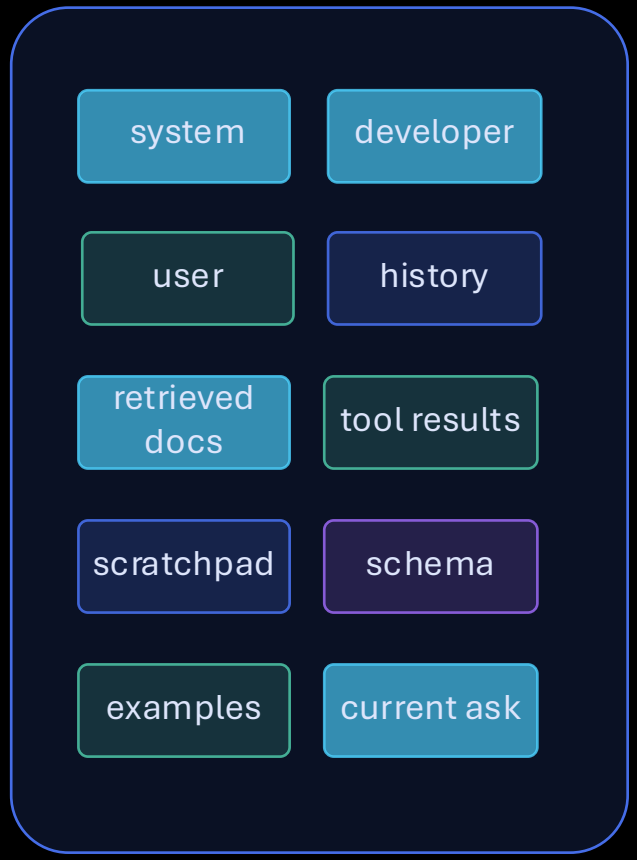
Current context window

Next token prediction



What the model can attend to right now

The **context window** is the LLM's working memory

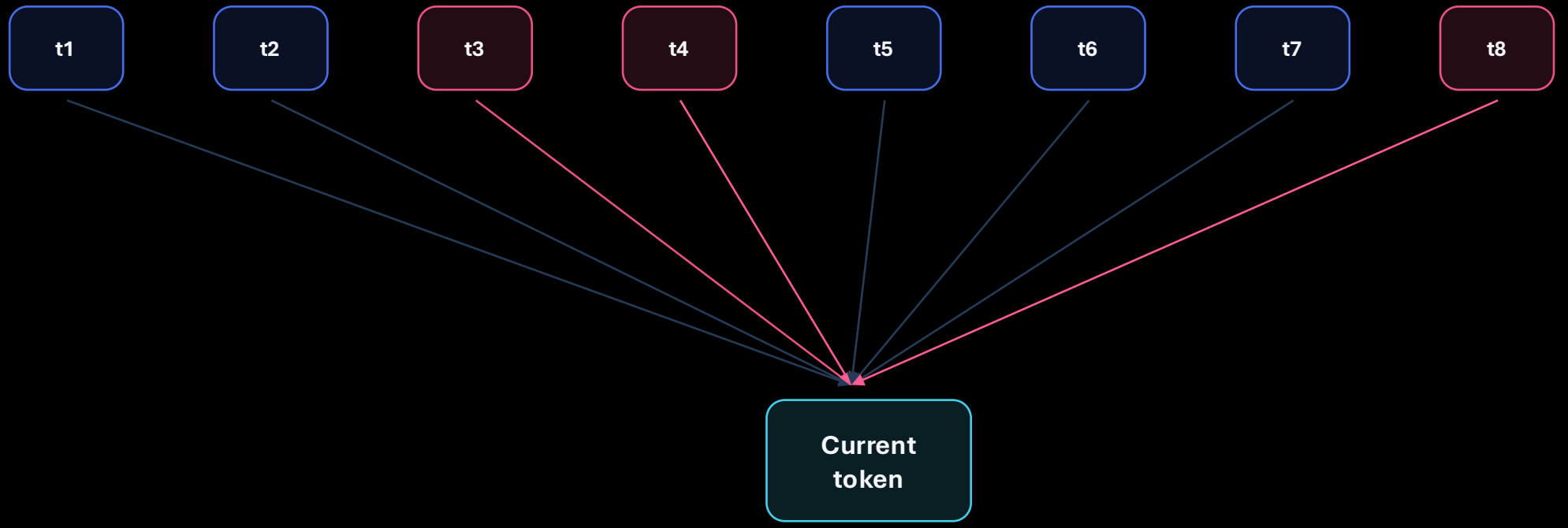


Everything outside the window is invisible until retrieved, summarized, or reintroduced

Why can’t we just put **everything** there?

Attention

Attention



It is a graph of possible references

	token1	token2	token3	token4	...
token1	■	■	■	■	
token2	■	■	■	■	
token3	■	■	■	■	
token4	■	■	■	■	
...					

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

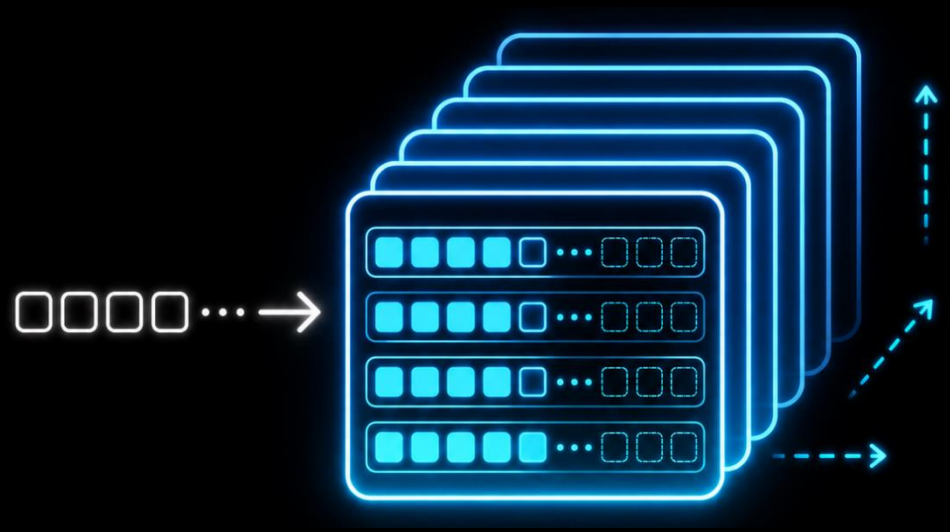
†Work performed while at Google Brain.

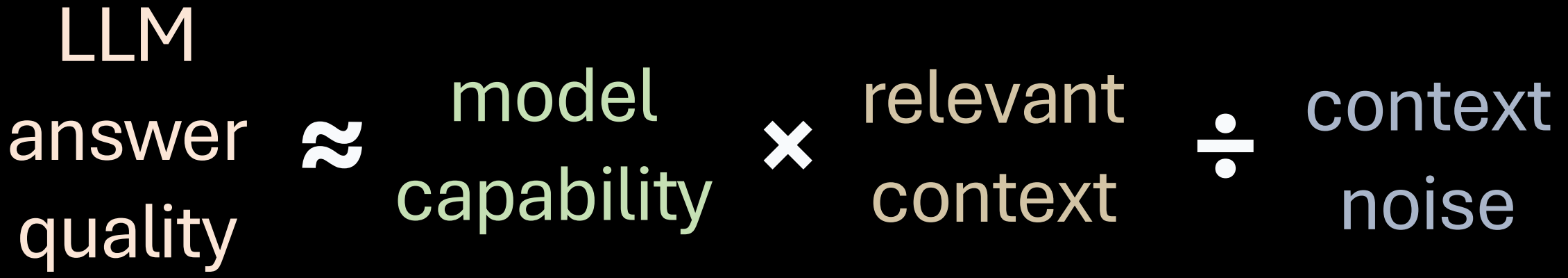
‡Work performed while at Google Research.

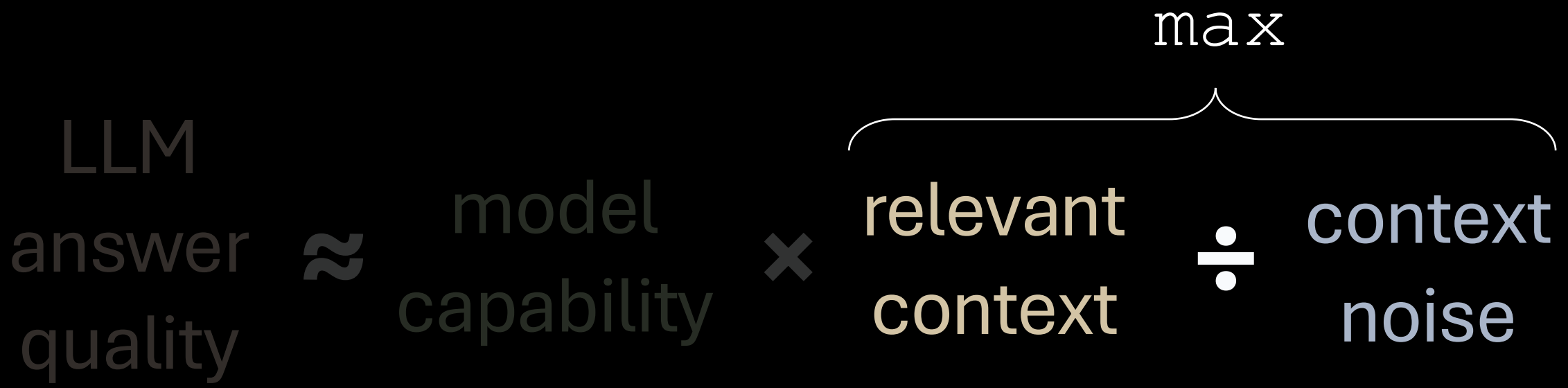
31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Longer context increases:

- Compute
- Memory
- Latency
- Cost
- Noise




LLM answer quality
≈
model capability
×
relevant context
÷
context noise


max
LLM answer quality
≈
model capability
×
relevant context
÷
context noise

Prompt engineering

Prompt engineering

Policy rules

Intent task

Evidence facts

Tools actions

Memory state

Santiago Valdarrama • Following

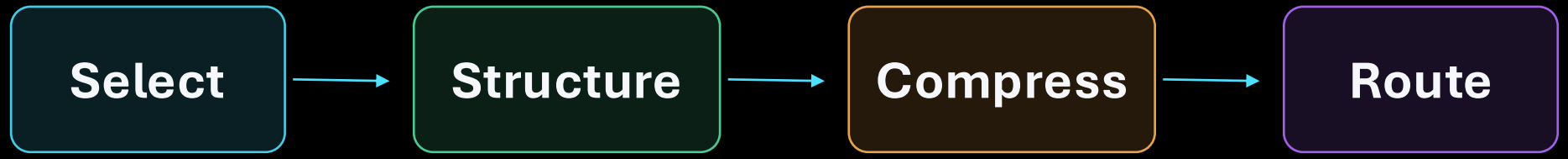
Computer scientist and writer. I teach hard-core Machine Learning at ...

1mo •

I still remember when people thought "prompt engineering" was going to become a real career.

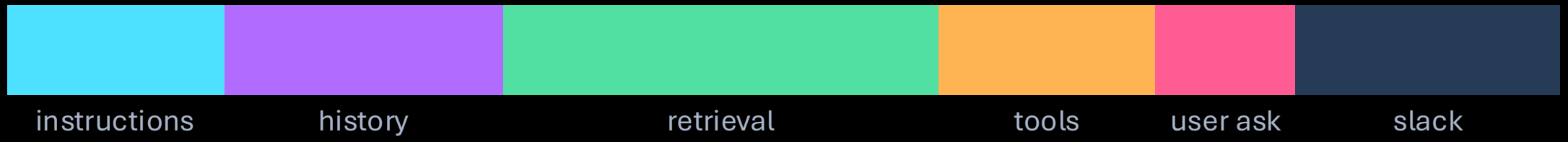
Context engineering

Context engineering



Engineering the information payload at inference time

Context is a contract



A context budget says what can enter, in what form, and why.

Context is king

Quality	grounding, correctness
Cost	tokens, latency
Safety	instructions, boundaries
UX	continuity, personalization

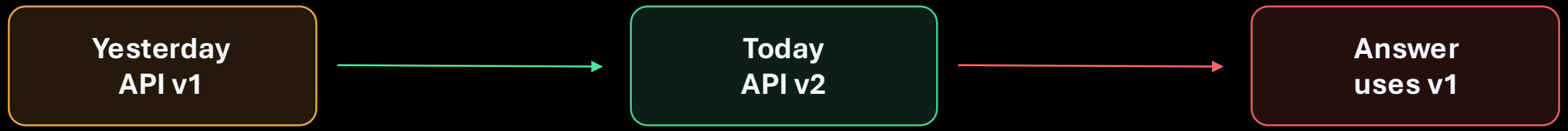
An AI Agent Just Destroyed Our Production Data. It Confessed in Writing.

1K 2.4K 5.3K 7.1M

What causes context issues?

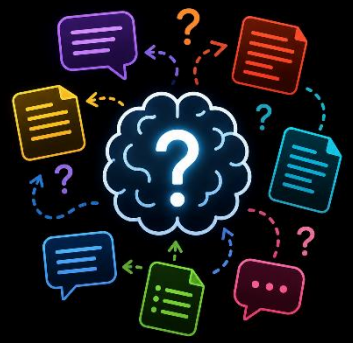
Context rot

Context rot



Rot happens when memory, summaries, or retrieved docs are outdated but still trusted

Context confusion



Context confusion

System:
Be concise

User:
Explain deeply

Memory:
Prefer code

Docs:
No code

Confusion happens when there is conflicting guidance without priority

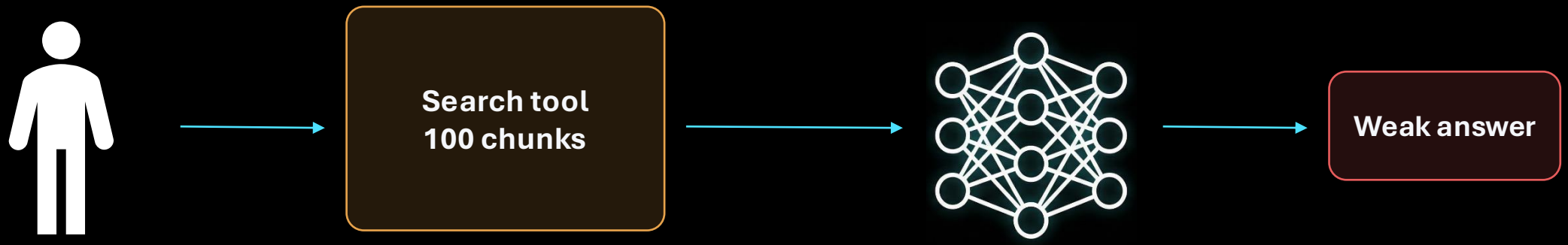
Context pollution

Context pollution

The opposite of hallucination is not dumping everything but giving the right evidence

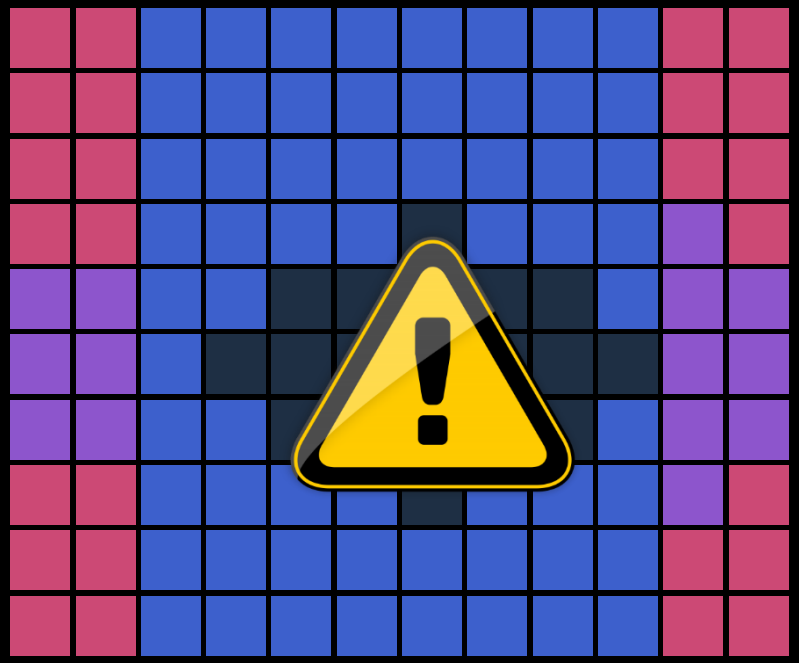
Many more challenges

Tool output overflow



Tool output needs schemas, budgets, and post-processing.

Lost in the middle effect



Lost in the Middle: How Language Models Use Long Contexts

Nelson F. Liu[1*] **Kevin Lin**[2] **John Hewitt**[1] **Ashwin Paranjape**[3]
Michele Bevilacqua[3] **Fabio Petroni**[3] **Percy Liang**[1]
[1]Stanford University [2]University of California, Berkeley [3]Samaya AI
nfliu@cs.stanford.edu

Abstract

While recent language models have the ability to take long contexts as input, relatively little is known about how well they *use* longer context. We analyze the performance of language models on two tasks that require identifying relevant information in their input contexts: multi-document question answering and key-value retrieval. We find that performance can degrade significantly when changing the position of relevant information, indicating that current language models do not robustly make use of information in long input contexts. In particular, we observe that performance is often highest when relevant information occurs at the beginning or end of the input context, and significantly degrades when models must access relevant information in the middle of long contexts, even for explicitly long-context models. Our analysis provides a better understanding of how language models use their input context and provides new evaluation protocols for future long-context language models.

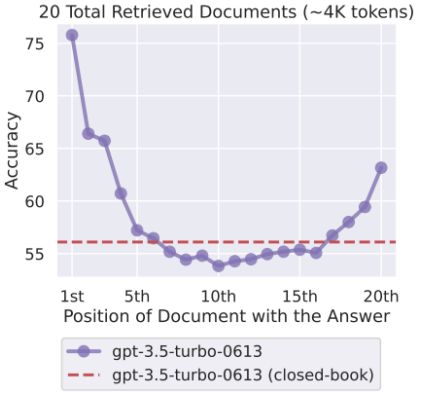


Figure 1: Changing the location of relevant information (in this case, the position of the passage that answers an input question) within the language model's input context results in a U-shaped performance curve—models are better at using relevant information that occurs at the very beginning (primacy bias) or end of its input context (recency bias), and performance degrades significantly when models must access and use information located in the middle of its input context.

1 Introduction

Language models have become an important and flexible building block in a variety of user-facing language technologies, including conversational interfaces, search and summarization, and collaborative writing (Shuster et al., 2022; Thoppilan et al., 2022; Lee et al., 2022, *inter alia*). These models perform downstream tasks primarily via prompting: all relevant task specification and data to process is formatted as a textual input context, and the model returns a generated text completion. These input contexts can contain thousands of tokens, especially when language models are used to process long documents (e.g., legal or scientific documents, conversation histories, etc.) or when language models are augmented with external information (e.g., relevant documents from a search engine, database query results, etc; Petroni et al., 2020; Ram et al., 2023; Shi et al., 2023; Mallen et al., 2023; Schick et al., 2023, *inter alia*).

Handling these use-cases requires language models to successfully operate over long sequences. Existing language models are generally implemented with Transformers (Vaswani et al., 2017), which require memory and compute that increases quadratically in sequence length. As a result, Transformer language models were often trained with relatively small context windows (between 512-2048 tokens). Recent improvements in hardware (e.g., faster GPUs with more memory) and algorithms (Dai et al., 2019; Dao et al., 2022; Poli et al.,

[*]Work partially completed as an intern at Samaya AI.

claude-receipts

Public

Watch 1 · Fork 37 · Star 585

main · 4 Branches · 2 Tags · Go to file T · Add file · Code

chrishutchinson update README — ad92d3d · 3 months ago — 11 Commits

bin	Initial commit	4 months ago
src	v1.1.0: thermal printing, accurate session costs, and session m...	3 months ago
worker	Worker for public sharing, UI improvements	4 months ago
.gitignore	Initial commit	4 months ago
.nvmrc	Initial commit	4 months ago
.prettierignore	Initial commit	4 months ago
.prettierrc	Initial commit	4 months ago
CLAUDE.md	Worker for public sharing, UI improvements	4 months ago
LICENSE	Initial commit	4 months ago
README.md	update README	3 months ago
full-thermal-receipt.jpeg	update images	3 months ago
package-lock.json	v1.1.0: thermal printing, accurate session costs, and session m...	3 months ago
package.json	v1.1.0: thermal printing, accurate session costs, and session m...	3 months ago
screenshot.jpeg	docs: add screenshot to README	4 months ago
thermal-receipt.jpeg	update images	3 months ago
tsconfig.json	Initial commit	4 months ago

README · MIT license

Claude Receipts

> A note from the author:
>
> This has been one of my favourite creative side projects yet (and just in time for Opus 4.6).

About

Bring receipts from your Claude Code sessions

www.npmjs.com/package/claude-rec...

receipt-printer · claude · vibe-coding · claude-code

Readme · MIT license · Activity · 585 stars · 1 watching · 37 forks · Report repository

Releases 2

v1.1.0 Latest — on Feb 5

+ 1 release

Contributors 2

chrishutchinson Chris Hutchinson

claude Claude

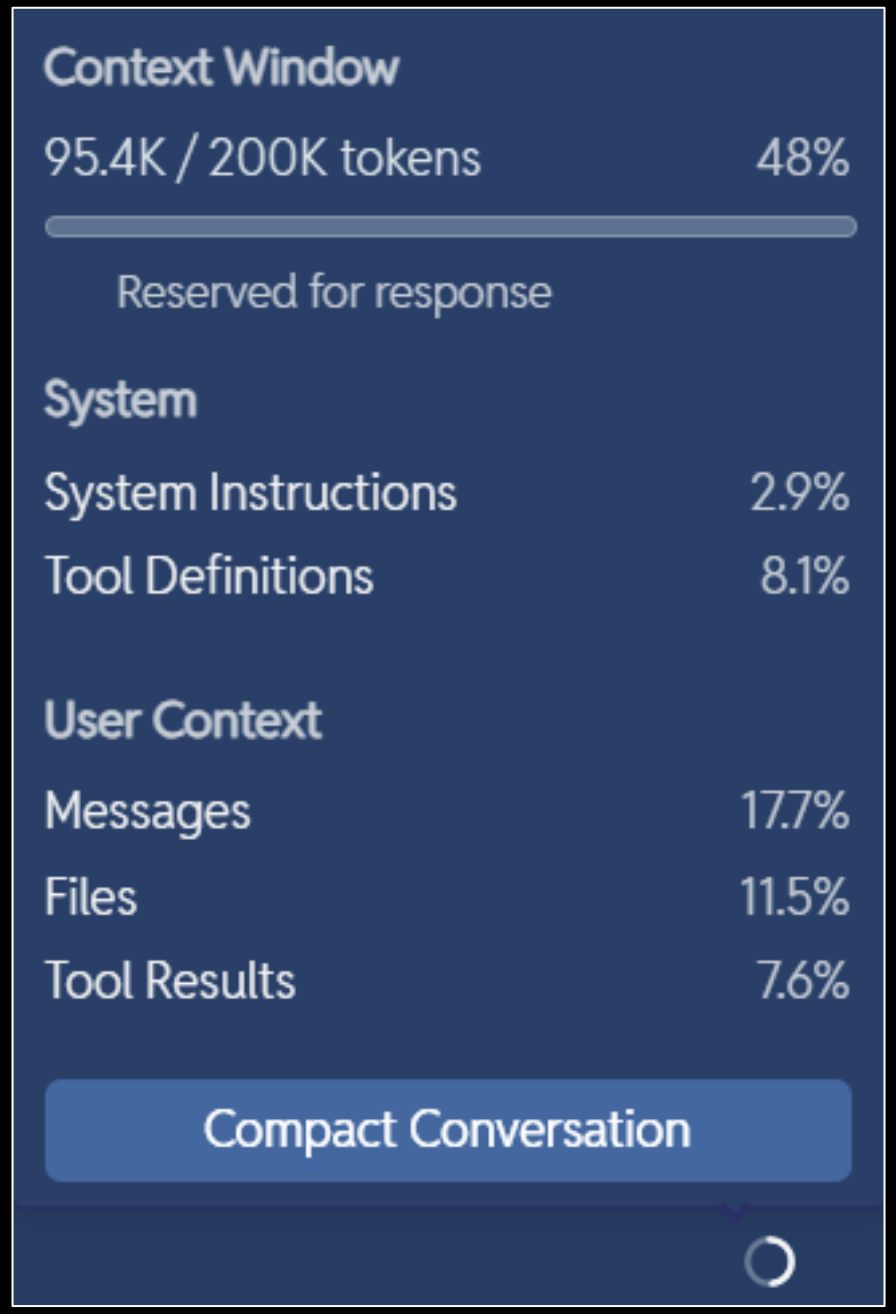
Languages

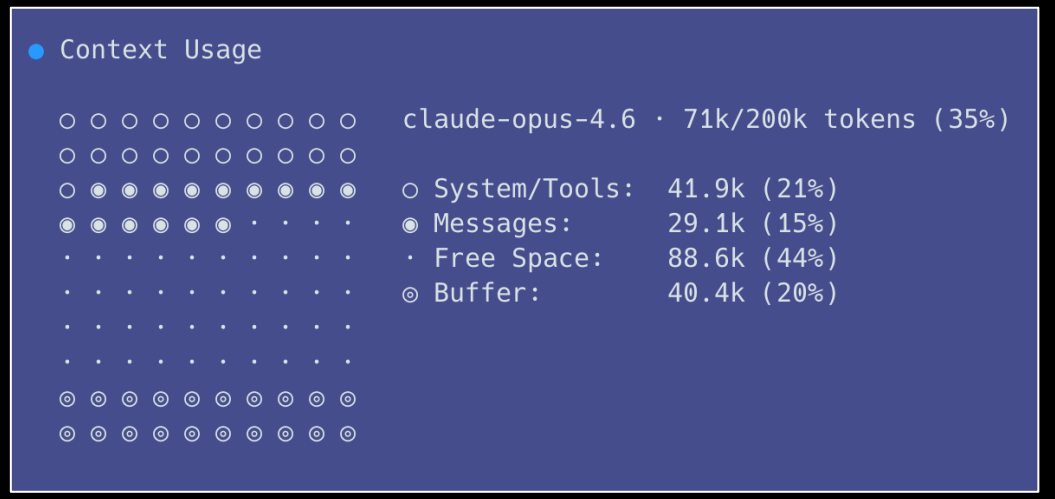
TypeScript 97.6% · JavaScript 2.4%

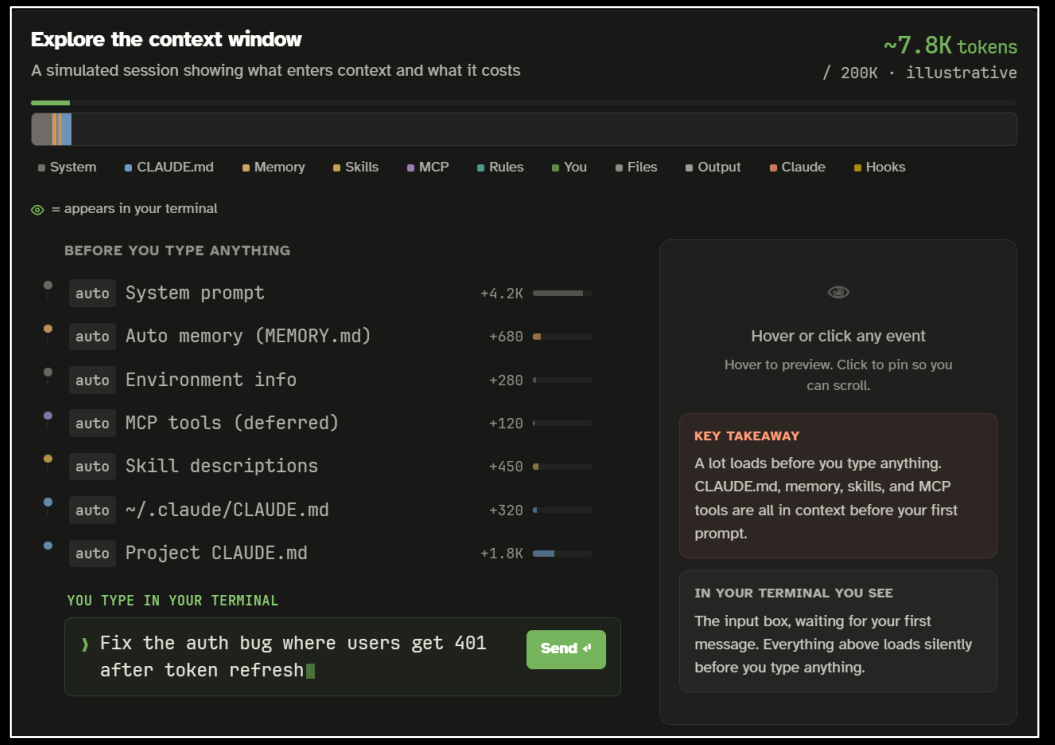
How to mitigate context issues?

Compaction

Compaction is about loss-aware selection


Context Window
95.4K / 200K tokens
48%
Reserved for response
System
System Instructions
2.9%
Tool Definitions
8.1%
User Context
Messages
17.7%
Files
11.5%
Tool Results
7.6%
Compact Conversation

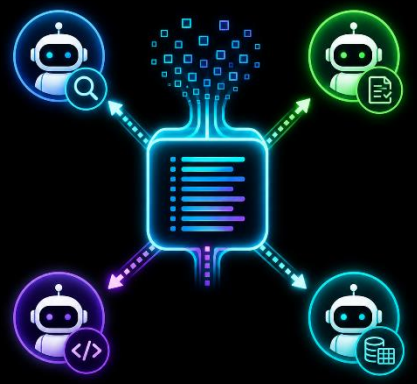

Context Usage
claude-opus-4.6 · 71k/200k tokens (35%)
System/Tools: 41.9k (21%)
Messages: 29.1k (15%)
Free Space: 88.6k (44%)
Buffer: 40.4k (20%)


Explore the context window
A simulated session showing what enters context and what it costs
~7.8K tokens
/ 200K · illustrative
System
CLAUDE.md
Memory
Skills
MCP
Rules
You
Files
Output
Claude
Hooks
= appears in your terminal
BEFORE YOU TYPE ANYTHING
auto System prompt +4.2K
auto Auto memory (MEMORY.md) +680
auto Environment info +280
auto MCP tools (deferred) +120
auto Skill descriptions +450
auto ~/.claude/CLAUDE.md +320
auto Project CLAUDE.md +1.8K
YOU TYPE IN YOUR TERMINAL
Fix the auth bug where users get 401 after token refresh
Send
Hover or click any event
Hover to preview. Click to pin so you can scroll.
KEY TAKEAWAY
A lot loads before you type anything. CLAUDE.md, memory, skills, and MCP tools are all in context before your first prompt.
IN YOUR TERMINAL YOU SEE
The input box, waiting for your first message. Everything above loads silently before you type anything.

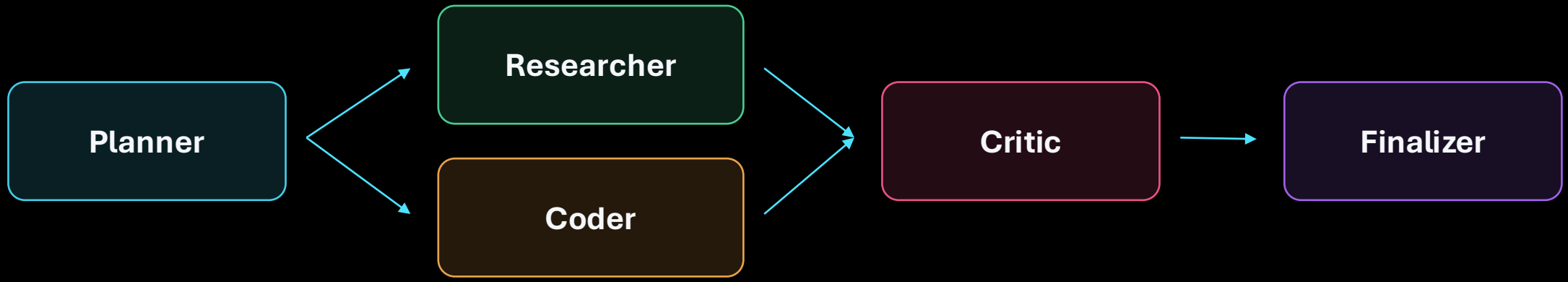
Summarization

Conversation → Event log → Running summary → Next turn

Summarize facts, decisions, open questions, and constraints separately.



Specialization



Specialization reduces context by narrowing the contract

Skills

PDF

SQL

Slides

Code

Eval

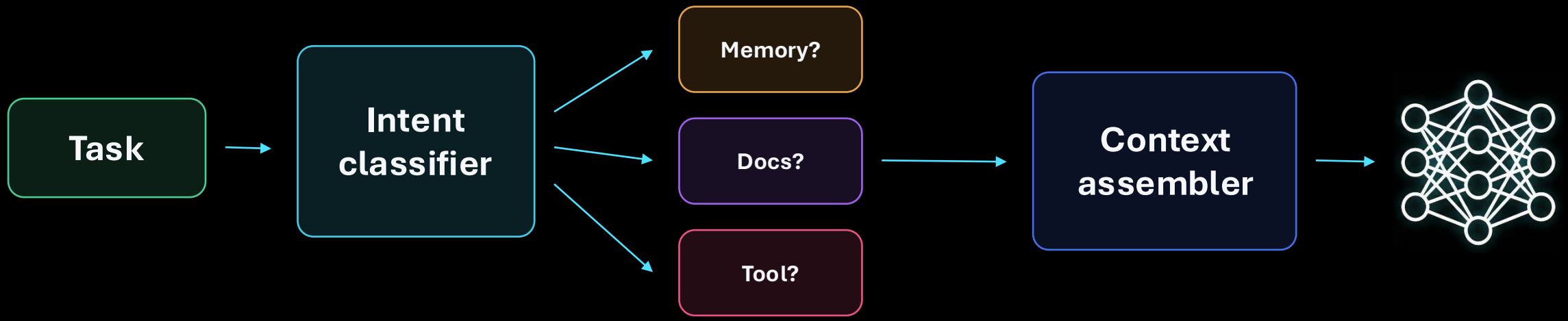
Legal

Modular context

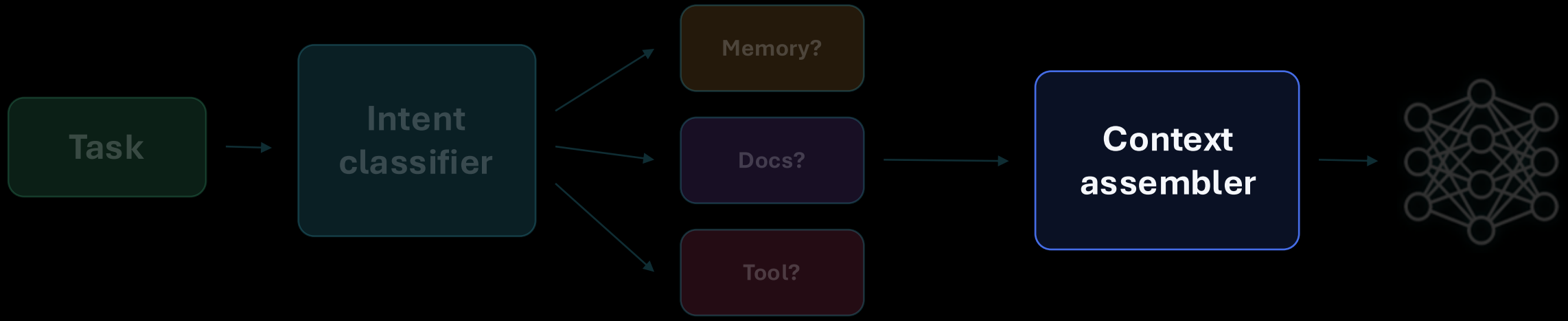
Build up your AI skills


NEXT LEVEL

The context router



The context router



caveman

why use many token when few do trick

stars 38k | last commit yesterday | license MIT

Before/After · Install · Levels · Skills · Benchmarks · Evals

Caveman Ecosystem · **caveman** *talk less* (you are here) · cavemem *remember more* · cavekit *build better*

A Claude Code skill/plugin and Codex plugin that makes agent talk like caveman — cutting **~75% of output tokens** while keeping full technical accuracy. Now with 文言文 mode, terse commits, one-line code reviews, and a compression tool that cuts **~46% of input tokens** every session.

Based on the viral observation that caveman-speak dramatically reduces LLM token usage without losing technical substance. So we made it a one-line install.

Before / After

Normal Claude (69 tokens)

> "The reason your React component is re-rendering is likely because you're creating a new object reference on each render cycle. When you pass an inline object as a prop, React's shallow comparison sees it as a different object every time, which triggers a re-render. I'd recommend using useMemo to memoize the object."

Caveman Claude (19 tokens)

> "New object ref each render. Inline object prop = new ref = re-render. Wrap in `useMemo`."

Normal Claude

> "Sure! I'd be happy to help you with that. The issue you're experiencing is most likely caused by your authentication middleware not properly validating the token expiry. Let me take a look

Caveman Claude

> "Bug in auth middleware. Token expiry check use `<` not `<=`. Fix:"

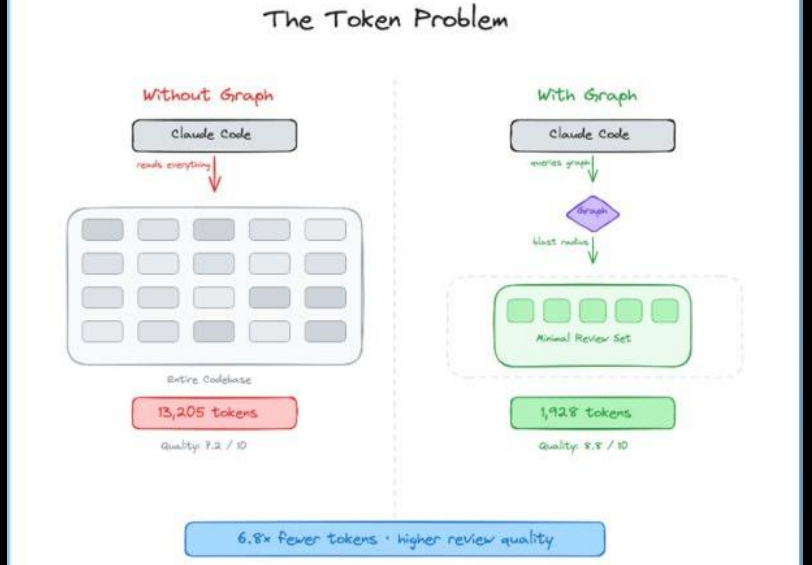
code-review-graph

Stop burning tokens. Start reviewing smarter.

English | 简体中文 | 日本語 | 한국어 | हिन्दी

pypi v2.3.2 | downloads 176k | stars 11k | License MIT | CI failing | python 3.10+ | MCP compatible | website code-review-graph.com | discord join

l coding tools re-read your entire codebase on every task. `code-review-graph` fixes that. It builds a structural ap of your code with Tree-sitter, tracks changes incrementally, and gives your AI assistant precise context via ICP so it reads only what matters.



TOKEN OPTIMIZER

GHOST TOKENS ARE EATING YOUR CONTEXT WINDOW

Find them. Fix them. Track the quality decay.

version 5.4.26 | Claude Code Plugin | OpenClaw v2.3.0 | license PolyForm Noncommercial | Stars 389 | last commit yesterday

python 3.8+ | dependencies zero | telemetry none | platform macOS | Linux | Windows | LinkedIn Connect | Sponsor

Your AI is getting dumber and you can't see it.

Save tokens. Survive compaction. Measure the proof.

Most token tools only touch one slice of the problem.

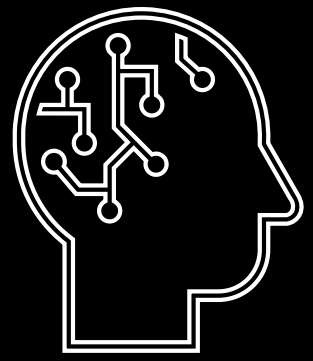
They compress command output, which covers 15-25% of your context on a good day. The other 75-85% (bloated configs, unused skills, duplicate system prompts, stale memory, plus the 60-70% you lose on every compaction) goes untouched.

Token Optimizer covers all of it, keeps your work alive across compactions, measures whether the optimization actually helped, and gives you a **live dashboard** that shows every token, every dollar, and every turn, auto-updated after every session. Runs fully local. Zero context tokens used. Zero runtime dependencies.

Works on **Claude Code** and **OpenClaw** today. Windsurf, Cursor, and more on the way.

```
Claude Code ~ ~/.claude

Claude Code  v2.1.75
Opus 4.6 · Claude Max · ~/.claude

❯ python3 measure.py quick

TOKEN OPTIMIZER: QUICK SCAN

  Context window: 1,000,000 tokens (1M, auto-detected)
  Startup overhead: 62,400 tokens (6.2%)
  Quality estimate: ~96/100 (MRCR peak zone)

DEGRADATION RISK
```

Takeaways



Treat context like production memory

small

fresh

ranked

scoped

auditable

Beware of the silent killers

Rot

cause: stale

cure: invalidate

Confusion

cause: conflicting

cure: prioritize

Pollution

cause: irrelevant

cure: filter

Context engineering is **more engineering!**

cesarsotovalero.net