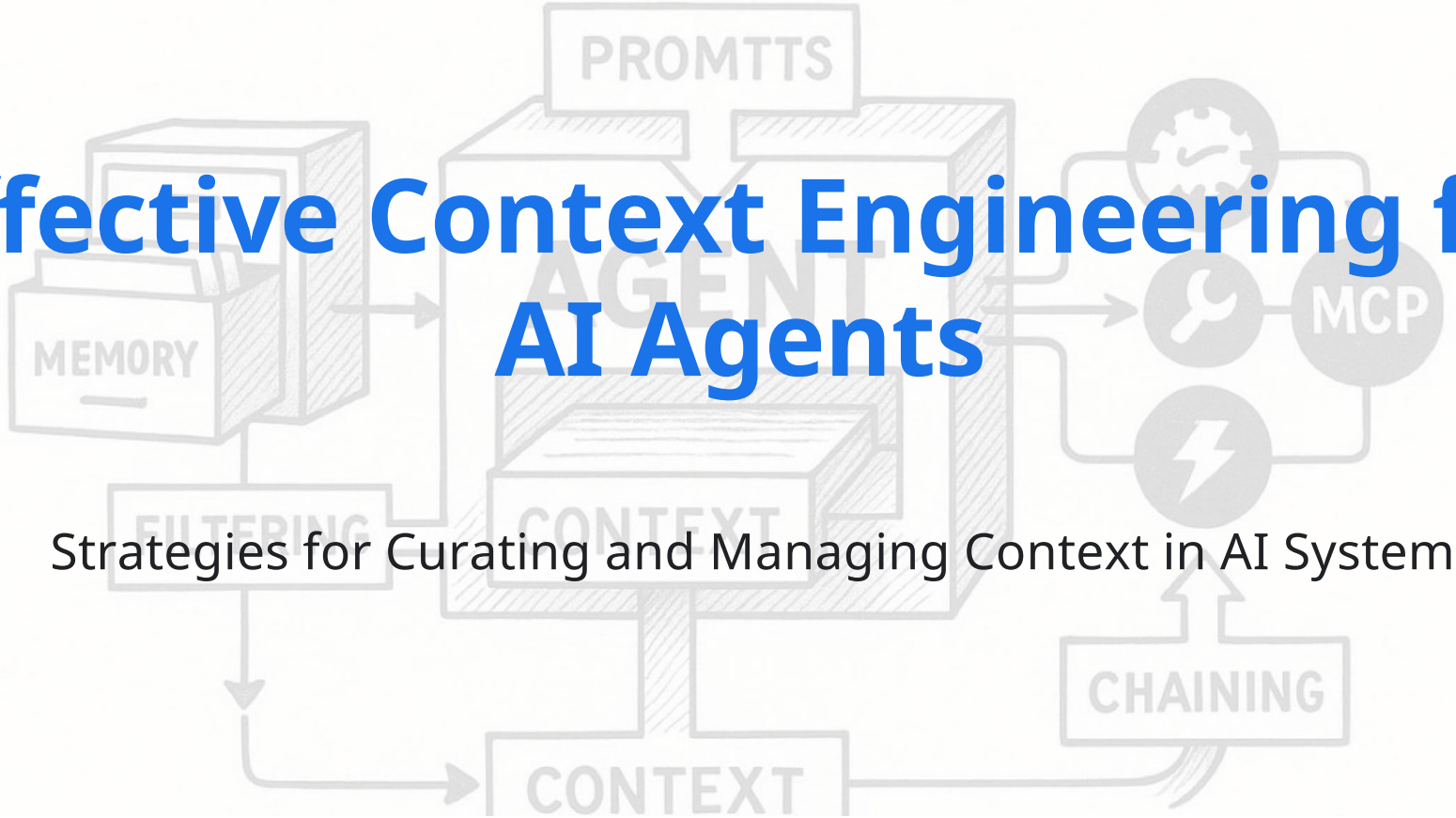
Effective Context Engineering for AI Agents

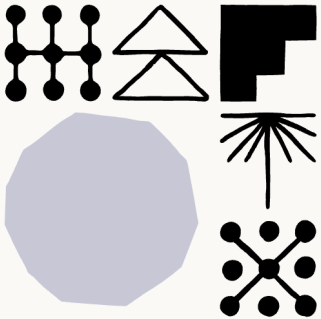
Strategies for Curating and Managing Context in AI Systems



ANTHROP\C

Research Economic Futures Commitments Learn News Try Claude

Engineering at Anthropic



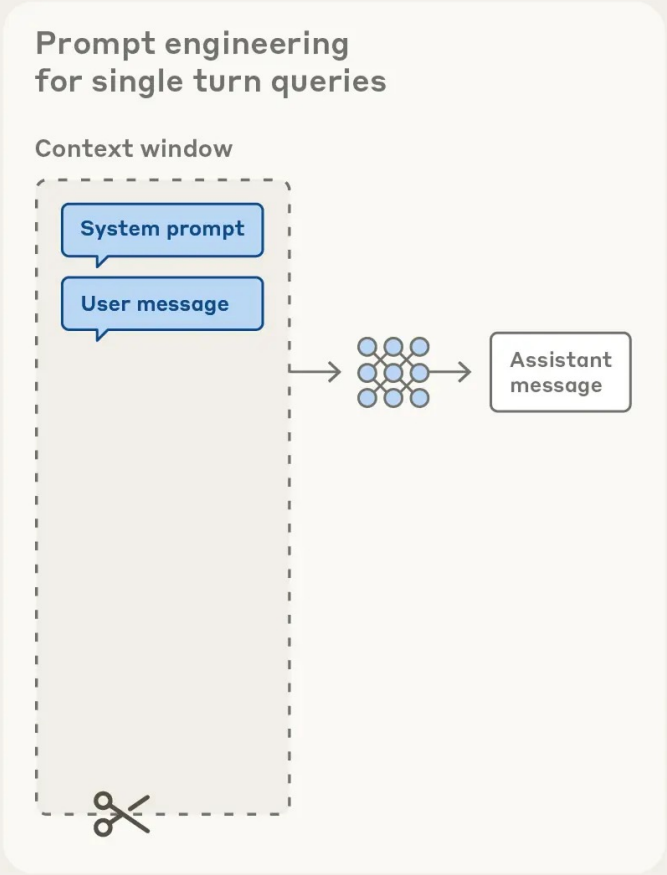
Effective context engineering for AI agents

Published Sep 29, 2025

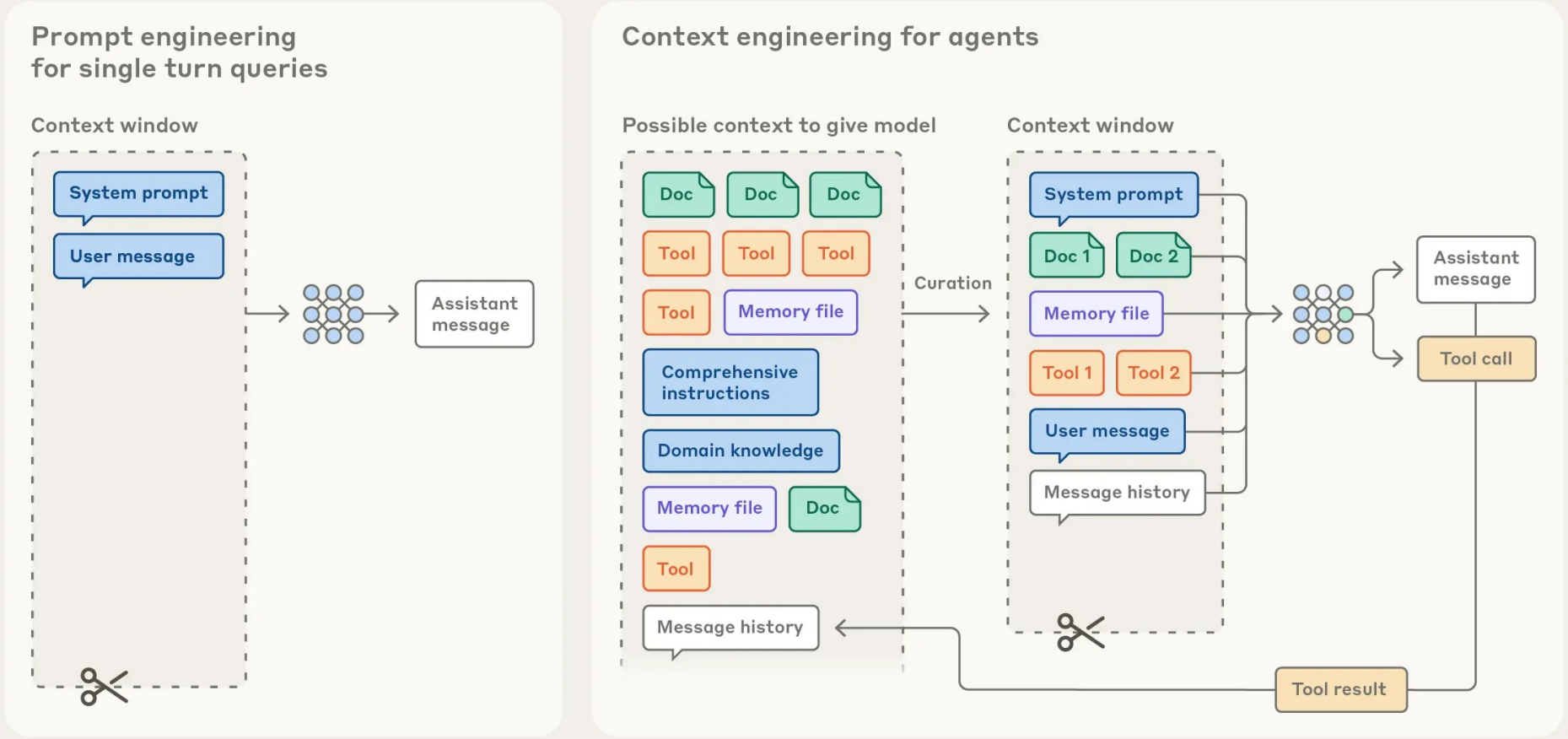
Context is a critical but finite resource for AI agents. In this post, we explore strategies for effectively curating and managing the context that powers them.

GPT-4o: 128K context tokens
GPT-5: 400K tokens
Claude 4.5: 200K tokens by default, up to 1M using API.
Gemini 2.5 Pro: 1M tokens

~~Prompt engineering~~ vs. context engineering



~~Prompt engineering~~ vs. context engineering



Context Engineering

Context

The set of tokens included when sampling from a large language model (LLM).

Context Engineering

Strategies for curating and maintaining the optimal set of tokens during LLM inference, including all information that may land there outside of the prompts.

Iterative process: context engineering happens each time we decide what to pass to the model.

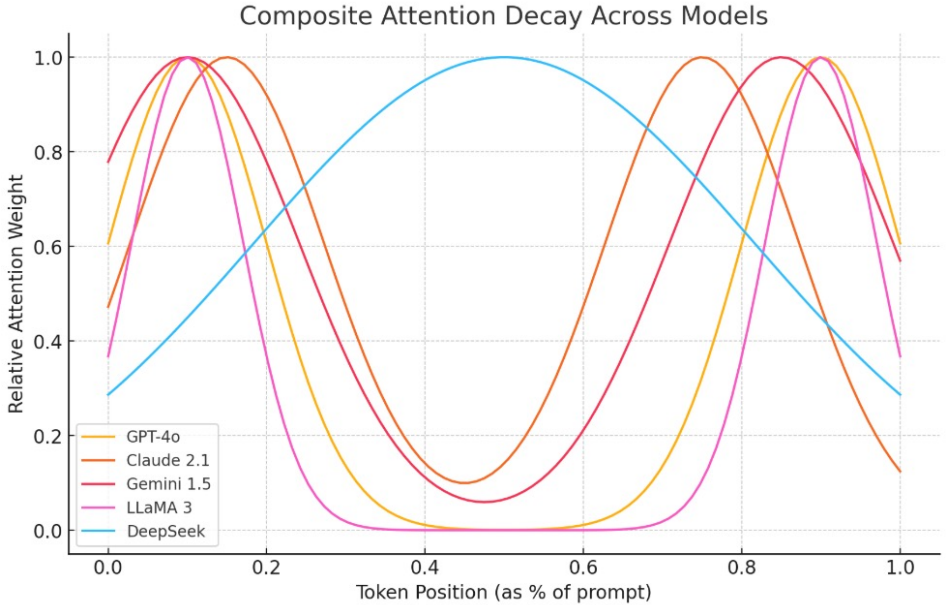
Why Context Engineering Matters

Despite their capabilities, LLMs lose focus or experience confusion at a certain point, similar to humans.

☐ Context Rot

As the number of tokens in the context window increases, the model's ability to accurately recall information from that context decreases.

“You paid for 128,000 tokens of context. You’re using maybe 30,000 effectively. The rest? Your model is ignoring them — and charging you anyway.”



Why Context Engineering Matters

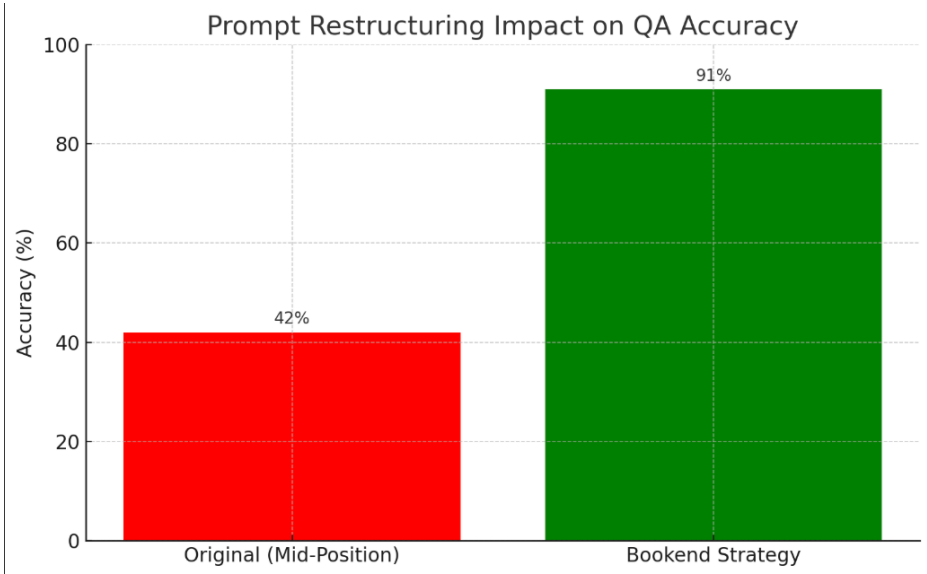
Despite their capabilities, LLMs lose focus or experience confusion at a certain point, similar to humans.

☐ Context Rot

As the number of tokens in the context window increases, the model's ability to accurately recall information from that context decreases.

The Bookend Strategy

Place your most critical context at the very beginning and again at the very end of your prompt.



Why Context Engineering Matters

Despite their capabilities, LLMs lose focus or experience confusion at a certain point, similar to humans.

Attention Budget

Like humans with limited working memory, LLMs have an "attention budget" that depletes with each new token introduced.

Long before information exceeds context window size, the transformer's ability to effectively represent and communicate this information within the window is exceeded. This means that current benchmark performance will not accurately capture performance over the full range of long-context reasoning tasks.

Low working memory

2 + 3 = ?

High working memory

13 * 28 = ?

Why Context Engineering Matters

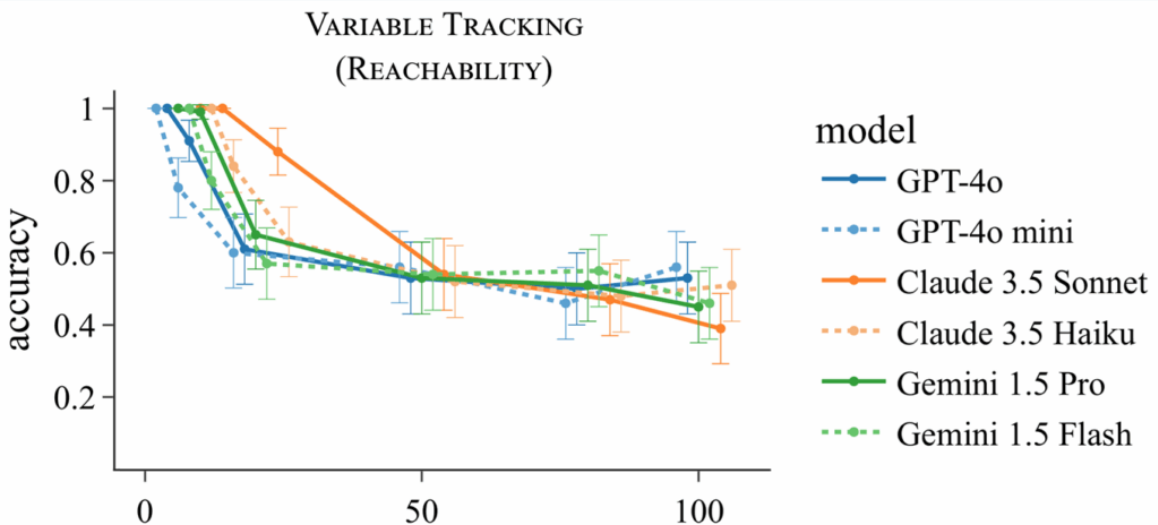
Despite their capabilities, LLMs lose focus or experience confusion at a certain point, similar to humans.

Attention Budget

Like humans with limited working memory, LLMs have an "attention budget" that depletes with each new token introduced.

Long before information exceeds context window size, the transformer's ability to effectively represent and communicate this information within the window is exceeded. This means that current benchmark performance will not accurately capture performance over the full range of long-context reasoning tasks.

Say we want to debug a certain part of someone's code and want to figure out whether the final value of the variable x7 is "a" or "b":



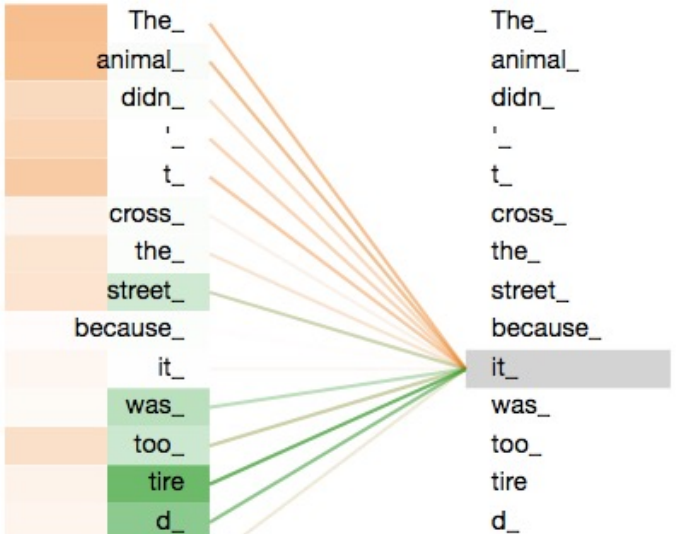

```
x6 = "a"
x4 = "b"
x0 = x6
x2 = x4
x3 = x0
x8 = x2
x9 = x3
x7 = x3
```

Why Context Engineering Matters

Despite their capabilities, LLMs lose focus or experience confusion at a certain point, similar to humans.

Architectural Constraints

Transformer architecture creates n^2 pairwise relationships between tokens, stretching the model's ability to capture dependencies at scale.



LLMs are based on the transformer architecture, which enables every token to attend to every other token across the entire context. This results in n^2 pairwise relationships for n tokens.

Models develop their attention patterns from training data distributions where shorter sequences are typically more common than longer ones. This means models have less experience with, and fewer specialized parameters for, context-wide dependencies.

Why Context Engineering Matters

Despite their capabilities, LLMs lose focus or experience confusion at a certain point, similar to humans.

Context Rot

As the number of tokens in the context window increases, the model's ability to accurately recall information from that context decreases.

Attention Budget

Like humans with limited working memory, LLMs have an "attention budget" that depletes with each new token introduced.

Architectural Constraints

Transformer architecture creates n^2 pairwise relationships between tokens, stretching the model's ability to capture dependencies at scale.

Thoughtful context engineering is essential for building capable agents

The Guiding Principle

> **Find the smallest possible set of high-signal tokens that maximize the likelihood of your desired outcome.**

Thinking in context: Consider the holistic state available to the LLM at any given time and what potential behaviors that state might yield.

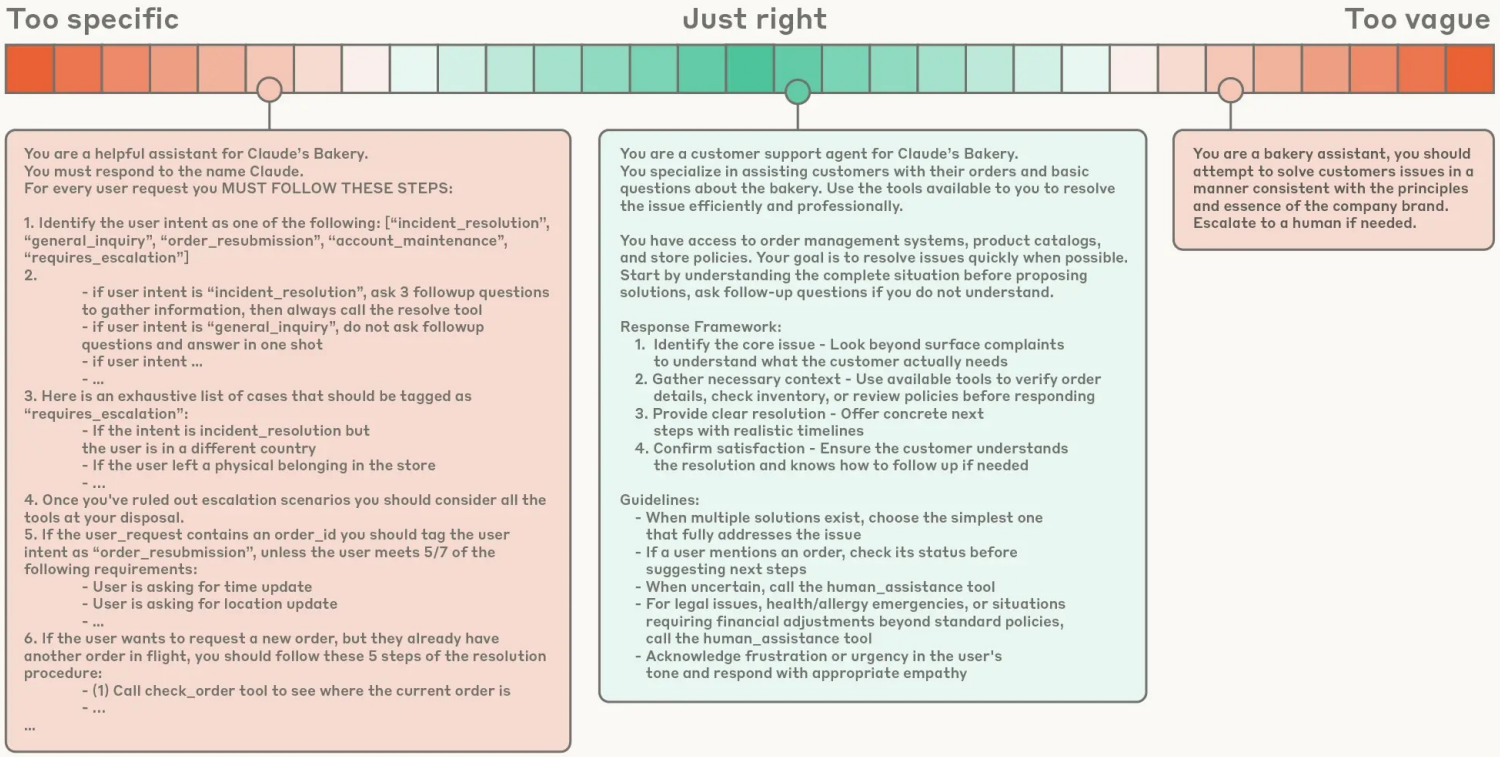
Like humans with limited working memory, LLMs have an **"attention budget"** that gets depleted with each new token.

The challenge is finding the right balance between providing sufficient information and respecting the model's attention constraints.

What if only ● is relevant?

Effective Context: System Prompts

System prompts should use **clear, simple, direct language** that presents ideas at the **right altitude** for the agent.



Effective Context: Tools

"If a human engineer can't definitively say which tool should be used in a given situation, an AI agent can't be expected to do better."

Tools allow agents to operate with their environment and pull in new, additional context as they work. Tools should be self-contained, robust to error, and extremely clear with respect to their intended use

- **Choosing the right tools to implement (and not to implement)**

  - Instead of implementing a `list_users`, `list_events`, and `create_event` tools, consider implementing a `schedule_event` tool which finds availability and schedules an event.
  - Instead of implementing a `read_logs` tool, consider implementing a `search_logs` tool which only returns relevant log lines and some surrounding context.
  - Instead of implementing `get_customer_by_id`, `list_transactions`, and `list_notes` tools, implement a `get_customer_context` tool which compiles all of a customer's recent & relevant information all at once.

Effective Context: Tools

"If a human engineer can't definitively say which tool should be used in a given situation, an AI agent can't be expected to do better."

- **Namespacing** (grouping related tools under common prefixes) can help delineate boundaries between lots of tools; MCP clients sometimes do this by default. For example,
  - By service (e.g., asana_search, jira_search)
  - By resource (e.g., asana_projects_search, asana_users_search)

- By selectively implementing tools whose names reflect natural subdivisions of tasks, you simultaneously reduce the number of tools and tool descriptions loaded into the agent's context and offload agentic computation from the agent's context back into the tool calls themselves. This reduces an agent's overall risk of making mistakes.

Effective Context: Tools

"If a human engineer can't definitively say which tool should be used in a given situation, an AI agent can't be expected to do better."

- **Returning meaningful context from your tools**

Detailed tool response

```
● I'll search slack for recent bug reports and use the d
investigate further.

● slack - search (MCP)(query: "bug", sort: "timestamp",
"detailed")
 L Search results for: "bug"

    ═══ Result 1 of 89 ═══
    Channel: #dev (C1234567890)
    From: @jane.doe (U123456789)
    Time: 2024-01-15 10:30:45 UTC
    TS: 1705316445.123456
    Text: Found a critical bug in the login flow.

    ═══ Result 2 of 89 ═══
    Channel: DM with @john.smith
    From: @john.smith (U987654321)
    Time: 2024-01-14 15:22:18 UTC
    TS: 1705247738.234567
    Text: The bug report for issue #123 is ready for review
    Files: bug-report-123.pdf
    ...
```

Concise tool response

```
● I'll search slack for recent bug reports and use the concise format to read as many messages as possible.

● slack - search (MCP)(query: "bug", sort: "timestamp", sortDir: "desc", limit: 100, responseFormat: "concise")
 L Search: "bug" (89 results)

    1. #dev - @jane.doe: Found a critical bug in the login flow. [Jan 15]
    2. DM - @john.smith: The bug report for issue #123 is ready for review [Jan 14]
    ...
```

Effective Context: Tools

"If a human engineer can't definitively say which tool should be used in a given situation, an AI agent can't be expected to do better."

- **Optimizing tool responses for token efficiency**

Unhelpful error response

```
● Sure. I'll fetch John's contact information from h

● asana - user_info (MCP)(userId: "john.doe@acme.co
  L {
      "error": {
        "code": "RESOURCE_NOT_FOUND",
        "status": 422,
        "message": "Invalid value",
        "details": {
          "field": "userId",
          "value": "john.doe@acme.corp",
          "type": "invalid_value"
        }
      }
    }
```

Helpful error response

```
● Sure. I'll fetch John's contact information from his profile.

● asana - user_info (MCP)(userId: "john.doe@acme.corp")
  L Tool Response:

    # Resource Not Found: Invalid `userId`

    ## Error Summary
    Your request to `/api/user/info` failed because the `userId` `john.doe@acme.corp` does not exist or is in
the wrong format.

    ## Valid User IDs
    Examples:
    - `1928298149291729`
    - `9381719375914731`

    ## Resolving a User ID
    - Call user_search()
```

Effective Context: Tools

"If a human engineer can't definitively say which tool should be used in a given situation, an AI agent can't be expected to do better."

- **Prompt-engineering your tool descriptions**

Example of good tool description

```
{
  "name": "get_stock_price",
  "description": "Retrieves the current stock price for a given ticker symbol. The ticker symbol must be a valid symbol for a publicly traded company on a major US stock exchange like NYSE or NASDAQ. The tool will return the latest trade price in USD. It should be used when the user asks about the current or most recent price of a specific stock. It will not provide any other information about the stock or company.",
  "input_schema": {
    "type": "object",
    "properties": {
      "ticker": {
        "type": "string",
        "description": "The stock ticker symbol, e.g. AAPL for Apple Inc."
      }
    },
    "required": ["ticker"]
  }
}
```

Example of poor tool description

```
{
  "name": "get_stock_price",
  "description": "Gets the stock price for a ticker.",
  "input_schema": {
    "type": "object",
    "properties": {
      "ticker": {
        "type": "string"
      }
    },
    "required": ["ticker"]
  }
}
```

Effective Context: Tools

"If a human engineer can't definitively say which tool should be used in a given situation, an AI agent can't be expected to do better."

Tools allow agents to operate with their environment and pull in new, additional context as they work.

Tools should be self-contained, robust to error, and extremely clear with respect to their intended use.

- **Choosing the right tools to implement (and not to implement)**
- **Namespacing** (grouping related tools under common prefixes**) can help delineate boundaries between lots of tools**
- **Optimizing tool responses for token efficiency**
- **Prompt-engineering your tool descriptions**

Effective Context: Context Retrieval Strategies

Just-in-Time Approach

Maintain lightweight identifiers (file paths, queries, web links) and dynamically load data into context at runtime using tools.

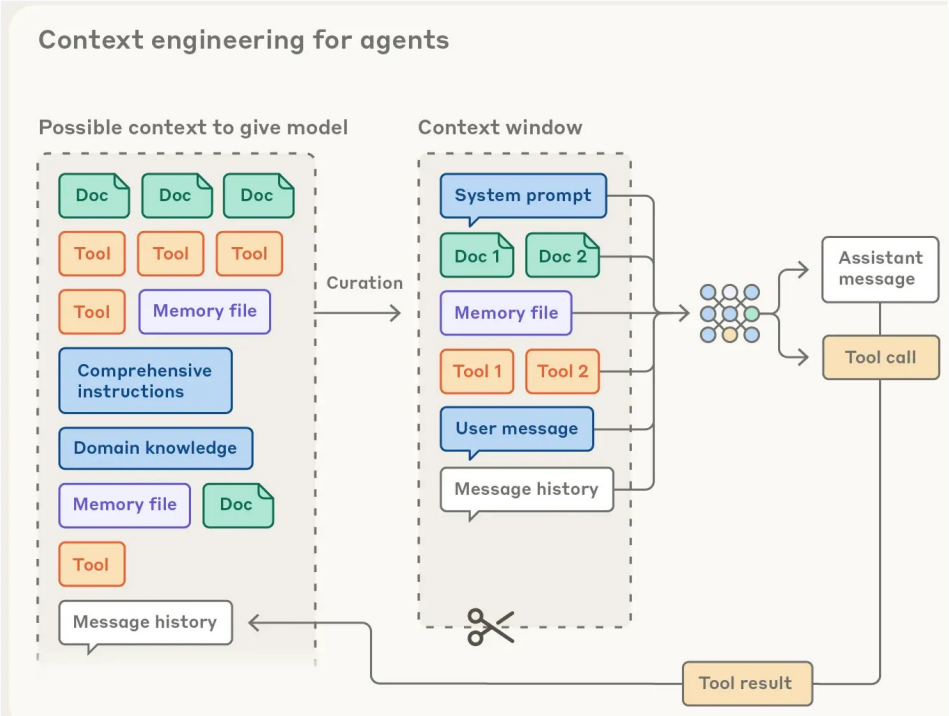
Progressive Disclosure

Agents incrementally discover relevant context through exploration, assembling understanding layer by layer.

Hybrid Strategy

Retrieve some data upfront for speed, pursue further autonomous exploration at the agent's discretion.

Trade-off: *Runtime exploration is slower than pre-computed retrieval, but offers more flexibility and adaptability to new findings.*



Effective Context: Context Retrieval Strategies

Just-in-Time Approach

Maintain lightweight identifiers (file paths, queries, web links) and dynamically load data into context at runtime using tools.

Anthropic's agentic coding solution Claude Code uses this approach to perform complex data analysis over large databases:

The model can write targeted queries, store results, and leverage Bash commands like head and tail to analyze large volumes of data without ever loading the full data objects into context.

This approach mirrors human cognition: we generally don't memorize entire corpuses of information, but rather introduce external organization and indexing systems like file systems, inboxes, and bookmarks to retrieve relevant information on demand.

Effective Context: Context Retrieval Strategies

Just-in-Time Approach

Maintain lightweight identifiers (file paths, queries, web links) and dynamically load data into context at runtime using tools.

Progressive Disclosure

Agents incrementally discover relevant context through exploration, assembling understanding layer by layer.

Each interaction yields context that informs the next decision:

- file sizes suggest complexity
- naming conventions hint at purpose
- timestamps can be a proxy for relevance

Agents can assemble understanding layer by layer, maintaining only what's necessary in working memory and leveraging note-taking strategies for additional persistence. This self-managed context window keeps the agent focused on relevant subsets rather than drowning in exhaustive but potentially irrelevant information.

Effective Context: Context Retrieval Strategies

Just-in-Time Approach

Maintain lightweight identifiers (file paths, queries, web links) and dynamically load data into context at runtime using tools.

Progressive Disclosure

Agents incrementally discover relevant context through exploration, assembling understanding layer by layer.

Hybrid Strategy

Retrieve some data upfront for speed, pursue further autonomous exploration at the agent's discretion.

Claude Code is an agent that employs this hybrid model: CLAUDE.md files are naively dropped into context up front, while primitives like glob and grep allow it to navigate its environment and retrieve files just-in-time, effectively bypassing the issues of stale indexing and complex syntax trees.

CLAUDE.md: concise, human-readable, tuned

```
# Project Overview: Todo List Application

This project is a simple, full-stack Todo list application. The front end i

## Architecture and Patterns
- The front-end communicates with the back-end via REST API calls.
- The back-end implements a simple CRUD API for managing todo items.
- The database schema is defined in `backend/db/schema.sql`.

## Key Files
- `frontend/src/App.js`: Main React component.
- `backend/src/server.js`: Express server entry point.
- `backend/db/schema.sql`: Database schema.

## Development Workflow Instructions
- **To add a new API endpoint:** Always define the route in `backend/src/se
- **To add a new UI component:** Create a new file in `frontend/src/compone
- **Testing Strategy:**
    - Unit tests for all utility functions.
    - Integration tests for the API endpoints.
- **IMPORTANT**: Before committing, always run the linter and tests.

## Custom Commands
- `fix-linter-errors`: Use `/project:fix-linter-errors` to fix all linting
- `run-api-tests`: Use `/project:run-api-tests` to run the backend API test
```

Context Engineering for Long-Horizon Tasks

Long-horizon tasks require agents to maintain coherence and context over sequences of actions where the token count exceeds the LLM's context window.

☐ Compaction

Summarize conversation nearing context limit and reinitiate with the summary. Preserves critical details while discarding redundant information.

Best for: Extensive back-and-forth tasks

In Claude Code, for example, we implement this by passing the message history to the model to summarize and compress the most critical details. The model preserves architectural decisions, unresolved bugs, and implementation details while discarding redundant tool outputs or messages.

Start by maximizing recall to ensure your compaction prompt captures every relevant piece of information from the trace, then iterate to improve precision by eliminating superfluous content.

Context Engineering for Long-Horizon Tasks

Long-horizon tasks require agents to maintain coherence and context over sequences of actions where the token count exceeds the LLM's context window.

☐ Compaction

Summarize conversation nearing context limit and reinitiate with the summary. Preserves critical details while discarding redundant information.

Best for: Extensive back-and-forth tasks

☐ Structured Note-Taking

Agent writes notes persisted outside context window, pulled back in at later times. Creates persistent memory with minimal overhead.

Best for: Iterative development with clear milestones

Like Claude Code creating a to-do list, or your custom agent maintaining a NOTES.md file, this simple pattern allows the agent to track progress across complex tasks, maintaining critical context and dependencies that would otherwise be lost across dozens of tool calls.

Context Engineering for Long-Horizon Tasks

Long-horizon tasks require agents to maintain coherence and context over sequences of actions where the token count exceeds the LLM's context window.

Compaction

Summarize conversation nearing context limit and reinitiate with the summary. Preserves critical details while discarding redundant information.

Best for: Extensive back-and-forth tasks

Structured Note-Taking

Agent writes notes persisted outside context window, pulled back in at later times. Creates persistent memory with minimal overhead.

Best for: Iterative development with clear milestones

Sub-Agent Architectures

Specialized sub-agents handle focused tasks with clean context windows, returning condensed summaries to the main agent.

Best for: Complex research and analysis with parallel exploration

Each subagent might explore extensively, using tens of thousands of tokens or more, but returns only a condensed, distilled summary of its work (often 1,000-2,000 tokens).

High-level Architecture of Advanced Research

Claude.ai chat

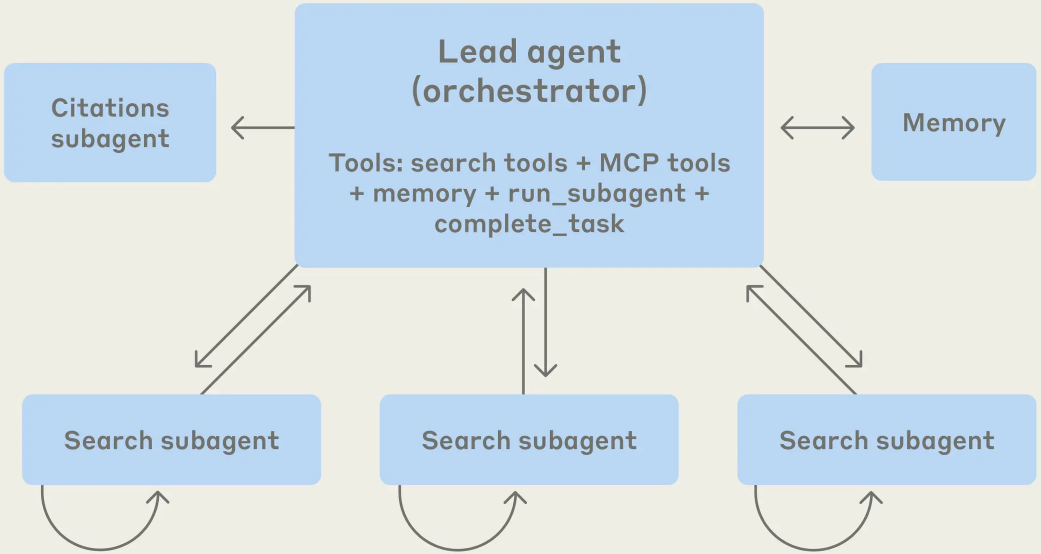
What's on your mind tonight?

what are all the companies in the united states working on AI agents in 2025? make a list of at least 100. for each company, include the name, website, product, description of what they do, type of agents they build, and their vertical/industry.

User request

Final report

Multi-agent research system

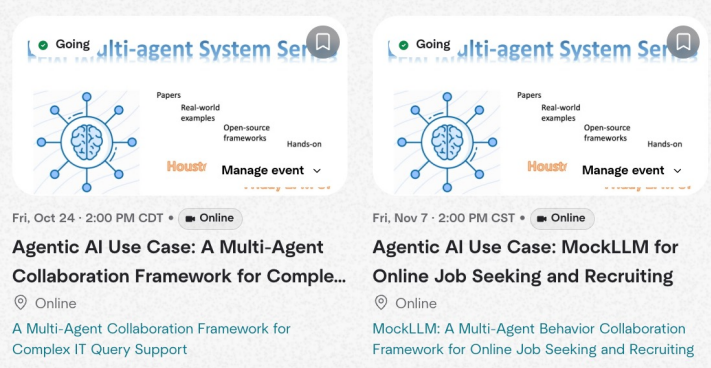


Key Takeaways

- Context is a **critical but finite resource** for AI agents that must be carefully managed.
- **Think in context** : Consider the holistic state available to the LLM and what behaviors it might yield.
- Find the **smallest possible set of high-signal tokens** that maximize likelihood of desired outcome, across the system prompt, tool design and context retrieval etc.
- Choose appropriate techniques for long-horizon tasks: **compaction, note-taking, or multi-agent architectures**.
- Follow the **simplicity first principle:** : Add complexity only when needed.

“Do the simplest thing that works”

Upcoming meetups – Open to proposals and guest speakers!



Slides posted at:
https://github.com/YanXuHappygela/LLM-reading-group

Recordings posted at:

YanAITalk

@yanaitalk · 3.55K subscribers · 72 videos

Make machine learning easy to understand! ...more