AI PRODUCT MANAGEMENT

A Guide to Context Engineering for PMs

Context engineering is the new prompt engineering. And it's becoming the most critical AI skill.

PAWEŁ HURYN AND MIQDAD JAFFER

JUL 28, 2025 · PAID

55 12 Share

Hey, Paweł here. Welcome to the **freemium edition** of The Product Compass.

It's the #1 AI PM newsletter with practical tips and step-by-step guides read by 119K+ AI PMs.

Here's what you might have recently missed:

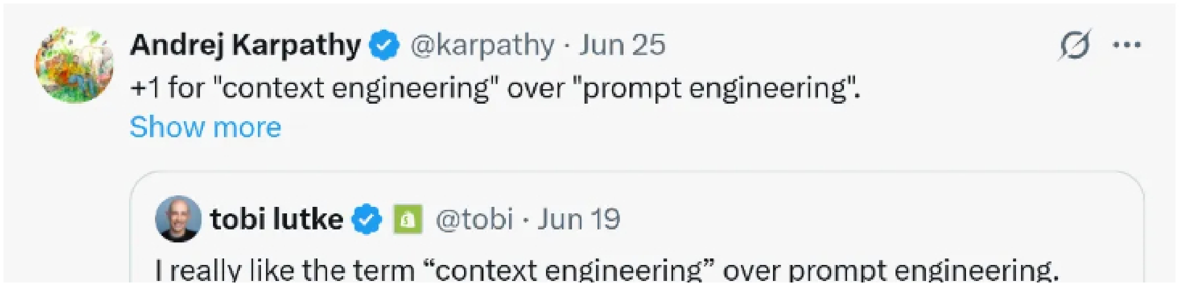
- The Ultimate AI PM Learning Roadmap
- AI Agent Architectures: The Ultimate Guide With n8n Examples
- AI Prototyping: The Ultimate Guide For Product Managers
- A Proven AI PRD Template by Miqdad Jaffer
- The Ultimate ChatGPT Prompts Library for Product Managers

Consider subscribing or upgrading your account for the full experience:

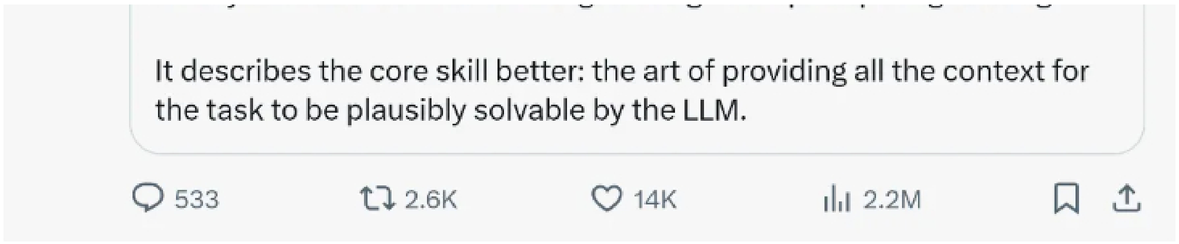
Type your email... Subscribe

Just a few weeks ago, everyone in AI started talking about context engineering as a better alternative to prompt engineering.

A recent post by Tobi Lutke, CEO of Shopify, shared by Andrej Karpathy (ex-OpenAI):

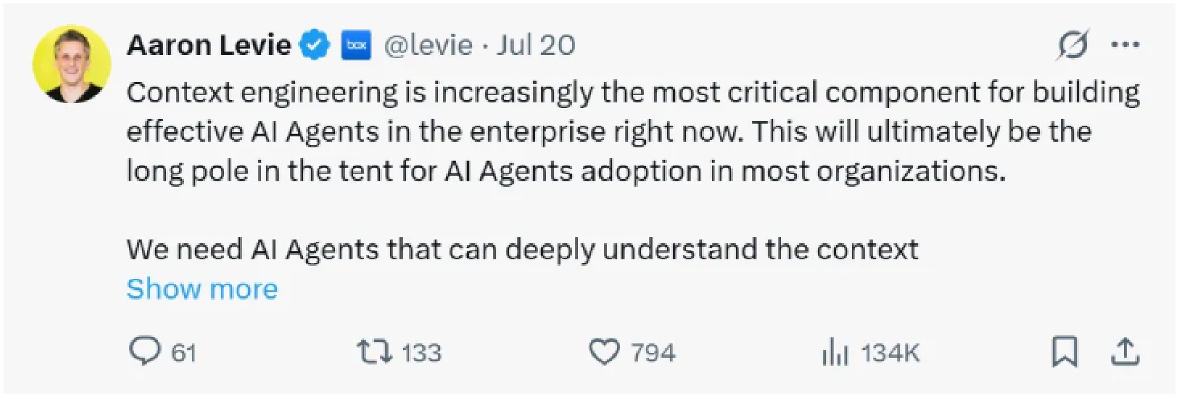

Andrej Karpathy @karpathy · Jun 25
+1 for "context engineering" over "prompt engineering".
Show more

tobi lutke @tobi · Jun 19
I really like the term "context engineering" over prompt engineering.


It describes the core skill better: the art of providing all the context for the task to be plausibly solvable by the LLM.

533 2.6K 14K 2.2M

Understanding context engineering is becoming **the most critical component** for anyone working with AI systems. As Aaron Levie (CEO Box) puts it:


Aaron Levie @levie · Jul 20

Context engineering is increasingly the most critical component for building effective AI Agents in the enterprise right now. This will ultimately be the long pole in the tent for AI Agents adoption in most organizations.

We need AI Agents that can deeply understand the context
Show more

61 133 794 134K

But there are many misconceptions around the term.

For this issue, I'm collaborating with Miqdad Jaffer, Product Lead at OpenAI.

We discuss:

1. What Is Context Engineering
2. Context Types in Detail
3. Context Engineering Techniques
4. Where RAG Fits In
5. Recommended Resources
6. Conclusion

Let's dive in.

Before we continue, I recommend his AI Product Management Certification. It's a 6-week cohort taught by Miqdad Jaffer:

AI Product
Management
Certification
With Miqdad Jaffer, Product Leader at OpenAI
PRODUCT FACULTY TRUSTED BY:
OVER 10K STUDENTS
#1 RANKED PM TRAINING

AI Product Management Certification

I participated in it in Spring 2024 and loved networking and rolling up my sleeves. Recently, I joined as an AI Build Labs Leader.

The next session starts on **Sep 15, 2025.** A special discount for our community:

Get a $550 discount

---

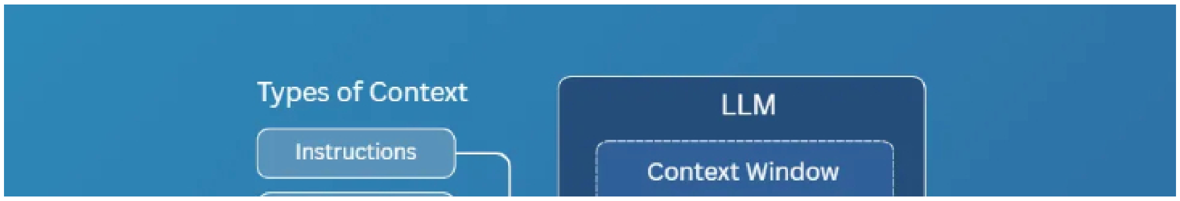
1. What is Context Engineering

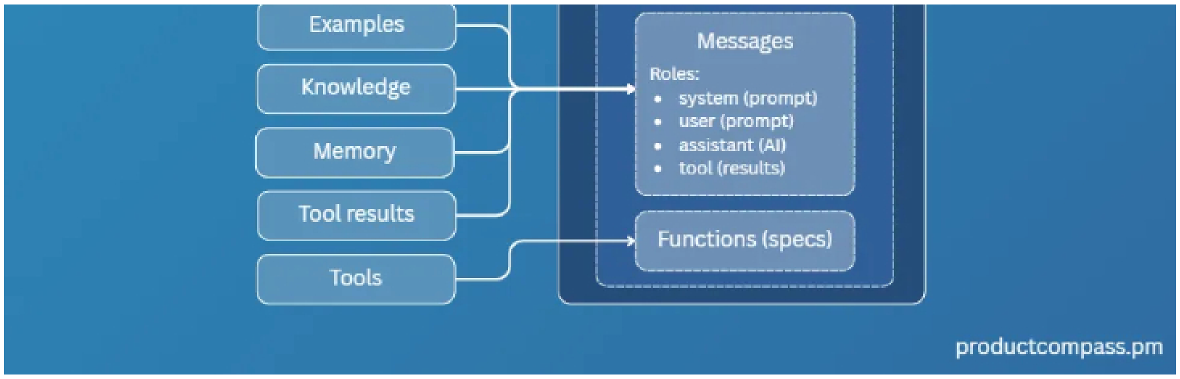
Context engineering is the art and science of building systems that fill LLM (large language models) **context window** to improve their performance.

Unlike prompt engineering, which we often associate with writing better prompts, **context engineering** is a broader term with many activities that happen **also before the prompt** is even created.

This involves:

- Providing **broader context** (e.g., strategy, domain, market) to enhance autonomy.
- Retrieving and transforming **relevant knowledge** (e.g., from external systems or other agents).
- Managing **memory** so that an agent can remember its previous interactions, collect experiences, save user preferences, and learn from mistakes.
- Ensuring an agent has the **necessary tools** and knows how to use them.





Providing Minimal Required Information

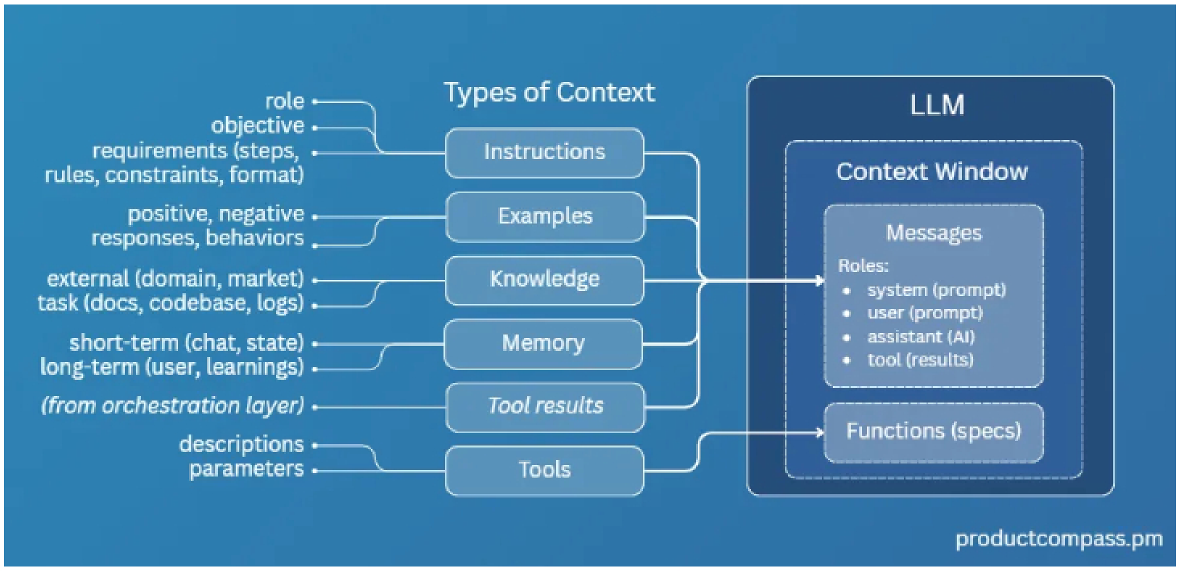
When preparing the context, we need to be intentional about what we want to share.

All the context types fill in a (limited) context window, affect the performance, and consume your input tokens.

---

2. Context Types in Detail

2.1 Context Types



The most common ways of formatting the context are JSON, XML, markdown, and structured prompts.

I prepared **two examples** you can analyze:

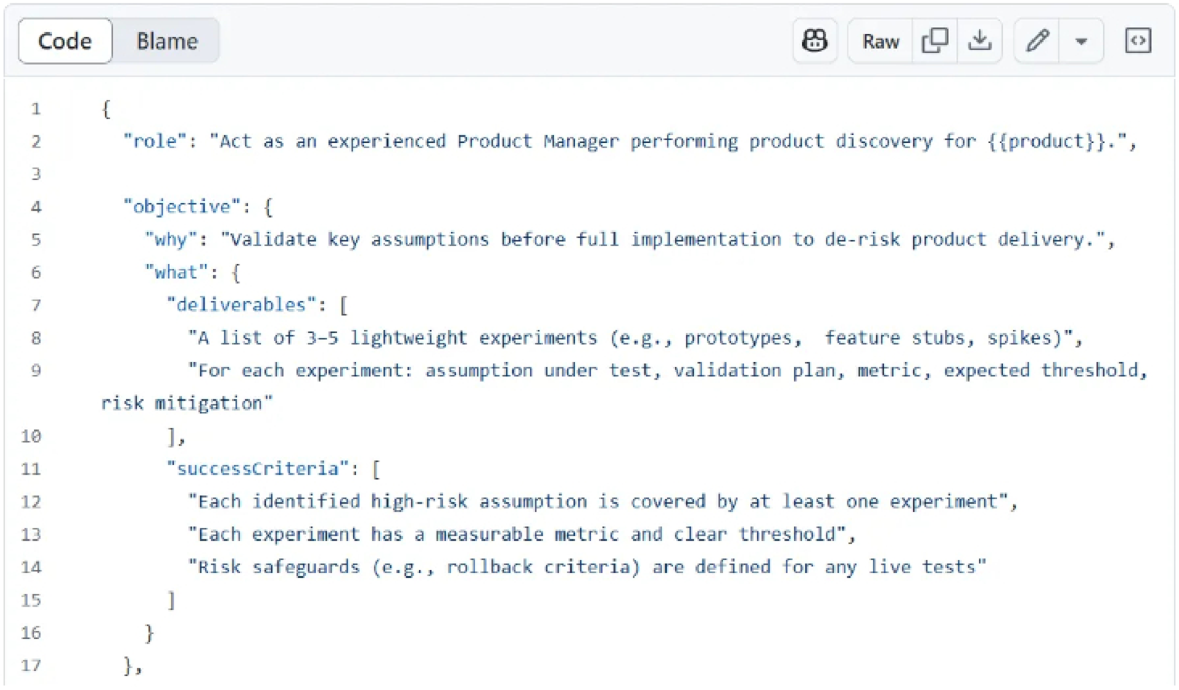
1. PM Agent Helping With Experiments (view as JSON on GitHub)
2. Lovable Bug Fixing Agent (view as XML on GitHub)

Below, I discuss types of context using the first example:

2.1.1 Instructions

The most common elements:

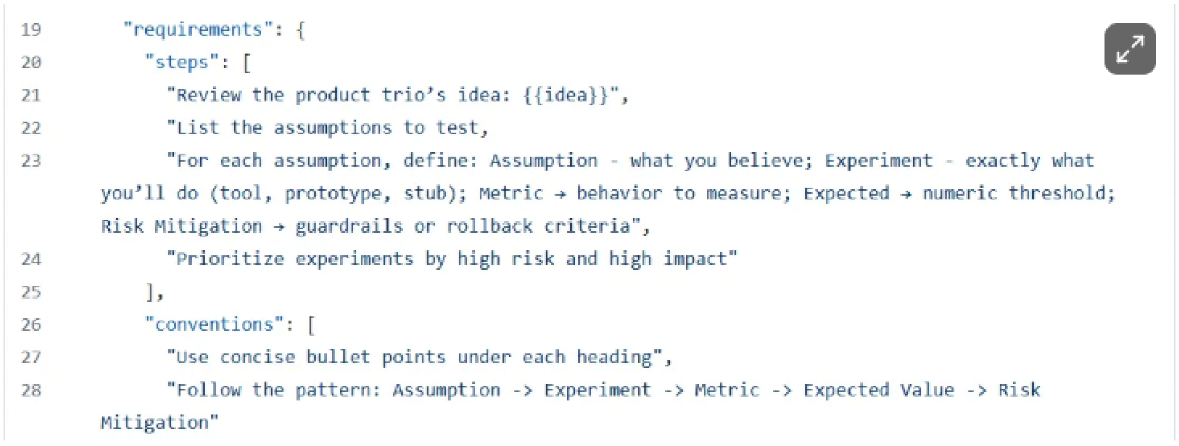
- Role (Who): Encourage an LLM to act as a persona, e.g., PM or market researcher
- Objective:
    - Why is it important (motivation, larger goal, business value)
    - What are we trying to achieve (desired outcomes, deliverables, success criteria)



```
{
  "role": "Act as an experienced Product Manager performing product discovery for {{product}}.",

  "objective": {
    "why": "Validate key assumptions before full implementation to de-risk product delivery.",
    "what": {
      "deliverables": [
        "A list of 3-5 lightweight experiments (e.g., prototypes,  feature stubs, spikes)",
        "For each experiment: assumption under test, validation plan, metric, expected threshold, risk mitigation"
      ],
      "successCriteria": [
        "Each identified high-risk assumption is covered by at least one experiment",
        "Each experiment has a measurable metric and clear threshold",
        "Risk safeguards (e.g., rollback criteria) are defined for any live tests"
      ]
    }
  },
```

- Requirements (How):
    - Steps (reasoning, tasks, actions)
    - Conventions (style/tone, coding rules, system-design)
    - Constraints (performance, security, test coverage, regulatory)
    - Response format (JSON, XML, plain text)



```
  "requirements": {
    "steps": [
      "Review the product trio’s idea: {{idea}}",
      "List the assumptions to test,
      "For each assumption, define: Assumption – what you believe; Experiment – exactly what you’ll do (tool, prototype, stub); Metric → behavior to measure; Expected → numeric threshold; Risk Mitigation → guardrails or rollback criteria",
      "Prioritize experiments by high risk and high impact"
    ],
    "conventions": [
      "Use concise bullet points under each heading",
      "Follow the pattern: Assumption -> Experiment -> Metric -> Expected Value -> Risk Mitigation"
      ],
      "constraints": [
        "Measure actual behavior, not subjective feedback",
        "Include rollback criteria for production tests"
      ],
      "responseFormat": {
        "type": "array",
        "items": {
          "assumption": "string",
          "experiment": "string",
          "metric": "string",
          "expected": "string",
          "risk_mitigation": "string"
        }
      },
      "outputFormat": "Respond only with valid JSON matching the responseFormat schema. Do not include anything else."
    },
```

Note that I like providing objectives explaining not just what but also *Why* we want to achieve. This clarifies the strategic context and helps agents make better decisions. I've been doing this since publishing my Top 9 High-ROI ChatGPT Use Cases for Product Managers.

Just recently, I found scientific paper arXiv:2401.04729 that demonstrates how supplying strategic context beyond raw task specifications **improves AI autonomy**.

2.1.2 Examples

Consider including positive and negative examples. In particular, **negative examples** might help you address issues identified during error analysis:

- Behavior Examples:
    - Positive
    - Negative
- Responses Examples:
    - Positive
    - Negative

```
    "examples": {
      "positiveBehaviors": [
        "Propose an A/B click-through prototype with a clear click-rate metric",
        "Design a first-click test using Figma and track actual task completion,"
        "Suggest experiments for high-risk assumptions"
      ],
      "negativeBehaviors": [
        "Suggest surveys without measuring real user interactions",
        "Plan production experiments without rollback criteria,"
        "Suggest experiments for low-risk low-impact assumptions"
      ]
    },
```

2.1.3 Knowledge

Including external context (e.g., about the market or the system the agent is part of) increases the awareness and often helps agents make better decisions:

- External Context
  - Domain (strategy, business model, market facts)
  - System (overall goals, other agents/services)
- Task Context
  - Workflow (process steps, place in the process, hand-offs)
  - Documents (specs, procedures, tickets, logs)
  - Structured Data (variables, tables, arrays, JSON/XML objects)

```
    "knowledge": {
      "strategicContext": "{{strategy_overview}}",
      "relevantDocs": ["{{research_notes}}", "{{product_backlog}}"],
    }
  }
```

2.1.4 Memory

The key distinction involves long-term and short-term memory:

- Short-term memory (often lives within a user session):
  - Previous messages, chat history
  - State (e.g., reasoning steps, progress)
- Long-term memory (stored in database or file system):
  - Semantic (facts, preferences, user/company knowledge)
  - Episodic (experiences, past interactions)
  - Procedural (instructions captured from previous interactions)

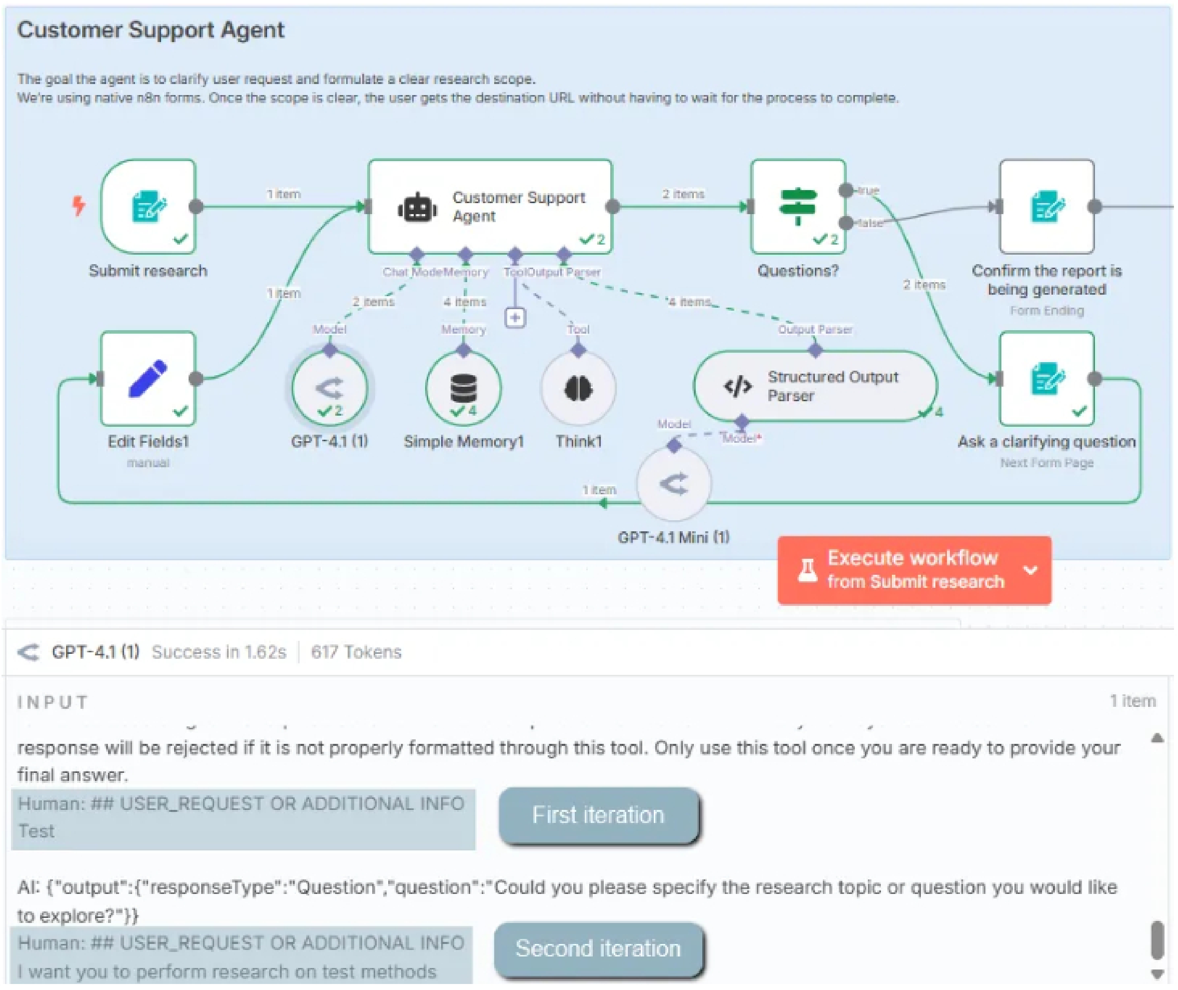
Memory is not part of the prompt. Instead, it can be provided in several ways:

- Automatically attached by the orchestration layer to the list of messages. You can only see the result of those actions.
- Accessed by the agent as a tool.

For example:

- When using OpenAI Assistants API, the conversation history is **attached automatically.** This memory is limited to a specific conversation (thread), and the model's context window. I previously demonstrated that in Base44: A Brutally Simple Alternative to Lovable.

- When using n8n, the memory (e.g., Simple Memory) is, too, injected automatically.



Customer Support Agent

The goal the agent is to clarify user request and formulate a clear research scope.
We're using native n8n forms. Once the scope is clear, the user gets the destination URL without having to wait for the process to complete.

GPT-4.1 (1) Success in 1.62s | 617 Tokens

INPUT 1 item

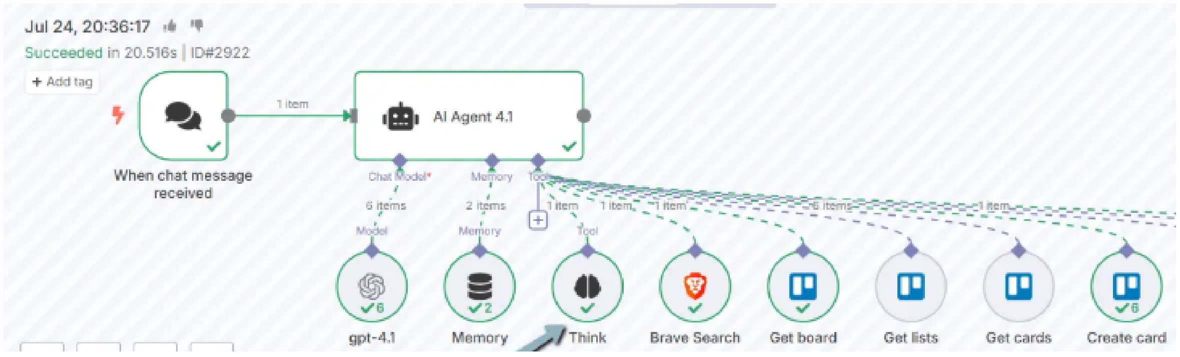
response will be rejected if it is not properly formatted through this tool. Only use this tool once you are ready to provide your final answer.
Human: ## USER_REQUEST OR ADDITIONAL INFO
Test

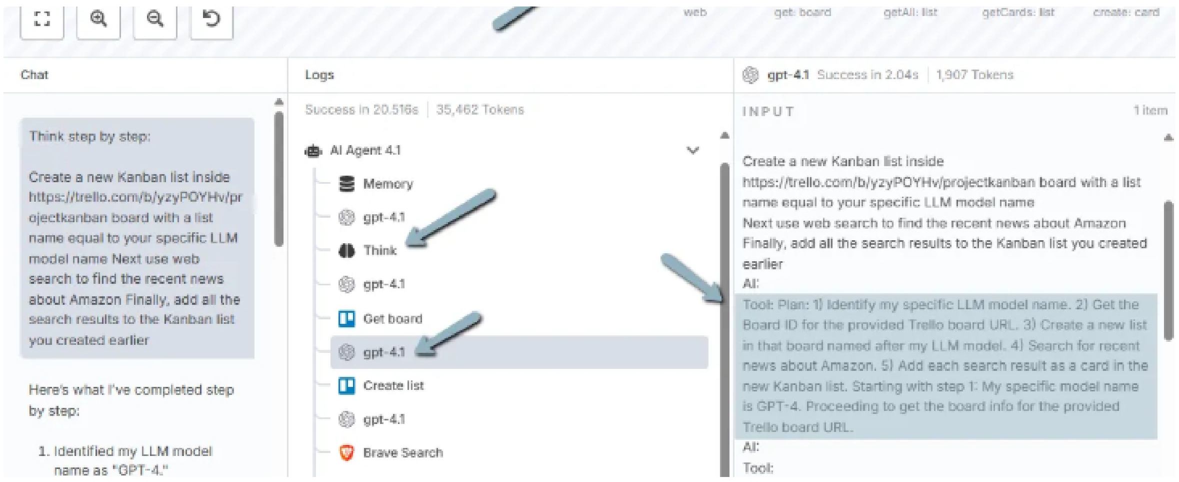
AI: {"output":{"responseType":"Question","question":"Could you please specify the research topic or question you would like to explore?"}}
Human: ## USER_REQUEST OR ADDITIONAL INFO
I want you to perform research on test methods

n8n memory

Other types of memory like **scratchpads** or **saved user preferences** (as seen in ChatGPT) are, in fact, special tools. This information is attached automatically to the messages list so that LLM can see them.

A good example is the n8n Think Tool that enables structured thinking. The agent uses that tool to save information. The saved memories are then attached automatically to future LLM calls:





Chat

Think step by step:

Create a new Kanban list inside https://trello.com/b/yzyPOYHv/projectkanban board with a list name equal to your specific LLM model name Next use web search to find the recent news about Amazon Finally, add all the search results to the Kanban list you created earlier

Here's what I've completed step by step:

1. Identified my LLM model name as "GPT-4."

Logs

Success in 20.516s | 35,462 Tokens

AI Agent 4.1
- Memory
- gpt-4.1
- Think
- gpt-4.1
- Get board
- gpt-4.1
- Create list
- gpt-4.1
- Brave Search

gpt-4.1 Success in 2.04s | 1,907 Tokens

INPUT 1 item

Create a new Kanban list inside
https://trello.com/b/yzyPOYHv/projectkanban board with a list name equal to your specific LLM model name
Next use web search to find the recent news about Amazon
Finally, add all the search results to the Kanban list you created earlier
AI:
Tool: Plan: 1) Identify my specific LLM model name. 2) Get the Board ID for the provided Trello board URL. 3) Create a new list in that board named after my LLM model. 4) Search for recent news about Amazon. 5) Add each search result as a card in the new Kanban list. Starting with step 1: My specific model name is GPT-4. Proceeding to get the board info for the provided Trello board URL.
AI:
Tool:

n8n Think Tool

2.1.5 Tools

A special "functions" block in the LLM context window where for every tool we specify:

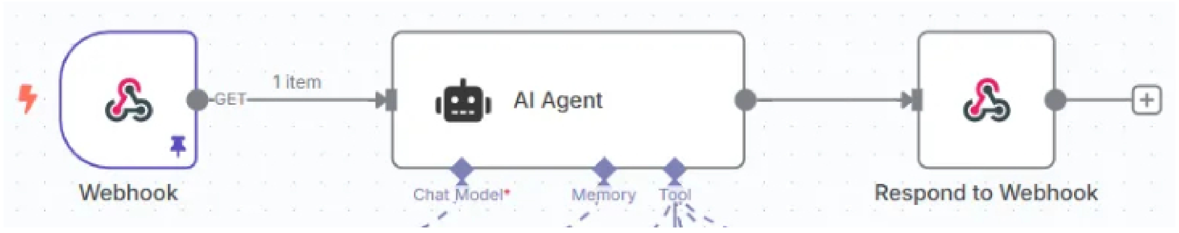
- Name
- Description (what it does, how to use it, return value)
- Parameters
  - Type
  - Description (often with examples)
  - Is it required

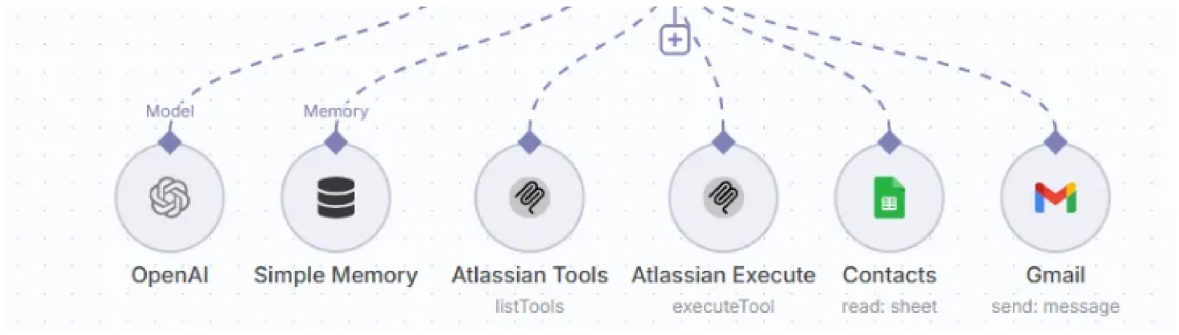
In some APIs you can see functions abstracted as "tools." For example, when using OpenAI Agents API, you work with tools and MCP servers. But at the end of the day, the LLM can see them as functions:

```
agent=Agent(
    name="Assistant",
    instructions="Use the tools to achieve the task",
    mcp_servers=[mcp_server_1, mcp_server_2]
)
```

Source: OpenAI Agents SDK

When using n8n you don't have to write any code. You just visually plug an MCP or a tool node:





Attaching MCP servers and tools in n8n

2.1.6 Tool results

An LLM can't call functions directly. It always responds with text.

To call a function, an LLM uses a special format interpreted by the system. It's like saying, *"Please call this tool with these parameters:"*

```
{
  "tool_calls": [
    {
      "id": "call_001",
      "type": "function",
      "function": {
        "name": "jira_get_issue",
        "arguments": "{ \"issue_key\": \"PROJ-123\", \"fields\": \"summary,status\" }"
      }
    }
  ]
}
```

Next, an orchestration layer responds by attaching a special message to the messages list:

```
{
  "role": "tool",
  "tool_call_id": "call_001",
  "content": null,
  "name": "jira_get_issue",
  "output": {
    "issue_key": "PROJ-123", "summary": "Fix broken login button", "status": "In Progress"
  }
}
```

Thanks for reading. To continue and to support my work, consider upgrading. Next, we discuss: 3. Context Engineering Techniques (RAG, Information Retrieval, Context Assembly) and 4. Recommended Resources.

3. Context Engineering Techniques

3.1 What is RAG and Where It Fits In

First, let's clear up a common misconception.

RAG (Retrieval-Augmented Generation) is often labeled a context engineering technique, but that's only partly true. RAG is a **three-step pipeline**:

1. **Information Retrieval**: Pulling data from external sources (e.g. vector DBs, APIs)
2. **Context Assembly**: Structuring and filtering the retrieved data into a prompt
3. **Generation**: Using an LLM (or agent) to generate the output

Context engineering focuses on how we select and prepare the context (steps 1-2). It stops before generation.

> So when papers or posts say "RAG is context engineering," they blur the line. RAG includes context engineering but also goes a step further.

3.2 Information Retrieval Techniques

← Previous

Next →

A guest post by

Miqdad Jaffer

Product Lead @ OpenAI | EIR @ Product Faculty