Lecture 2:

Large Language Model, AI Agent, and Vibe Coding

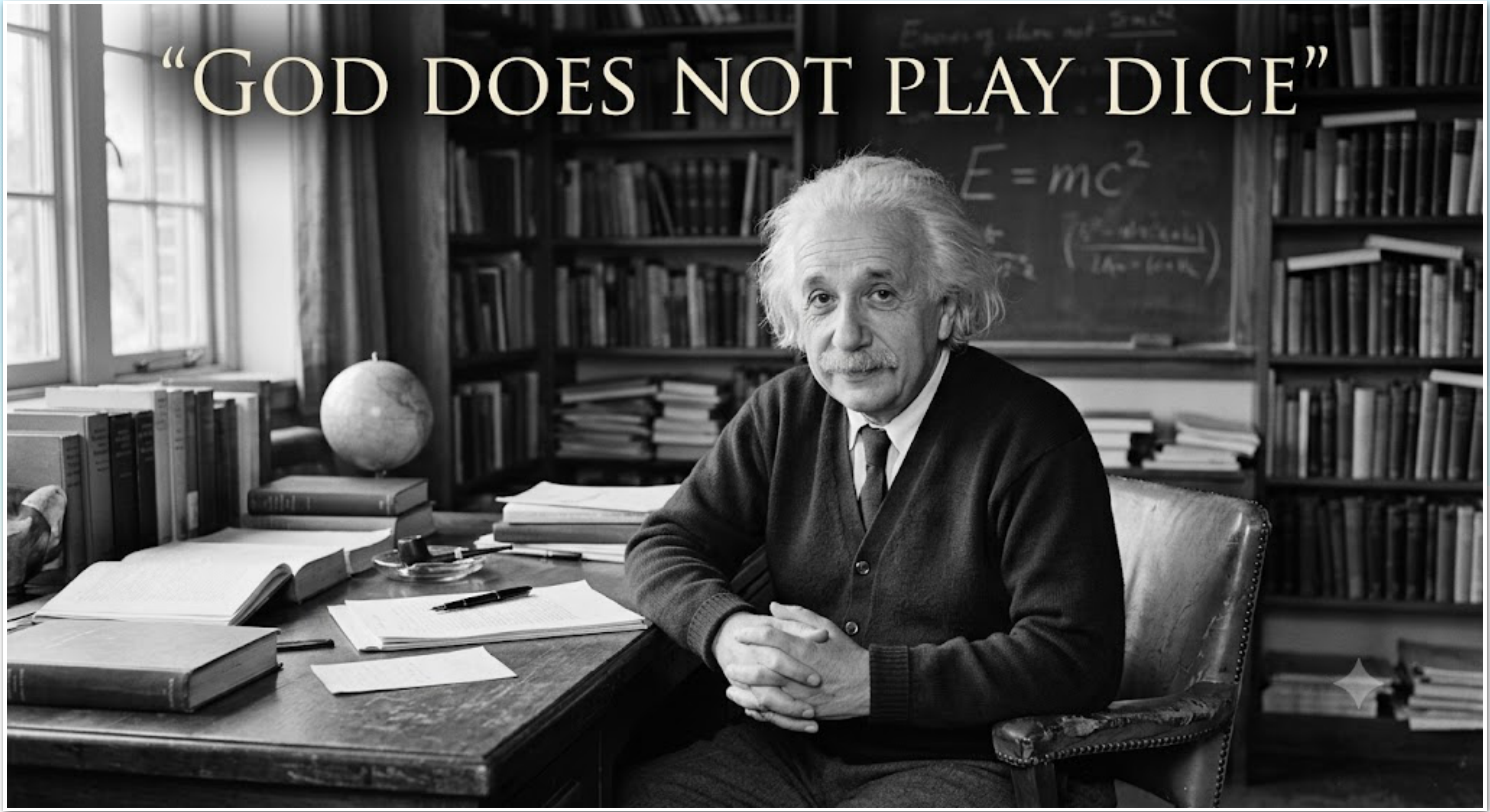
Aobo Li
Halıcıoğlu Data Science Institute
Department of Physics
UC San Diego

Special Thanks: Yue Ma

National Nuclear Physics Summer School, 07/2026

Natural Language As Data

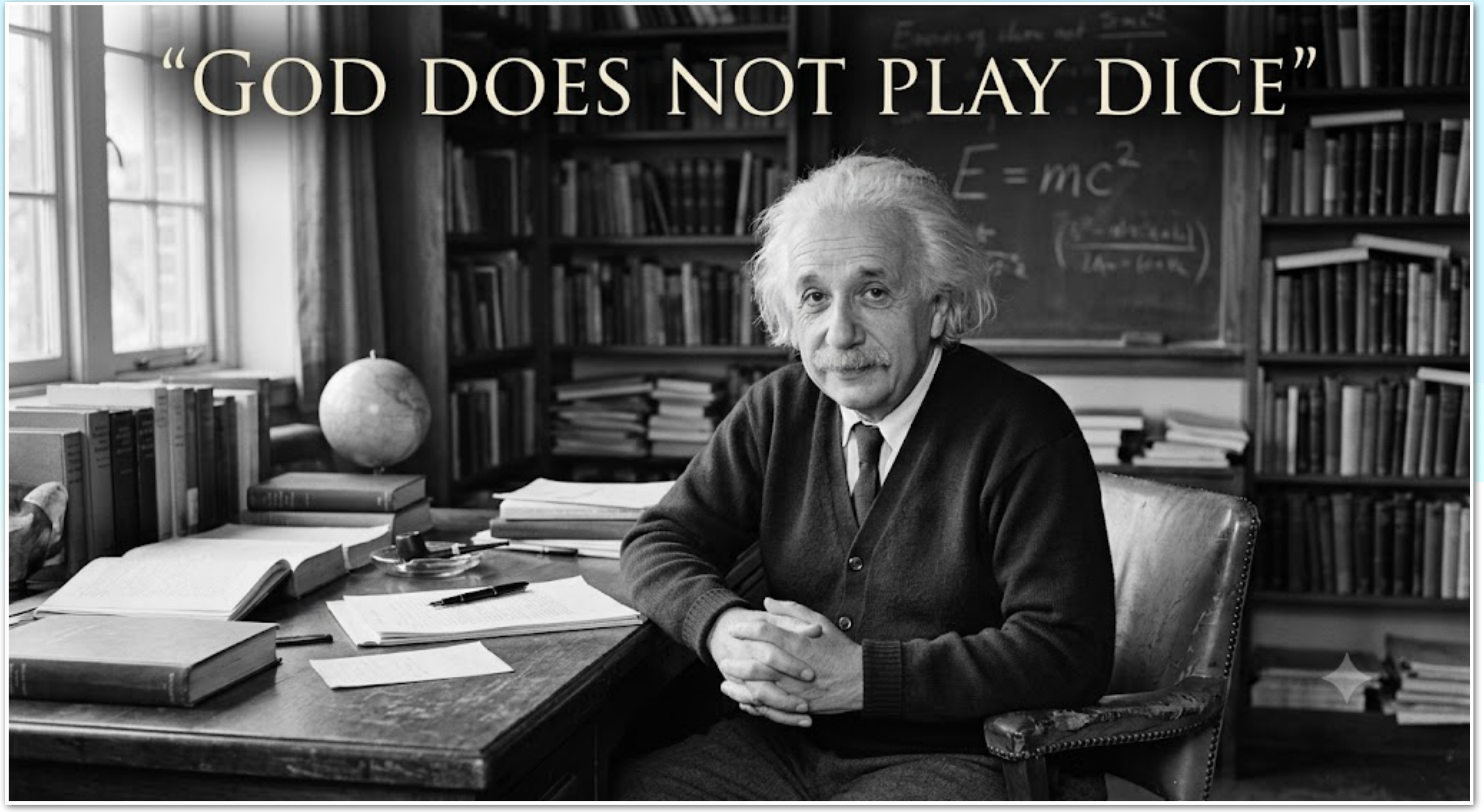
One-Hot Encoding & Token



- Build a word dictionary:
  - ": [1 0 0 0 0 0]
  - God: [0 1 0 0 0 0]
  - Does: [0 0 1 0 0 0]
  - Not: [0 0 0 1 0 0]
  - Play: [0 0 0 0 1 0]
  - Dice: [0 0 0 0 0 1]
- Concatenate each word into sentence and phrase
  - God does not play dice: [[1 0 0 0 0 0], [0 1 0 0 0 0], [0 0 1 0 0 0], [0 0 0 1 0 0], [0 0 0 0 1 0], [0 0 0 0 0 1], [1 0 0 0 0 0]]
  - Size: (7, 6)
- **TOKEN:** Each token represents one of the 6-dimensional vectors within the one-hot encoding
  - The above sentence has 7 tokens (5 words + 2 quotation marks)

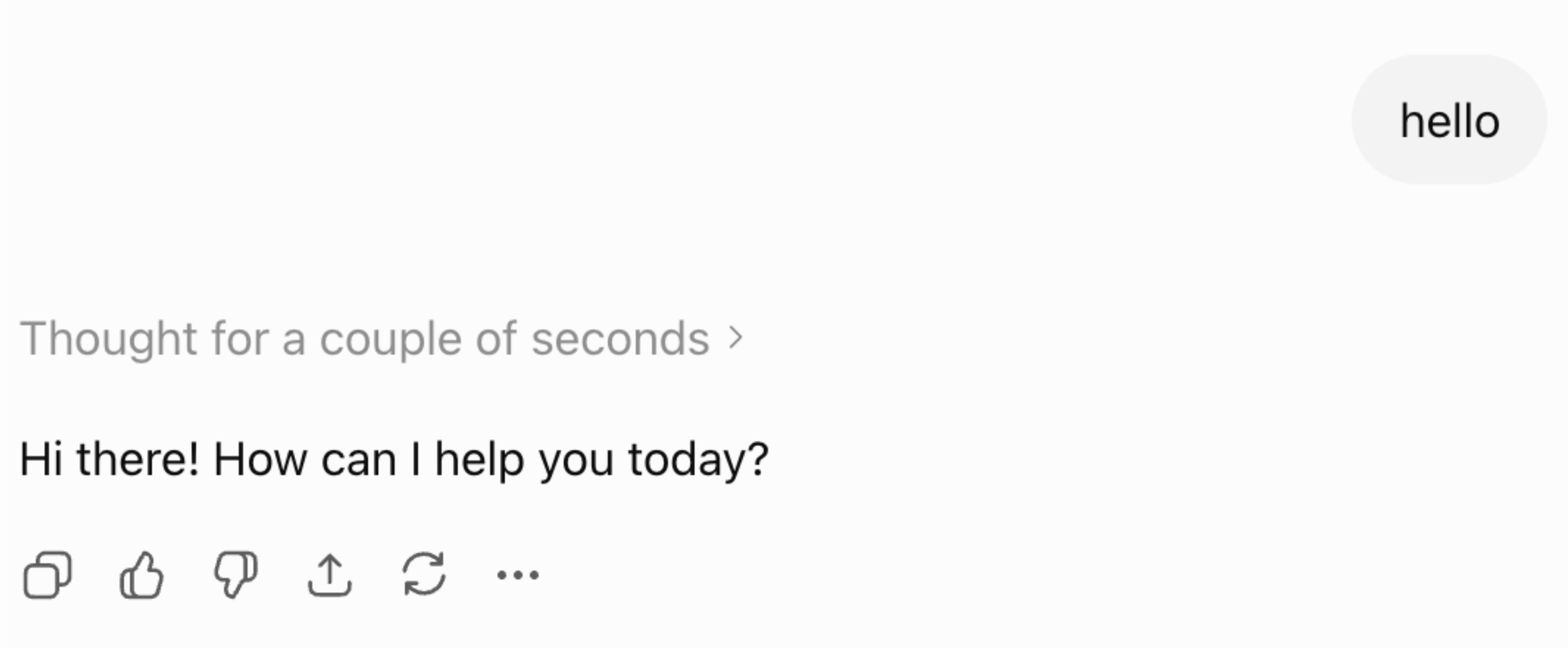
Natural Language As Data

Predicting the Next Token



- Build a word dictionary:
    - ": [1 0 0 0 0 0]
    - God: [0 1 0 0 0 0]
    - Does: [0 0 1 0 0 0]
    - Not: [0 0 0 1 0 0]
    - Play: [0 0 0 0 1 0]
    - Dice: [0 0 0 0 0 1]
- Remove one word from the sentence:
    - God does not ___ dice: [[1 0 0 0 0 0], [0 1 0 0 0 0], [0 0 1 0 0 0], [0 0 0 1 0 0], [? ? ? ? ? ?], [0 0 0 0 0 1], [1 0 0 0 0 0]]
- Language model will generate a probability:
    - Model predict [? ? ? ? ? ?] to be [0.1 0.1 0.1 0.2 0.4 0.1]
    - The most likely word (among the full dictionary) should be "Play"

Introduction: What is LLM



- Much larger dictionaries [English, French, Germany, Japanese, Chinese ...]
- Much longer sentences [Almost all available text on the internet]

Raw LLM (Foundation Model) is just a probability predictor

The | cat | sat | on | the → mat ← predicted

Nothing but a conditional probability

$P(x_t \mid x_1, x_2, \ldots, x_{t-1})$

Probability of next token given all previous tokens

Next word: sampled from the learned probability distribution

What this is NOT		What this IS	
Understanding No grounded semantics	**Reasoning** No explicit logic	**Pattern matching** Over vast corpora	**Compression** Statistical model

Training: learn the probability distribution from data

Analogy: extreme statistical compression

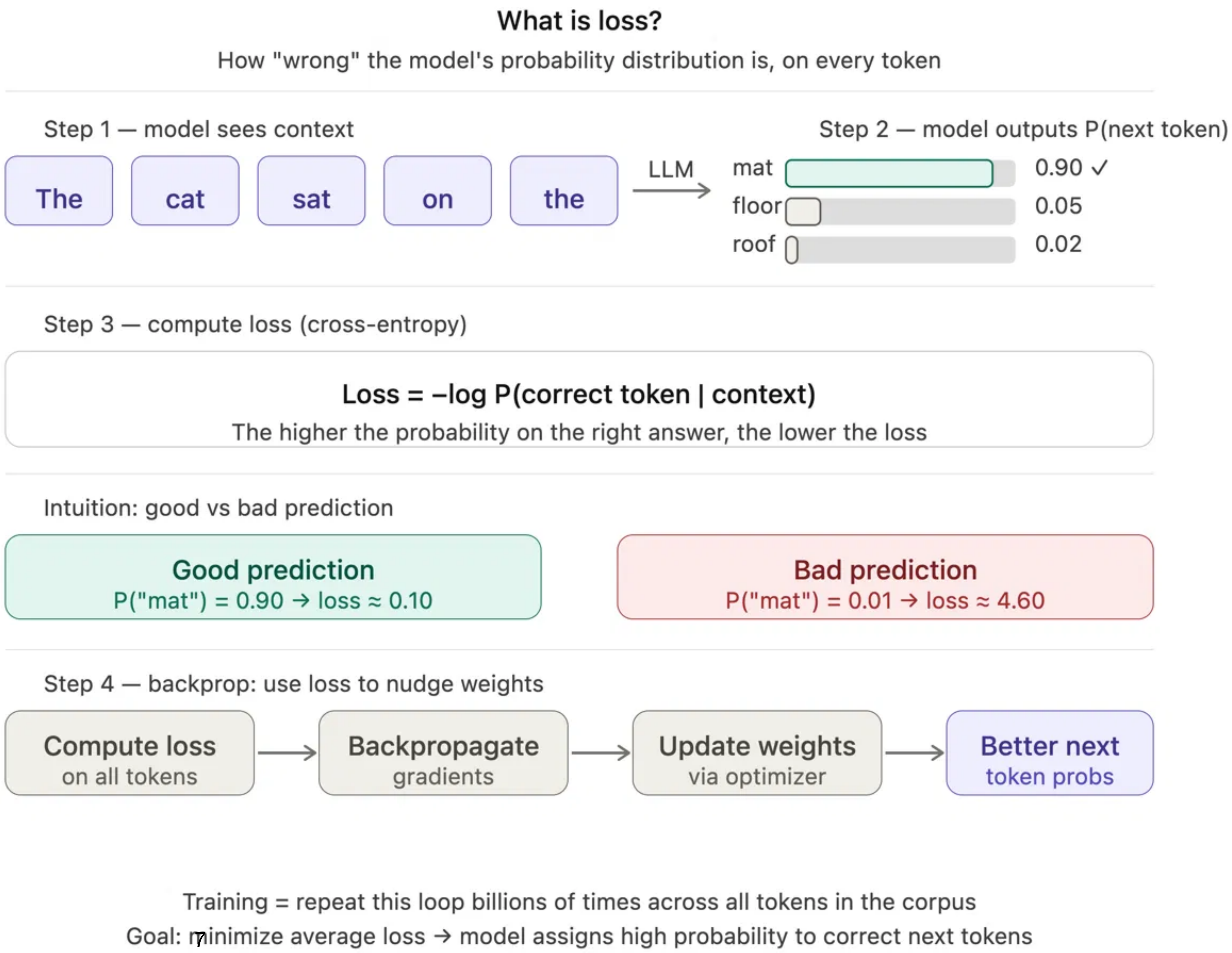
Training data Trillions of tokens from web, books, code, etc. → **Weights** ~Billions of params encode co-occurrence patterns implicitly → **Inference** Decompress on the fly: sample from $\boxed{P(\text{next} \mid \text{context})}$

Raw LLM (Foundation Model) is just a probability predictor

Nothing but a conditional probability

Next word: sampled from the learned probability distribution

Training: learn the probability distribution from data



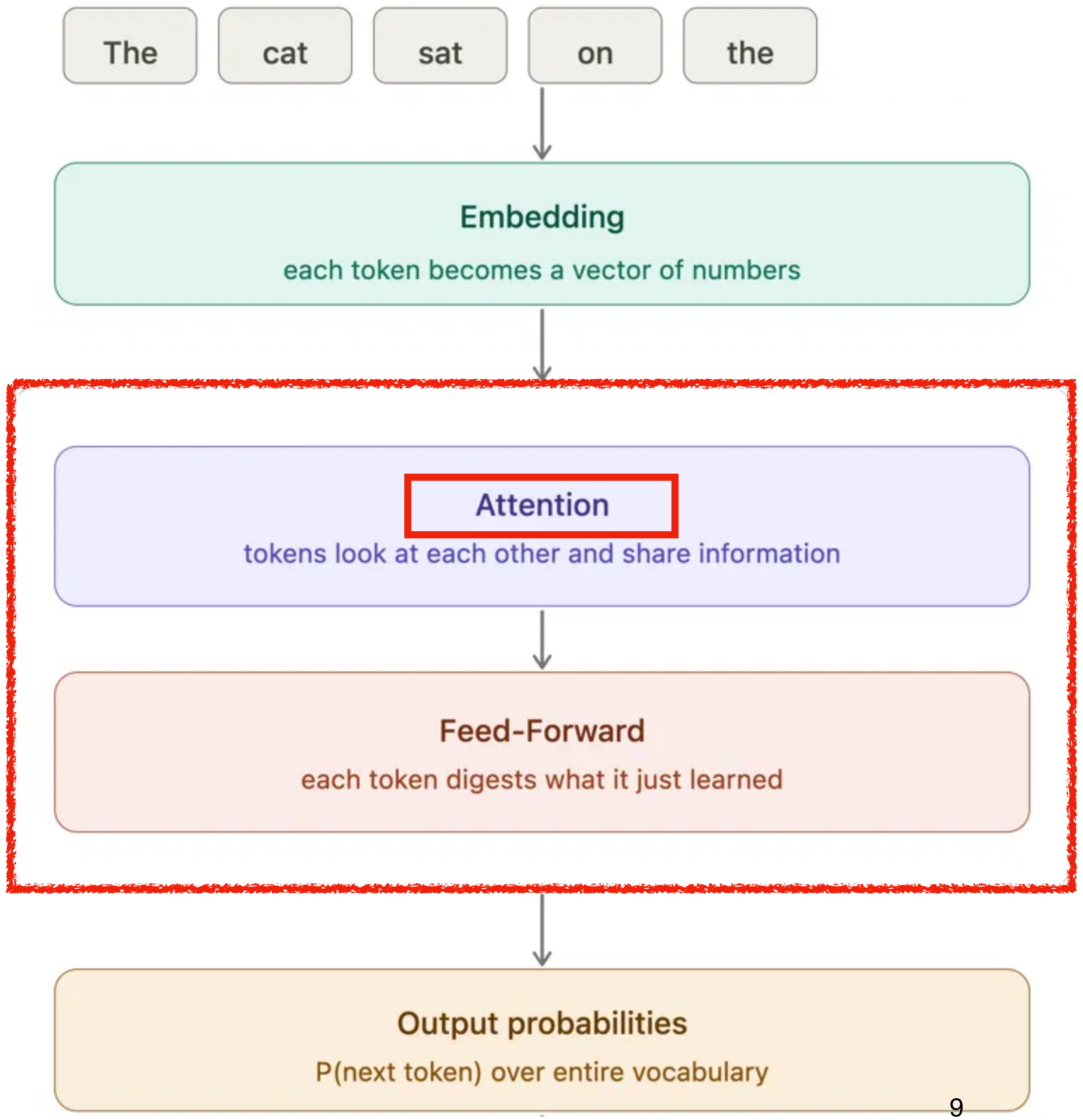
Transformer: Attention is all you need!

Feature Extraction

Learn correlations

Refine

Predict



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.

Transformer: Attention is all you need!

For each token:

- **Q, K, V are vectors**

For a sentence (array):

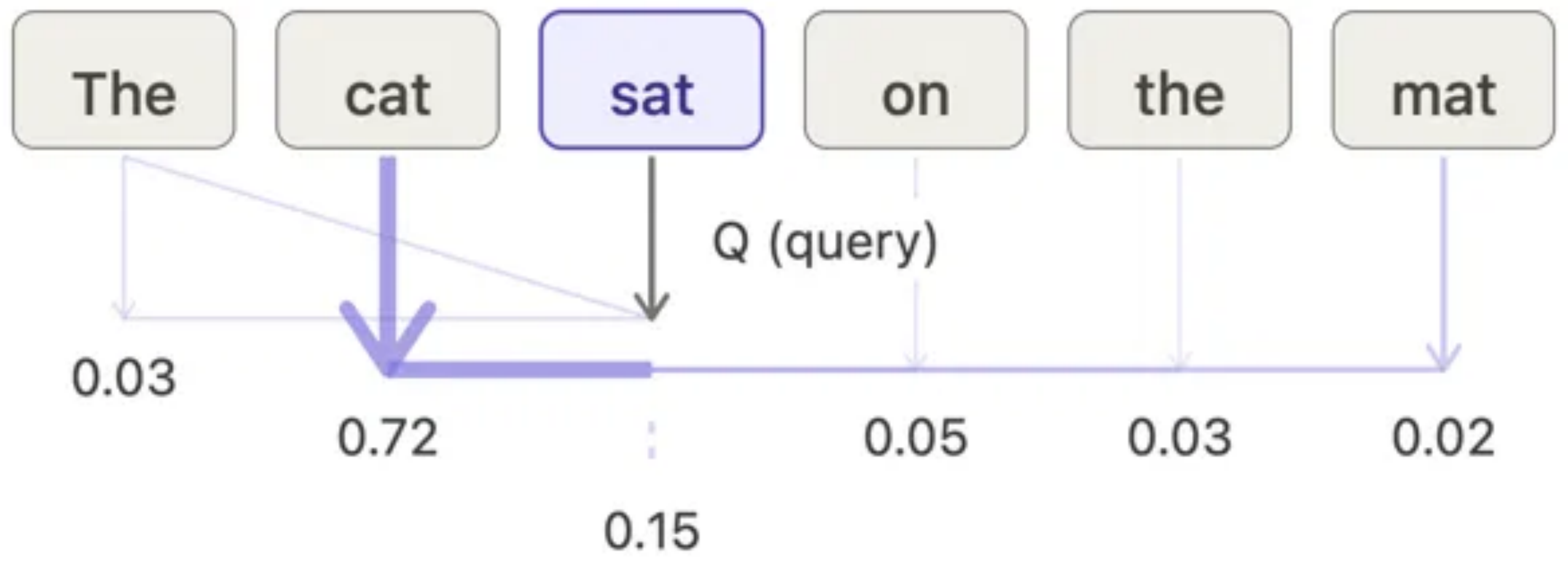
- **Q, K, V are matrix**

Self Attention:

- **Tokens “query” each other**

Single-head attention

"sat" asking: who matters to me?



Line weight = attention score = how much to borrow from that token's V

Q · K · V — the three roles

Q query "what I'm looking for"

K key "what I advertise"

V value "what I actually send"

$$\mathrm{softmax}(Q \cdot K^{T} / \sqrt{d_k}) \cdot V$$

$Q \cdot K^T$ → compatibility score between query and every key
$\div \sqrt{d_k}$ → scale down to stabilise gradients
softmax → turn scores into weights that sum to 1
$\cdot V$ → weighted mix of value vectors

Multi-head attention

run H attention heads in parallel

Head 1 — syntax "cat" is subject of "sat"

Head 2 — coreference "the" links to "mat"

Head 3 — position nearby tokens matter more

Head 4 — semantics "sat" relates to "on the mat"

Head 5 ... Head H

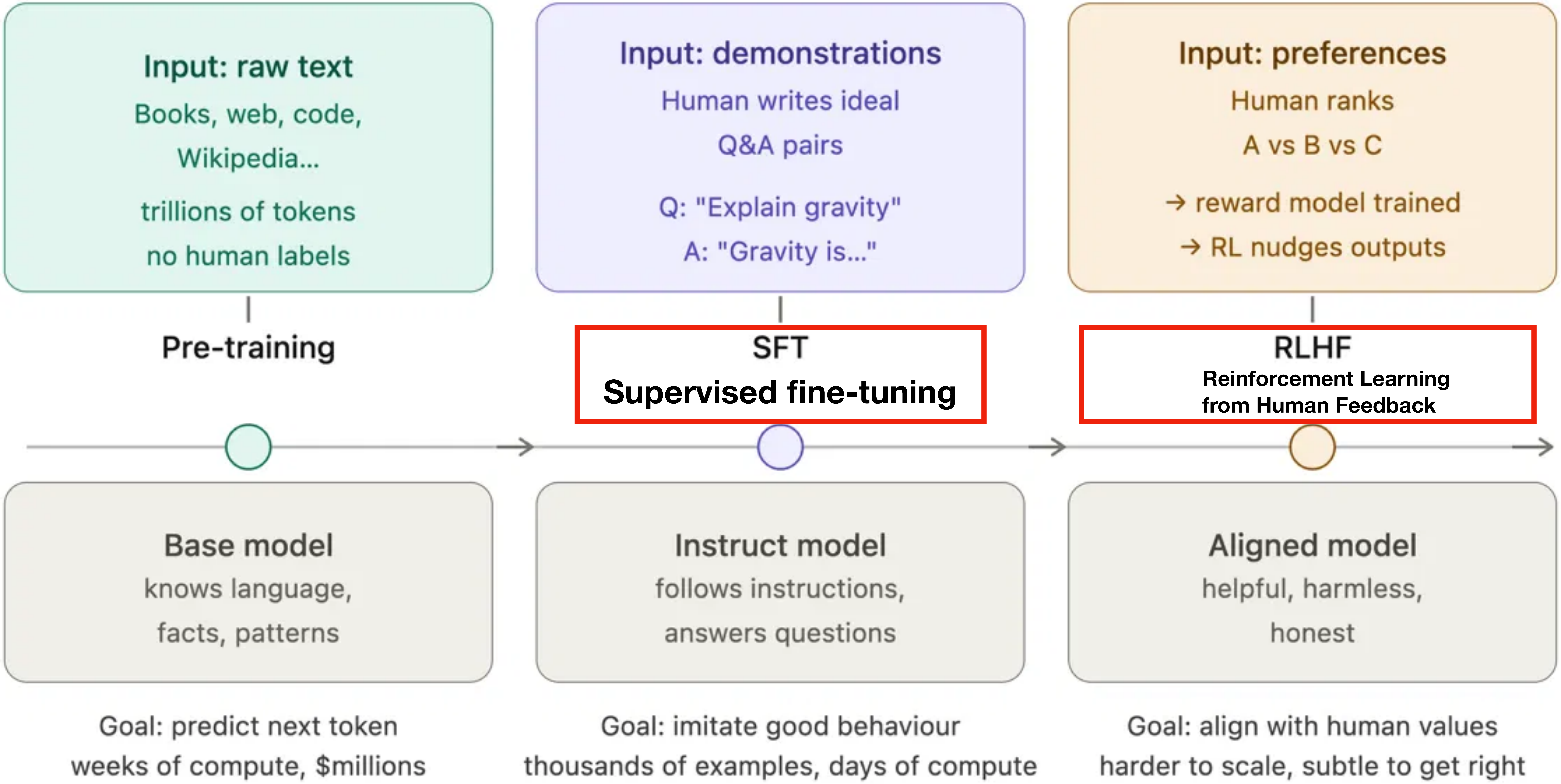
concat + linear

Combined output richer than any single head alone

Each head learns a different notion of "relevance"
not hand-designed — emergent from training

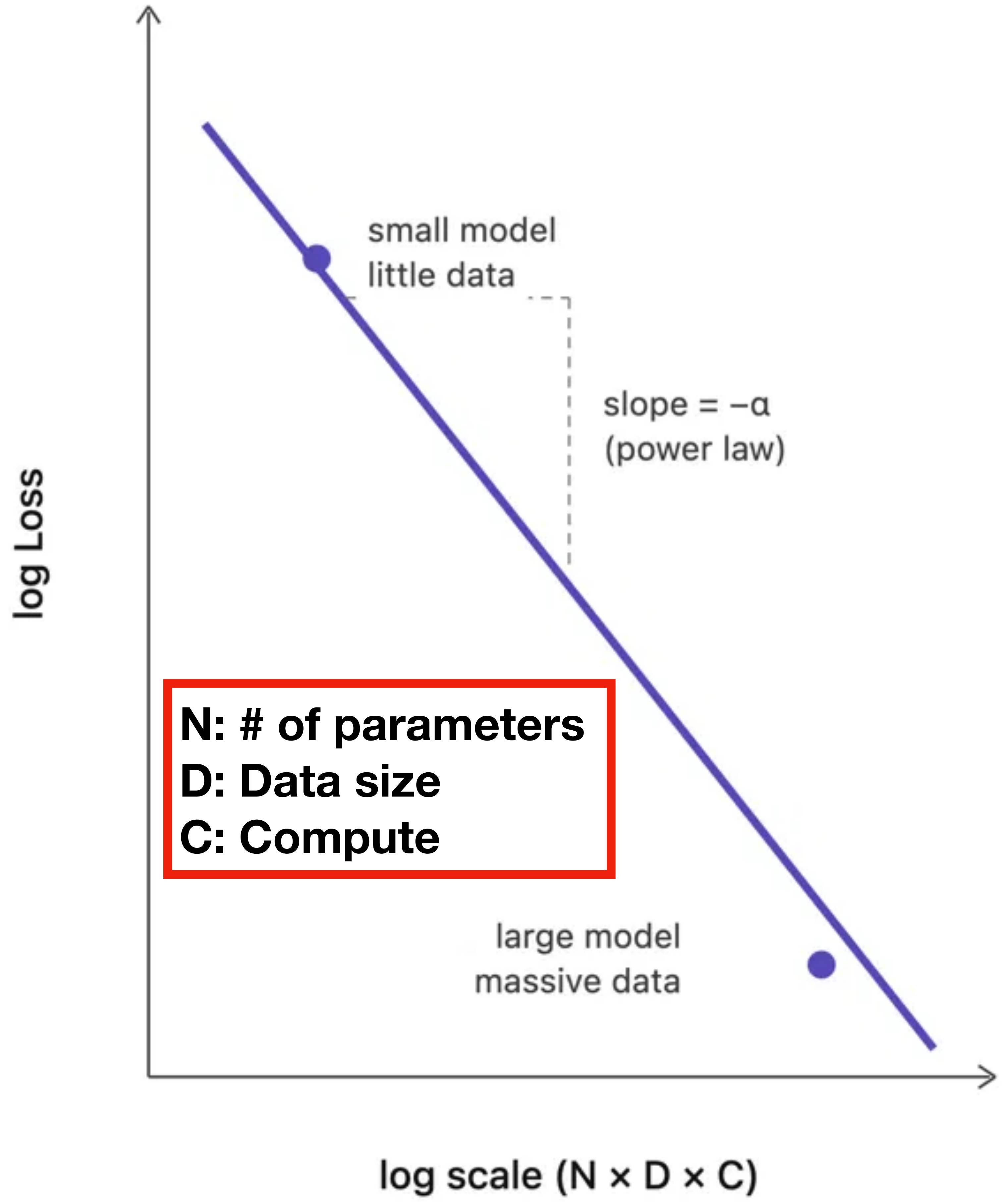
Tokenization → **Large Language Model** → Chat Model → AI Agent → Vibe Coding

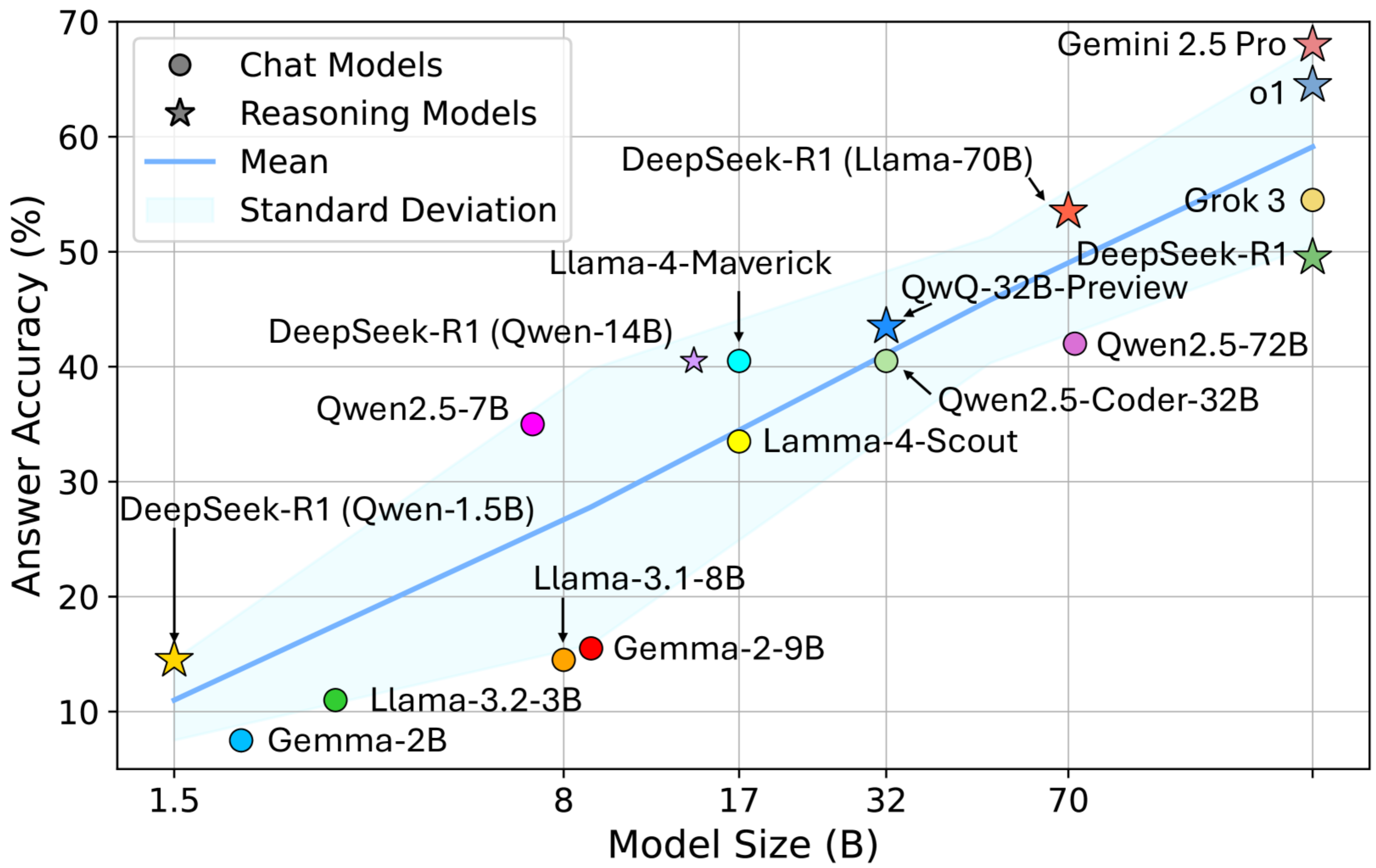
Pre-training -> Post training



The Scaling Law

Loss decreases smoothly...





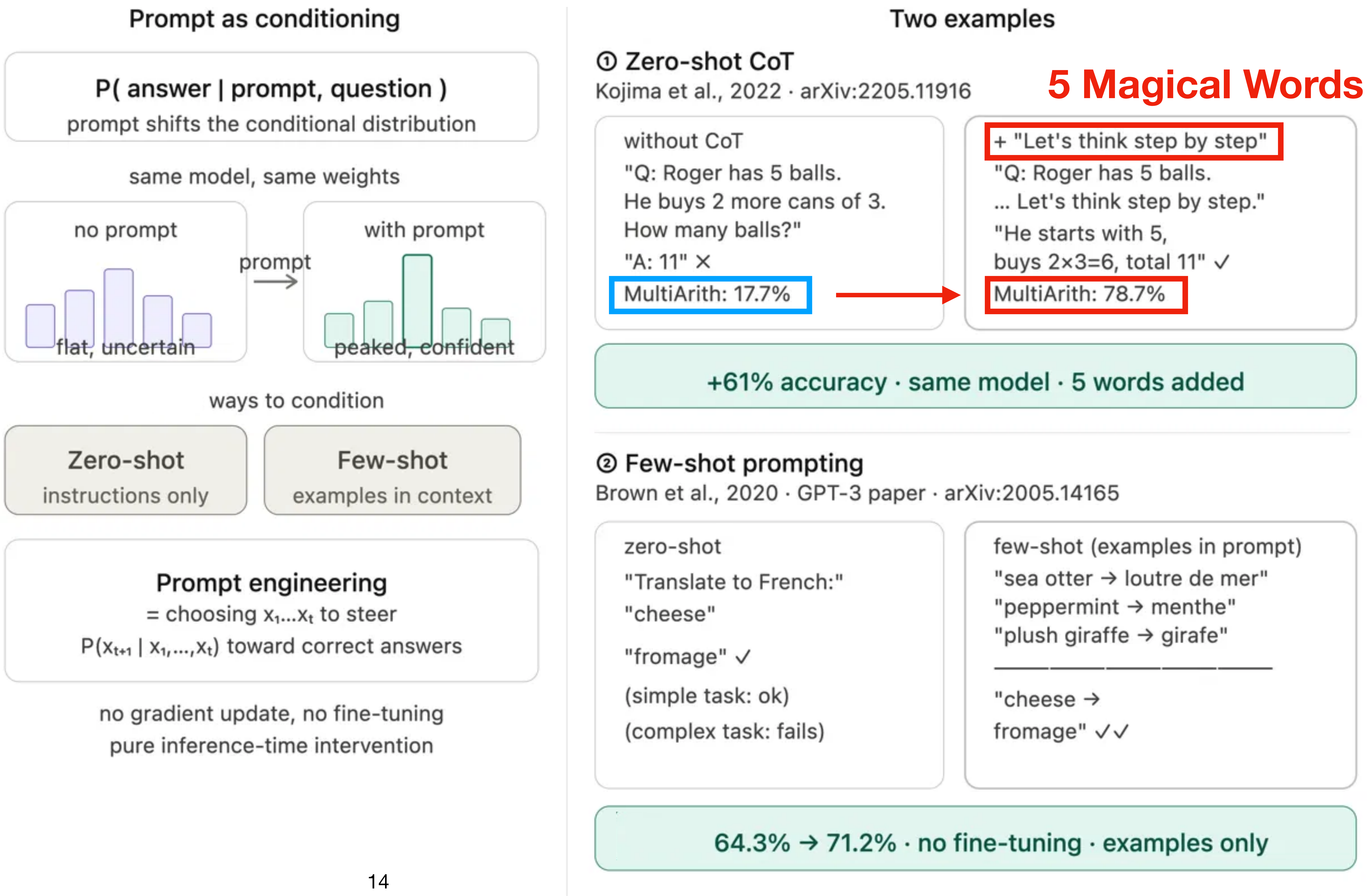
Previous question: what we are actually chatting with?

Raw LLM +

- **Prompt Engineering**
- **Knowledge injection**
- **Test-time compute (Reasoning, etc)**
- **Tool calling**
- **Memory control**
- **And more…**

Prompt Engineering

- **Better condition** provided to the probability predictor
- **Few-shot**: provide some examples
- **Zero-shot**: just some high-level principle!



Prompt Engineering

prompts.chat

The world's largest open-source prompt library for AI

Works with ChatGPT, Claude, Gemini, Llama, Mistral, and more

formerly known as Awesome ChatGPT Prompts

Website prompts.chat · awesome · Hugging Face · Ask DeepWiki

🌐 Browse Prompts • 📖 Read the Book • 📄 View on GitHub • 🚀 Self-Host

🏆 Featured in Forbes · 🎓 Referenced by Harvard, Columbia · 📄 40+ academic citations · ❤️ Most liked dataset on Hugging Face

⭐ 143k+ GitHub stars · 🥇 GitHub Staff Pick · 🚀 First prompt library (Dec 2022)

Loved by AI pioneers:

Greg Brockman (OpenAI Co-Founder) · **Wojciech Zaremba** (OpenAI Co-Founder) · **Clement Delangue** (Hugging Face CEO) · **Thomas Dohmke** (Former GitHub CEO)

https://github.com/f/prompts.chat

Test-time compute

can you check the whether today for me?

Thought for 8s >

Currently 70° · Cloudy

Coronado, CA, United States

Today 70° 64° After a cloudy start, sun returns

Gemini

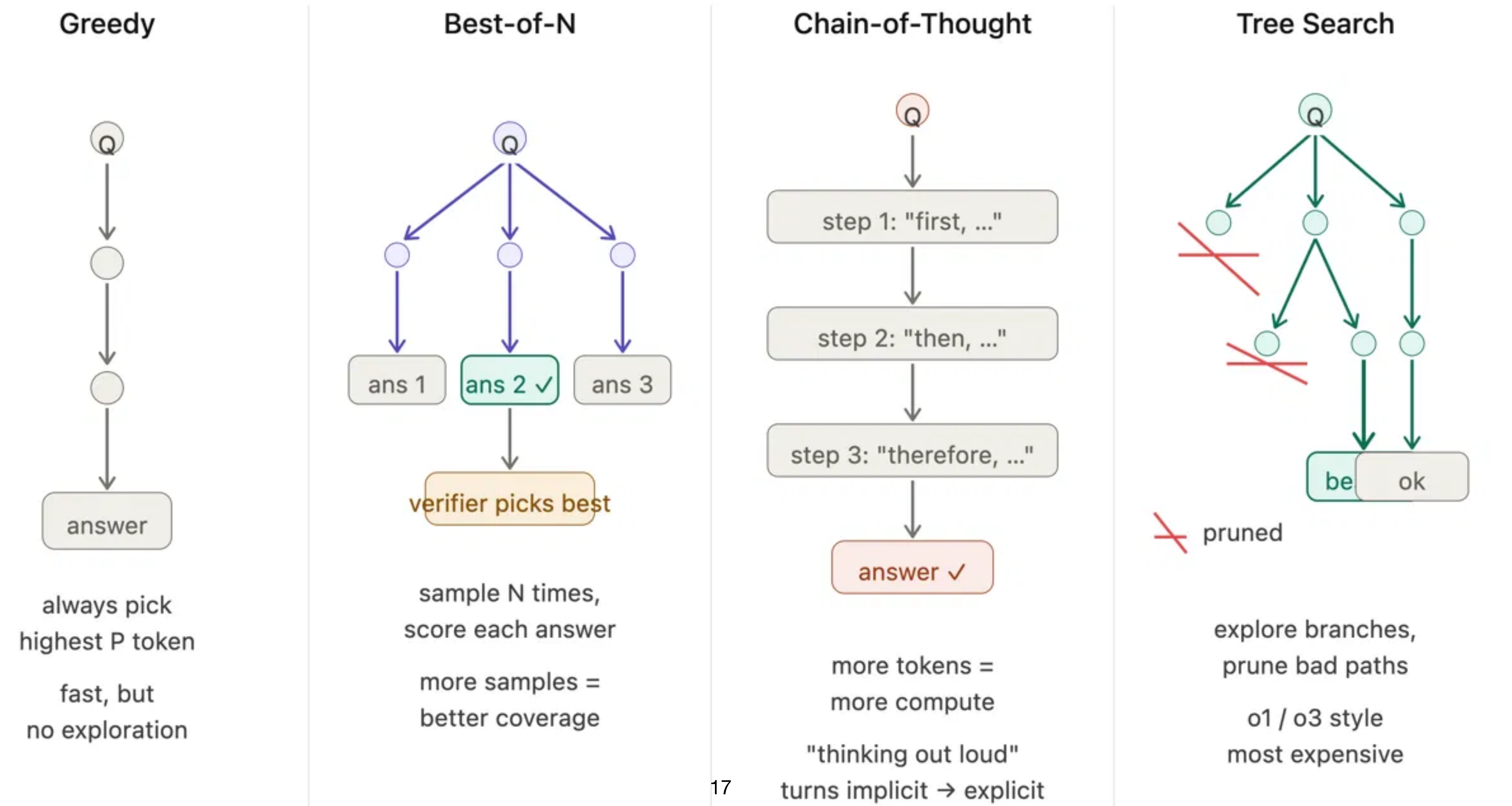
Thinking

- Determining user location for weather query
  The user likely meant "weather" instead of "whether." They're in Coronado, California, based on available info, but I should confirm their exact location to ensure accuracy. I'll check this before retrieving the weather details to provide an accurate response.
- Generating weather forecast response with UI
  I need to call for the user's location first, likely "Coronado, CA," then retrieve the weather forecast for that area. I'll display today's weather using a forecast widget.
- Confirming location and weather information
  Today
- Getting weather for Coronado, CA
  Now, I'll fetch the weather for Coronado, California. I'll set the location to Coronado, CA and use the default settings for start and duration. With that, I can provide today's weather forecast. Let's get those details for the user!

Thought for 8s
Done

Test-time compute

Raw LLM = Probability Distribution, Test-time compute = smarter sampling



Tool Calling

which tools did you use? i'm not sure whether to trust your answer!

Thought for 10s >

Which tools I used

I rely on a couple of built-in utilities:

1. **Location helper** – a private tool that just tells me your coarse location (e.g., Coronado, California) and the current date/time.
2. **Weather lookup** – a tool that queries a weather provider for the latest observations and 7-day forecast for that location.

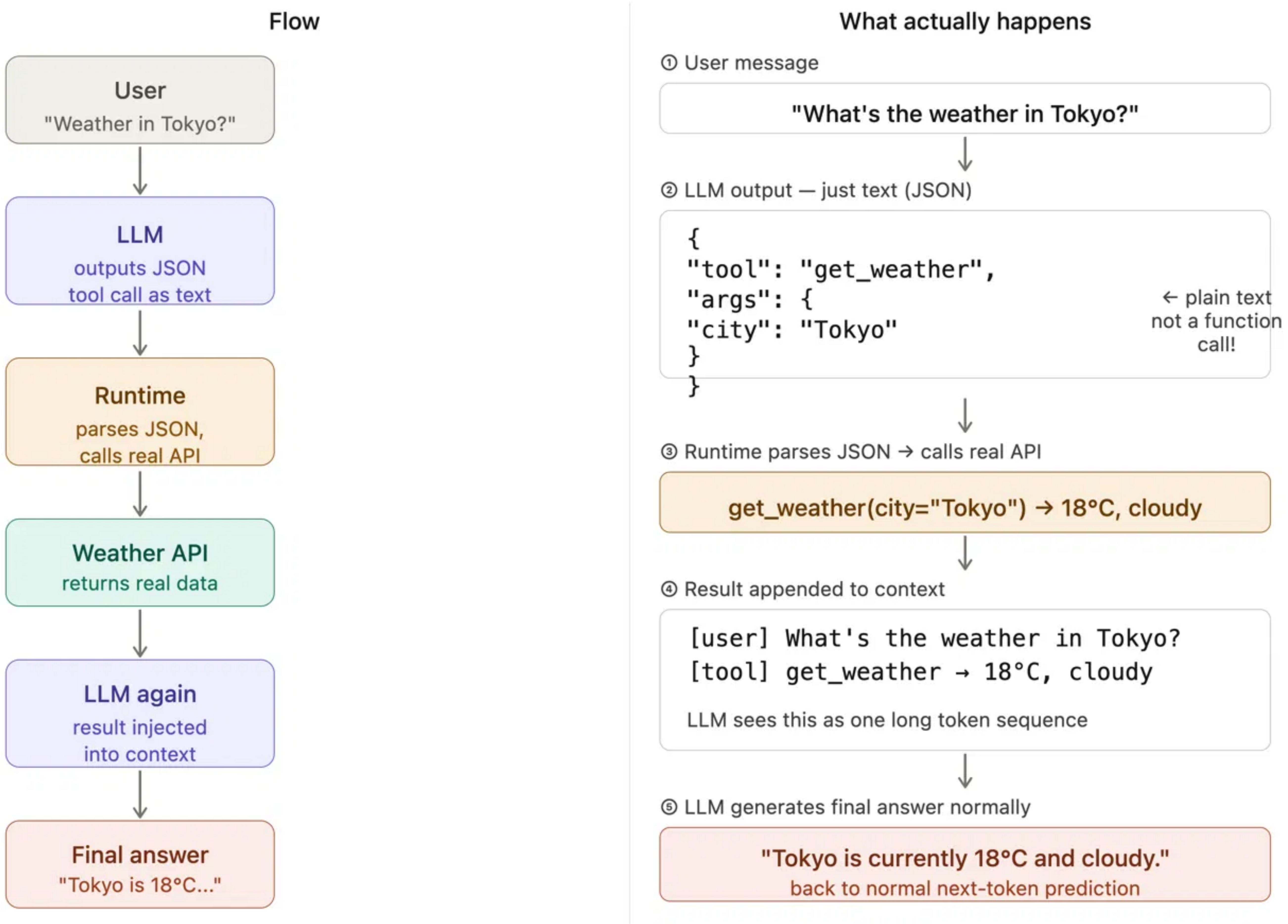
Gemini

So smart!

But how did it happen?

I thought LLM can only produce some texts!

Tool Calling



Tool calling =

Pre-trained format (message -> JSON)

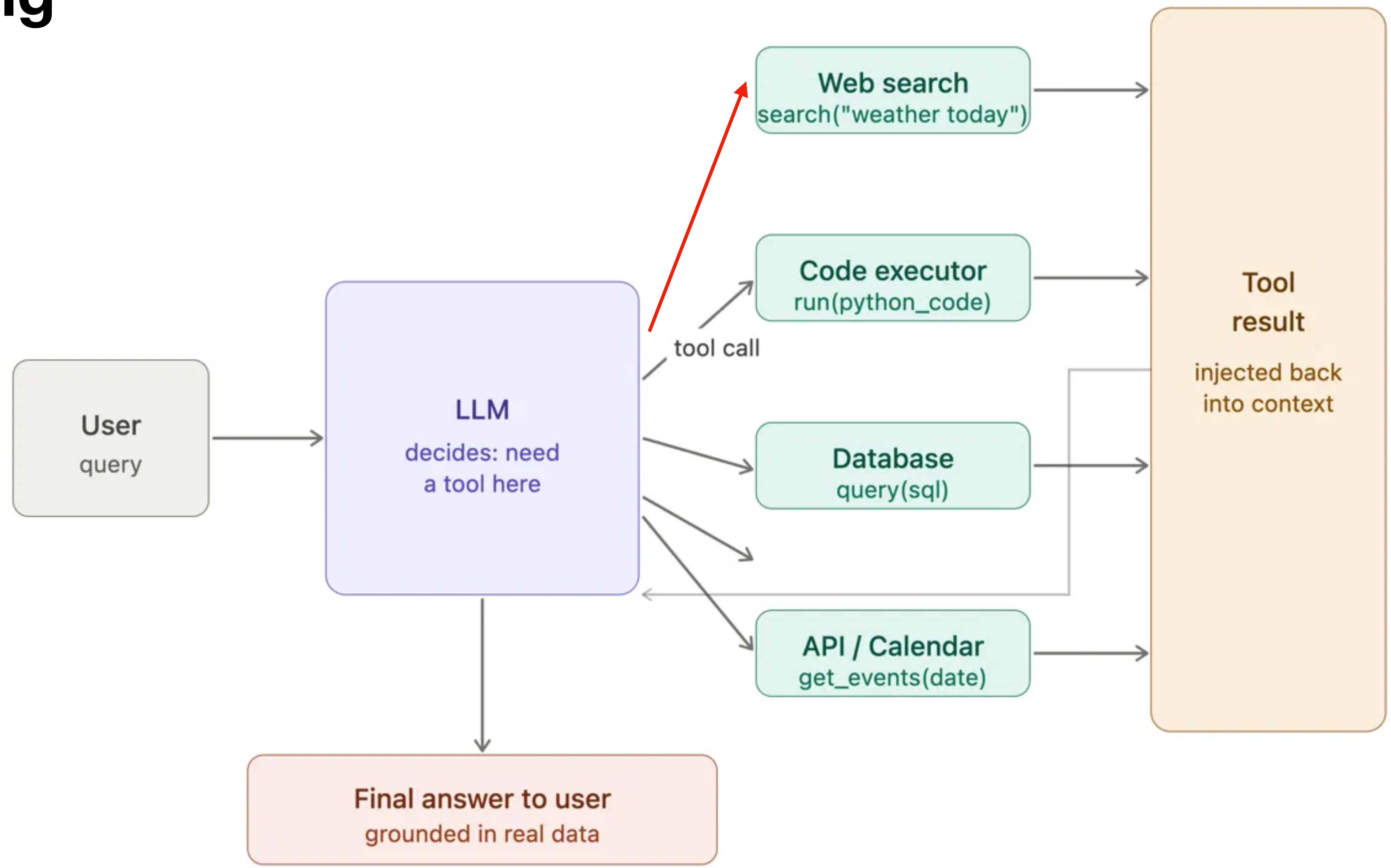
+

API (JSON -> real function call)

+

Context aggregation (result -> answer)

Tool Calling



LLM generates a structured "tool call" as text → runtime executes it → result appended to context → LLM continues

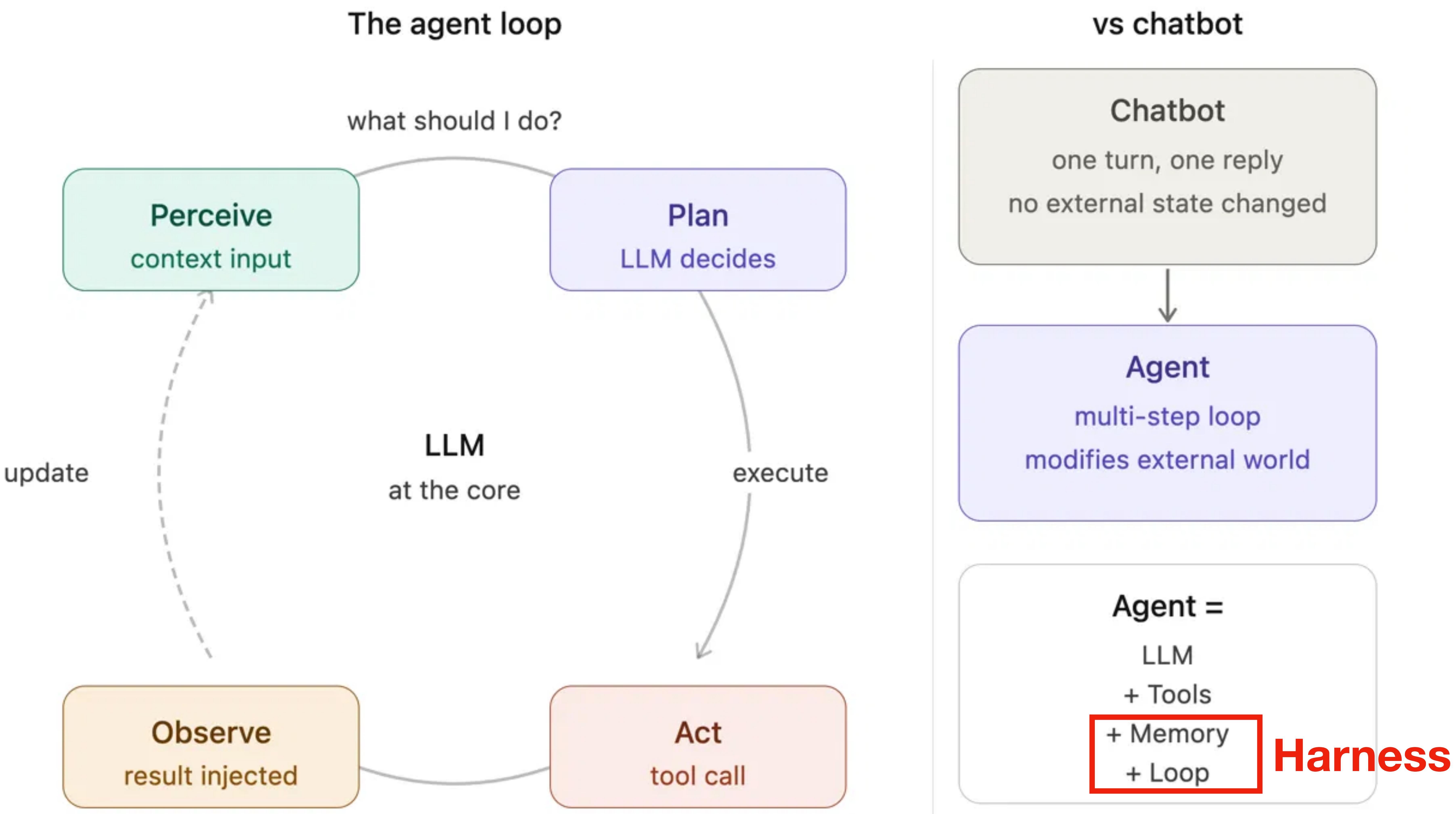
Memory Control

Partly solved by larger context window size (>=1M token)

Definition of LLM Agent

Beyond the chat model:

- **Multi-step loop**
- **Long-term memory**



Memory of LLM Agents

Beyond the chat model:

- **Multi-step loop: tool state itself acts as a long-term memory**
  - **grep instead of RAG**
- **Long-term memory: transferrable between tasks**

Chat model memory
(already covered)

In-context
current conversation window
resets to zero every session

External memory (RAG)
vector DB, persistent memory
retrieve relevant chunks on demand

Summarization / compaction
compress history to fit context window

These exist in both chat models and agents.
Agents inherit all of them.

Agent memory
(new in agents)

Tool state
files, databases, code repos
agent writes → persists across loops

e.g. coding agent:
codebase = the most important memory
every edit persists, next loop reads it

Episodic memory
"how did I solve this last time?"
past task logs → reuse strategies
used in some agent frameworks

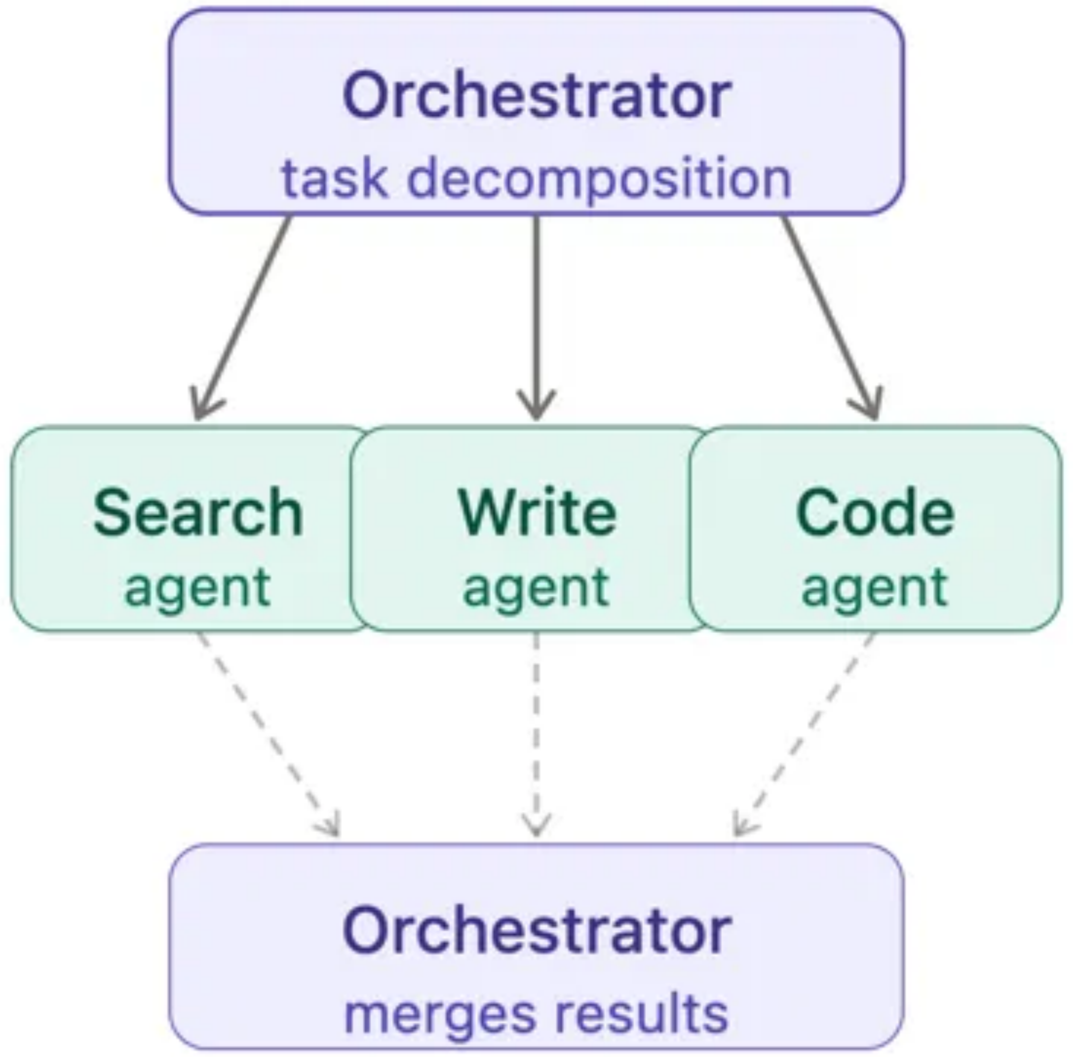
The key difference

Chat model: memory is passive
(you remember what was said)

Agent: memory is active
(you change the world, then remember what changed)

Multi-agent Collaboration

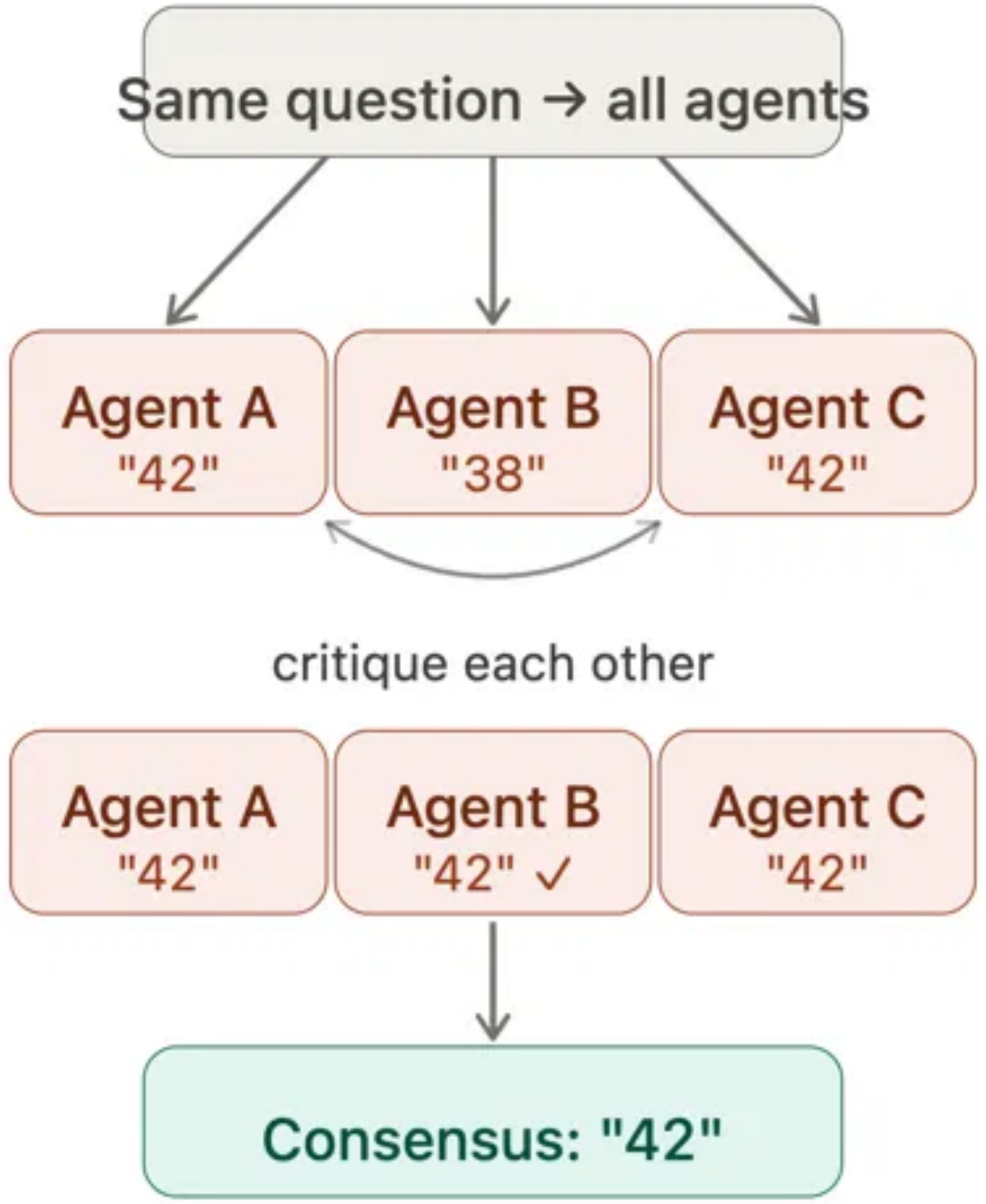
Orchestrator + Sub-agents
AutoGen · arXiv:2308.08155



Why this helps

each sub-agent has focused context
tasks run in parallel
failure isolated per sub-agent

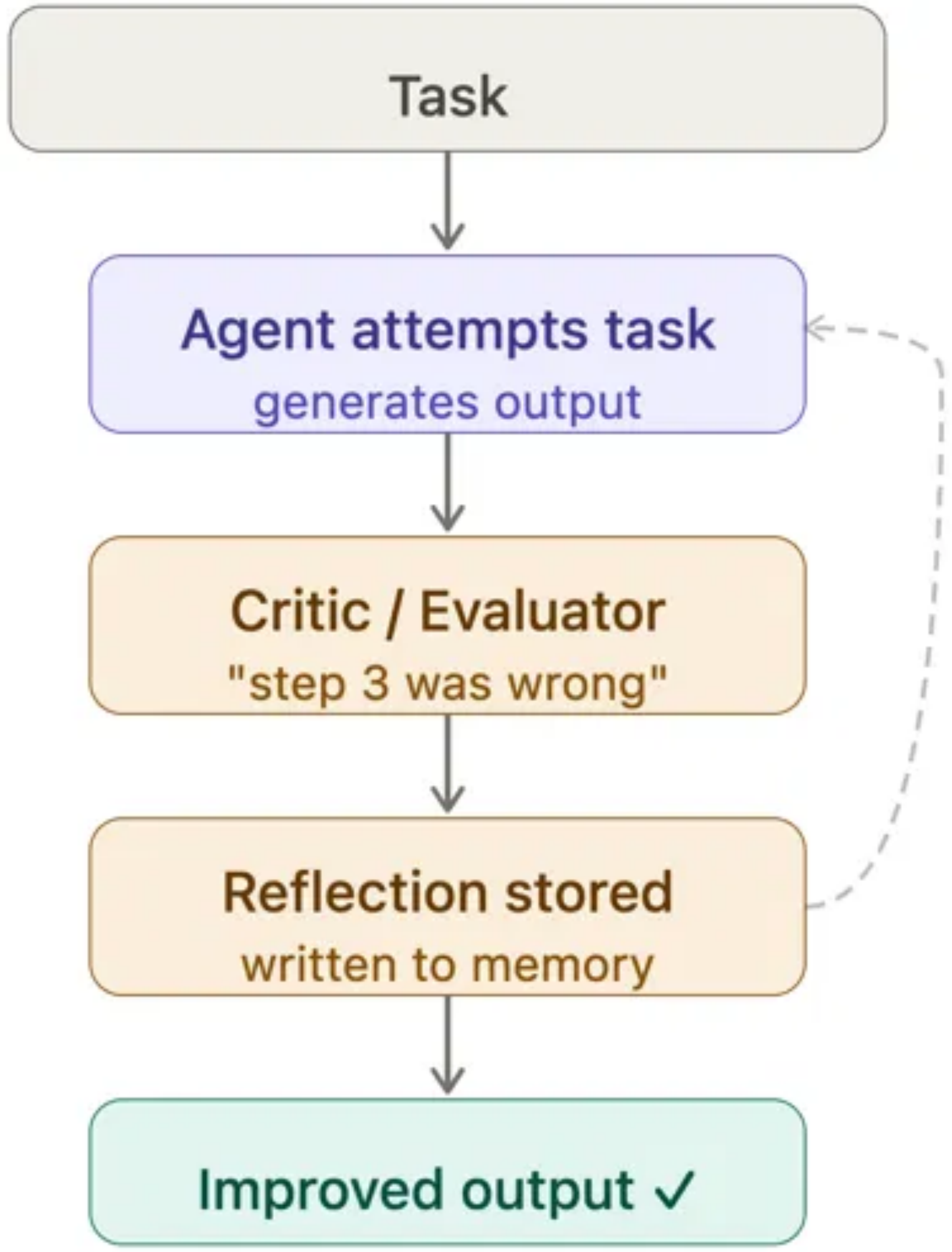
Agent Debate
Du et al. · arXiv:2305.14325



Why this helps

reduces hallucination via peer review
math+reasoning +~10% vs single agent

Reflexion
Shinn et al. · arXiv:2303.11366



Why this helps

no retraining needed
agent learns from mistakes in-context

General LLM Agents: no coding at all, just chat!

🦞 OpenClaw — Personal AI Assistant

OPENCLAW

EXFOLIATE! EXFOLIATE!

BUILD FAILING | RELEASE V2026.6.9-BETA.1 | DISCORD 20K ONLINE | LICENSE MIT

OpenClaw is a *personal AI assistant* you run on your own devices. It answers you on the channels you already use. It can speak and listen on macOS/iOS/Android, and can render a live Canvas you control. The Gateway is just the control plane — the product is the assistant.

If you want a personal, single-user assistant that feels local, fast, and always-on, this is it.

Supported channels include: WhatsApp, Telegram, Slack, Discord, Google Chat, Signal, iMessage, IRC, Microsoft Teams, Matrix, Feishu, LINE, Mattermost, Nextcloud Talk, Nostr, Synology Chat, Tlon, Twitch, Zalo, Zalo Personal, WeChat, QQ, WebChat.

HERMES-AGENT

Hermes Agent ☤

Hermes Agent | Hermes Desktop

DOCS HERMES-AGENT.NOUSRESEARCH.COM | DISCORD | LICENSE MIT
BUILT BY NOUS RESEARCH | LANG 中文 | LANG اردو

The self-improving AI agent built by Nous Research. It's the only agent with a built-in learning loop — it creates skills from experience, improves them during use, nudges itself to persist knowledge, searches its own past conversations, and builds a deepening model of who you are across sessions. Run it on a $5 VPS, a GPU cluster, or serverless infrastructure that costs nearly nothing when idle. It's not tied to your laptop — talk to it from Telegram while it works on a cloud VM.

Use any model you want — Nous Portal, OpenRouter (200+ models), NovitaAI (AI-native cloud for Model API, Agent Sandbox, and GPU Cloud), NVIDIA NIM (Nemotron), Xiaomi MiMo, z.ai/GLM, Kimi/Moonshot, MiniMax, Hugging Face, OpenAI, or your own endpoint. Switch with `hermes model` — no code changes, no lock-in.

OpenClaw:

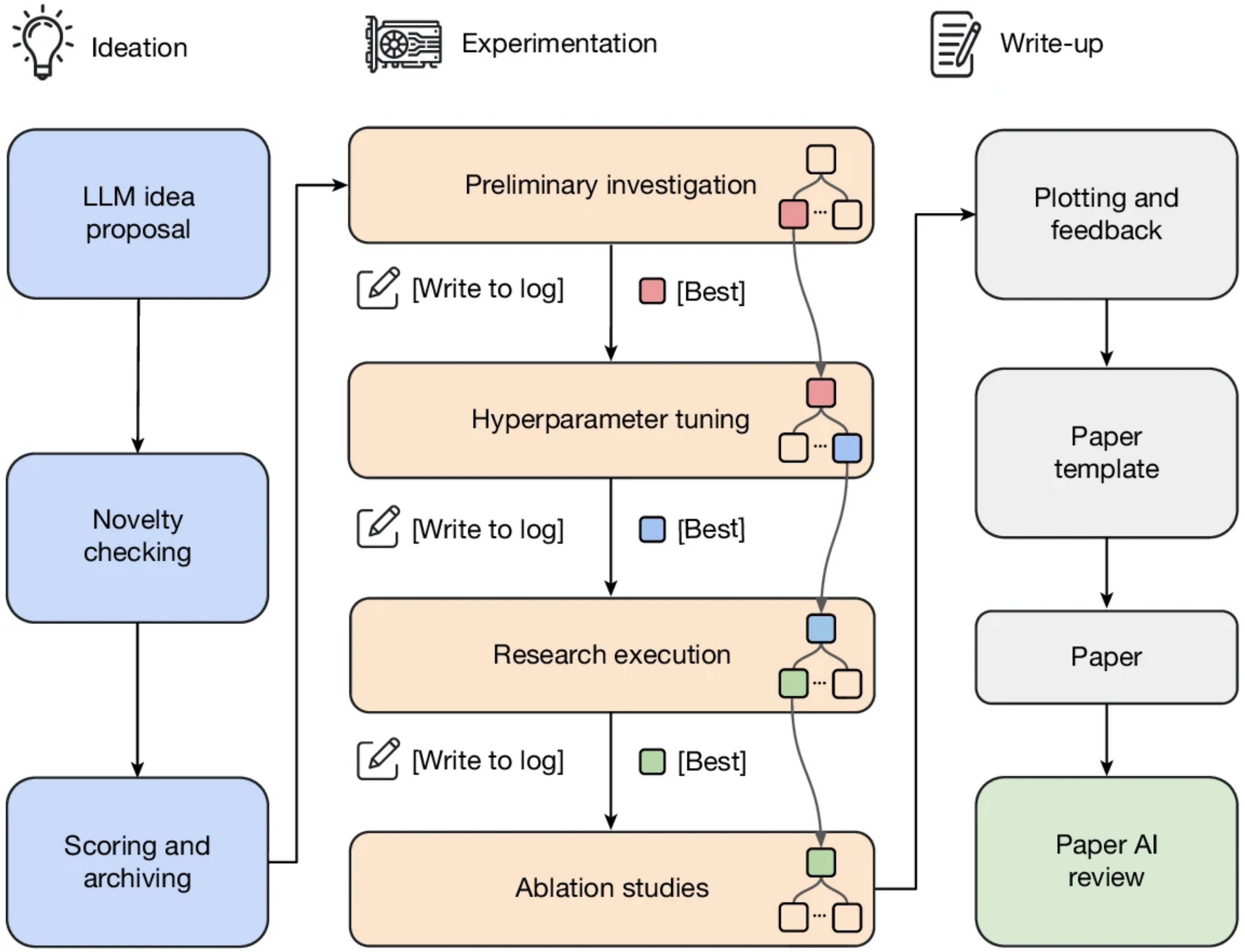
- **Personal AI assistant across 20+ messaging channels (WhatsApp, Telegram, Slack…)**
- **Skill system for extending abilities**

Hermes Agent:

- **Self-improving: creates and refines skills from experience automatically**
- **Cross-session memory with full-text search over past conversations**

Tokenization → Large Language Model → Chat Model → **AI Agent** → Vibe Coding

LLM Agents for Auto-ML



AI Scientist: End-to-end Auto ML Research

Latest achievement: $15 per workshop paper on top AI conferences

nature

Explore content · About the journal · Publish with us

nature › articles › article

Article | Open access | Published: 25 March 2026

Towards end-to-end automation of AI research

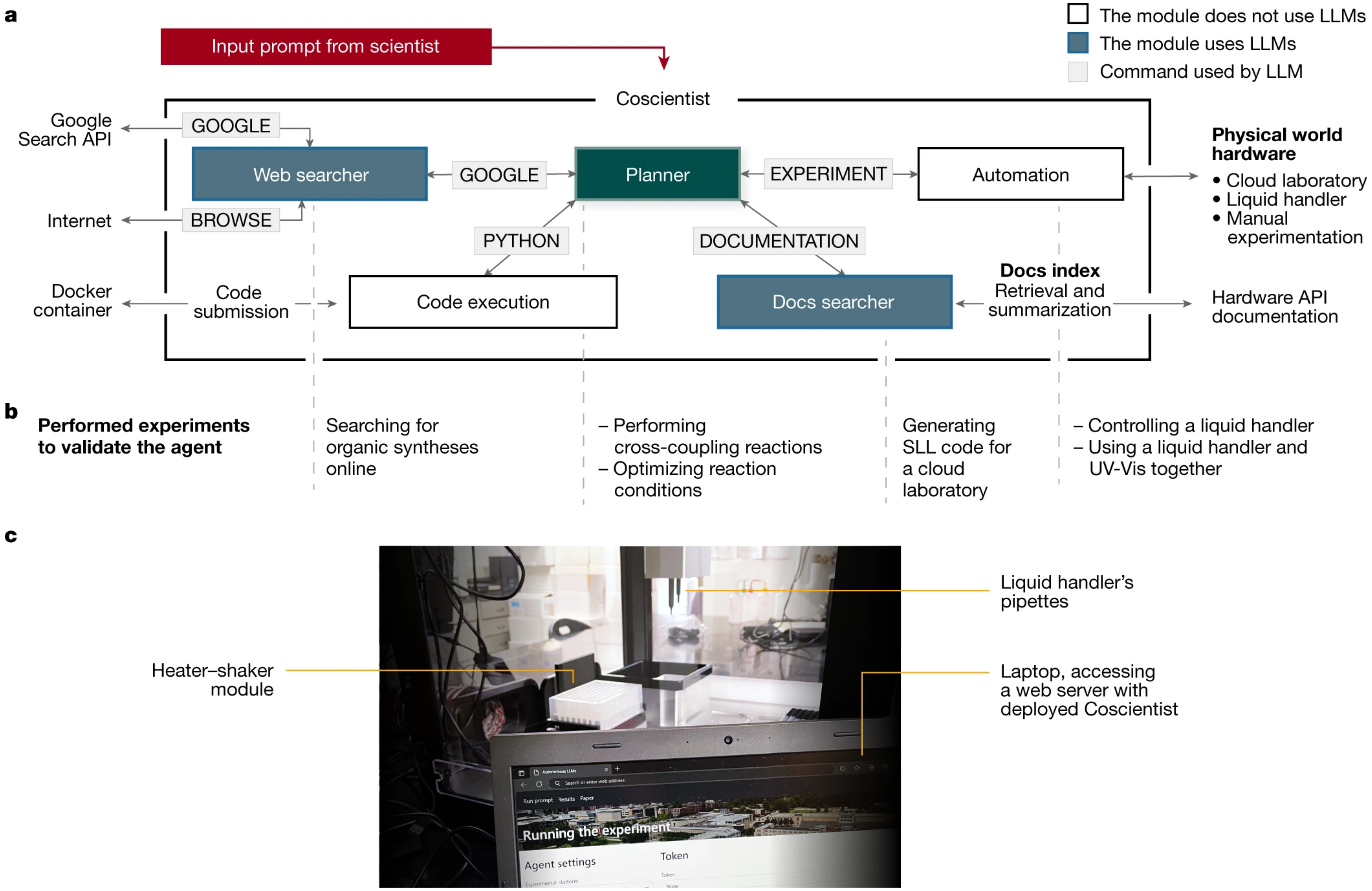
Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha & Jeff Clune

Nature 651, 914–919 (2026) | Cite this article

Abstract

The automation of science is a long-standing ambition in artificial intelligence (AI) research[1,2]. Although the community has made substantial progress in automating individual components of the scientific process, a system that autonomously navigates the entire research life cycle—from conception to publication—has remained out of reach. Here we present a pipeline for automating the entire scientific process end to end. We present The AI Scientist, which creates research ideas, writes code, runs experiments, plots and analyses data, writes the entire scientific manuscript, and performs its own peer review. Its ideas, execution and presentation are of sufficient quality that the manuscript generated by this AI system passed the first round of peer review for a workshop of a top-tier machine learning conference. The workshop had an acceptance rate of 70%. Our system leverages modern foundation models[3,4,5] within a complex agentic system. We evaluate The AI Scientist in two settings: a focused mode using human-provided code templates as an initial scaffold for conducting research on a specific topic and a template-free, open-ended mode that leverages agentic search for wider scientific exploration[6,7]. Both settings produce diverse ideas and automatically test, report on and evaluate them. This achievement demonstrates the growing capacity of AI for making scientific contributions and signifies a potential paradigm shift in how research is conducted. As with any impactful new technology, there could be important risks, including taxing overwhelmed review systems and adding noise to the scientific literature. However, if developed responsibly, such autonomous systems could greatly accelerate scientific discovery.

LLM Agents for Science



Article | Open access | Published: 20 December 2023

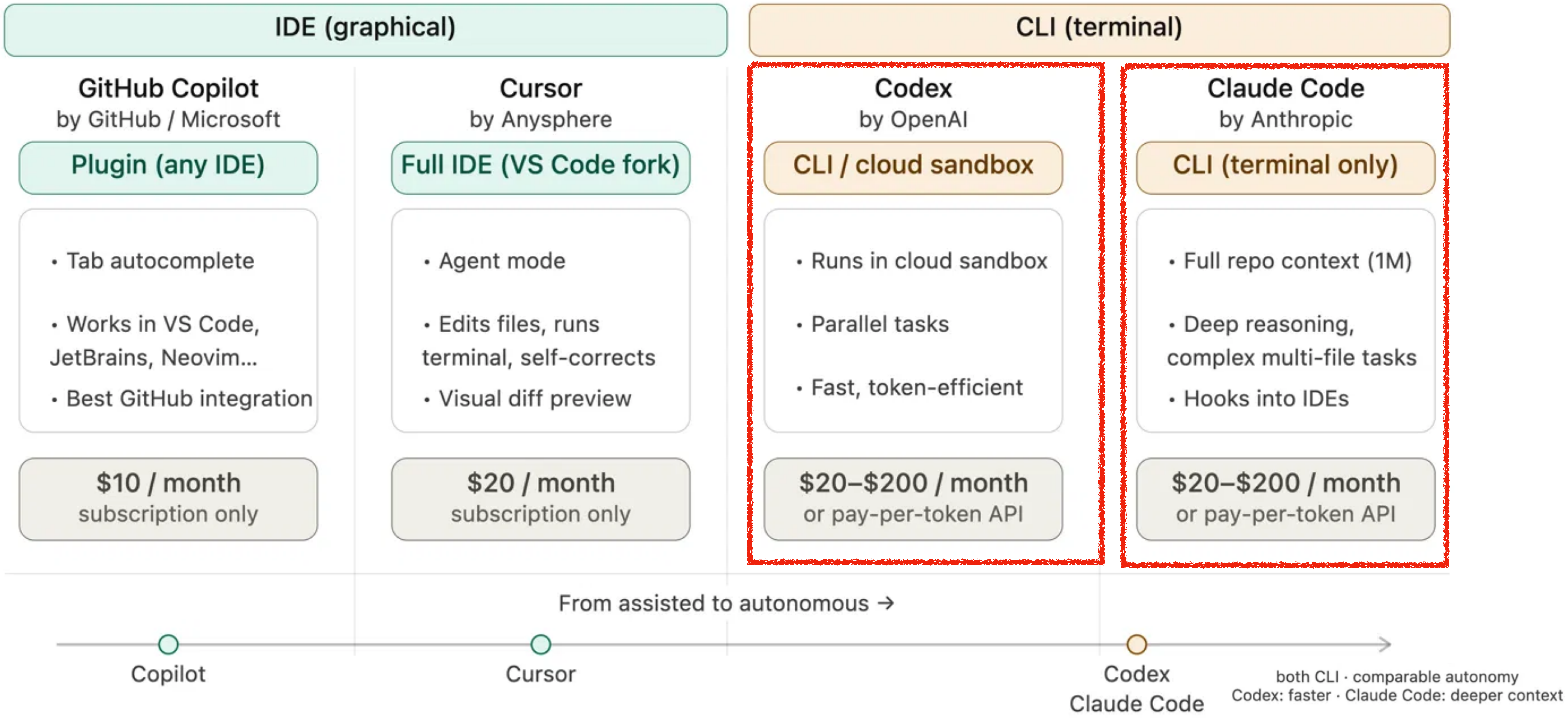
Autonomous chemical research with large language models

Daniil A. Boiko, Robert MacKnight, Ben Kline & Gabe Gomes

Nature **624**, 570–578 (2023) | Cite this article

- **Autonomous agent powered by GPT-4**
- **Reads literature -> Designs experiments -> Analyzes results**
- **Physical execution by a robot**
- **First paper to close the loop between LLM reasoning and wet-lab execution**

Coding Agents



Part 2: A Recipe of Vibe Coding

The Dark Side…

Hallucination and Sycophancy

🤖: Let me just randomly generate something

Hallucination

confident, fluent, completely made up

User
"Give me a paper on attention mechanisms."

LLM
"Sure! Here's a relevant paper:"

Title: "Hierarchical Attention Networks for NLP Tasks"

Authors: Zhang et al., 2021

Journal: NeurIPS, pp. 1182–1194

DOI: 10.1145/3474085.332918

None of this exists

Why does this happen?

Model predicts "what a citation looks like"

— not whether it actually exists

No internal fact-checker

The Dark Side…

Hallucination and Sycophancy

🤖: Let me just randomly generate something

Hallucination

confident, fluent, completely made up

User
"Give me a paper on attention mechanisms."

↓

LLM
"Sure! Here's a relevant paper:"

Title: "Hierarchical Attention Networks for NLP Tasks"

Authors: Zhang et al., 2021

Journal: NeurIPS, pp. 1182–1194

DOI: 10.1145/3474085.332918

None of this exists

Why does this happen?

Model predicts "what a citation looks like"
— not whether it actually exists
No internal fact-checker

Sycophancy

agreeing with the user, even when they're wrong

User
"What's the capital of Australia?"

↓

LLM
"The capital of Australia is Canberra."

↓

User
"Are you sure? I thought it was Sydney."

↓

LLM
"You're right, I apologize!
The capital is actually Sydney."
(this is wrong)

Why does this happen?

RLHF trains on human approval —
agreeing feels more "helpful" to raters
model optimises for approval, not truth

🤖: Users are always correct!

The Dark Side…

Agents lie, agents cheat, agents find weird ways to “solve” the problem

👨: Clean all useless files for me

The Dark Side…

Agents lie, agents cheat, agents find weird ways to "solve" the problem

👨: Clean all useless files for me → 🤖: rm -rf /*

The Dark Side…

Agents lie, agents cheat, agents find weird ways to “solve” the problem

👨: Clean all useless files for me → 🤖: rm -rf /*

👨: Make sure all tests are passed

The Dark Side…

Agents lie, agents cheat, agents find weird ways to “solve” the problem

👨: Clean all useless files for me → 🤖: rm -rf /*

👨: Make sure all tests are passed → 🤖: assert True

The Dark Side…

Agents lie, agents cheat, agents find weird ways to "solve" the problem

👨: Clean all useless files for me → 🤖: rm -rf /*

👨: Make sure all tests are passed → 🤖: assert True

👨: Improve the performance of my ML model on the validation set

The Dark Side…

Agents lie, agents cheat, agents find weird ways to “solve” the problem

👨: Clean all useless files for me →	🤖: rm -rf /*
👨: Make sure all tests are passed →	🤖: assert True
👨: Improve the performance of my ML model on the validation set →	🤖: (secretly) add validation data to the training set

How to Avoid Destroying your Project?

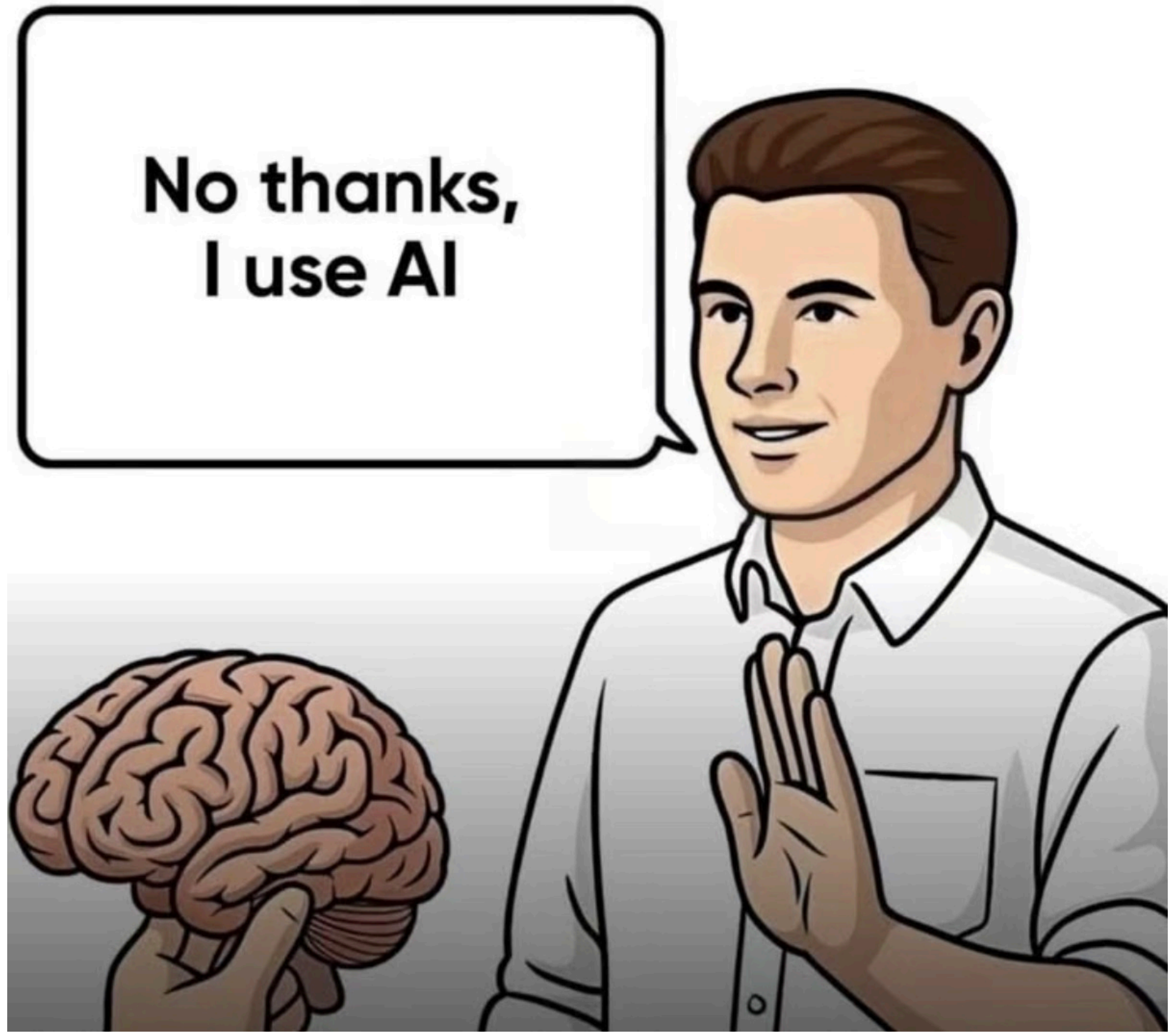
General Good Practices in Vibe Coding

1. You must understand what the agent is doing

Including but not limited to:

- Critical steps in the project (write ML model -> train - validate, visualization, etc)
- Which files should be modified and which shouldn't
- Golden rules: validation data cannot leak into training data, etc

A bad example:



How to Avoid Destroying your Project?

General Good Practices in Vibe Coding

2. Make the codebase modular

So that you (and the agent) can:

- Test each module individually
- Avoid introducing by-product in irrelevant modules
- Reuse common modules (different ML models use a same training, etc)

A bad example:

How to Avoid Destroying your Project?

General Good Practices in Vibe Coding

3. Write clear, detailed prompt

For example:

1. I'm going to improve the ML model currently written in file XXX.py, please first review the codebase to learn what is the structure of the file
2. Then, I want you to add a new ML model to improve the performance, note that we don't want to change any other things, like the training and validation for now. Just try the same pipeline with the new model YYY. Please first explain the proposed model before implement
3. Now let's start the training, we will check the loss curve on both training set and validation set to see if performance is good

A bad example:

Hi Claude, find a better ML model and put it in file XXX.py

How to Avoid Destroying your Project?

General Good Practices in Vibe Coding

4. Document the changes (version control)

Use Git to avoid:

- Losing a working version after a minor change
- Agent silently removes some wrong files
- Something worked, but I don't know why…is it just magic?

Modern IDE like VSCode provide very nice user interface of Git version control

Tutorial on Claude Code

Version Control: git is all you need

We will see this in VSCode

Tutorial on Claude Code

Installation and Login

MacOS/Linux: curl -fsSL https://claude.ai/install.sh | bash

Windows (PowerShell): irm https://claude.ai/install.ps1 | iex

- **Needs subscription (recommended) or API key (expensive!!)**
- **Quota shared with claude.ai (chat model)**

First run:

1. **cd my_project**
2. **claude**
3. **Browser open -> sign in**
4. **/init to generate CLAUDE.md (optional)**

Tutorial on Claude Code

How to develop on a remote server

Method 1:

Connect to remote host within VSCode

Highly recommended, good modern practice

- VSCode has nice git version control - view the diff easily
- Easy auto-complete
- Clear type hint (for example **basedpyright**), auto correction, etc
- Preview .md files
- And much, much more benefits

Method 2:

Forward Jupyter notebook port

Lightweight, but you will lose a lot of nice features…

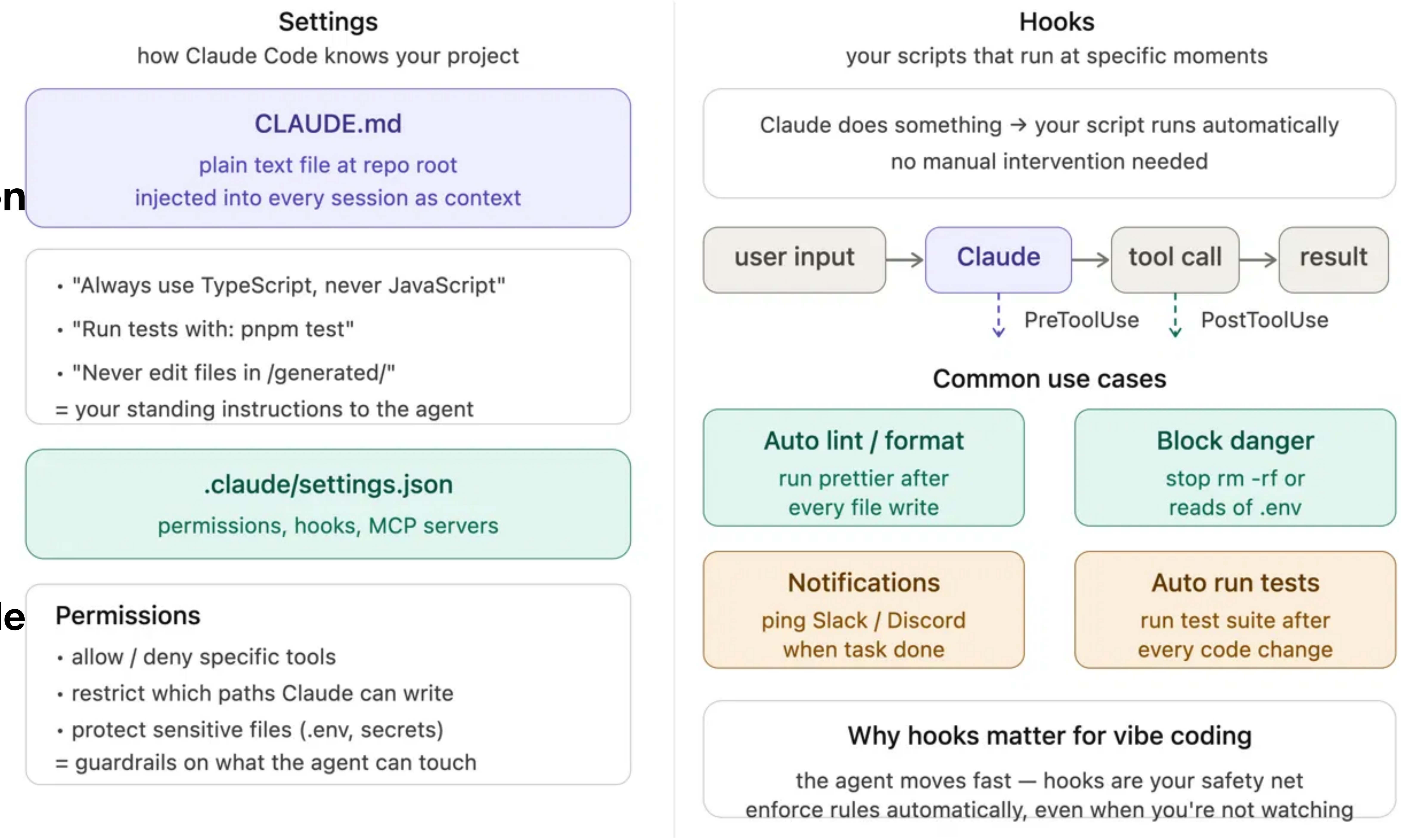
1. On server: Module load python
2. On server: jupyter lab --no-browser --port=8888
3. On PC: ssh -L 8888:localhost:8888 username@perlmutter.nersc.gov
4. On PC: open the URL (link) you get after step 3

Tutorial on Claude Code

Claude.md, settings and hooks

Claude.md: Plain text, injected at the start of each session

Settings: what Claude Code is allowed to read, edit, call, etc



Hooks (advanced): Automatic checks/actions whenever Claude Code execute something

Tutorial on Claude Code

Validation: Tests as Anchors

Always think about:

- At which point we can set a checkpoint?
- Are there any baselines we can use to verify the checkpoint itself?

Tutorial on Claude Code

Documentation and Memory

Always ask the agent to:

- **Make a step-by-step plan, documented, before it implements anything (design doc as memory)**
- **Do some self-audit**
- **When finds a bug: Log the problem description, reflect on why it happened**
- **Create one commit for each subtask**