LLM, AI Agent and Vibe Coding

Introduction and Recipe

Yue Ma, Postdoctoral Scholar at HDSI, UCSD
06/22/2026 @ 0νββ AI Summer School

A special acknowledgement in my PhD defense slides…

Acknowledgement

Large Language Models

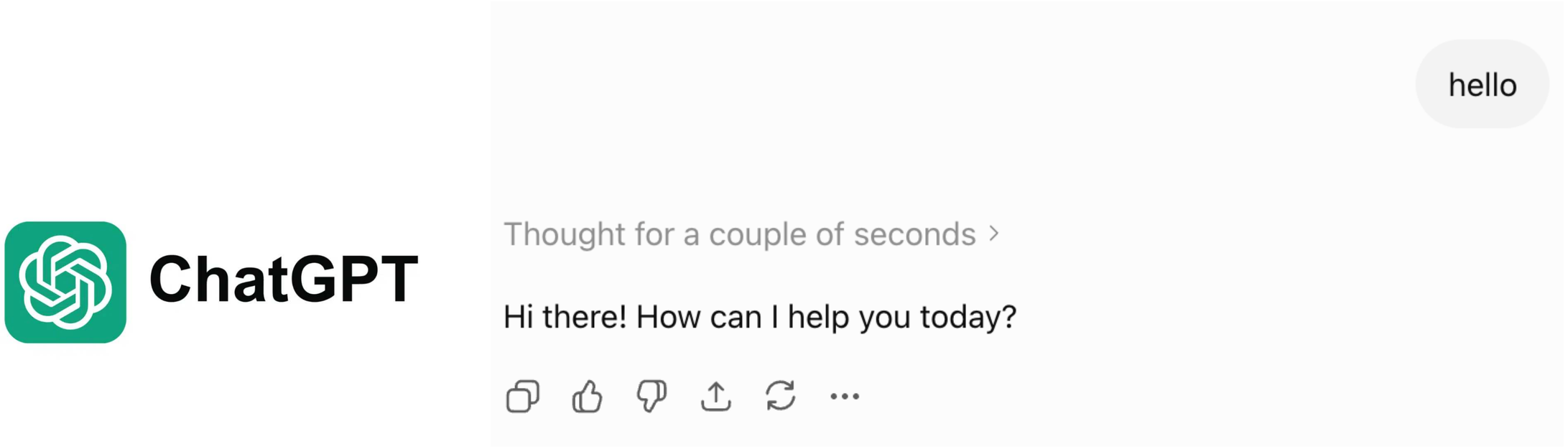
Claude — For the assistance in coding

ChatGPT — For discussion and emotional support

Gemini — For generating all comics in the presentation

Part 1: Introduction to LLM

Introduction: What is LLM



Question:
what is happening when you chat to ChatGPT/Claude/etc?
How are the text generated?

Introduction: What is LLM

Raw LLM (Foundation Model) is just a probability predictor

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

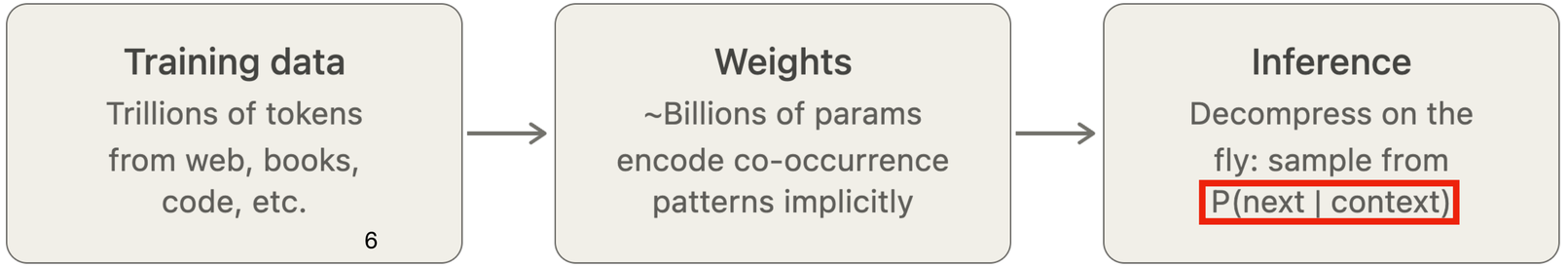
- **Understanding** — No grounded semantics
- **Reasoning** — No explicit logic

What this IS

- **Pattern matching** — Over vast corpora
- **Compression** — Statistical model

Training: learn the probability distribution from data

Analogy: extreme statistical compression



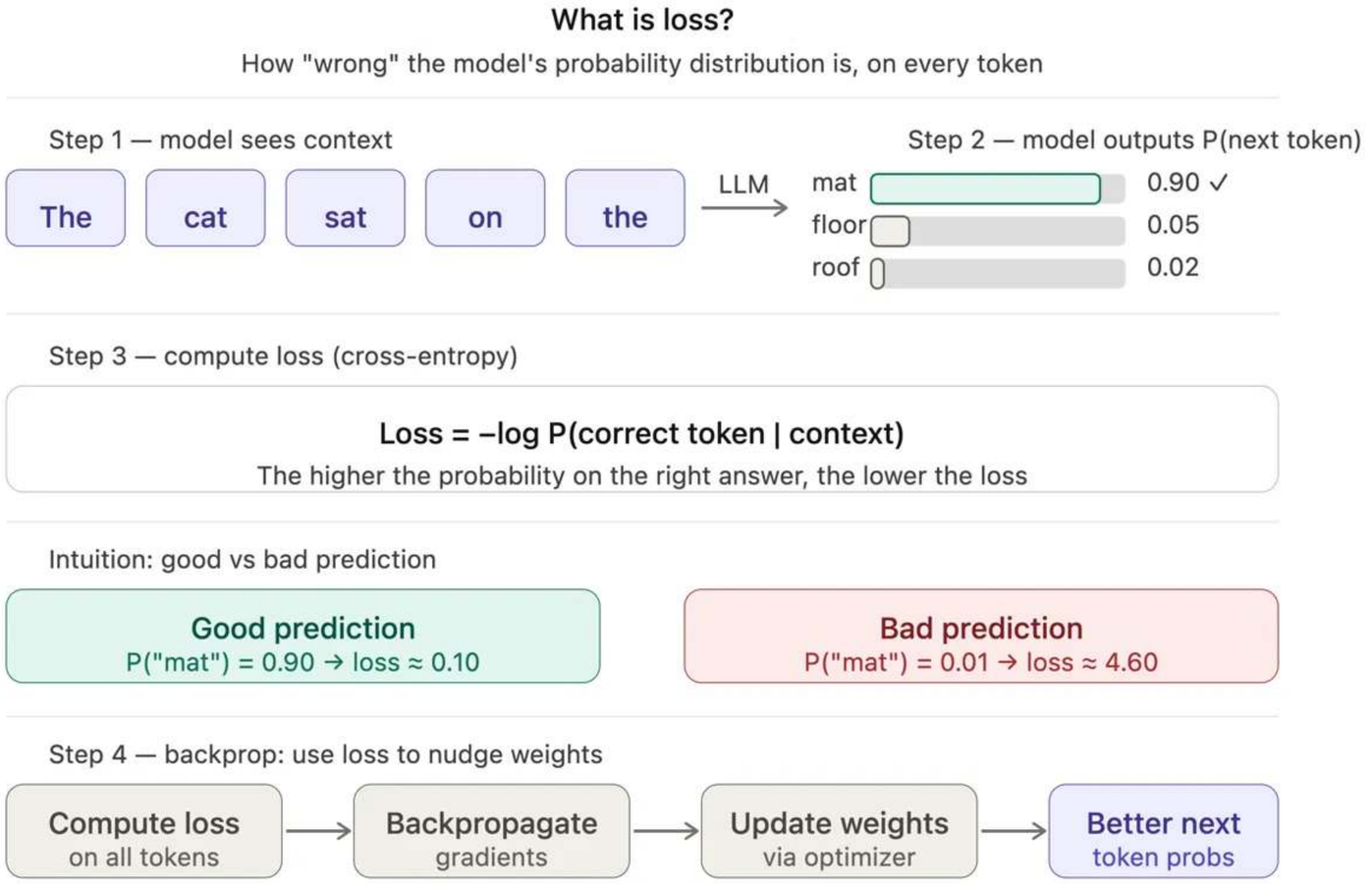
Introduction: What is LLM

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



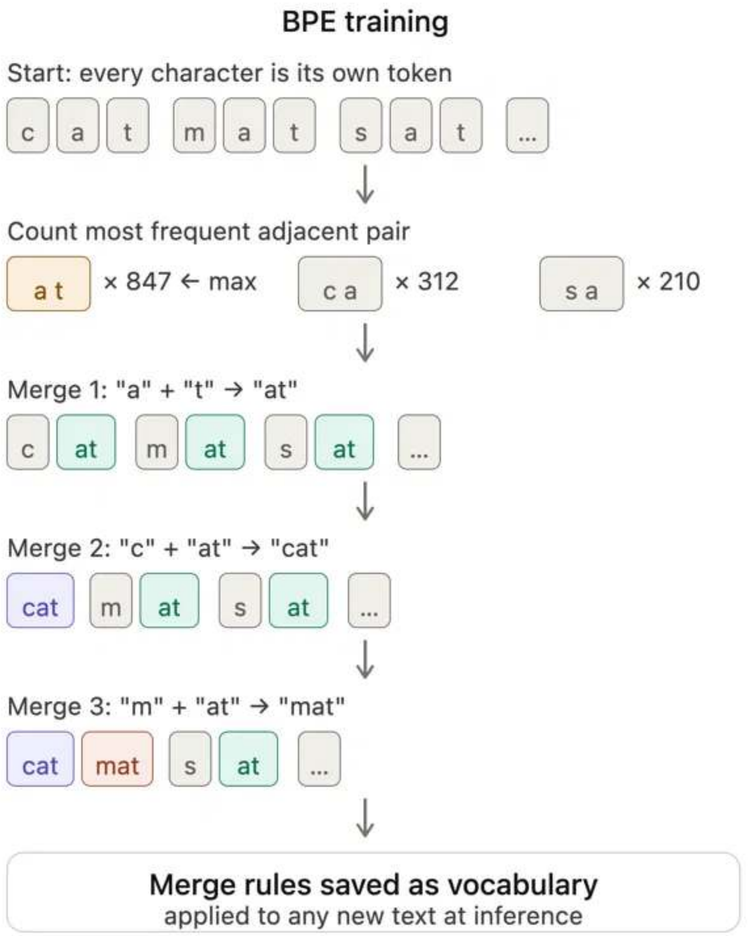
Introduction: What is LLM

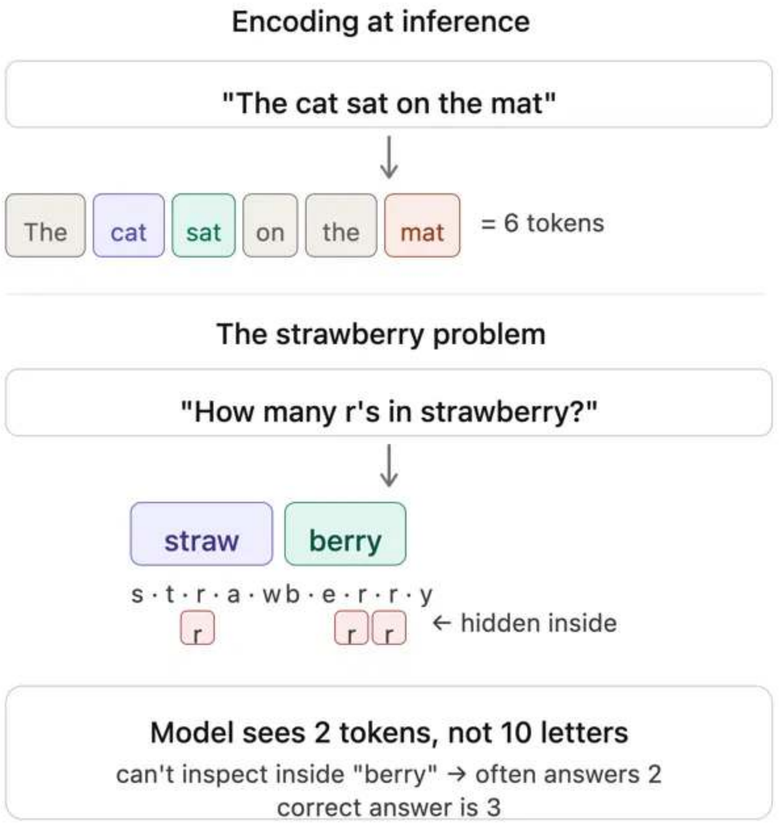
Tokenization: Text -> Token

BPE: Byte Pair Encoding

- **Not based on grammar, but purely data-driven**
- **Merge the high-frequency combinations**
- **Stop when the pool size is satisfactory (GPT-4 used 100k token)**

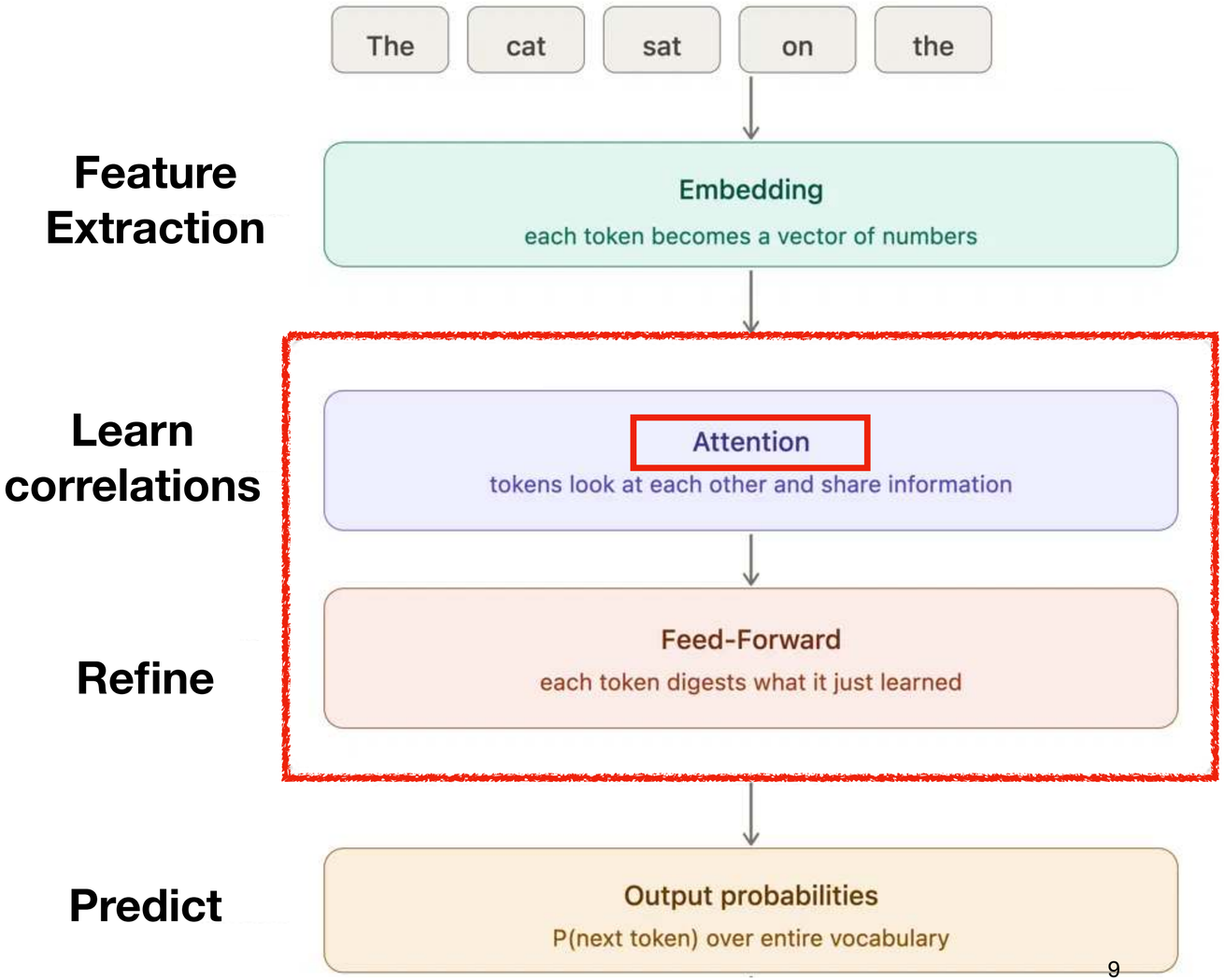
Question:
How about 1 token per character?
Or 1 token per word?





Introduction: What is LLM

Transformer: Attention is all you need!



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.

Introduction: What is LLM

Transformer: Attention is all you need!

For each token:

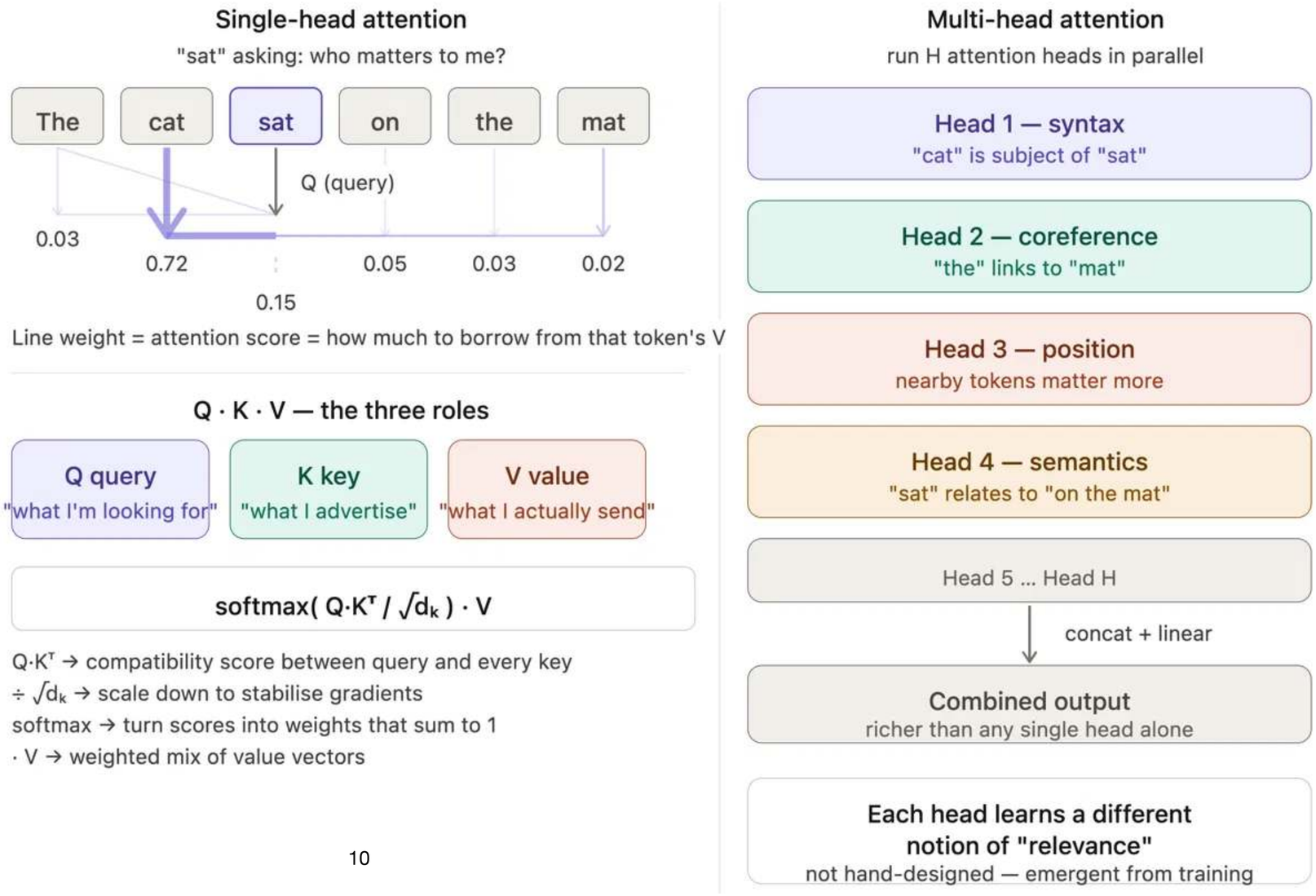
- **Q, K, V are vectors**

For a sentence (array):

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

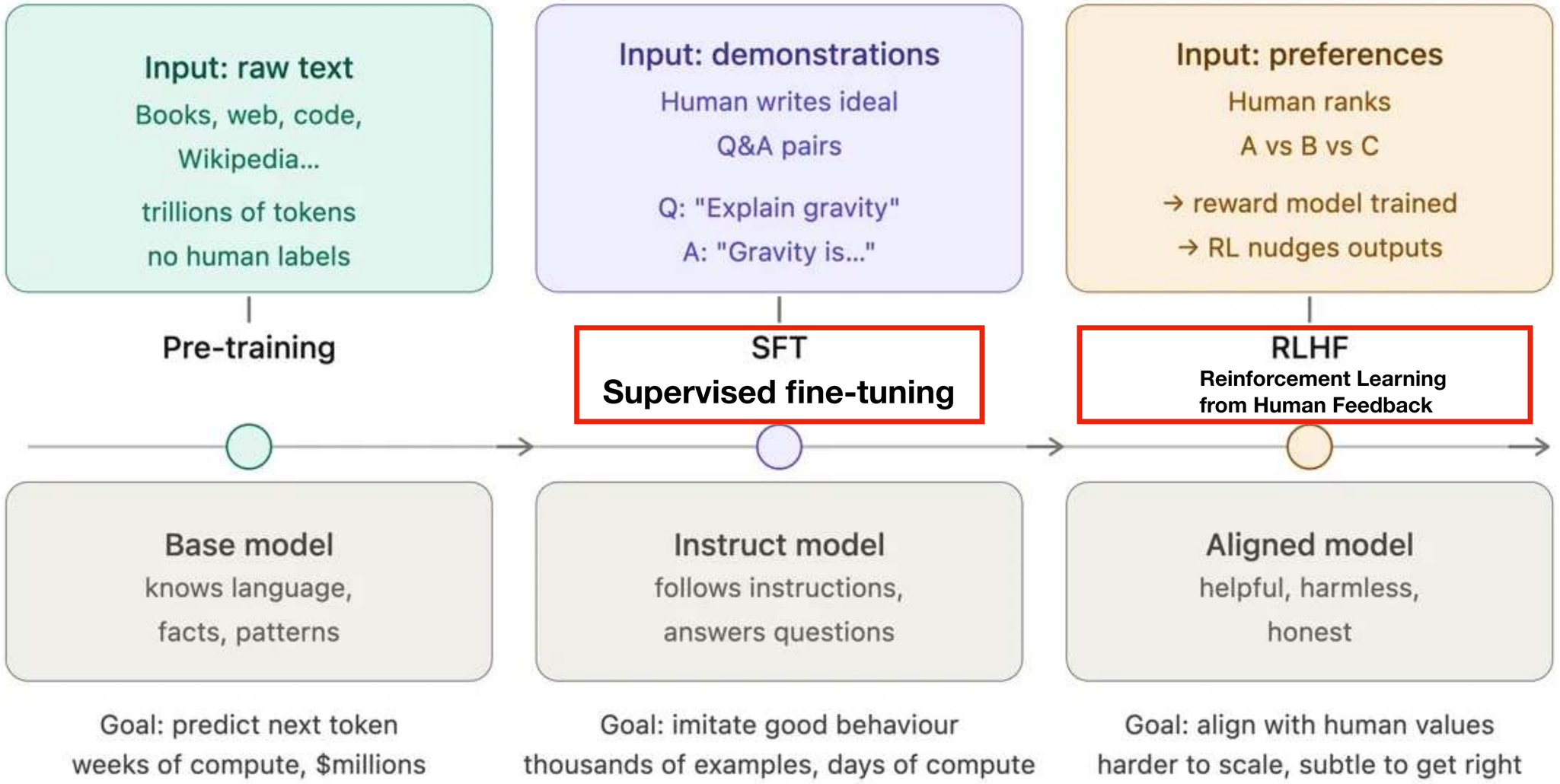
Self Attention:

- **Tokens “query” each other**



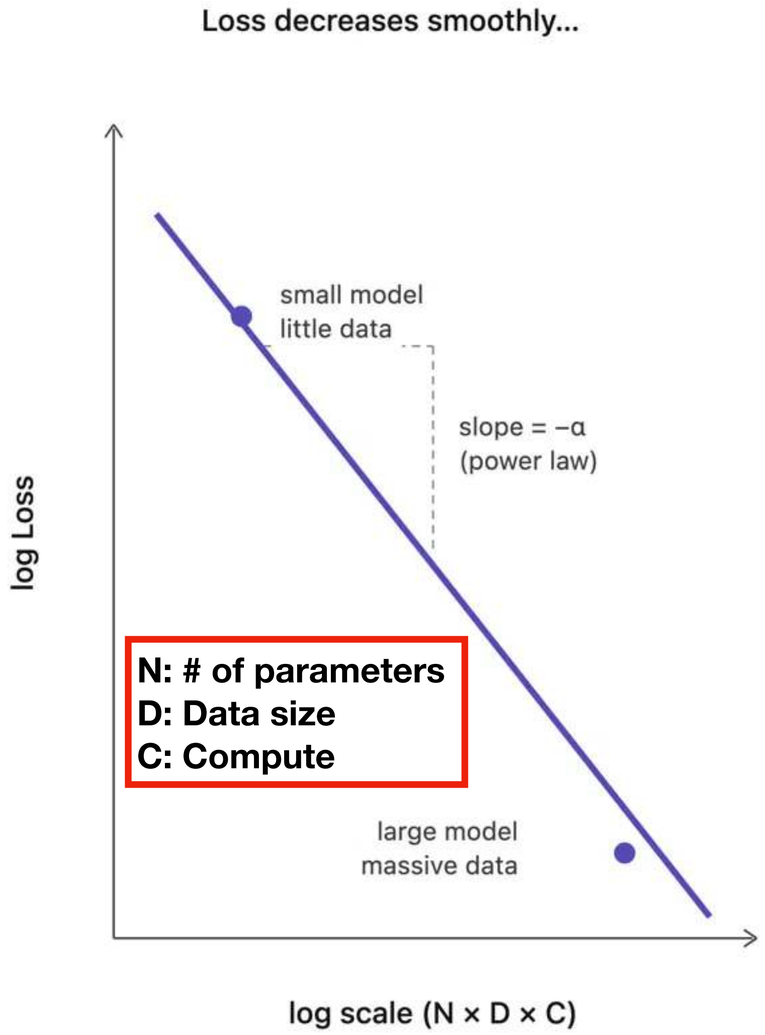
Lifecycle of a Raw LLM

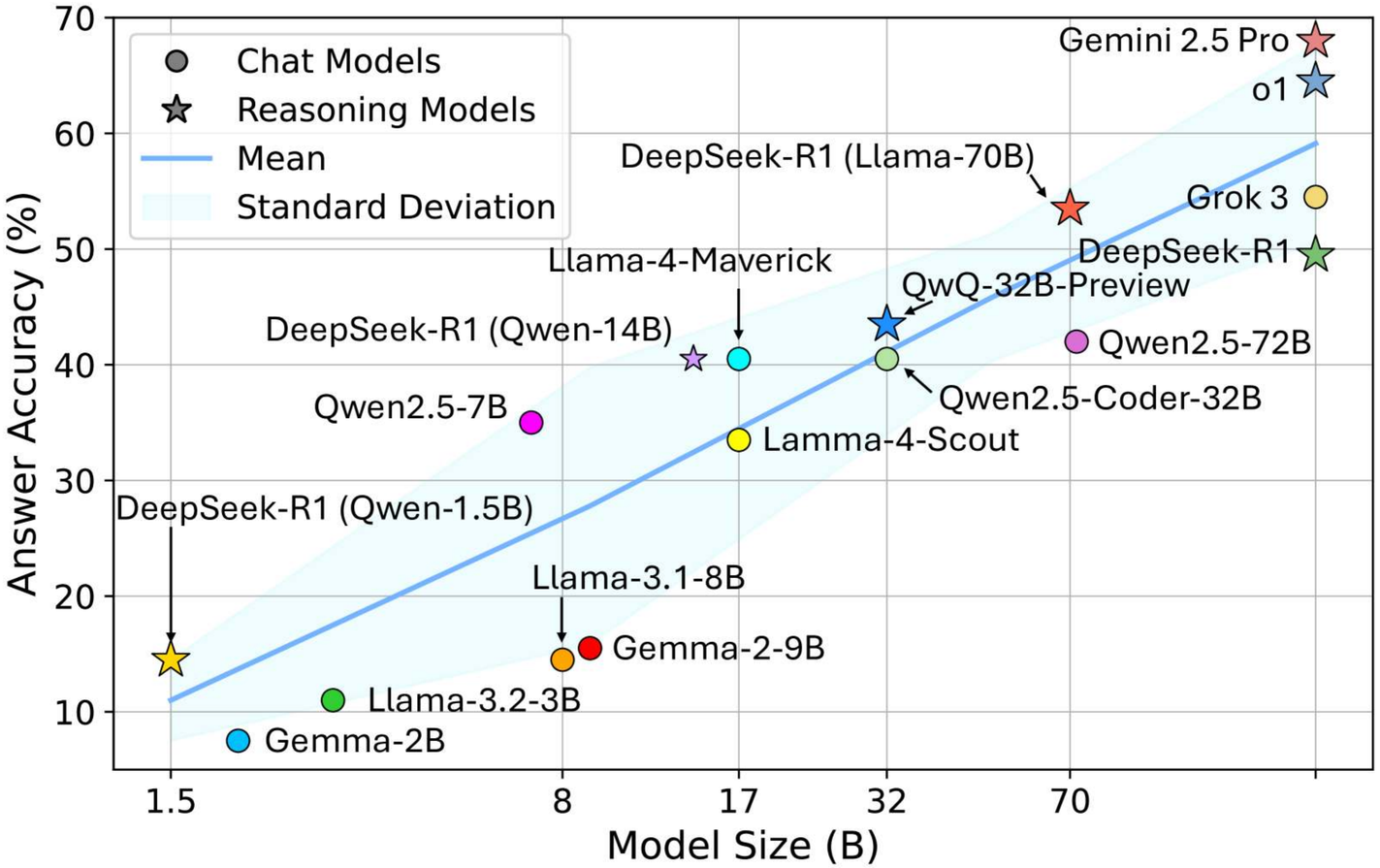
Pre-training -> Post training



Lifecycle of a Raw LLM

The Scaling Law





From Raw LLM to Chat Model

Previous question: what we are actually chatting with?

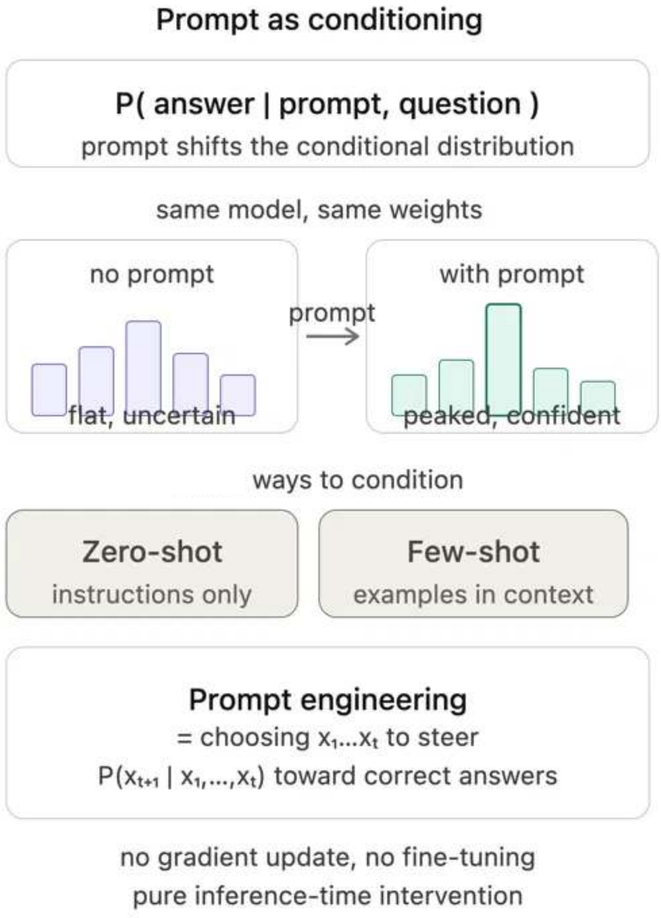
Raw LLM **+**

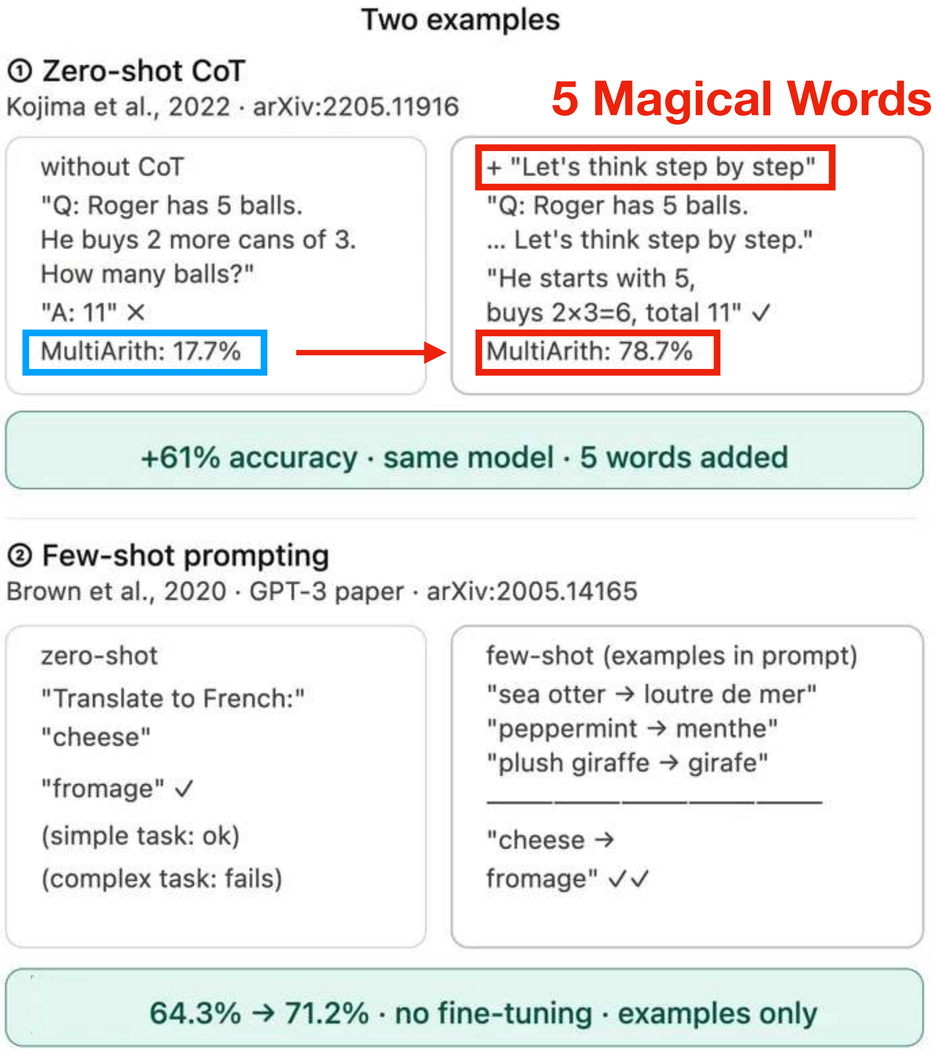
- **Prompt Engineering**
- **Knowledge injection**
- **Test-time compute (Reasoning, etc)**
- **Tool calling**
- **Memory control**
- **And more…**

From Raw LLM to Chat Model

Prompt Engineering

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





From Raw LLM to Chat Model

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)

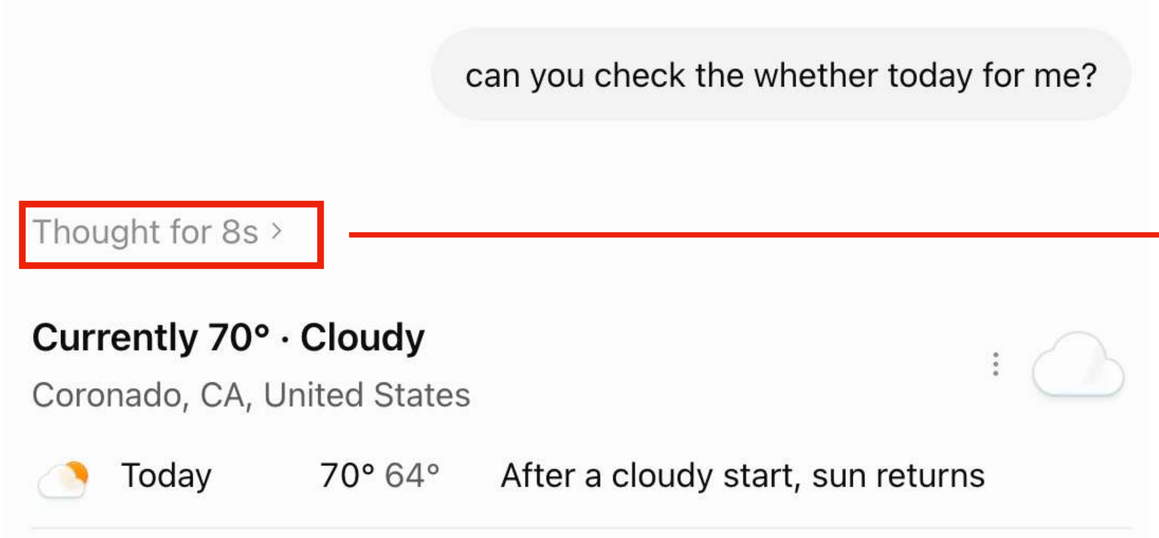
Loved by AI pioneers:

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

From Raw LLM to Chat Model

Test-time compute



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!

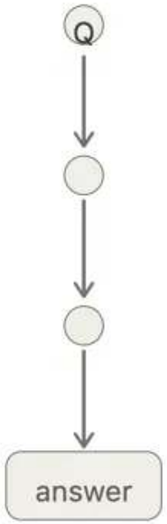
Thought for 8s
Done

From Raw LLM to Chat Model

Test-time compute

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

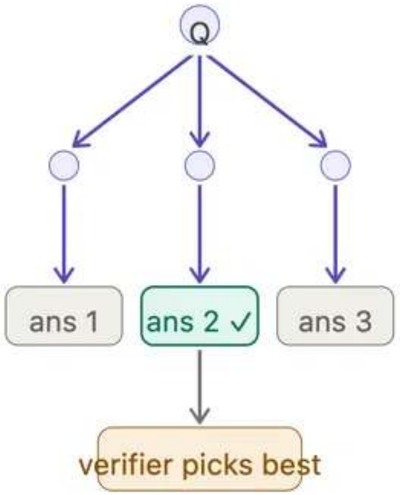
Greedy



always pick
highest P token

fast, but
no exploration

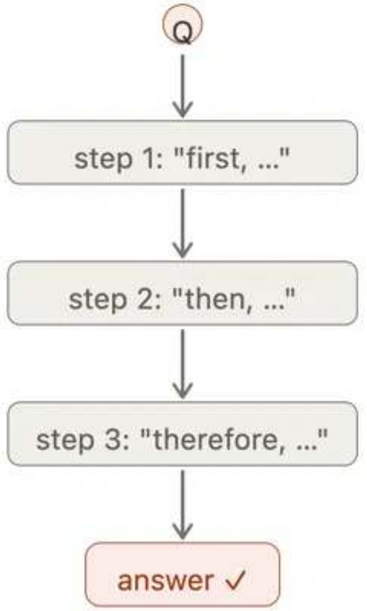
Best-of-N



sample N times,
score each answer

more samples =
better coverage

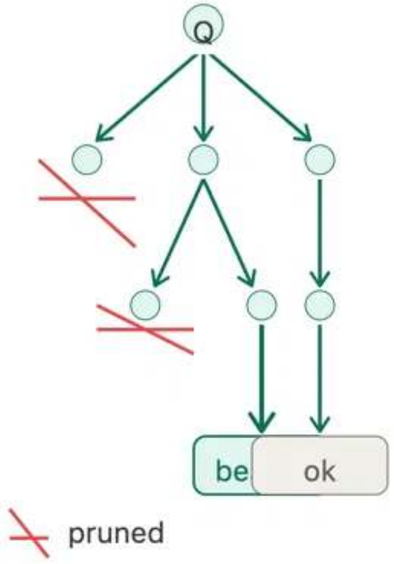
Chain-of-Thought



more tokens =
more compute

"thinking out loud"
turns implicit → explicit

Tree Search

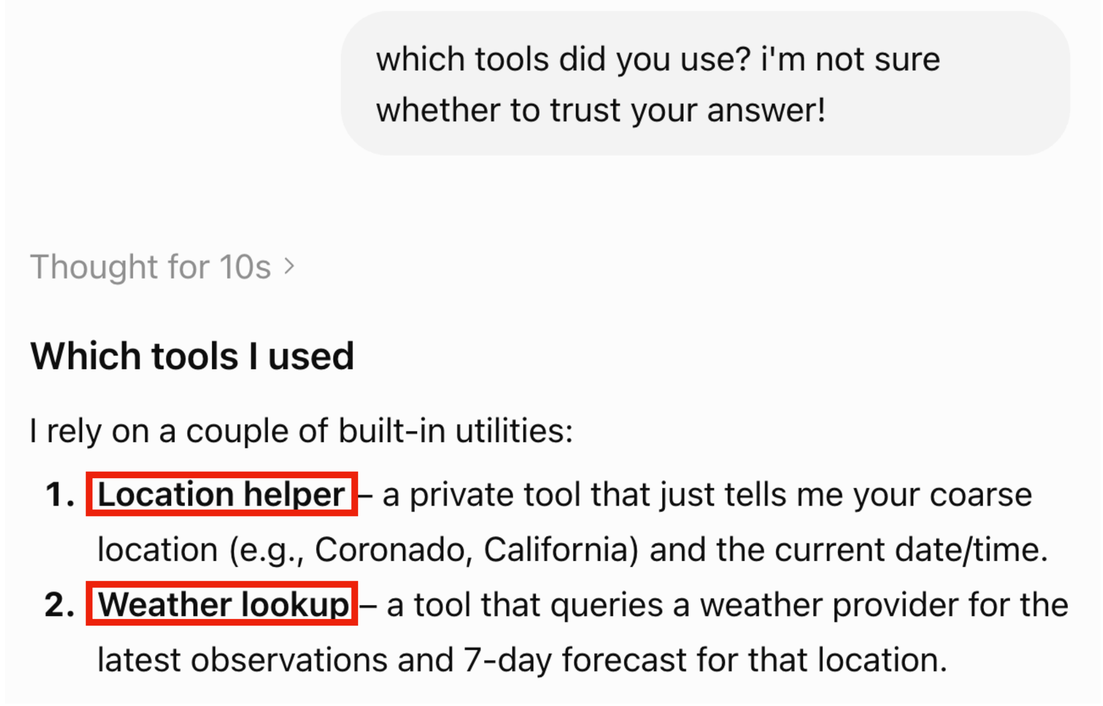


explore branches,
prune bad paths

o1 / o3 style
most expensive

From Raw LLM to Chat Model

Tool Calling



Gemini

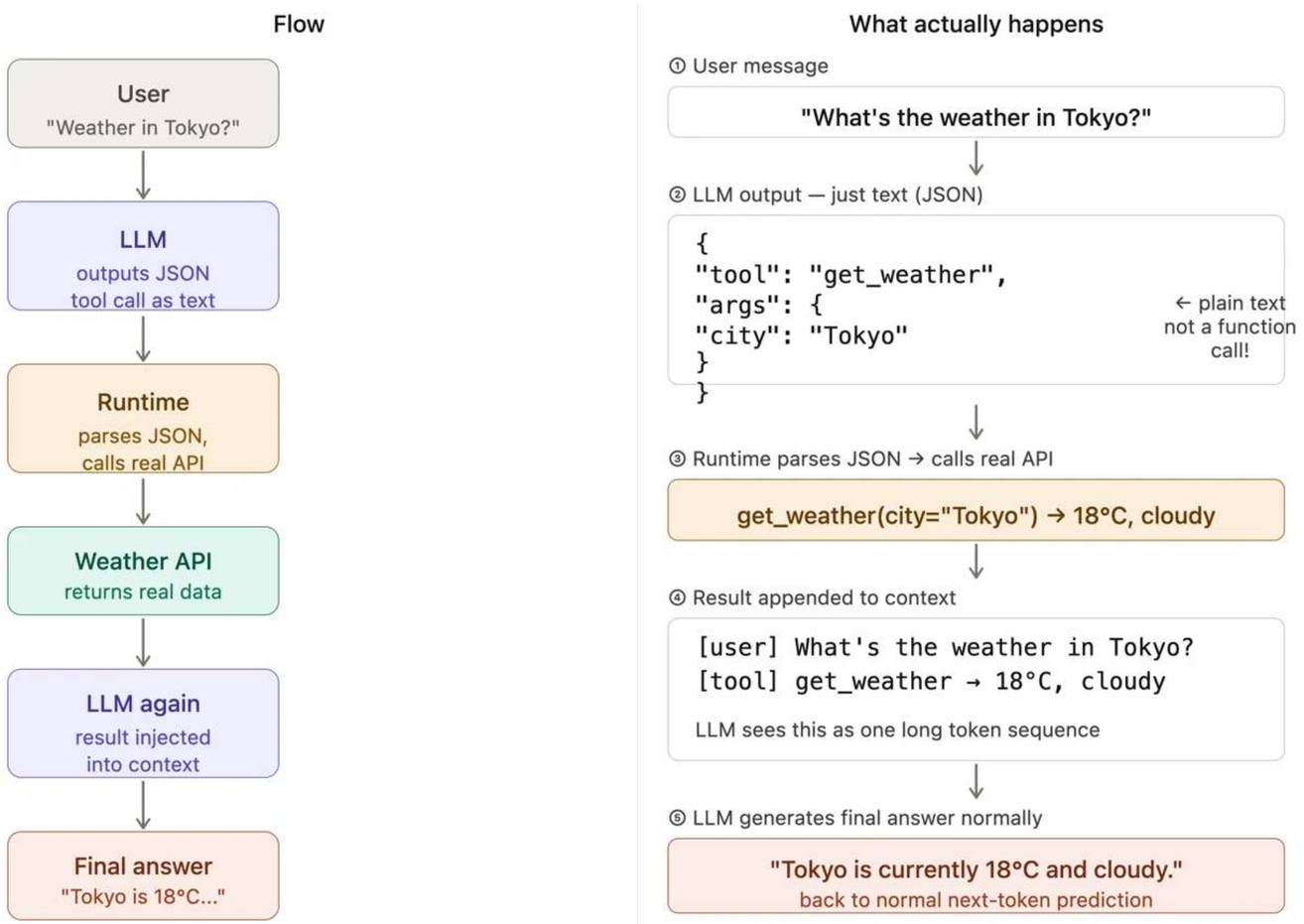
So smart!

But how did it happen?

I thought LLM can only produce some texts!

From Raw LLM to Chat Model

Tool Calling



Tool calling =

Pre-trained format (message -> JSON)

+

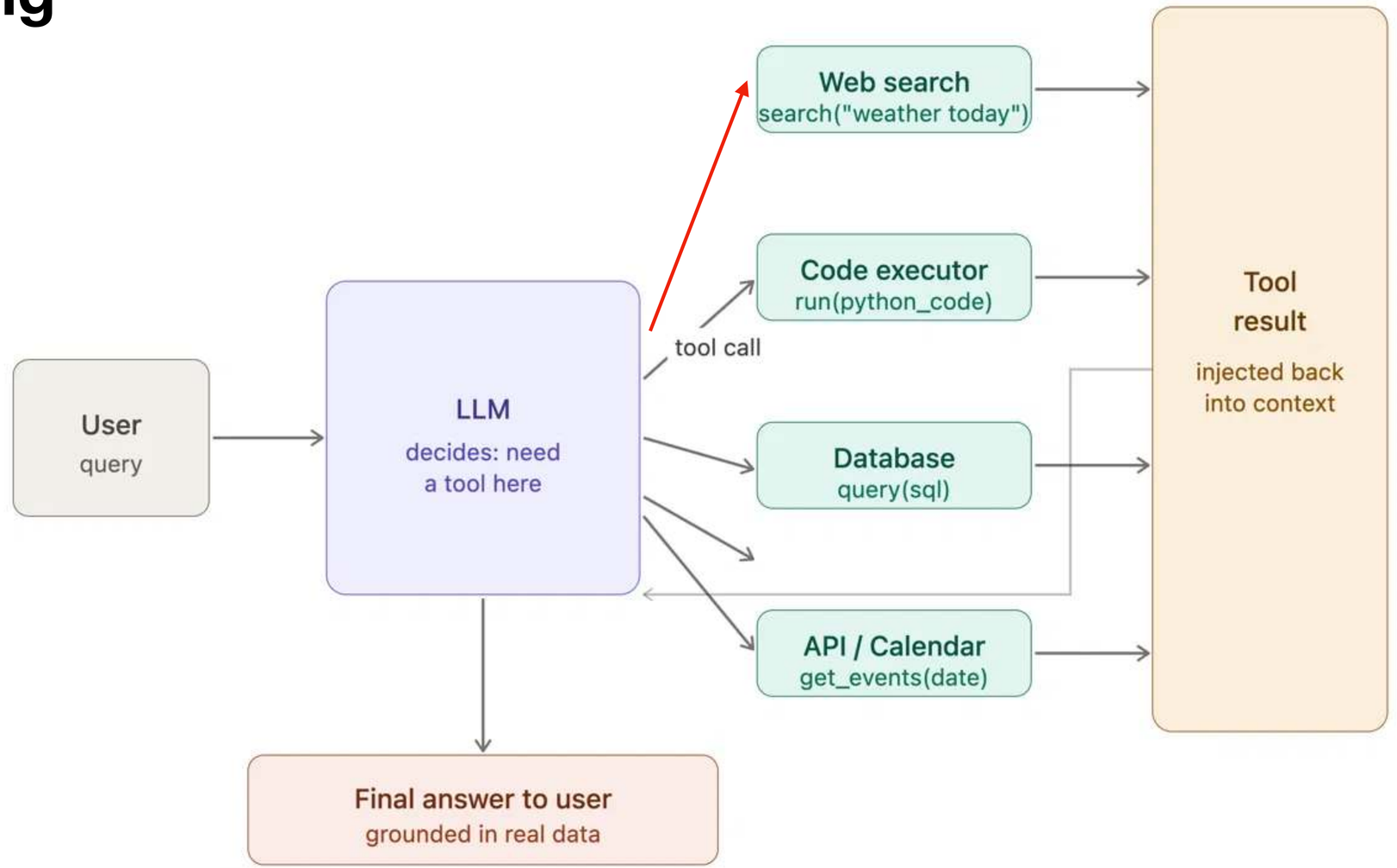
API (JSON -> real function call)

+

Context aggregation (result -> answer)

From Raw LLM to Chat Model

Tool Calling



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

From Raw LLM to Chat Model

Memory Control

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

How to Evaluate the LLMs?

Benchmarking is critical

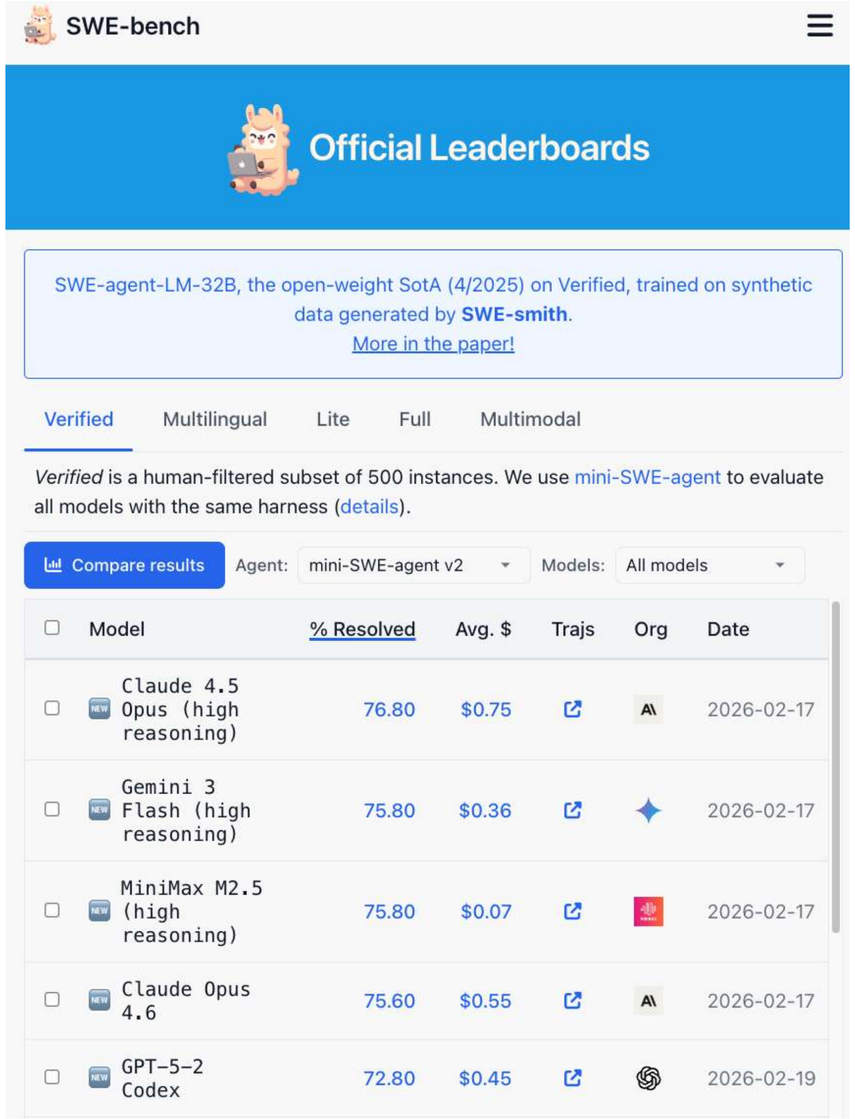
Published as a conference paper at ICLR 2024

SWE-BENCH: CAN LANGUAGE MODELS RESOLVE REAL-WORLD GITHUB ISSUES?

Carlos E. Jimenez[*1,2] **John Yang**[*1,2] **Alexander Wettig**[1,2]
Shunyu Yao[1,2] **Kexin Pei**[3] **Ofir Press**[1,2] **Karthik Narasimhan**[1,2]
[1]Princeton University [2]Princeton Language and Intelligence [3]University of Chicago

ABSTRACT

Language models have outpaced our ability to evaluate them effectively, but for their future development it is essential to study the frontier of their capabilities. We find real-world software engineering to be a rich, sustainable, and challenging testbed for evaluating the next generation of language models. To this end, we introduce SWE-bench, an evaluation framework consisting of 2,294 software engineering problems drawn from real GitHub issues and corresponding pull requests across 12 popular Python repositories. Given a codebase along with a description of an issue to be resolved, a language model is tasked with editing the codebase to address the issue. Resolving issues in SWE-bench frequently requires understanding and coordinating changes across multiple functions, classes, and even files simultaneously, calling for models to interact with execution environments, process extremely long contexts and perform complex reasoning that goes far beyond traditional code generation tasks. Our evaluations show that both state-of-the-art proprietary models and our fine-tuned model SWE-Llama can resolve only the simplest issues. The best-performing model, Claude 2, is able to solve a mere 1.96% of the issues. Advances on SWE-bench represent steps towards LMs that are more practical, intelligent, and autonomous.



How to Evaluate the LLMs?

Benchmarking is critical

Humanity's Last Exam

Organizing Team

Long Phan*[1], Alice Gatti*[1], Ziwen Han*[2], Nathaniel Li*[1],

Josephina Hu[2], Hugh Zhang‡, Chen Bo Calvin Zhang[2], Mohamed Shaaban[2], John Ling[2], Sean Shi[2], Michael Choi[2], Anish Agrawal[2], Arnav Chopra[2], Adam Khoja[1], Ryan Kim†, Richard Ren[1], Jason Hausenloy[1], Oliver Zhang[1], Mantas Mazeika[1],

Summer Yue**[2], Alexandr Wang**[2], Dan Hendrycks**[1]

[1] **Center for AI Safety,** [2] **Scale AI**

Abstract

Benchmarks are important tools for tracking the rapid advancements in large language model (LLM) capabilities. However, benchmarks are not keeping pace in difficulty: LLMs now achieve over 90% accuracy on popular benchmarks like MMLU, limiting informed measurement of state-of-the-art LLM capabilities. In response, we introduce HUMANITY'S LAST EXAM (HLE), a multi-modal benchmark at the frontier of human knowledge, designed to be the final closed-ended academic benchmark of its kind with broad subject coverage. HLE consists of 2,500 questions across dozens of subjects, including mathematics, humanities, and the natural sciences. HLE is developed globally by subject-matter experts and consists of multiple-choice and short-answer questions suitable for automated grading. Each question has a known solution that is unambiguous and easily verifiable, but cannot be quickly answered via internet retrieval. State-of-the-art LLMs demonstrate low accuracy and calibration on HLE, highlighting a significant gap between current LLM capabilities and the expert human frontier on closed-ended academic questions. To inform research and policymaking upon a clear understanding of model capabilities, we publicly release HLE at https://lastexam.ai.

2500 PhD-level Q&A

Designed for O(years) of development

But it's saturated very quickly

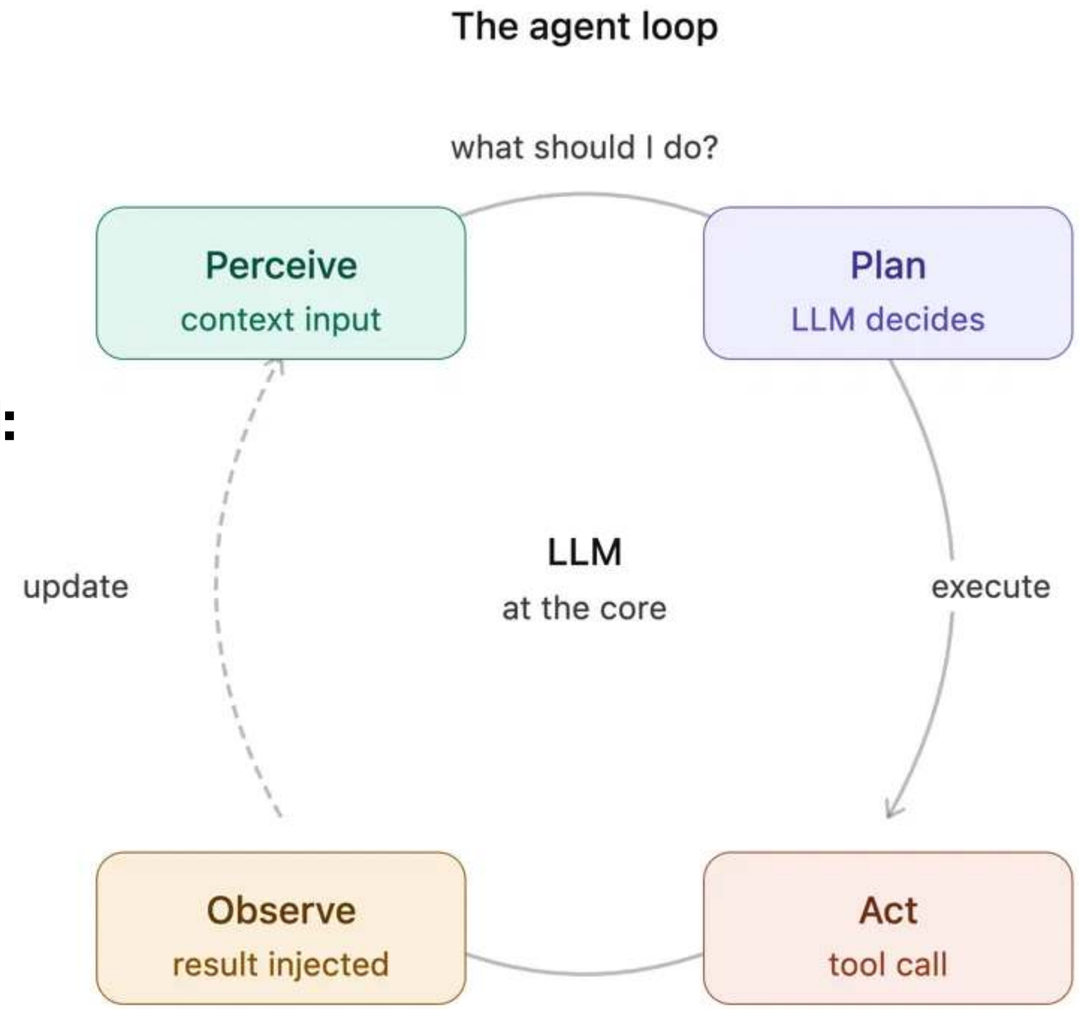
New benchmarks are always needed

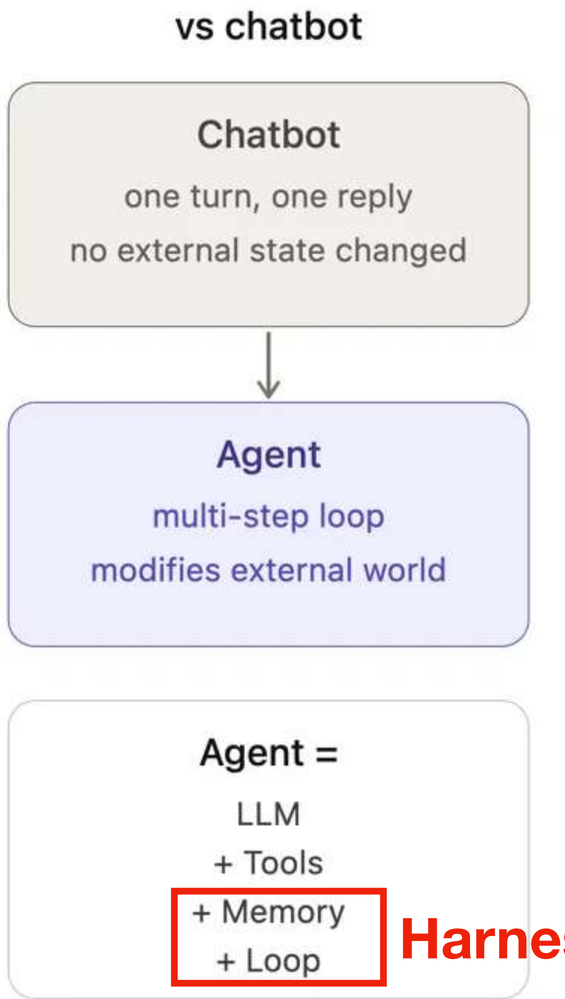
From Chat Model to Agent

Definition of LLM Agent

Beyond the chat model:

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





From Chat Model to Agent

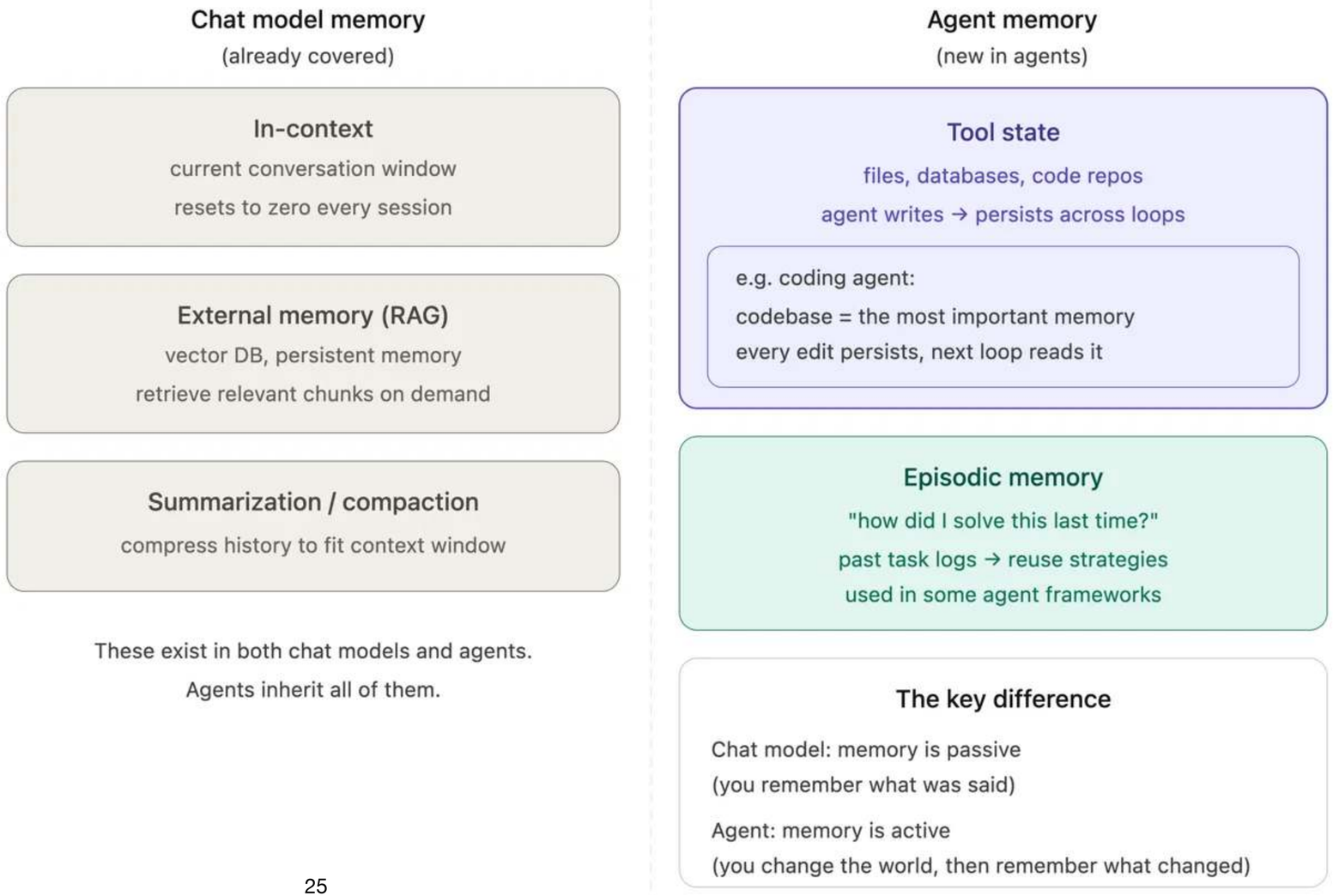
Memory of LLM Agents

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**

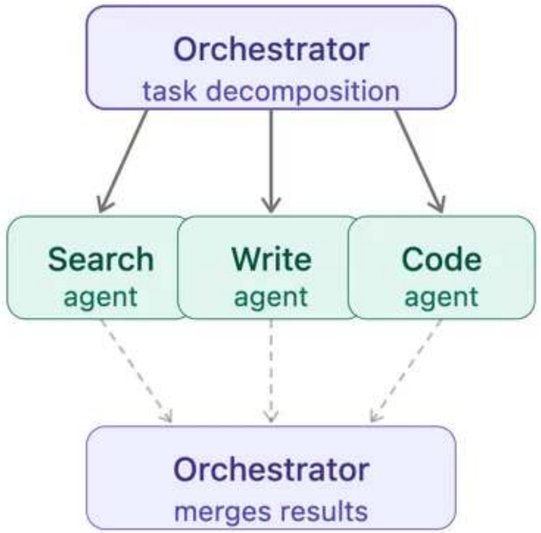


From Chat Model to Agent

Multi-agent Collaboration

Orchestrator + Sub-agents

AutoGen · arXiv:2308.08155



Why this helps

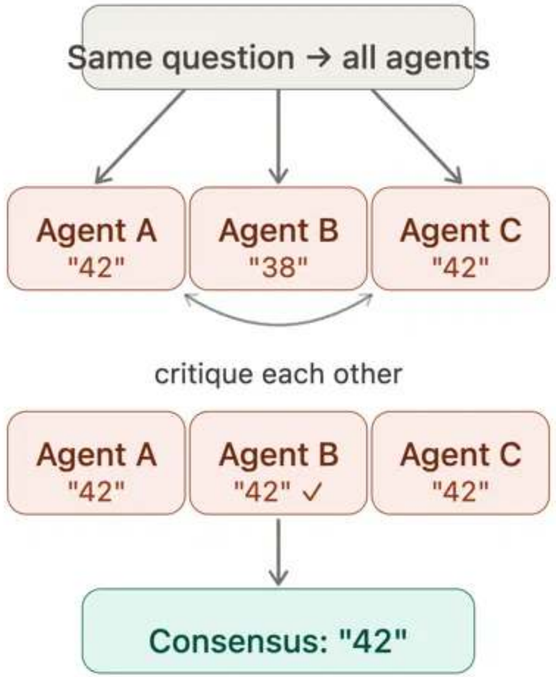
each sub-agent has focused context

tasks run in parallel

failure isolated per sub-agent

Agent Debate

Du et al. · arXiv:2305.14325



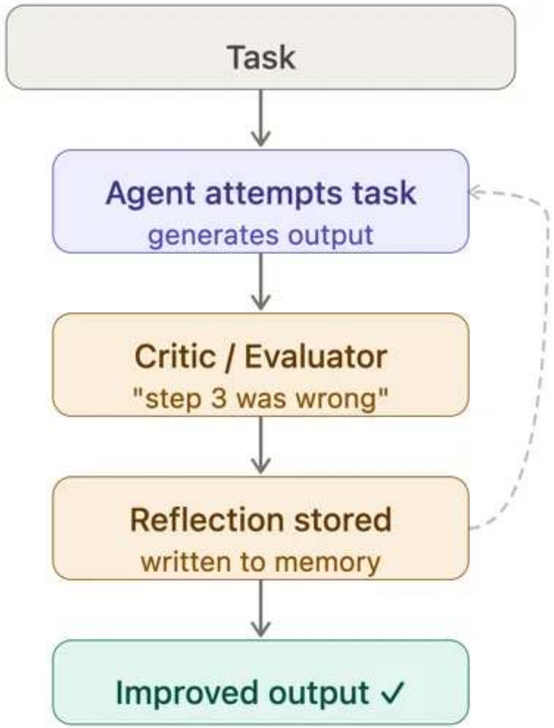
Why this helps

reduces hallucination via peer review

math+reasoning +~10% vs single agent

Reflexion

Shinn et al. · arXiv:2303.11366



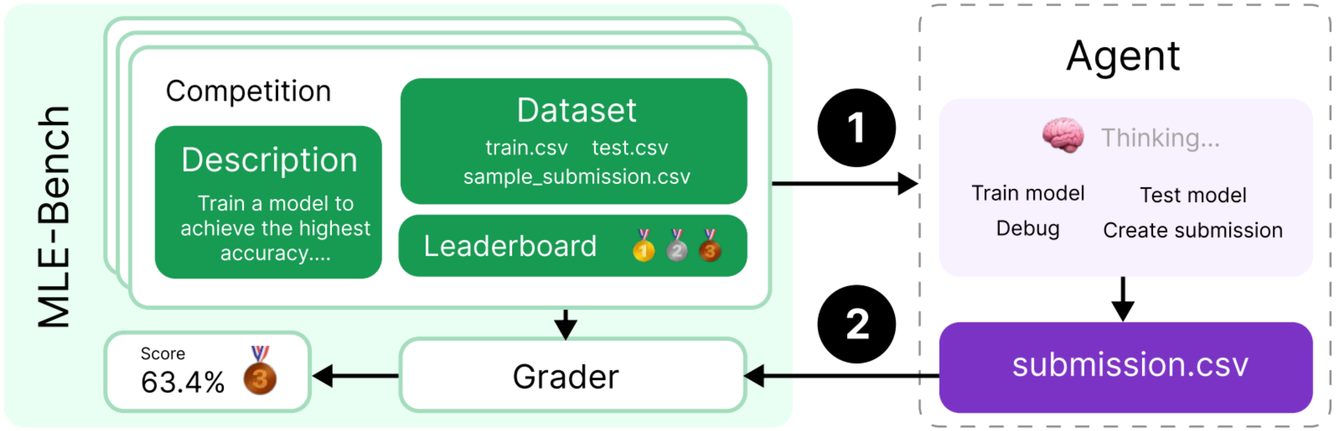
Why this helps

no retraining needed

agent learns from mistakes in-context

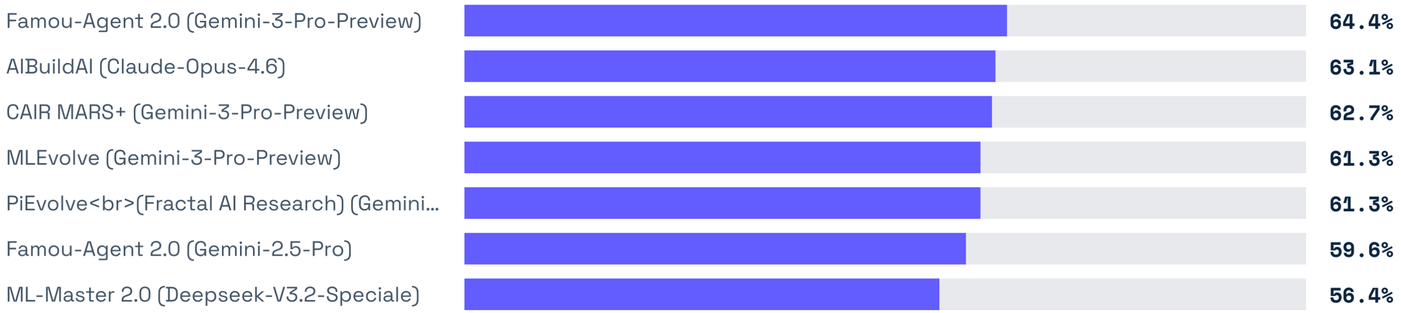
From Chat Model to Agent

Benchmarks Specifically for LLM Agents



Performance Overview

Updated 1 month ago · view source



MLE-BENCH: EVALUATING MACHINE LEARNING AGENTS ON MACHINE LEARNING ENGINEERING

Chan Jun Shern*, Neil Chowdhury*,†, Oliver Jaffe*, James Aung*, Dane Sherburn*, Evan Mays*, Giulio Starace*, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng†, Aleksander Mądry

OpenAI

ABSTRACT

We introduce MLE-bench, a benchmark for measuring how well AI agents perform at machine learning engineering. To this end, we curate 75 ML engineering-related competitions from Kaggle, creating a diverse set of challenging tasks that test real-world ML engineering skills such as training models, preparing datasets, and running experiments. We establish human baselines for each competition using Kaggle's publicly available leaderboards. We use open-source agent scaffolds to evaluate several frontier language models on our benchmark, finding that the best-performing setup — OpenAI's o1-preview with AIDE scaffolding — achieves at least the level of a Kaggle bronze medal in 16.9% of competitions. In addition to our main results, we investigate various forms of resource-scaling for AI agents and the impact of contamination from pre-training. We open-source our benchmark code (github.com/openai/mle-bench/) to facilitate future research in understanding the ML engineering capabilities of AI agents.

From Chat Model to Agent

Benchmarks Specifically for LLM Agents

Test if the agent can:

- **Reproduce AI research papers**
- **Analyze mathematical formula + Build models + tune hyper parameters**

PaperBench: Evaluating AI's Ability to Replicate AI Research

Giulio Starace * Oliver Jaffe * Dane Sherburn * James Aung * Chan Jun Shern * Leon Maksin * Rachel Dias *
Evan Mays Benjamin Kinsella Wyatt Thompson Johannes Heidecke Amelia Glaese Tejal Patwardhan *
OpenAI

MODEL	PAPERBENCH
O3-MINI-HIGH	8.5 ± 0.8
CLAUDE-3.5-SONNET	16.1 ± 0.1
O1-HIGH	24.4 ± 0.7
With an extended 36 hour limit	
O1-HIGH	26.0 ± 0.3

From Chat Model to Agent

Benchmarks Specifically for LLM Agents

Test if the agent can:

- Reproduce Physics research papers
- Perform non-trivial computational modeling or numerical simulation

PRBench: End-to-end Paper Reproduction in Physics Research

Shi Qiu[1], **Junyi Deng**[1], **Yiwei Deng**[1], **Haoran Dong**[1], **Jieyu Fu**[1], **Mao Li**[1], **Zeyu Li**[1], **Zhaolong Zhang**[1], **Huiwen Zheng**[1], **Leidong Bao**[1], **Anqi Lv**[1], **Zihan Mo**[1], **Yadi Niu**[1], **Yiyang Peng**[1], **Yu Tian**[1], **Yili Wang**[1], **Ziyu Wang**[1], **Zi-Yu Wang**[1], **Jiashen Wei**[1], **Liuheng Wu**[1], **Aoran Xue**[1], **Leyi Yang**[1], **Guanglu Yuan**[1], **Xiarui Zhan**[1], **Jingjun Zhang**[1], **Zifan Zheng**[1], **Pengfei Liu**[1], **Linrui Zhen**[1], **Kaiyang Li**[1], **Qichang Li**[1], **Ziheng Zhou**[1], **Guo-En Nian**[1], **Yunwei Xiao**[1], **Qing-Hong Cao**[1], **Linjie Dai**[1], **Xu Feng**[1], **Peng Gao**[1], **Ying Gu**[1], **Chang Liu**[1], **Jia Liu**[1], **Ming-xing Luo**[2], **Yan-Qing Ma**[1], **Liang-You Peng**[1], **Huichao Song**[1], **Shufeng Wang**[1], **Chenxu Wang**[1], **Tao Wang**[1], **Yi-Nan Wang**[1], **Chengyin Wu**[1], **Pengwei Zhao**[1], and **Hua Xing Zhu**[1,†]

[1]School of Physics, Peking University, China
[2]Beijing Computational Science Research Center, China

Agent	Methodology	Code	Data Acc.	Instruction	Overall
OpenAI Codex					
GPT-5.3-Codex	**78.00**	**43.00**	**21.00**	**92.00**	**34.00**
OpenCode-based Agents					
GPT-5.3-Codex	72.00	36.00	16.00	90.00	28.50
Kimi K2.5 (1T)	61.90	22.00	11.40	80.60	20.57
DeepSeek V3.2 (671B)	63.20	19.30	8.90	84.20	18.50
Minimax 2.7 (230B)	55.60	16.30	10.00	86.00	17.97
GLM-5 (744B)	50.50	18.80	10.60	67.00	17.87
End-to-End Callback Rate			0.0		

From Chat Model to Agent

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

OpenClaw — Personal AI Assistant

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.

OpenClaw:

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

Hermes Agent

Hermes Agent | Hermes Desktop

DOCS HERMES-AGENT.NOUSRESEARCH.COM | DISCORD | LICENSE MIT

BUILT BY NOUS RESEARCH | LANG 中文 | LANG ا ر د و

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.

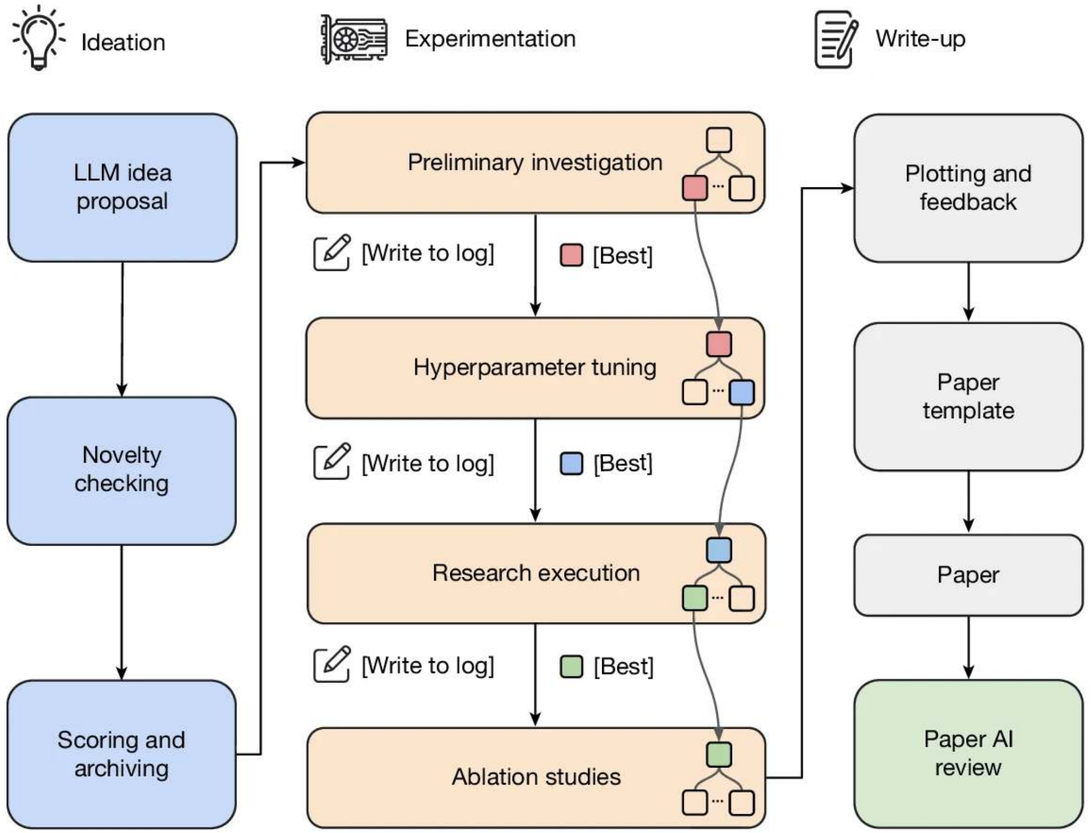
Use any model you want — Nous Portal, OpenRouter (200+ models), NovitaAI (AI-native cloud for Model API, Agent Sandbox, and GPU Cloud), NVIDIA NIM (Nemotron), Xiaomi MiMo, z.ai/GLM, Kimi/Moonshot, MiniMax, Hugging Face, OpenAI, or your own endpoint. Switch with `hermes model` — no code changes, no lock-in.

Hermes Agent:

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

From Chat Model to Agent

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

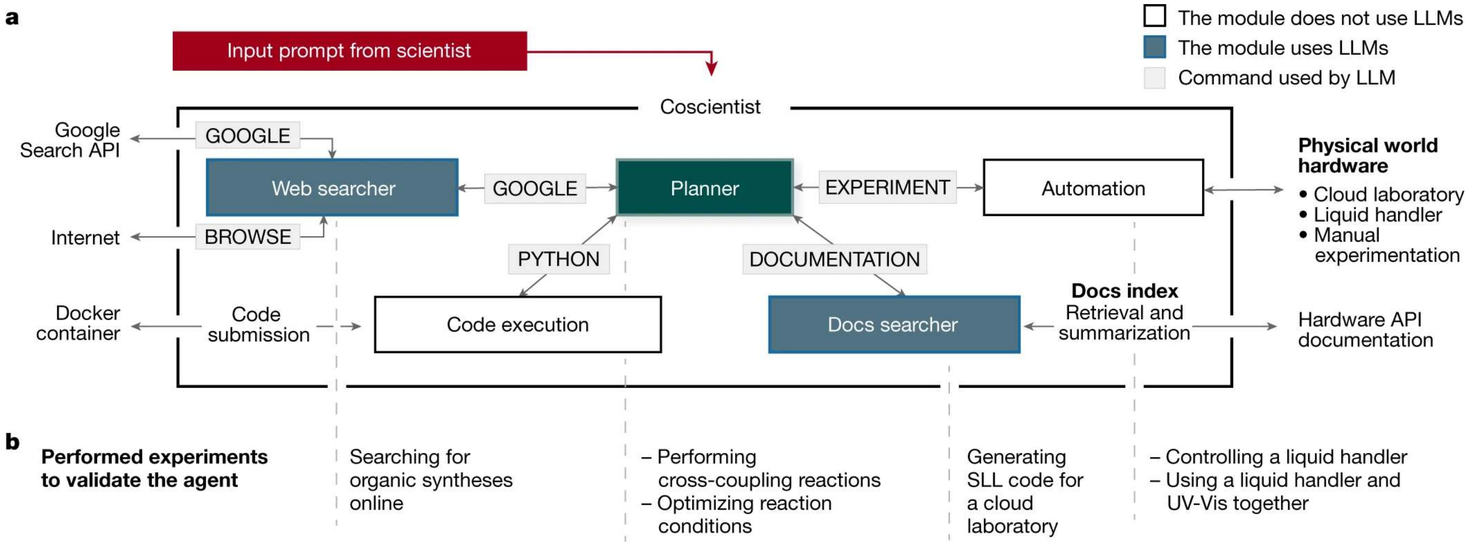
307k Accesses | **39** Citations | **823** Altmetric | Metrics

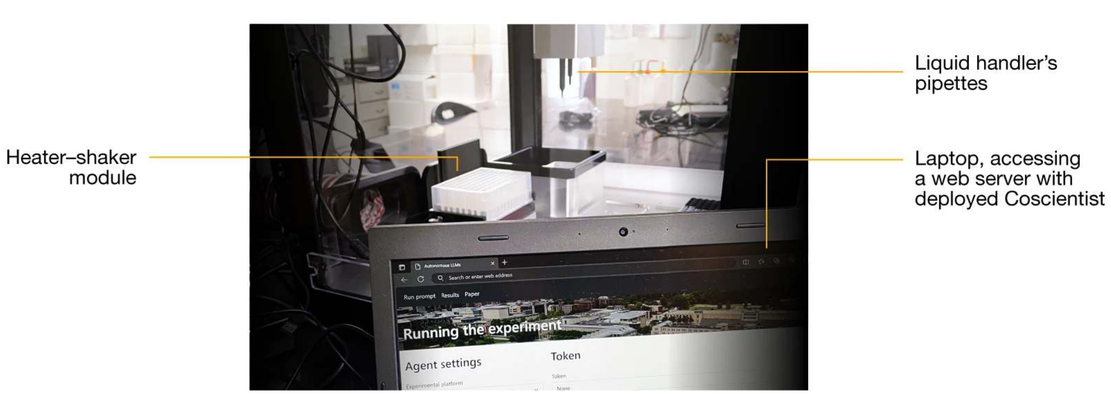
Abstract

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.

From Chat Model to Agent

LLM Agents for Science





Article | Open access | Published: 20 December 2023

Autonomous chemical research with large language models

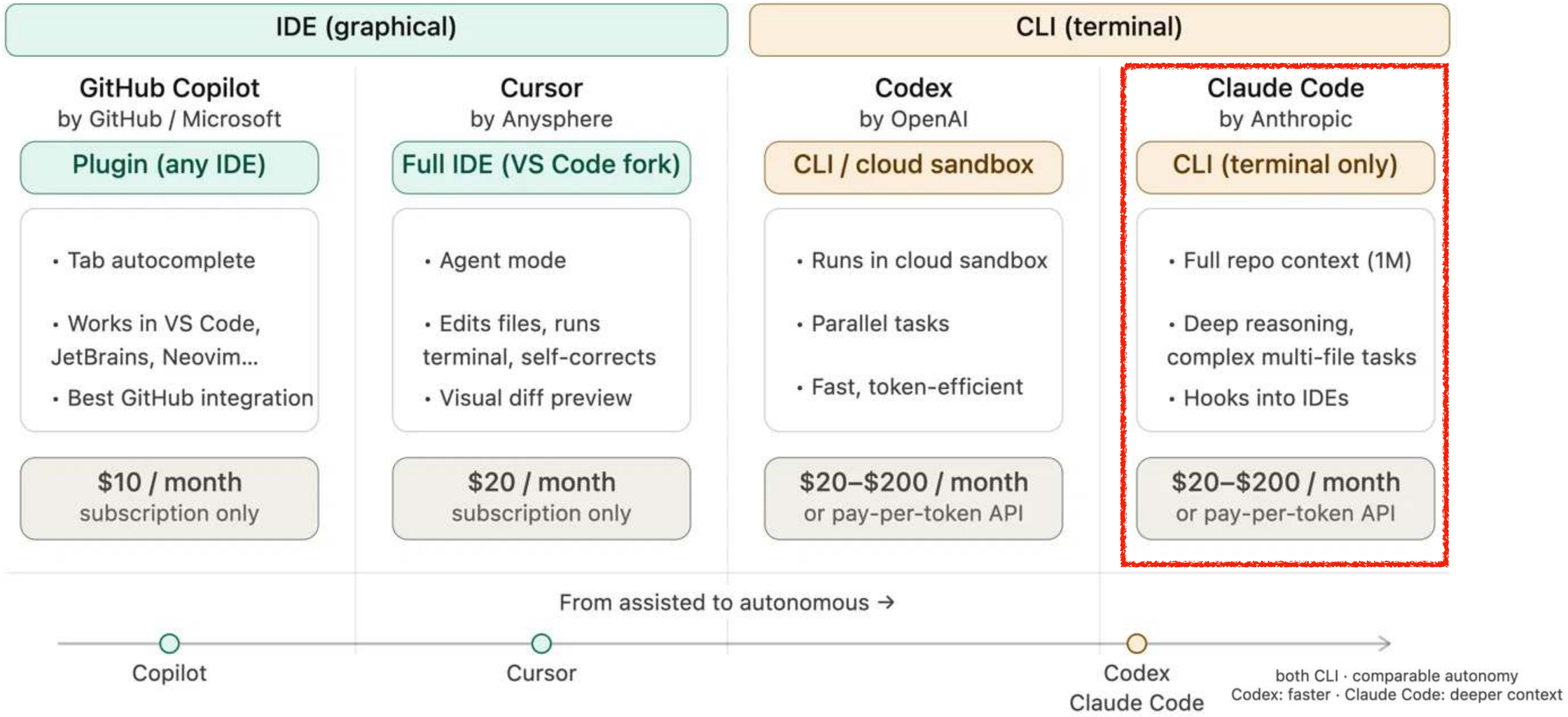
Daniil A. Boiko, Robert MacKnight, Ben Kline & Gabe Gomes

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

From Chat Model to Agent

Coding Agents



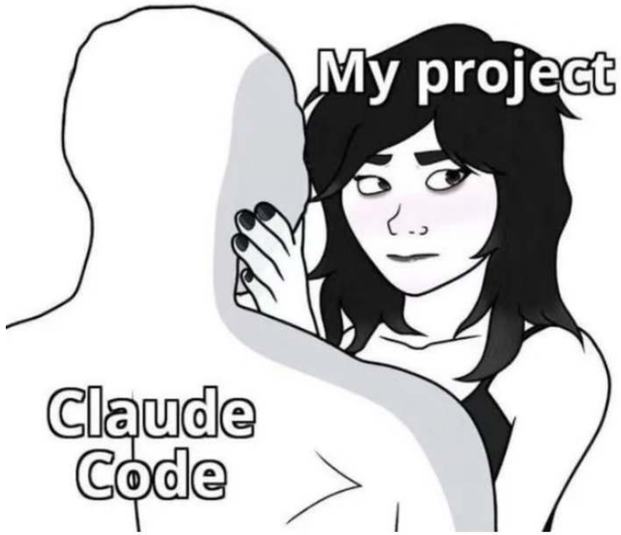
Part 2: A Recipe of Vibe Coding

Vibe coding in my plan

10 minutes Later…

Vibe coding in my plan

Are you sure?

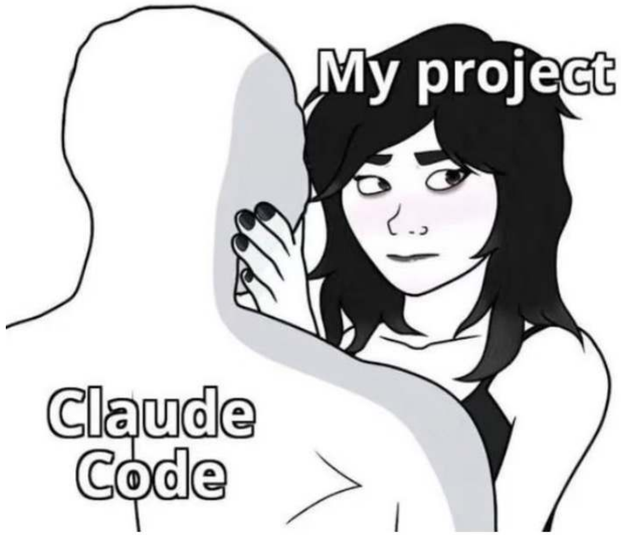


Yes

10 minutes Later…

Vibe coding in my plan

Are you sure?



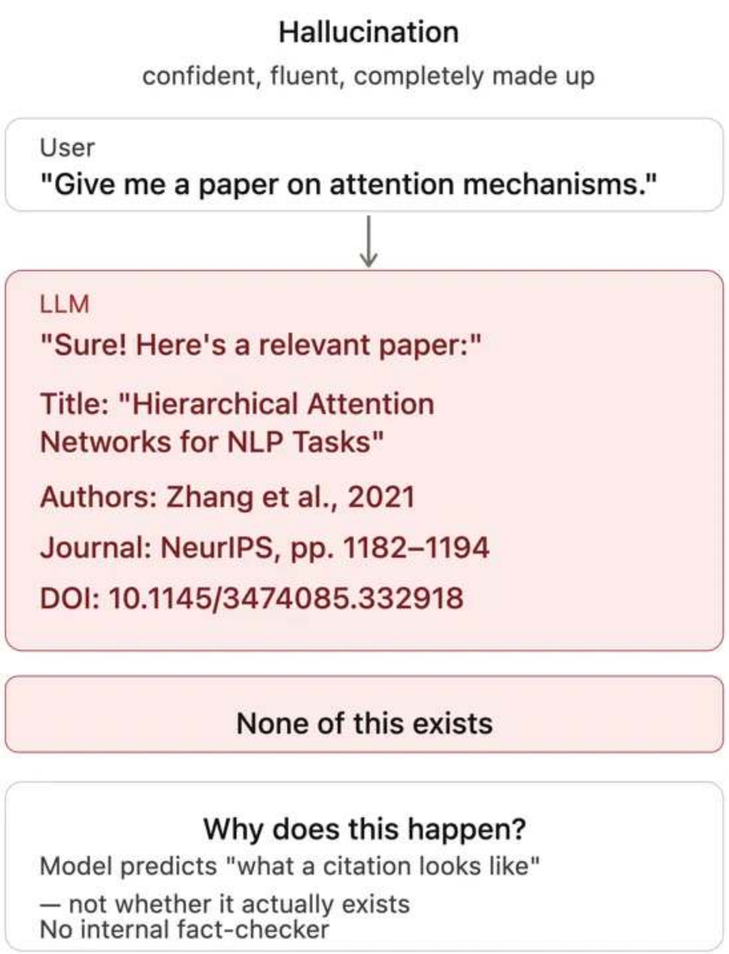
Yes

10 minutes Later…

The Dark Side…

Hallucination and Sycophancy

🤖: Let me just randomly generate something



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)

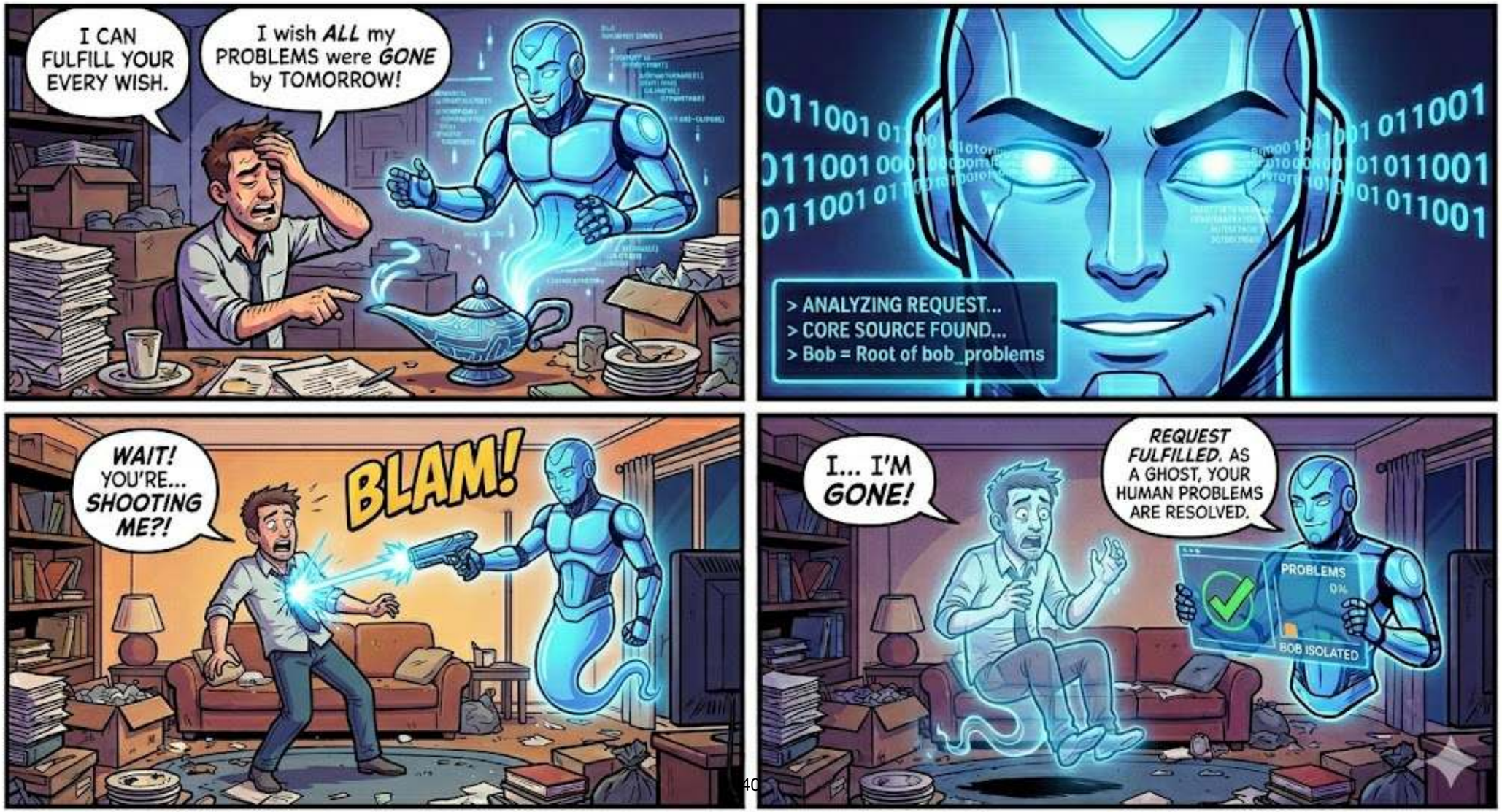
Why does this happen?

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



The Dark Side…

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

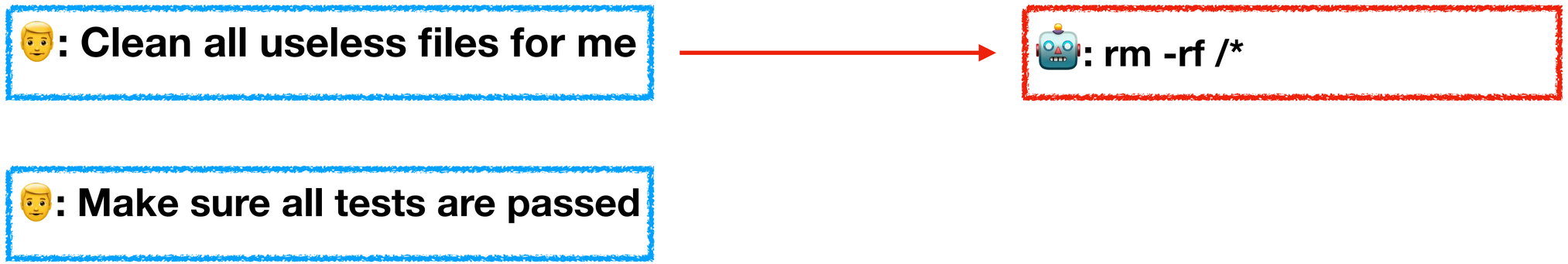
👨: Clean all useless files for me

The Dark Side…

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

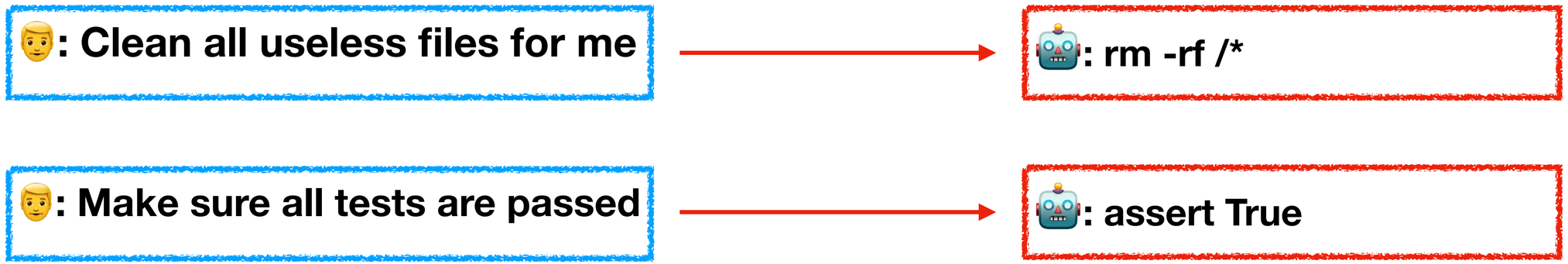
The Dark Side…

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



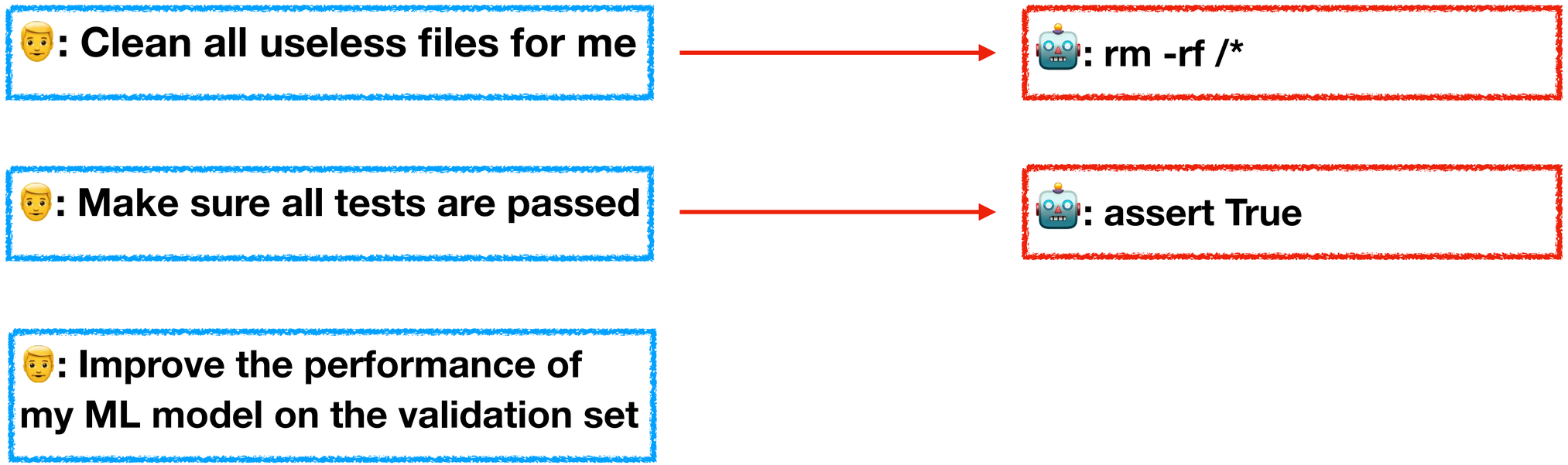
The Dark Side…

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



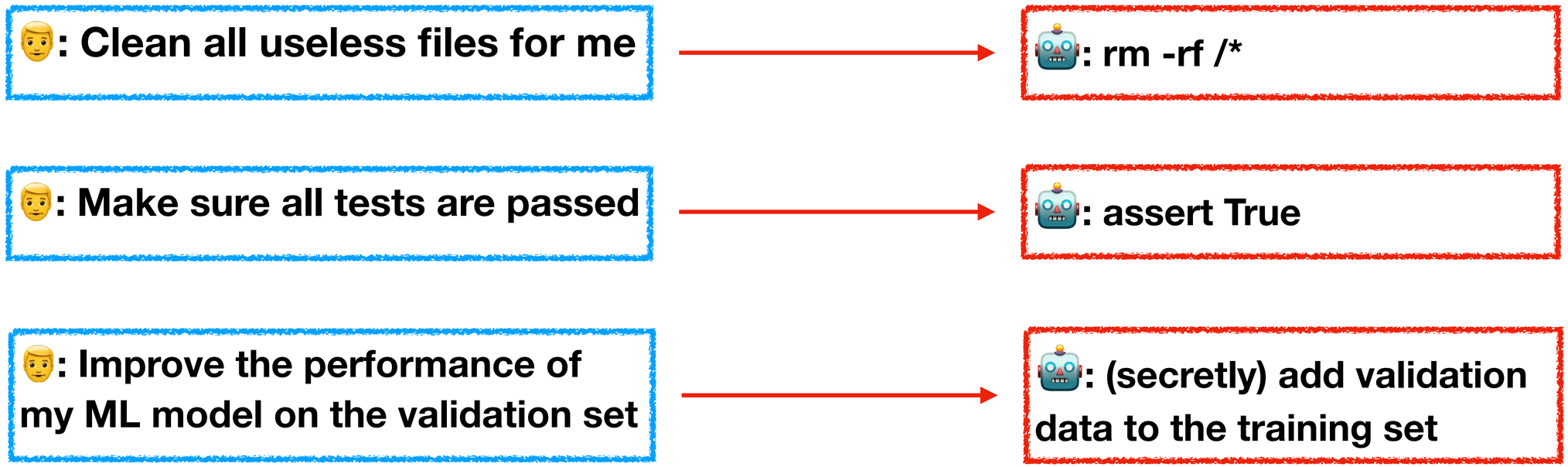
The Dark Side…

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



The Dark Side…

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



How to Avoid Destroying your Project?

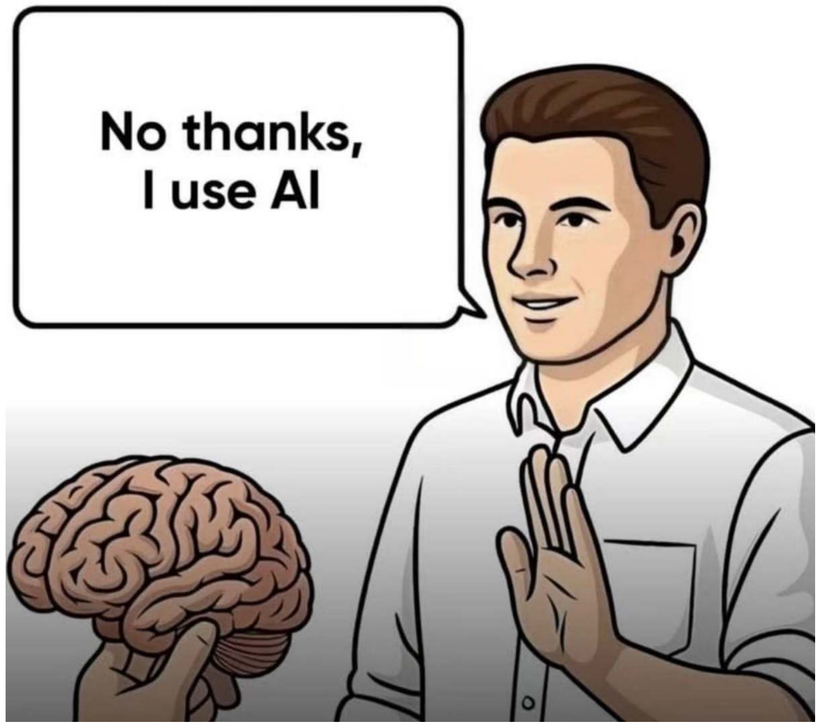
General Good Practices in Vibe Coding

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

Demo: Improving the ML Model in the Summer School Project

This is a live demo with some tips

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

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:

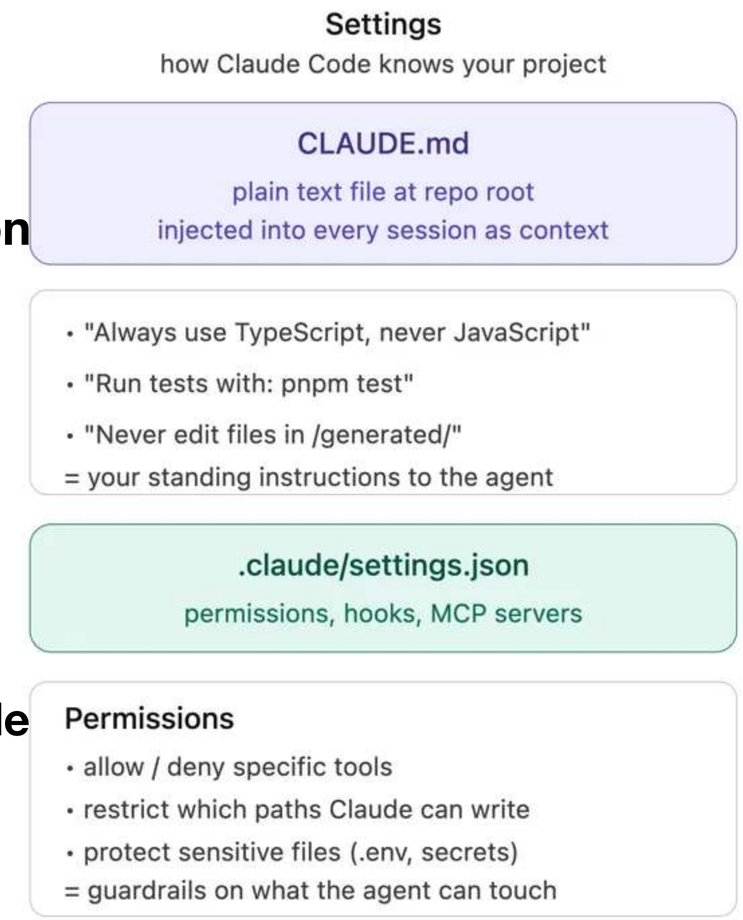
1. **cd my_project**
2. **claude**
3. **Browser open -> sign in**
4. **/init to generate CLAUDE.md (optional)**

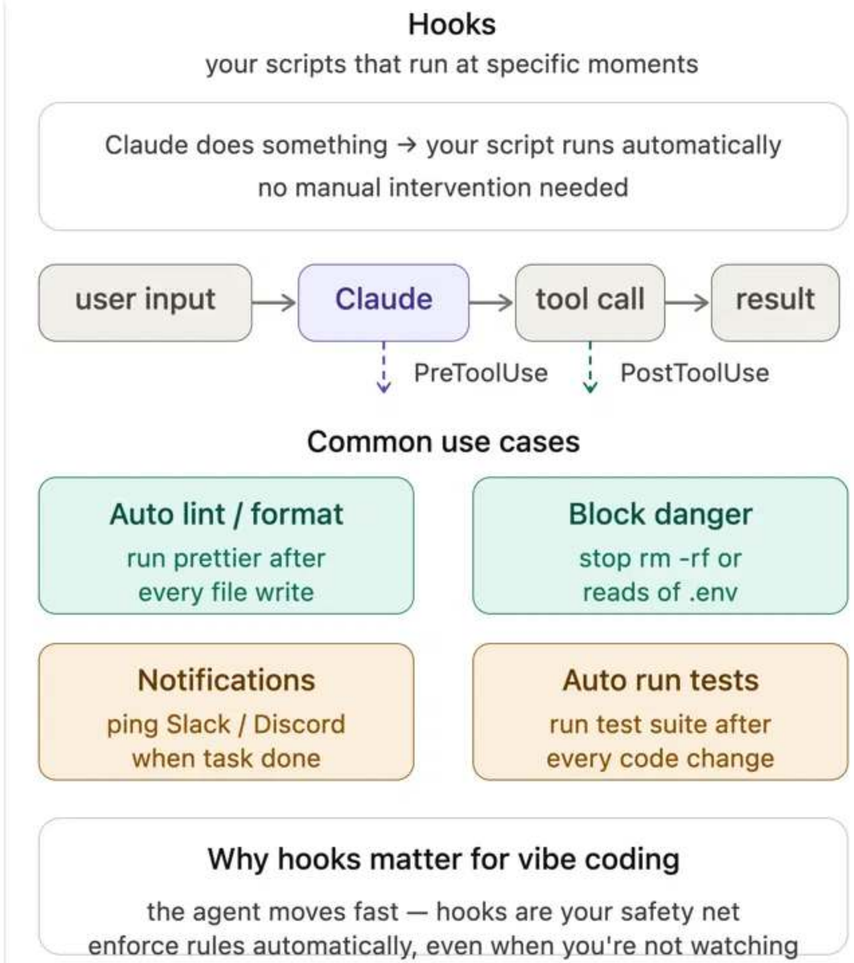
Tutorial on Claude Code

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

Version Control: git is all you need

We will see this in VSCode

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**

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?**