MYKHAYLO.DEV

How does this *AI thing* work

How I explain large language models to family, friends, and anyone curious but not technical.

MYKHAYLO OLEFIRENKO

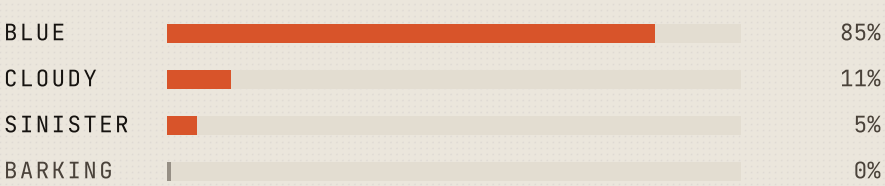
INSIDE

Contents

01

LLMs are *not magic*

- LLMs are statistical models trained to **predict words, one after another**.
- They were trained on a huge amount of human text: basically the digitized text we could scrape (mostly English, mostly web).
- They work by predicting the next word. Write *"the sky is ..."* and the model leans on what it saw humans write next: most often `BLUE`, less often `CLOUDY`, rarely `SINISTER`, never `BARKING`.



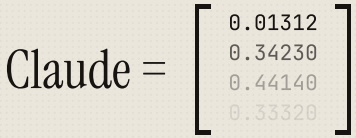
- **Important:** it doesn't always pick the most likely word. It **rolls weighted dice and samples** from the probabilities.
- That's why the same prompt gives different answers every time, and why it can sometimes produce something it has basically seen **0 times**.

02

Tokens

- LLMs run on **tokens**.
- The question you ask is converted into tokens.
- The answer the LLM writes is a bunch of tokens.
- The companies that own the model **bill you by how many tokens** it reads and writes. So understanding tokens explains both how the model "sees" your words and why you pay what you pay.
- Tokens are apparently random fragments that make up the LLM vocabulary (like `"the"`, `"lo"`, `"est"`).
- They're just the set of character chunks that lets the model represent all human text while being **as efficient as possible** in the math behind word prediction.

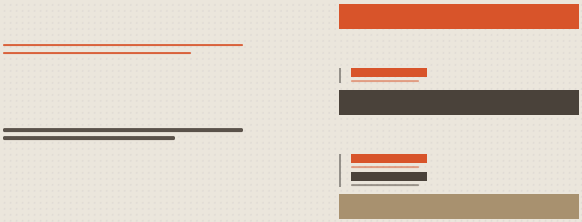
> *You don't pay per word or per question. You pay per token, in and out.*



03

How are LLMs *made*

- A model is just a giant collection of **parameters**: pure numbers. They aren't set by hand, they're **"discovered" during training**.
- **Training is a loop:** each step it measures how wrong it is, computes which way to nudge every parameter, and takes a small step that way. Repeated millions of times until it settles into values that predict well.
- This runs for **weeks or months**, until the engineers decide it's not improving anymore and stop it.
- Parameter count is chosen before training: **small** = faster and cheaper but dumber; **big** = smarter but expensive and slow.
- The models we all use are in the **trillions** of parameters. Loading 1 trillion numbers needs ~1 TB of memory; your home computer has maybe 16–32 GB. That's why we pay for Claude or ChatGPT instead of running them ourselves.
  - **Bigger** (Claude Opus, GPT-5): smarter, better at planning, but slower and pricier (more VRAM).
  - **Smaller** (Claude Haiku, GPT-5 mini): less smart but faster and cheaper, good for summarizing or simple fact retrieval.
  - **Thinking** (Claude extended thinking, OpenAI o-series): reasons step by step before answering, which boosts ability. But that monologue is still billed tokens, so you pay in time and usage.



04

Memory

- The model itself has **NO memory**. It doesn't remember past conversations, or even the earlier messages of this one. It has no concept of a conversation.
- When you send your 5th message, your previous 4 messages and answers are **quietly attached** to it. THAT is the only reason it "follows" along.
- So how does ChatGPT/Claude "remember" you across chats? **It doesn't.** The product saves facts about you (name, job, preferences) and injects them into the context at the start of each chat. You feel remembered; the app is just re-feeding text every time.
- Longer conversation = more messages appended each turn = **token usage grows** alongside the chat.
- Every model has a token limit per turn called the **context window**. Long enough chats hit it, and then it has 2 options:
  - **Ignore** older messages to make space for the latest ones.
  - **Compact** the conversation into a summary, keeping only the key facts.

05

Tools

- The model is **frozen in time**. It was trained up to a certain date and then locked. It knows nothing that happened after that date.
- A bare model only **guesses** from frozen training data. A model **with tools** can actually go act: search the web, read a file you uploaded, run code.
- This is the **counterweight to hallucination and cutoff**: instead of inventing, it retrieves real current information, or runs real code and pastes the result into its context.
- **Practical rule:** if facts or recency matter, attach documents or tell it to explicitly search online or do the math.

> *Tools turn a confident guesser into something that can check its work.*

06

Are LLMs *intelligent?*

- LLMs are mathematical machines that combine tokens based on statistical patterns seen in training. They have **no real understanding** of reality or the laws that govern it.
- They appear intelligent thanks to the humans who understood reality and wrote it down. LLMs achieve **"second-hand intelligence"** by mimicking human language.
- Every time you ask, you get the **most likely-sounding answer**: a blend of what showed up most in training and what human raters later scored as "good." Popularity, shaped by raters.
- The problem stays the same: you don't know where the answer comes from, how many books or articles are mixed together in it, or who is behind it.

> *With AI I could publish a realistic article on "skateboarding during pregnancy improves the baby's coordination" in 10 minutes. One article won't move the needle, but enough junk like it slowly tips what the model treats as "normal."*

07

A better way *to use it*

- Best approach: find **real humans who got real-world results**, formalize their teaching as text files, feed those to the LLM, then ask for personal advice.
- Ask *"how to make $1000 as fast as possible"* and let the model answer from its **own internal knowledge**: it digs through its training data, finds a pile of articles and books by people who made money fast, and **mashes them into one coherent-sounding answer**.
  - The **source is unclear**: who are these people? You can't tell the experts from the scammers, criminals, or liars blended in.
  - It's **many strangers' answers fused together**: a theoretical average with no single owner, possibly with no roots in reality.
- Model it after specific humans instead:
  - The source is **one real human's experience**: known reputation, values, history.
  - It still generates statistically and can drift, but you've biased it hard toward that human's material and made the source **auditable**.

> *The real win is provenance and control: you know exactly whose experience you're pulling from.*

08

Privacy

- Whatever you paste **goes to the provider**. It may be logged and, depending on the product or plan, used to train future models.
- So don't dump **secrets, passwords, or other people's private or client data** blindly. Assume anything you type could resurface somewhere.
- For total privacy you can run LLMs **locally** on your own computer. But the smart models we all love are gigantic, so local means smaller, weaker models.

> *If you wouldn't put it on a billboard, think twice before pasting it.*

09

The question > *the answer*

- How you phrase the question matters a lot, for two reasons: LLMs are great at writing **plausible-sounding text**, and they're inclined to **agree with you**.
- Share your CV and ask:
  - **"Is my CV good?"** biases toward the good stuff.
  - **"Is my CV bad?"** biases toward the bad stuff.
  - **"What do you think of this CV?"** less leading, but still nice by default.
  - **"What would a senior dev think?"** a lens instead of a desired answer. Sharper, less flattery.
  - **"What would make a senior dev reject this CV?"** flips the bias against you and pulls out the truth the default would hide.
- **General rule:** because the default is to agree, the highest-value move is to make it **argue against you**. "Steelman the opposite", "the 3 weakest points", "why would this fail".

IN SHORT

The whole thing, *in one breath.*

It guesses the next word. It has no memory and it's frozen in time. It mimics human intelligence, so feed it real humans' work, give it tools, and ask the question that pulls out the truth. Then check what matters.

Want to learn more, or work with me?

MYKHAYLO.DEV →