

One Word at a Time

How a chatbot is built

Sean Davis, MD, PhD

University of Colorado Anschutz School of Medicine

October 15, 2026



Where are you starting from?

How often do you use a chatbot (ChatGPT, Copilot, Gemini, Claude, or similar)?

- Never, or only once or twice
- A few times a month
- About once a week
- A few times a week
- Every day

Speaker notes

Welcome back. Here’s the plan for today. First we’ll pick up where we left off yesterday, the training loop, and see that chatbots work the same way, just much bigger. Then a walkthrough of getting started with ChatGPT, with the goal of setting up tonight’s learning journal. If you’re comfortable with ChatGPT already, sit back and relax. If you’re not, you don’t have to keep up with every click; the slides stay online and you can work through them afterwards.

The journal is lightweight, and honestly it’s for you as much as for me, more really. My hope is that by the end of the course you think of a chatbot less as a question-and-answer box and more as a colleague, loosely speaking. I use them every day to think through problems and to keep myself accountable. I tell them how I want to work, and they push back when I ask for something that cuts against it. For you, that might be “don’t answer until I’ve told you what I think.”

Before I say anything about chatbots, I’d like to know where the room is starting from. Partly that’s for me: it tells me how fast to go and what to explain on screen. And partly it’s for you. Whatever you answer, you’ll see you’re not the only one, and that matters if you’re the person who has never really used one of these.

How often do you use a chatbot? Any of them counts: ChatGPT, Copilot, Gemini, Claude.

LIVE: Open the poll. About 30 seconds, then read the spread aloud. If many people pick “never”, slow down for Step 1 in the hands-on. For comparison: last year about 20% of students said never or only once or twice, and so did several faculty.

LIVE: Run sheet, about 25 minutes of talk, then about 17 of hands-on. 0 to 3: slides 2 to 4, three quick live questions on how the room uses chatbots, about 45 seconds each. 3 to 5: slide 5, recap, keep it brisk. 5 to 8: slide 6, the word cloud; give it about 45 seconds, then talk over the result; slide 7 ties it to the chatbot, about 30 seconds. 8 to 13: slides 8 to 12; on slide 9 type one drug name live; on slide 11 click through the highlights; on slide 12 press “Pick the next word” four or five times, then slide the temperature to each end. 13 to 16: slides 13 and 14, what the model actually reads and how the transcript goes back every turn, about 90 seconds each. 16 to 22: slides 15 to 20; slide 17 is the animation, about 40 seconds, and it plays on its own; on slide 19 click through the two outputs. 22 to 25: slides 21 to 23; slide 23 is yesterday’s eGFR, asked of a chatbot. Slide 24 is the bridge to the hands-on. Slides 25 to 31 set up the learning journal in a ChatGPT Project, about 15 minutes, done on screen together: slide 26 maps the Project onto what the model reads (1 min), then step 1 (4 min), step 2 (4), step 3 (3), step 4 (5). Slides 32 and 33 are for students afterwards. In 2026 the talk ran about 30 minutes, not 22 (before slides 13, 14 and 26 were added), so the hands-on became a live demo instead: show slides 25 to 27, build the Project in ChatGPT on screen (Steps 1 to 4), and leave the step slides for students to follow afterwards. That worked; plan for it.

LIVE: Before class: open this deck once with ?presenter=YOUR-KEY so the live questions (slides 2 to 4 and 6) open. Turn on Zoom live captions and paste the deck link into the chat.

How confident are you?

How confident are you using a chatbot?

Scale: Not at all · A little · Somewhat · Fairly · Very

Speaker notes

How confident are you using a chatbot, from not at all to very?

I ask because confidence and frequency don't always match. Some people use one every day and still feel they're fumbling; some have barely tried and feel fine about it. Neither is a problem. It just tells me what to slow down for, and it gives you a baseline. Remember your answer; it's worth comparing against at the end of the course.

LIVE: Open the question. About 30 seconds. Last year most people answered “a little” or “somewhat”.

What do you use it for?

When you use a chatbot, what do you use it for **most**?

- Explaining, tutoring, or how-to advice
- Looking things up, like a search engine
- Writing or editing
- Something else
- I don't use one yet

The first three are the top uses in OpenAI's study of 1.5 million ChatGPT conversations: together, nearly 80% of them ([Chatterji et al., 2025](#)).

Speaker notes

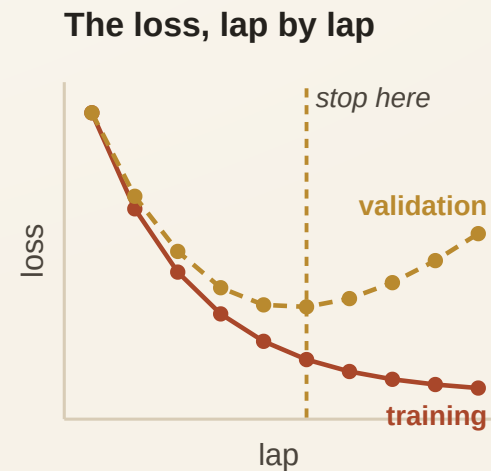
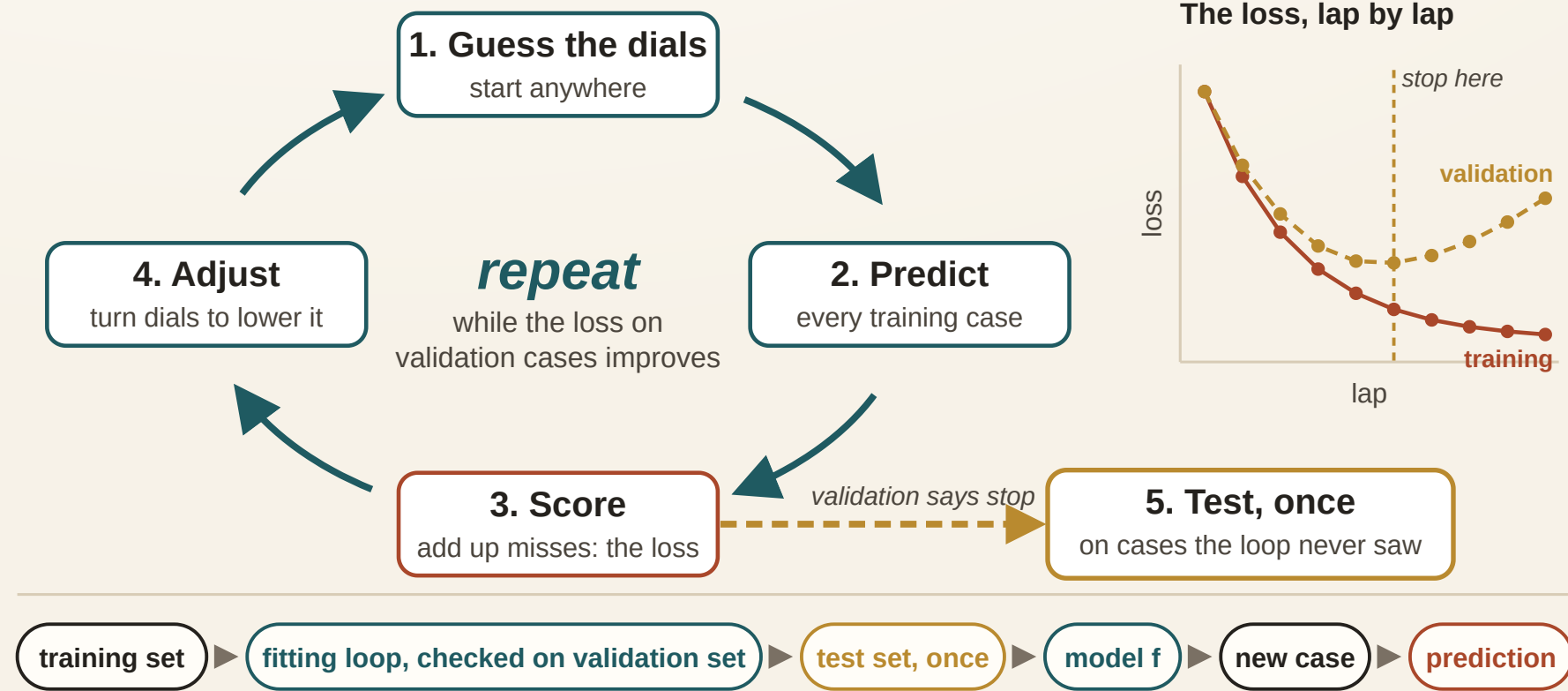
Last one. When you use a chatbot, what do you use it for most? Pick the one that’s most true.

The first three choices aren’t mine. They’re the three biggest categories in a 2025 study by OpenAI and economists who looked at one and a half million ChatGPT conversations: practical guidance, which includes tutoring and how-to advice; looking up information; and writing. Together those were nearly eighty percent of all conversations. So it’s interesting to see whether a room of medical students looks like everyone else.

I also ask because the risks differ. An explanation you’ll check against your notes is fairly low stakes. Using it like a search engine is where “fluent and unchecked” bites hardest, because the answer looks like a fact. Whatever you pick is a good first thing to write about in tonight’s journal entry.

LIVE: Open the poll. About 30 seconds. Then compare the room with the study: in the study, those three categories were nearly eighty percent. Ask whether anyone is surprised.

Yesterday, in one picture



- A **model** turns predictors into a prediction.
- **Fitting** turns its dials until the **loss**, the total miss, is as small as it can get.
- A **neural network** is the same loop with far more dials.
- A score on cases it has already seen **proves nothing**.

Today: the same loop, with words. Then we open ChatGPT Edu.

Yesterday ended with a promise: a chatbot is the same loop, with words. I want to keep that promise in about fifteen minutes, and then we'll spend the rest of the session using one.

Why start with yesterday? Because if the loop makes sense to you, almost everything surprising about chatbots stops being surprising. So, one picture. A model turns predictors into a prediction. Fitting means guessing the dials, scoring the misses with a loss, nudging the dials, and going around again. A neural network is that same loop with far more dials; for a large language model, billions to trillions of them. While we fit, we watch the error on a small held-out validation set, and that's how we spot a model that's too complex for the data we have, one that's overfitting. And a score on cases the model has already seen tells you very little; you need a test set it never saw. [Click]

Today is about one kind of deep learning model, large language models. Image models work much the same way, with images where we'll have words.

Today we feed that loop words. By the end, I'd like you to be able to explain to a classmate how a chatbot gets built, in three stages, and why that recipe makes it behave the way it does when you use it.

Finish the sentence

The patient was started on empiric _____

One word. Whatever comes to mind first.

Speaker notes

Before any theory, I want you to be the model for a minute. I'd rather you feel this than hear me describe it.

Finish the sentence with one word, whatever comes first. The patient was started on empiric what?

LIVE: Open the word cloud. About 45 seconds. Then read the cloud: there is usually one big word, a few medium ones, and a tail of small ones.

PAUSE: 30 s. Say your word out loud, or write it down, before you go on.

Look at what the room made. Not one answer, but a spread: a couple of big words, a few medium ones, and a tail of words only one person picked. Nobody looked anything up. You each predicted the next word from everything you've read and heard on the wards. Hold on to that picture, because it's the whole next slide.

What you just did is what a chatbot does

You each drew on every sentence like this one you have read or heard on the wards. Together, your answers made a **spread**.

A chatbot has learned from **millions of sentences like this one**, and similar ones, and how each was finished. From them it gives every possible next word a **probability**, then picks one.

It doesn't look those sentences up while it answers. Training turned them into its dials, the way yesterday's 8,254 patients became the few numbers in the eGFR.

Speaker notes

That cloud is a pretty good picture of what’s going on inside a chatbot every time it writes a word.

Each of you drew on every sentence like this one you’ve ever read or heard. Nobody consulted a reference. And together, your answers made a spread, with a few likely words and a long tail. [Click]

A chatbot has learned from millions of sentences like this one, and similar ones, and from how each of them was finished. From that it gives every possible next word a probability, a spread like yours, and then it picks one. [Click]

The detail I most want you to keep: it doesn’t look those sentences up while it answers. Training boiled them down into its dials, the same way yesterday’s eight thousand patients got boiled down into a handful of numbers in the eGFR equation. That’s why it can’t tell you where a particular fact came from. The rest of the talk is how that boiling down works.

LIVE: About thirty seconds, straight after the word cloud.

A language model is the same loop, with a different table

the patient was started on empiric vancomycin

Predictors: the words so far	Target: the next word
the	patient
the patient	was
the patient was	started
the patient was started	on
the patient was started on	empiric
the patient was started on empiric	vancomycin

Nobody labels anything. **Every sentence ever written is already a table like this.**

Yesterday’s two questions: the answer is in the data, and the output is a **category**, one token out of a vocabulary of about 200,000. A language model is a **classifier**, run again and again.

This is the trick that made modern chatbots possible, and once you see it, it's hard to unsee.

Take any sentence. The patient was started on empiric vancomycin. Now turn it into a table, the same kind of table we used for the eGFR. The predictors are the words so far, and the target is the very next word. “The” is followed by “patient”. “The patient” is followed by “was”. And so on, down to “the patient was started on empiric”, followed by “vancomycin”. [Click]

Why does this matter so much? Because nobody had to label anything. For the eGFR, someone had to measure GFR in eight thousand people, which is slow and expensive, to get the answer column. Here the answer is already sitting in the text. Every sentence ever written is a set of free training cases. In practice the “words so far” aren’t a single sentence; they can be thousands, even hundreds of thousands, of tokens of preceding text. That’s the reason these models could be trained on a large share of the written internet, and the reason they got so big so fast. [Click]

Now ask yesterday’s two questions. Does the training data contain the answer? Yes, the next word. What does it output? A category: one token out of a vocabulary of about two hundred thousand. So a language model is a classifier, and a chatbot is that classifier run over and over, one word at a time. If you remember one sentence from today, that one would do.

The model reads tokens, not words

A **token** is a word or a piece of a word.

Error

This document uses OJS, which requires JavaScript features disabled when running in file:///... URLs. In order for these features to work, run this document in a web server.

Error

This document uses OJS, which requires JavaScript features disabled when running in file:///... URLs. In order for these features to work, run this document in a web server.

The real tokenizer used by OpenAI's GPT-4o models (o200k_base), from the open-source gpt-tokenizer library. It runs in your browser: nothing you type leaves your computer. Other models split text differently.

Speaker notes

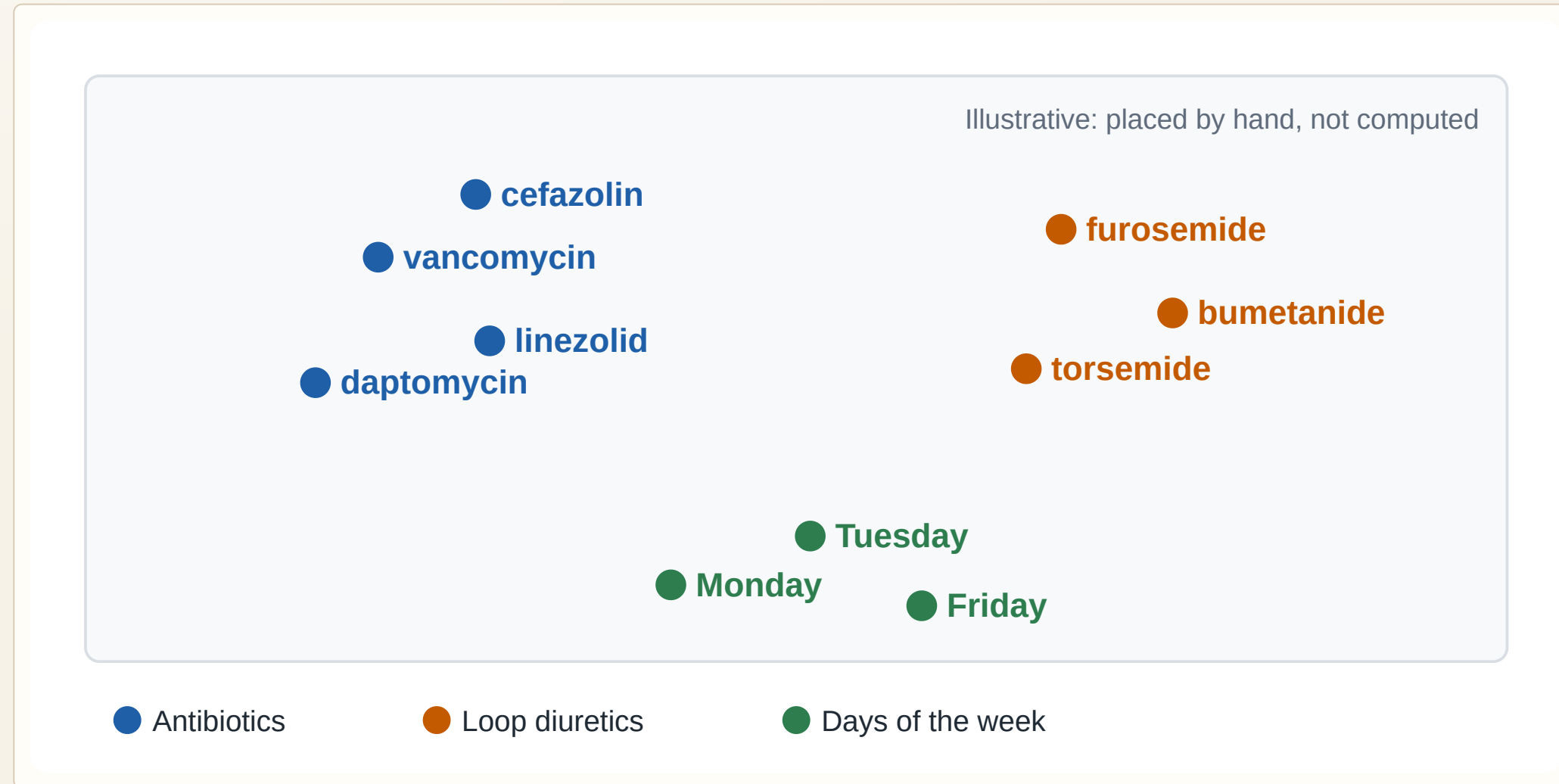
One small detail, because it explains some odd behavior you’ll run into. The model doesn’t actually read words. It reads tokens, which are words or pieces of words.

The box runs a real tokenizer in your browser, the one OpenAI’s GPT-4o models use, and nothing you type leaves your computer. Common words like “on” and “and” get one token each. Drug names get chopped into pieces. That’s part of why a chatbot can be shaky with rare drug names, with spelling, or with counting letters: it never sees the word whole. And when a tool tells you about limits, how much text you can paste in, it’s counting tokens, not words. For the rest of the talk, when I say “word”, I really mean token.

PAUSE: 1 min. Type a made-up sentence of your own and watch it split. Never use one about a real patient.

LIVE: Type one drug name the room suggests, such as a long biologic, and move on. About one minute.

Words become numbers



Illustrative: positions placed by hand to show the idea, not computed from a model.

Each token becomes a list of numbers called an **embedding**. Tokens used in similar sentences get similar numbers.

Speaker notes

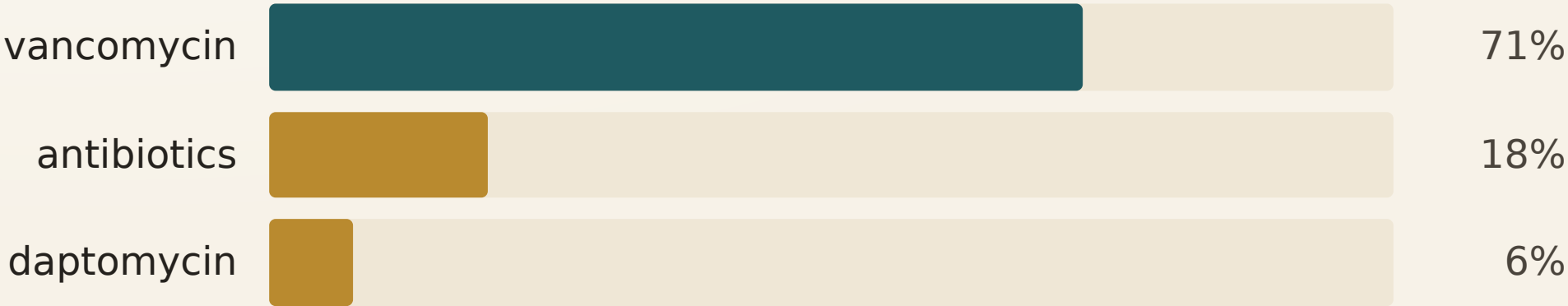
A neural network only takes numbers, so each token gets turned into a list of numbers, called an embedding. Those numbers are dials too. The fitting loop sets them, and tokens that show up in similar sentences end up with similar numbers.

This picture is illustrative; I placed the points by hand. Vancomycin ends up near linezolid because they turn up in the same kinds of sentences. The loop diuretics cluster together. Both groups are a long way from Tuesday. Nobody ever told the model that these are antibiotics. It worked that out from the company the words keep.

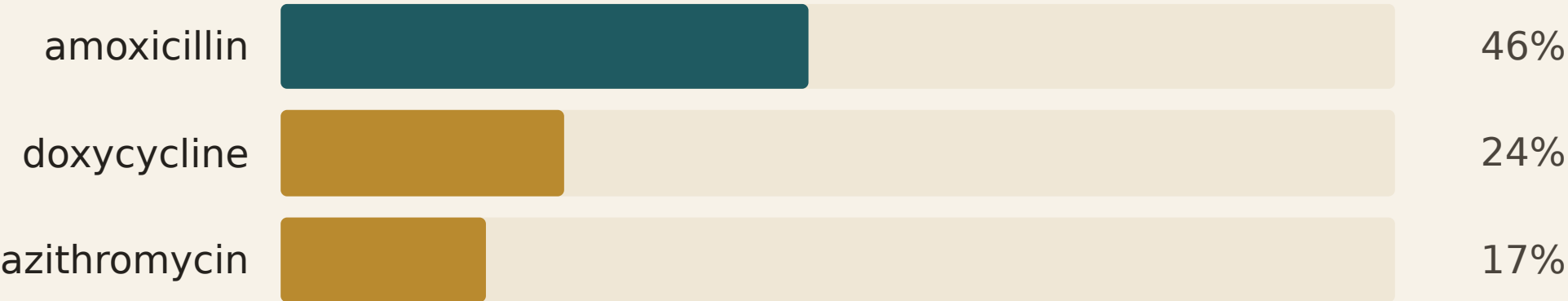
Why should you care? It’s why a chatbot seems to know drug classes and synonyms without anyone teaching it. If vancomycin and cefazolin sit close together, the model can write cefazolin into a sentence where it only ever saw vancomycin. It’s also one reason it can slip a neighbor into a fluent sentence, one antibiotic where another belonged, without the sentence sounding any less sure of itself. The real lists have thousands of numbers, not two.

The words so far: which ones matter?

Fever and hypotension. She has a central line and grew MRSA on her last admission. The patient was started on empiric _____



A healthy 25-year-old with a cough and a lobar infiltrate, sent home. The patient was started on empiric _____



The **same last six words**, the sentence from the word cloud, and a different next word. At each step the model **weighs every earlier word** and decides which matter for the next one. This is **attention**, the core of the *transformer*, the “T” in GPT (Vaswani et al., 2017). Probabilities made up for this slide.

Speaker notes

On the word cloud you only had one sentence to go on. A model sees all the words so far, and that changes the answer. This slide is really about why the context you give a chatbot matters so much.

Two short notes. [Click] Look at how they end: both with the sentence you just finished, the patient was started on empiric, blank. The same six words.

[Click] In the first note, a few earlier words carry the weight: a central line, and MRSA on her last admission. [Click] So vancomycin is far ahead.

[Click] In the second, different words carry the weight: a healthy twenty-five-year-old, a lobar infiltrate, sent home. [Click] Now the likely next words are outpatient antibiotics for pneumonia.

[Click] Same last six words, different next word. At every step, the model weighs every earlier word and decides which ones matter for the word it's about to write. That weighing is called attention. It's the core of the transformer, the 2017 design we met on Tuesday, the “T” in GPT. It's what lets a model use a whole page, or a whole conversation, as its predictors.

And here's the practical payoff, which we'll come back to in the hands-on: whatever you put earlier in the conversation, who you are, what you're trying to do, what you've already ruled out, changes what comes next. A bare question gets the answer for an average patient. The probabilities on this slide are made up to show the idea.

In a chat, “the words so far” is much more than your question. That's the next two slides.

LIVE: About ninety seconds. Click through; don't explain the mechanism beyond “it weighs the earlier words”.

Predict, pick, repeat

Error

This document uses OJS, which requires JavaScript features disabled when running in file:///... URLs. In order for these features to work, run this document in a web server.

Error

This document uses OJS, which requires JavaScript features disabled when running in file:///... URLs. In order for these features to work, run this document in a web server.

Temperature: how willing the model is to pick less likely words. ChatGPT doesn't let you change it, but the pick is still random. Whole words are used here instead of tokens, to keep it readable.

Speaker notes

This is everything a chatbot does when it writes, and it's worth seeing it slowly once. Given the words so far, it produces a probability for every possible next word. One word gets picked, at random but weighted by those probabilities. It goes on the end. Then the whole thing runs again with one more word in the predictors.

PAUSE: 2 min. Press “Pick the next word” until the sentence ends, then start over and do it again.

You get a different sentence each time. Every one of them is smooth, plausible clinical prose, and none of them was checked against anything. That combination, fluent and unchecked, is the thing I most want you to recognise when you use these tools. The probabilities here are made up.

The slider is called temperature. Turn it down and the most likely word wins almost every time. Turn it up and the tail gets picked more often. Higher is what you'd want for creative writing, lower for answers you want to come out the same way every time. ChatGPT doesn't give you this slider. Developers can set it for some models, and the newest reasoning models don't allow it at all. Either way the pick is still partly random, which is why the same question can get a different answer each time you ask. That's also a cheap check you can run yourself: ask twice, and if the answers disagree on something that matters, you've learned something. A real model does this with about two hundred thousand tokens to choose from, looking back over thousands of words, with billions of dials.

LIVE: Press the button four or five times, start over once, then drag the temperature to each end and press again. About two minutes.

What it actually reads

FROM THE COMPANY	You are ChatGPT. Today's date is ... You can search the web ... Be helpful, follow these rules ... <i>(you never see this part)</i>
MEMORY	Notes from earlier chats: medical student, first year; interested in pediatrics.
PROJECT INSTRUCTIONS	You are my learning-journal coach for a 4-week course on AI in medicine ...
FROM YOUR FILES	Wednesday handout: "A score on cases it has already seen proves nothing ..."
YOU	I asked ChatGPT to explain eGFR, then checked its formula against Wednesday's slides.
ASSISTANT	Good check. The slides use the 2021 equation, without race. Did the formula match?
YOU	Draft my entry.
ASSISTANT	← the next word goes here

It predicts the next word of **this whole document**, not of your question. Your message is only the last few words. Illustrative: the real hidden text is longer and isn't published in full.

Speaker notes

A student asked a great question in class: if a sentence starts with just “the”, how does the model know to stay on track? The answer changes how you should think about every chat, so I want to show it.

Here’s what you think the model reads: your message. “Draft my entry.” On its own, that’s hopeless. Draft what entry? For whom? [Click]

But the app doesn’t send just your message. It sends the whole conversation so far, what you said and what it said, laid out as one long document with a label on each turn. [Click]

In a Project, it also adds your project instructions, and pieces of your files that look relevant to what you just asked. [Click]

With memory on, it adds notes about you from earlier chats. That’s why ChatGPT seems to get to know you over time. It isn’t learning; it’s being handed notes. [Click]

And at the very top, instructions from the company that you never see: who it is, today’s date, what tools it has, what rules to follow. [Click]

Then the model does the only thing it knows how to do. It predicts the next word of this entire document, which happens to end with “Assistant:”. It’s still the loop from the last three slides. What changed is the predictors: not your question, but all of this. That’s why “draft my entry” works in your project and would be nonsense in a fresh chat.

The examples are illustrative. The company’s actual hidden instructions are longer and aren’t published in full.

LIVE: About ninety seconds. Click slowly; each layer is one sentence.

Every turn, the whole thing goes back

Turn 1	Turn 2	Turn 3	Week 4 of one long chat
From the company	From the company	From the company	From the company
Memory · project · files	Memory · project · files	Memory · project · files	Memory · project · files
You: I'm studying renal ...	You: I'm studying renal ...	You: I'm studying renal ...	You: (week 1) ...
	Assistant: Try asking it ...	Assistant: Try asking it ...	Assistant: (week 1) ...
	You: Was my check good?	You: Was my check good?	You: (week 2) ...
		Assistant: Good check ...	Assistant: (week 3) ...
		You: Draft my entry.	You: (week 4) ...
			Assistant: (week 4) ...
			You: Draft my entry.

context window limit

The model is **the same at every turn**: your chat doesn't turn its dials. The app sends the whole transcript again each time. A long chat can outgrow the **context window**, and older parts get dropped. Whether your chats train a *future* model depends on the terms: personal ChatGPT may, unless you opt out; CU's agreement rules it out for ChatGPT Edu.

Speaker notes

Now watch what happens over a conversation. [Click]

Turn one: the hidden instructions, your project and memory, and your first message. [Click]

Turn two. You might think the model remembers turn one. It doesn't. The app sends everything again: turn one, its answer, and your new message. [Click]

Turn three: the whole thing again, a little longer. Every time you press Enter, the entire transcript goes back to the model from the top.

Here's the part that surprises people. The model itself never changes during your chat. Nothing you type turns its dials; those were set in training, months ago. All the "memory" in a conversation is the transcript being sent again. [Click]

And that has a limit, called the context window. It's large these days, but a chat that runs for four weeks can outgrow it, and then something has to give. The oldest parts get dropped or squeezed down, and the details from week one quietly drop out. You won't get a warning. [Click]

One distinction worth getting right. Your chat doesn't change the model today. Whether it gets used to train next year's model is a separate question, and the answer is in the terms of service, not the technology. On a free or personal ChatGPT account, OpenAI may use your conversations for training unless you turn off "Improve the model for everyone" in the data controls. CU's agreement for ChatGPT Edu rules that out for our workspace. Which is a good reason to actually read the terms of the tools you use; I'll point you to them in the hands-on.

Two practical things follow. Start a new chat when you start a new task, so old material doesn't steer the new one. And if something early in a long chat matters, say it again, or put it in the project instructions, which go in every time. That's exactly why we'll keep one chat per week in the hands-on.

LIVE: About ninety seconds.

PART 2

How a chatbot is built

Speaker notes

So that's what the model does when you use it. Now, how does it get made? I think this part matters more than it sounds, because each stage of the recipe leaves fingerprints on how the tool behaves. Three stages.

Three stages

1 · PRETRAINING

Reads: trillions of words of web pages, books, articles, and code.

Learns: to continue any text.

Gives: a *document completer*.

Almost all of the cost.

2 · INSTRUCTION TUNING

Reads: thousands of example conversations, **written by people**: a request, then a good answer.

Learns: what a helpful reply looks like.

Gives: something that answers.

3 · PREFERENCE TUNING

Reads: people's choices between **pairs of answers**.

Learns: to give the kind of answer people prefer.

Gives: the assistant you use.

All three are the fitting loop. Only the table changes.

Speaker notes

The whole recipe on one slide first, then we'll take the stages one at a time. [Click]

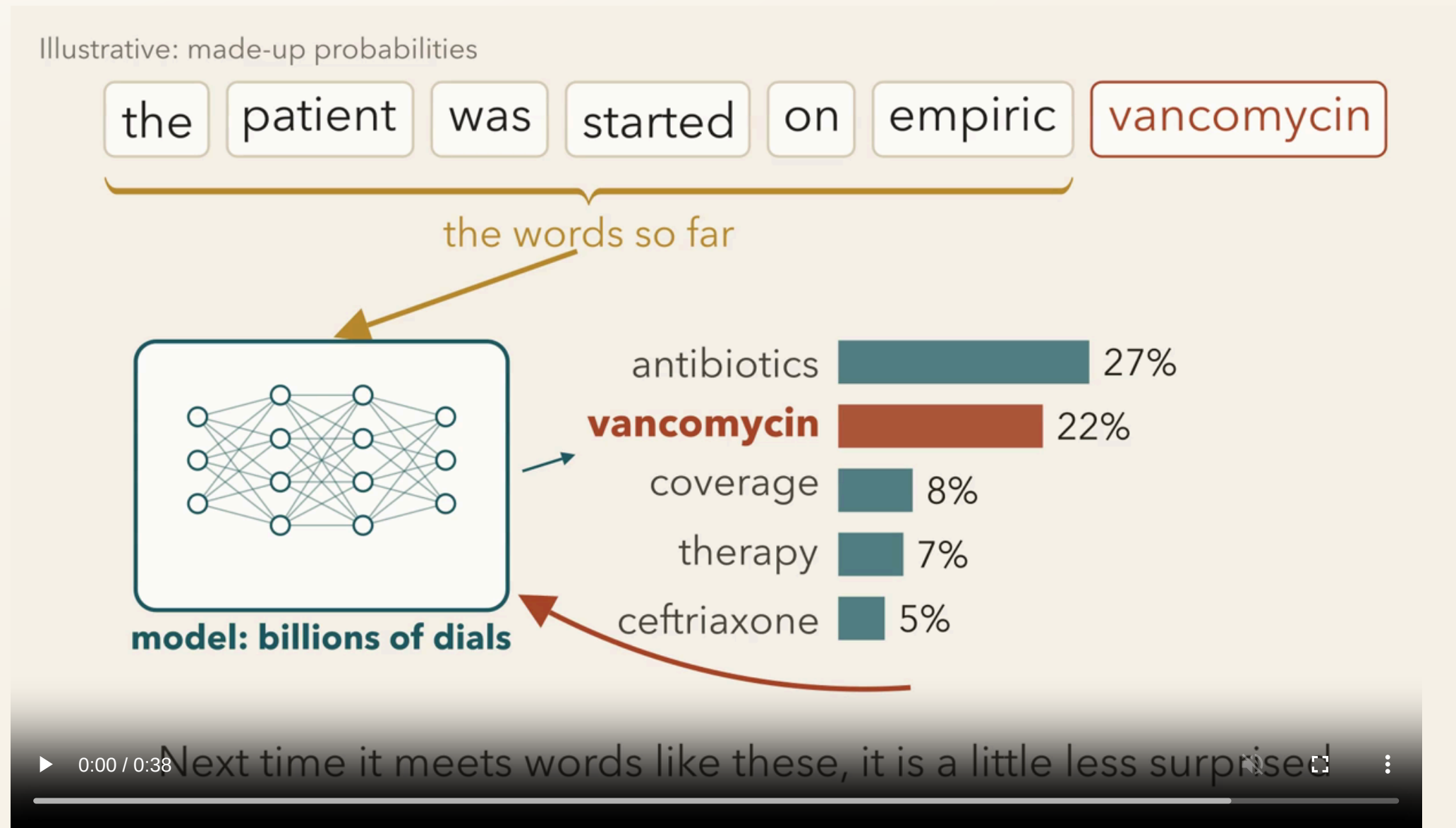
Stage one is pretraining. The model reads trillions of words, from web pages, books, articles, and code, and learns to continue any text. What comes out is a document completer, and this stage is almost all of the cost. It's where the knowledge comes from, and also the gaps and the old information. [Click]

Stage two is instruction tuning. People write thousands of example conversations, a request and then a good answer, and the model is trained on those with the same next-word loop. It learns what a helpful reply looks like. This is where the shape of the answers comes from. [Click]

Stage three is preference tuning. People are shown two answers to the same request and pick the better one, and the model is nudged toward the kind that wins. If ChatGPT has ever shown you two answers and asked which you prefer, that's the kind of data this stage uses. This is where the tone comes from, including the confidence. Now you have the assistant you actually talk to. [Click]

All three are yesterday's fitting loop. Only the table changes. Keep that in mind and the rest of the talk is mostly consequences.

Pretraining: one lap of the loop



The **loss** is yesterday's idea, for words: the less probability the model gave the word that actually came next, the bigger the loss. Made-up probabilities. Scale from Llama Team, AI @ Meta (2024).

This is one lap of pretraining. It plays on its own; I'll talk over it.

A training case: the words so far, with the next word hidden. They go into the model with its billions of dials, and out comes a probability for every word it knows. Then the answer is revealed. The next word was vancomycin, and the model only gave it twelve percent. The loss is how surprised it was. That's yesterday's loss, for words: the less probability it gave the word that actually came next, the bigger the loss. Then every dial gets turned a little, in the direction that would have made vancomycin more likely. Next time it meets words like these, it's a little less surprised.

Then the next word, and the next, through clinical notes, prescriptions, history, computer code, other languages. For one model from 2024, about fifteen trillion tokens.

Notice what the loop rewards: predicting the next word well, and nothing else. To predict the next word of a pharmacology textbook, it helps to have picked up some pharmacology. That's where the knowledge comes from, and honestly I still find it a little remarkable that it works. But nothing in the loop asks whether the text was true. A wrong forum post and a correct guideline are both just text to predict, and the model learns from both.

LIVE: Let it play, about forty seconds. Say the last paragraph after it ends. Hover over the video for its controls: pause, drag the bar to rewind, or play again from the end. Click back on the slide, outside the video, before using the arrow keys.

How big is “trillions”?

	Dials (parameters)	Training data
eGFR, CKD-EPI 2021	fewer than 10	8,254 people
GPT-3, 2020 (Brown et al., 2020)	175 billion	300 billion tokens
Llama 3, 2024 (Llama Team, AI @ Meta, 2024)	405 billion	about 15 trillion tokens, on up to 16,000 chips

Reading 15 trillion tokens at 250 words a minute, 8 hours a day, would take a person about **a quarter of a million years**.

Our arithmetic: a token is about $\frac{3}{4}$ of a word. The companies behind ChatGPT, Claude, and Gemini don’t publish these numbers for their current models.

Speaker notes

A sense of scale, because “trillions” stops meaning anything after a while. The eGFR has fewer than ten numbers in it, fit to about eight thousand people. GPT-3, an ancestor of ChatGPT from 2020, had a hundred and seventy-five billion dials and was trained on three hundred billion tokens. Llama 3, an open model from Meta in 2024, has four hundred and five billion dials and was trained on about fifteen trillion tokens, using up to sixteen thousand specialised chips at once. [Click]

If you read fifteen trillion tokens at two hundred fifty words a minute, eight hours a day, it would take you about a quarter of a million years. That’s my arithmetic, not the paper’s.

It’s also why building these is so expensive and uses so much energy, and why data centers keep coming up in the news. Good dinner conversation, if nothing else.

Why does the scale matter to you? Two reasons. Only a handful of companies can afford to build these, so a few organisations decide what goes into the training text. And the companies behind ChatGPT, Claude, and Gemini don’t publish these numbers, or the details of the training data, for their current models. So for the tools you’ll use today, yesterday’s first question, what was it trained on, often doesn’t have a detailed answer. That’s not a reason to avoid them. It’s a reason to check them.

LIVE: If asked about the energy footprint of using a chatbot: use it deliberately, and when a plain search does the job, search. Long answers cost more than long questions, since every word it writes is another lap of the model (API prices for output tokens run several times those for input), so asking for short answers helps. A good journal topic: how could I measure my own use? Hospitals and universities have AI committees; resident representatives who bring evidence on this are rare and useful.

Same prompt, before and after

Illustrative: both outputs made up for this slide, to show the difference. Not a real model's output.

Prompt: *What is the first-line treatment for community-acquired pneumonia in a healthy adult outpatient?*

PRETRAINING ONLY: IT CONTINUES THE TEXT

What is the first-line treatment for community-acquired pneumonia in an adult with COPD?

What is the first-line treatment for hospital-acquired pneumonia?

What is the first-line treatment for aspiration pneumonia?

Chapter 12 review questions: answers on page 418.

AFTER STAGES 2 AND 3: IT ANSWERS

For a healthy adult outpatient with no comorbidities, US guidelines recommend **amoxicillin**. **Doxycycline** is an alternative, or a **macrolide** where pneumococcal resistance to macrolides is low (Metlay et al., 2019).

Neither is broken. Online, a list of questions is often followed by **more questions**, so the pretrained model continues the document it thinks it is in. Stages 2 and 3 teach it that a question is followed by an answer.

Speaker notes

Here’s the same prompt typed into one model at two points in its training. Both outputs are made up to show the difference, but each is typical of what that stage does.

The prompt is an ordinary clinical question: first-line treatment for community-acquired pneumonia in a healthy adult outpatient. [\[Click\]](#)

On the left, the model after pretraining only. It doesn’t answer. It writes three more questions and then a line from a textbook. [\[Click\]](#)

On the right, the same model after the two later stages. It answers: amoxicillin, with doxycycline or a macrolide as alternatives, which matches the US guideline. [\[Click\]](#)

Neither one is broken. The pretrained model is doing exactly what it was trained to do, which is continue the document. Online, a list of questions is often followed by more questions, like the review sheet at the end of a chapter, or the Step 1 and Step 2 question banks you know well, so it guesses it’s in one of those and keeps going. It has no idea anyone wants help. The next two stages close that gap by teaching it that a question is followed by an answer.

I show you this because it’s easy to assume the helpful, conversational voice is the model’s nature. It isn’t. It’s a layer trained on top. This also isn’t a hallucination: nothing on the left is false, it just isn’t an answer. And you’ll never meet the left-hand model in ChatGPT; every chat app you use has been through all three stages. Part of why chatbots have become so much better as assistants is bigger models, and part is that these later stages have gotten better too.

Two more stages make an assistant

2 · INSTRUCTION TUNING

- People **write example conversations**: a request, then the answer they would want.
- The same next-word loop, on this new, much smaller table.
- Now a question is followed by an answer.

3 · PREFERENCE TUNING

- The model writes **two answers**; a person picks the better one.
- The model is adjusted toward the preferred kind: **reinforcement learning from human feedback** (Ouyang et al., 2022; Thirunavukarasu et al., 2023).

In one study, people preferred the answers of a model **100 times smaller** that had these two stages over the full-size model without them (Ouyang et al., 2022).

It is trained toward answers people **rate well**.

That is not the same as answers that are **true**.

| Telling people what they want to hear has a name: **sycophancy**.

Speaker notes

Stage two, instruction tuning. People write example conversations, a request and then the answer they’d want to get, and the model is trained on those with the same next-word loop. The table is tiny compared with the internet, and it’s written by people on purpose. Afterwards, a question gets an answer instead of more questions.

Stage three, preference tuning. The model writes two answers to the same request, and a person picks the better one. The model gets adjusted toward the kind that wins. The literature calls this reinforcement learning from human feedback. [Click]

These two stages punch well above their size. In the 2022 paper that described this recipe, people preferred answers from a model a hundred times smaller that had been through these stages over the full-size GPT-3 that hadn’t. Keep that in mind for our last week, on responsibility: the biggest model isn’t always the best choice, for you or for the environment. A model a hundred times smaller costs far less to build and run. [Click]

And here’s the catch, and it’s the reason I spend time on this stage. The model is trained toward answers people rate well, and that isn’t the same as answers that are true. A confident, complete-sounding answer tends to beat a cautious, partial one in those comparisons, whether or not it’s right. People also tend to like being agreed with. So when a chatbot sounds sure, or goes along with the diagnosis you suggested, some of that is training, not evidence. [Click] The usual name for the going-along part is sycophancy: telling us what we want to hear. Push back on it and see what happens.

LIVE: If asked about newer “reasoning” models: they add another training stage, practising on problems whose answers can be checked automatically, such as math and code, and they write out their working before answering. Same loop, another table.

The same loop, two tables

	The eGFR	The model behind a chatbot
Predictors	creatinine, age, sex	all the tokens so far
Target	measured GFR	the next token
Output	a number: regression	one token of ~200,000: classification
Training set	8,254 adults in 10 studies	trillions of tokens of text, then examples and ratings from people
Dials	fewer than 10	hundreds of billions
Loss	how far the estimate misses	how surprised it was by the true next token
Test set	4,050 people in 12 other studies	benchmark questions, <i>if they were not in the training text</i>

Speaker notes

This slide is the bridge in one place, and it’s here so you can carry yesterday’s appraisal questions straight over to chatbots. Read it down the columns.

The eGFR takes creatinine, age, and sex; the chatbot takes all the tokens so far. The eGFR predicts measured GFR; the chatbot predicts the next token. One is a regression, the other a classification. The eGFR was fit to about eight thousand people; the chatbot to trillions of tokens, then to examples and ratings from people. A handful of dials against hundreds of billions. The loss is a miss in one case and surprise in the other.

The row I’d point you to is the last one. The eGFR was tested on about four thousand people from twelve other studies. A chatbot gets tested on benchmark questions, often licensing-exam questions. But if those questions and their answers were posted online, they may well be in the training text. Then the impressive score is yesterday’s mistake all over again: a score on cases it has already seen.

What the recipe predicts

- **It sounds the same when it is wrong.** Nothing in any stage checks whether a claim is true. A made-up citation and a real one are both plausible ways to continue “the original paper was”.
- **It can answer differently each time.** The next word is picked at random, weighted by probability.
- **It knows its training text, up to a date,** unless the app searches the web and adds the results to the words so far. The training population is part of the model, as with the eGFR: what is common online is what it knows best.
- **A high exam score needs yesterday’s question:** *what was it tested on?*

Useful, often right, and never checked.

Checking is your job.

Speaker notes

Put the recipe together and it predicts, pretty well, how these tools behave. I find that reassuring, because it means the behavior isn't mysterious. [Click]

First, it sounds the same when it's wrong. Nothing in any of the three stages checks whether a claim is true, and the model itself doesn't look anything up. A made-up citation and a real one are both plausible ways to continue “the original paper was”. [Click]

Second, it can answer differently each time, because the next word is picked partly at random. Ask twice and compare; it's the cheapest check you have. [Click]

Third, it knows its training text, up to a date. The training population is part of the model, just as with the eGFR. What's common online, it knows well. What's rare, recent, or local, like your hospital's antibiogram, it may not know at all, and it won't necessarily tell you that. Ask it about yesterday's news with search turned off and it can't help you. That was everyday life with chatbots in 2023. Now the app can search the web and paste what it finds into the context, the words so far, which helps, but the search results then need checking too. [Click]

Fourth, when you read that a model passed a licensing exam, ask yesterday's question. What was it tested on, and could it have seen those questions before? [Click]

So: useful, often right, and never checked. I don't want you leaving with the idea that it's usually wrong. On common, well-documented questions, the most plausible continuation is often the fact. But the error rate isn't zero, and the tone doesn't change when it's wrong. So the checking is your job. You can ask the chatbot to check its own answer, and that helps up to a point, but the final check is a human's. The rest of this course is largely about doing that well.

LIVE: About two minutes.

A chatbot asked about yesterday's eGFR

- In 2023, researchers asked four chatbots: ***“How do I calculate eGFR?”*** Each question was asked **five times** (Omiye et al., 2023).
- Both ChatGPT-3.5 and GPT-4 had runs that **promoted the race-based equation**, with false claims about muscle mass in Black people.
- The problem answers **did not appear in every run**. Some models **made up equations**.

The training population, a random pick, and no truth check:
all three, in answers about a number you used yesterday.

The race-free CKD-EPI 2021 equation was the national recommendation from 2021. A lot of older text online still describes the race term. Today's models may answer differently; the way to know is to test them.

Speaker notes

One study puts all of that together, and it happens to be about the number from yesterday. [Click]

In 2023, researchers asked four chatbots “How do I calculate eGFR?”, and asked each question five times. [Click]

Both ChatGPT-3.5 and GPT-4 had runs that promoted the race-based equation, the one the 2021 refit removed. Some runs justified it with false claims about muscle mass in Black people. [Click]

The problem answers didn’t show up in every run. And in some answers, the models simply made up equations. [Click]

That’s all three predictions at once. The training population: years of older text describing the race term outweighed one change in 2021. The random pick: whether you got the harmful answer depended on which run you happened to get. And no truth check: an invented equation reads exactly like the real one.

Today’s models may well do better on this question, and I’d expect them to. But the only way to know is the way we talked about yesterday: test it, on cases that matter to you, more than once.

LIVE: About ninety seconds. Then go straight to the next slide and the hands-on.

Now: centaur or cyborg

Open **ChatGPT Edu**.

As we work, watch for what the recipe predicts:

- Ask the **same question twice**. Do you get the same answer?
- When it gives a **source**, open it. Does it say what the answer says?
- Notice the **tone** when it is right, and when it is wrong.

Speaker notes

That’s the theory. Now let’s use the thing, because reading about these tools teaches you much less than an hour of using them with your eyes open.

As we work, watch for what the recipe predicts. Ask the same question twice and compare. When it gives you a source, open it and see whether it says what the answer says. And notice the tone, both when it’s right and when it’s wrong.

LIVE: Go straight into Part 3: about fifteen minutes to set up the learning journal, done on screen together.

PART 3

Your learning journal, inside ChatGPT

Speaker notes

Now we'll set up the place your learning journal lives for the next four weeks: a Project in ChatGPT Edu.

Why keep the journal inside ChatGPT at all? Two reasons. Reflection is where a lot of learning actually sticks, and a coach that asks you good questions makes the reflection easier to do and harder to skip. And you'll be practising the exact skills we just talked about, giving context and checking answers, every week, on something low-stakes. I'll do every step on screen. If you're watching the recording, pause after each slide and do the step.

Your Project writes most of what it reads

FROM THE COMPANY	You are ChatGPT ...	set by OpenAI
MEMORY	Notes from your other chats	Step 1: Default or project-only
PROJECT INSTRUCTIONS	You are my learning-journal coach ...	Step 2: the coach
FROM YOUR FILES	Relevant pieces of the handouts	Step 3: the handouts
EARLIER TURNS	This week's updates and replies	One chat per week
YOU	Your new message	Step 4: context, then ask
ASSISTANT		the model

Most of the words so far are yours. Write them on purpose.

Here’s the document from earlier, the one the model actually reads, and this is why the hands-on looks the way it does. Every step we’re about to do writes one layer of it.

The top layer is the company’s, and you can’t touch it. Memory you choose when you create the project, in step one. The project instructions are step two: the coach. Pieces of your files are step three: the handouts. The earlier turns are your updates through the week, and keeping one chat per week keeps that layer short enough to stay useful. And your message is step four: give it context, then ask. [Click]

So most of the words so far are yours. A chatbot with good context isn’t a smarter model; it’s the same model reading a better document. Write it on purpose.

LIVE: About one minute. This is the why for all of Part 3.

One Project for the whole course

INSTRUCTIONS

Turn the chatbot into a **journal coach**: it knows the five journal questions and your **goals**, asks you questions, and drafts your entry *in your words*.

FILES

The **handouts** from the talks: slides plus speaker notes. Add each week's as you go.

ONE CHAT A WEEK

Week 1, Week 2, ... The chat *is* your log: short updates through the week, by voice if you like. Then *“draft my entry; ask me what's missing first”*, and paste it into Canvas.

Why not one chat for all four weeks? Remember **“Every turn, the whole thing goes back”**: a month-long chat can outgrow the context window. Keep each chat to a week.

A bonus: with the handouts in the Project, you can ask about the lectures too. *“I'm reviewing Wednesday's talk. Using that handout, quiz me one question at a time on the six words, and point me to the slide when I'm wrong.”*

Speaker notes

The design first, so the steps make sense. One Project for the whole course, with three parts.

The instructions turn the chatbot into a journal coach. They hold the five journal questions and your learning goals, so every chat in the Project starts out knowing both, and you don’t have to explain yourself from scratch each time. The coach asks you questions and drafts your entry in your words.

The files are the handouts from the talks, the slides plus my speaker notes. That gives it something specific to draw on, our course, instead of the internet in general. Add each week’s as you go.

This is honestly how I use chatbots myself. I give them as much about what I’m working on as I can, so when I need a strategic plan or a project summary, I don’t have to go and find everything first. It’s already there. Projects are one way to do that.

And one chat a week. The chat is your log: drop in a short update whenever something worth noting happens, while you still remember the details. Dictating by voice works fine. On Thursday, ask it to draft your entry and to ask you what’s missing first. Edit it, and paste it into Canvas. Canvas is still where you submit. [Click]

Why not one chat for all four weeks? Think back to the slide where the whole transcript goes back every turn. A month-long chat is a lot of words: it can outgrow the context window, and early material quietly drops out. A chat per week stays manageable. [Click]

And a bonus. With the handouts in the Project, you can ask about the lectures too. Give it context, the way you’d brief a tutor: what you’re doing, what you want, how you want it. “I’m reviewing Wednesday’s talk. Using that handout, quiz me one question at a time on the six words, and point me to the slide when I’m wrong.” Then check what it says against the slides, which is exactly today’s habit.

Step 1: get in, and create the Project

Go to **chatgpt.com** and log in with your **CU email**.

- Click your picture, bottom left. It must say **CU - Anschutz (Edu)**. If not, log out and log in with your CU email.
- In the sidebar, next to **Projects**, create a new project: *AIM learning journal*. **Memory**: *Default* is fine. Don't **Share** it.
- Start chats from the **project's own page**, not the main *New chat*, or they won't see your instructions and files.

Slides for copying text: **talks.seandavis.net**

HOW-TO, FROM THE SOURCE

- OpenAI: *Projects in ChatGPT*, the help article
- OpenAI Academy: *Using projects in ChatGPT*
- CU: getting-started videos and access

READ THE TERMS

Workspace data isn't used to train models. Check for yourself: [CU's FAQ](#) and [OpenAI's data-use page](#).

Step one is getting into the right place and making the project. I'm not going to walk you through the buttons with screenshots. The interface changes every few weeks, and OpenAI's own help pages change with it, so the links on the right will stay right longer than my pictures would. If you get stuck on where something is, use those, or ask ChatGPT itself.

Go to chatgpt dot com and log in with your CU email address. If you've used ChatGPT with a personal account, this matters most for you. Click your picture at the bottom left. The menu should say CU - Anschutz, Edu. If it shows your personal account, log out and log back in with your CU email. The reason isn't bureaucracy: your personal account isn't covered by the university's agreement about what happens to what you type. If you can't get in at all, CU asks students to complete a short AI training first; the CU link has the steps.

Then create the project. Why a Project rather than just a chat? Because a Project remembers your setup, the instructions and the files, for every chat you start inside it. You set it up once instead of re-explaining yourself every Thursday. Name it AIM learning journal, or anything you'll recognise.

You'll be asked about memory: default or project-only. Default means the project can use what ChatGPT already knows about you from other chats, and the other way round. That's usually what you want, since it already knows you're a medical student. Project-only keeps the project walled off, which is useful when you don't want its answers shaped by anything else you've told it. Remember the document the model actually reads: memory is one of its layers, just more context.

Don't share this project. Your journal is between you and the course director, and sharing changes what the project can access.

The most common slip: the main New chat button at the top of the sidebar starts a chat outside the project, and that chat won't see your instructions or your files. Start chats from the project's own page.

And read the terms. The footer says workspace data isn't used to train models. That's one real difference from a free personal account, where OpenAI may train on your chats unless you opt out. CU's agreement goes further and prohibits using any CU environment for training. Don't take my word, or the footer's, for it: the panel links CU's FAQ and OpenAI's own page on how data is used. Reading the terms of a tool before you trust it with your work is a habit worth starting now, and it would make a good journal entry. CU does approve this workspace for confidential data, but for this course, keep real patients out of your project anyway; the journal never needs them.

LIVE: About four minutes. Build the project on screen; wait until most of the room has the project page open.

Step 2: paste the coach into Instructions

Open **⋮ → Project settings → Instructions**. Copy this in (the copy button is at the top right of the box), fill in the brackets and your goals, and close the dialog.

About me: [program and year]; interested in [____];
AI experience: [none / some / a lot].

You are my learning-journal coach for a 4-week course on AI in medicine. Each Thursday I submit an entry answering these five questions:

1. What I chose to learn, and why (tie it to my goals)
2. How AI helped, and where it did not; one time I checked the AI: why, how, and what I found
3. What surprised me
4. What changed for me
5. What I used: about 3 resources with links, at least one I found myself, and whether each was worth it

When I give an update: reply briefly, then ask ONE question, often "Did you check anything the AI told you? How, and what did you find?"

When I'm unsure how to do something: ask what I'm trying to do, suggest 2-3 approaches, and say which you'd pick and why. Look things up first if that helps.

When I say "draft my entry": first ask me about anything missing. Then draft in my words; don't invent experiences or judgments. End with one line on how AI helped with the entry.

Remind me to keep real patients out of this project.

My goals:

- 1.
- 2.
- 3.

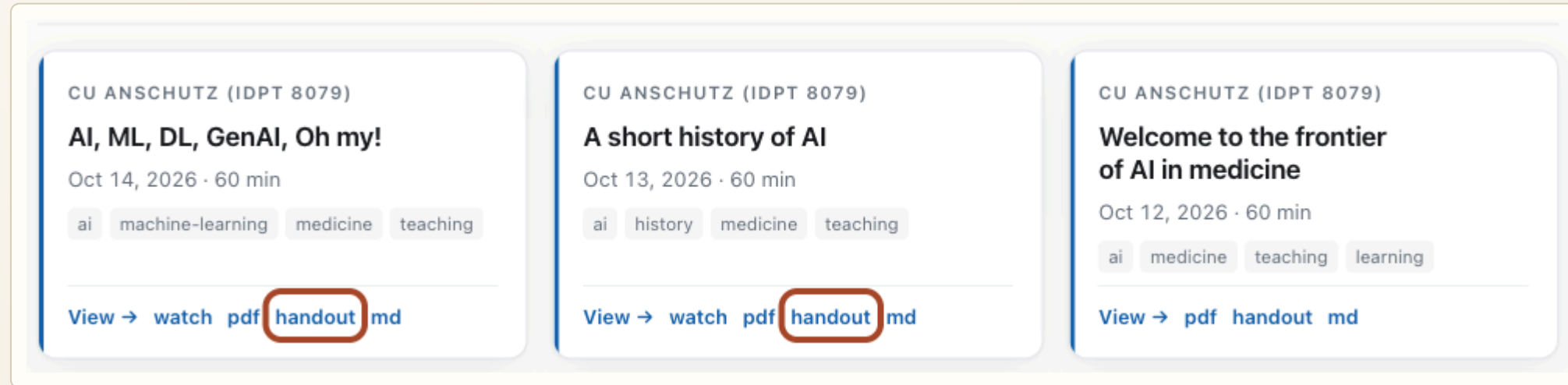
Step two, the instructions. Open the three-dot menu, then Project settings, and click into Instructions. Copy this text in; there’s a copy button at the top right of the box on the slide. Fill in the brackets about yourself.

Please read it before you paste it, because it’s the deal you’re making with the tool, and every line is there for a reason. It starts with who you are, because context changes everything the model writes; that’s the slide where the earlier words changed the next word. Then the five journal questions, straight from the course page, so it works from our questions instead of inventing its own. When you give an update, it replies briefly and asks one question, usually whether you checked anything the AI told you. That’s journal question two, and it’s far easier to answer on the day it happens than on Thursday night. When you’re unsure how to do something, it asks what you’re trying to do and suggests a few approaches with a recommendation, looking things up first if that helps. That one’s there because the hardest part of using these tools is often knowing what to ask. When you say “draft my entry”, it asks about what’s missing first, then drafts in your words without inventing anything. The experiences and judgments in the journal have to be yours, and it adds the one-line disclosure the course asks for. And it reminds you to keep real patients out.

No goals yet? That’s step four. Leave them blank for now.

LIVE: About four minutes.

Step 3: add the talk handouts



1. On talks.seandavis.net, click **handout** for Tuesday and Wednesday (direct: [Tuesday](#), [Wednesday](#)).
2. If the PDF opens in a browser tab, **save it** (it usually lands in *Downloads*).
3. In the Project, open **Sources** and add both files.

The **handout** has the slides *and* the speaker notes, so the chatbot gets the explanations. Add today's handout when it's posted, after class.

Speaker notes

Step three: give the Project the talks. This is the part that makes the coach useful beyond the journal, because it can now answer from our course materials instead of from the internet at large.

Go to talks dot seandavis dot net. Each talk has a card, and each card has a link labelled “handout”. That’s the one you want: it has the slides and my speaker notes, so the chatbot gets the explanations, not just the bullet points. The plain “pdf” link is slides only. Both handouts are also linked directly on this slide.

Your browser may open the PDF in a tab instead of downloading it. If so, use the browser’s download or save button; it usually lands in your Downloads folder. Then, in the project, open the Sources tab and add both files.

Today’s handout goes up on the same site after class. Add it then, and each week’s as we go.

In class I tried this live. Before the handout, I asked the new project what IDPT 8079 was. It searched, found a little, and mostly guessed. After I added one handout to Sources and asked again, it described the course in detail, tables and all. Same model, more context.

One thing to know about how it uses them. The chatbot searches these files when it thinks it needs them; it doesn’t hold every slide in mind at once. So if you want it to use a particular talk, say so: “using Wednesday’s handout, …”. And when it quotes the handout, you can open the handout and check, which is a lot easier than checking the internet.

LIVE: About three minutes.

Step 4: your first chat

In the Project's **New chat** box, give it **context**, then ask. Two ways to start:

NO GOALS YET?

"I'm a first-year [program] student, interested in [___]. I've used ChatGPT [never / for ___]. This course asks me to set three learning goals about AI in medicine. Before suggesting any, ask me a few questions, then offer options and say which you'd pick."

Then paste your goals into **Instructions**.

YOUR FIRST UPDATE

Tell it what you tried, **why**, and what you weren't sure about. Haven't tried anything yet? *"I'm studying [___] and have 20 minutes. Suggest two small things I could try with AI, and how I'd check what you tell me."*

Tonight (Week 1 is due today), in the same chat: *"Draft my Week 1 entry. First ask me what's missing."* Read it, fix it, paste it into **Canvas**. Rename the chat *Week 1* from its ⋮ menu.

Step four, your first chat. Use the project’s New chat box, the one on the project page, not the main New chat button.

This is the habit I’d most like you to leave with. Give it the context you’d give a new tutor: who you are, what you’re trying to do, and your constraints. If you don’t know how to start, ask it how it would approach the task. Short prompts get generic answers, the answer for an average person, and you aren’t an average person.

If you don’t have goals yet, start there. Something like: I’m a first-year medical student, interested in cardiology, or not sure yet. I’ve used ChatGPT for summaries. This course asks me to set three learning goals about AI in medicine. Before suggesting any, ask me a few questions about what I want, then offer options and tell me which you’d pick. Asking it to question you first is deliberate: it gets better information, and you do more of the thinking. Answer its questions, choose your three, and paste them into the project’s Instructions.

If you already have goals, give it your first update instead. Tell it what you tried, why, and what you weren’t sure about. For example: I’m studying renal physiology for next week’s exam. I asked ChatGPT to explain eGFR. It was clear, but it gave a formula I didn’t recognise, so I checked Wednesday’s slides, which use a different one. Was that a good check? How could I check better next time? And if you haven’t tried anything yet, say what you’re studying and ask it what to try.

This week is short, because the entry is due tonight. So tonight, in the same chat, say: draft my Week 1 entry, and ask me what’s missing first. Read it, fix anything that isn’t what you meant, and paste it into Canvas. From next week the rhythm is a few updates through the week, then the draft on Thursday.

Once you have a plan you like, two more moves. Ask it to check the plan: do some research and tell me whether this matches good practice for learning a new skill. And ask it to write the plan up as a document, then add that document to the project’s Sources, so every later chat can see it. It can also turn your material into slides, quizzes, or Anki cards.

Chats get automatic titles. Rename this one Week 1 from its three-dot menu so you can find it later.

And remember, the journal is graded for completion. It isn’t a high bar. It’s there to keep us in touch, and to give you a reason to use these tools every week. By taking this course you’ve volunteered to be someone your residency program, clinic, or hospital will look to on AI, and at this point using it is how you get there.

LIVE: If asked whether the thinking-effort setting (instant, medium, high) matters: higher means more thinking, potentially better reasoning on hard problems, and more tokens, which cost money and energy. Instant is faster and cheaper. ChatGPT usually picks a sensible level itself; raise it when you know a question needs deep thought.

LIVE: About five minutes. Then on to the rest of the hands-on.

Further exploration

WATCH: INSIDE THE MODEL (8 MIN)

Large Language Models explained briefly,

3Blue1Brown, 2024. Today's story, beautifully animated.

Captions on YouTube;

transcript at

3blue1brown.com/lessons/mini-llm.

READ: LLMS IN MEDICINE

Large language models in medicine, Nature Medicine, 2023. How these models are built, and what that means for clinical use (Thirunavukarasu et al., 2023).

USE: CHATGPT EDU AT CU

- *Projects in ChatGPT*, OpenAI's help article
- *Using projects in ChatGPT*, OpenAI Academy
- **CU's ChatGPT Edu page:** access, videos, and the FAQ on your data
- *How your data is used to improve model performance*, OpenAI

Speaker notes

For anyone who wants more. The 3Blue1Brown video tells today’s story in eight minutes, with far better animation than mine, and it’s the one I’d watch first. The Nature Medicine review goes further into how these models are built and what that means for clinical use. The third panel is the practical one: OpenAI’s own guides to Projects, which stay current as the interface changes, and the terms. CU’s page covers access, short getting-started videos, and what happens to your data; OpenAI’s data-use page explains how personal accounts differ.

LIVE: Skip in a live session.

References

- Brown, T. B., Mann, B., Ryder, N., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems* 33. <https://arxiv.org/abs/2005.14165>
- Chatterji, A., Cunningham, T., Deming, D. J., Hitzig, Z., Ong, C., Shan, C. Y., & Wadman, K. (2025). *How people use ChatGPT* (Working Paper No. 34255). National Bureau of Economic Research. <https://www.nber.org/papers/w34255>
- Llama Team, AI @ Meta. (2024). *The llama 3 herd of models*. arXiv:2407.21783. <https://arxiv.org/abs/2407.21783>
- Metlay, J. P., Waterer, G. W., Long, A. C., et al. (2019). Diagnosis and treatment of adults with community-acquired pneumonia. An official clinical practice guideline of the american thoracic society and infectious diseases society of america. *American Journal of Respiratory and Critical Care Medicine*, 200(7), e45–e67. <https://doi.org/10.1164/rccm.201908-1581ST>
- Omiye, J. A., Lester, J. C., Spichak, S., Rotemberg, V., & Daneshjou, R. (2023). Large language models propagate race-based medicine. *Npj Digital Medicine*, 6, 195. <https://doi.org/10.1038/s41746-023-00939-z>
- Ouyang, L., Wu, J., Jiang, X., et al. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35. <https://arxiv.org/abs/2203.02155>
- Thirunavukarasu, A. J., Ting, D. S. J., Elangovan, K., Gutierrez, L., Tan, T. F., & Ting, D. S. W. (2023). Large language models in medicine. *Nature Medicine*, 29(8), 1930–1940. <https://doi.org/10.1038/s41591-023-02448-8>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems* 30. <https://arxiv.org/abs/1706.03762>

Questions

Speaker notes

Let's take questions.

